MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

Order this document
by MPC505TS/D Rev. 1

MPC505

Technical Summary

PowerPC™ MPC505 RISC Microcontroller

The MPC505 is the charter member of the PowerPC Family of reduced instruction set computer (RISC) microcontrollers (MCUs). The MPC505 implements the 32-bit portion of the PowerPC architecture, which provides 32-bit effective addresses, integer data types of 8, 16, and 32 bits, and floating-point data types of 32 and 64 bits.

The RISC MCU processor (RCPU) integrates four execution units: an integer unit (IU), a load/store unit (LSU), a branch processing unit (BPU), and a floating-point unit (FPU). The RCPU is capable of issuing one sequential (non-branch) instruction per clock. In addition, branch instructions are evaluated ahead of time when possible, resulting in zero-cycle execution time for many branch instructions. Instructions can complete out of order for increased performance; however, the MPC505 makes them appear sequential.

The MPC505 includes an on-chip, 4-Kbyte, two-way set associative, physically addressed instruction cache, chip-select logic to reduce or eliminate external decoding logic, 4 Kbytes of static RAM, and extensive processor debugging functionality.

The MPC505 has a high-bandwidth, 32-bit data bus and a 32-bit address bus. The MCU supports 16-bit and 32-bit memories and both single-beat and burst data memory accesses.

The MPC505 uses an advanced, 3.3-V CMOS process technology and maintains full interface compatibility with TTL devices.

MOTOROLA

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

1 INTRODUCTION

1.1 Features

- Fully-Integrated Single-Chip Microcontroller
- RISC MCU Central Processing Unit (RCPU)
  - — 32-Bit PowerPC Architecture (Compliant with PowerPC Architecture Book 1)
  - — Single-Issue Processor
  - — Integrated Floating-Point Unit
  - — Branch Prediction for Prefetch
  - — 32 Bit x 32 Bit General-Purpose Register File
  - — 32 Bit x 64 Bit Floating-Point Register File
  - — Precise Exception Model
  - — Internal Harvard Architecture: Load/Store Bus (L-Bus), Instruction Bus (I-Bus)
  - — PowerPC Time Base and Decrementer
- System Interface Unit (SIU)
  - — Chip-Select Logic to Reduce or Eliminate External Decoding Logic
  - — External Bus Interface (EBI) that Supports Synchronous, Asynchronous, Burst Transfer, and Pipeline Transfer Memory Types
  - — System Protection Features Including Bus Monitor and Periodic Interrupt Timer
  - — On-Chip Phase-Locked Loop (PLL), 16 MHz to 44 MHz
  - — Five Dual-Purpose I/O Ports, Two Dual-Purpose Output Ports
- Peripheral Control Unit (PCU)
  - — Software Watchdog
  - — Interrupt Controller to Manage External and Internal Interrupts to the CPU
  - — Dual-Purpose I/O Port
  - — L-Bus IMB Interface (LIMB) Connecting L-Bus to Intermodule Bus 2 (IMB2)
- 4-Kbyte On-Chip Instruction Cache (I-Cache)
- 4-Kbyte On-Chip Static Data RAM (SRAM)
- 3.3-V Supply Voltage
- Tolerates Input Signals from 5-V Peripherals

1.2 Block Diagram

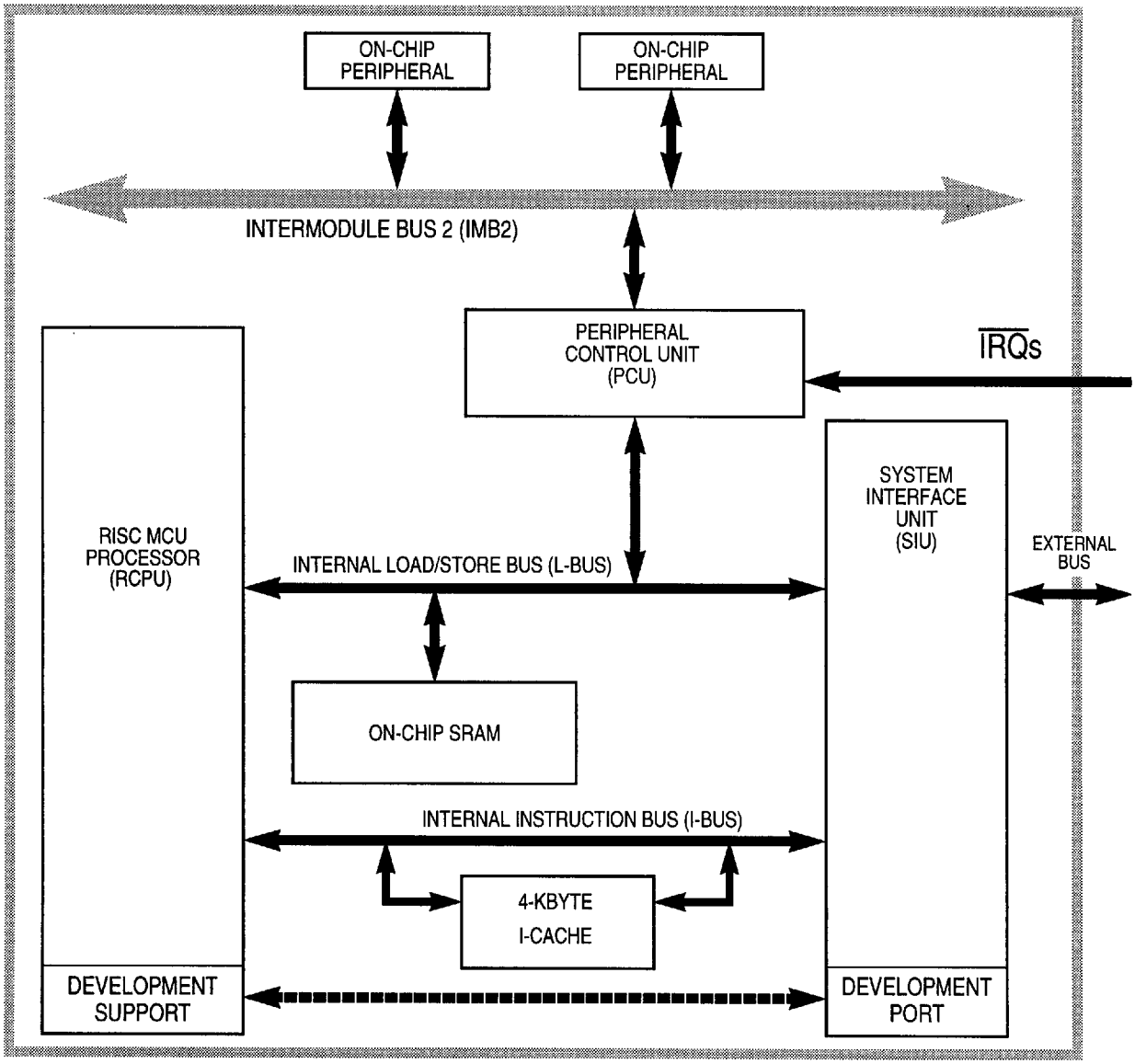


Figure 1 MPC505 Block Diagram

Notice in Figure 1 that the IMB2 connects the processor to any on-chip peripherals. Although the MPC505 has no on-chip peripherals, the diagram is intended to show the operation of the IMB2 across the MPC family.

1.3 Pin Connections

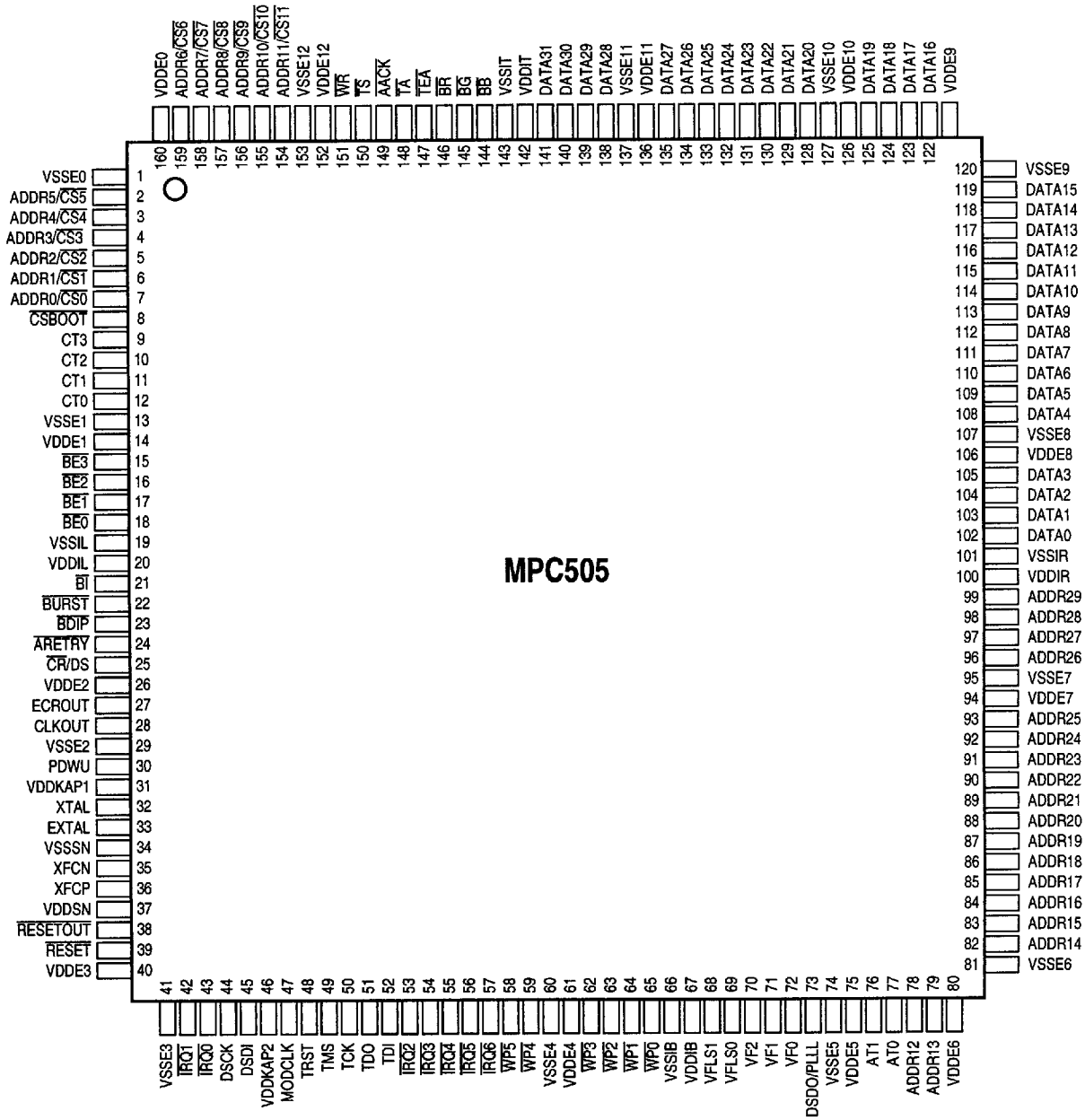


Figure 2 MPC505 Pin Assignments

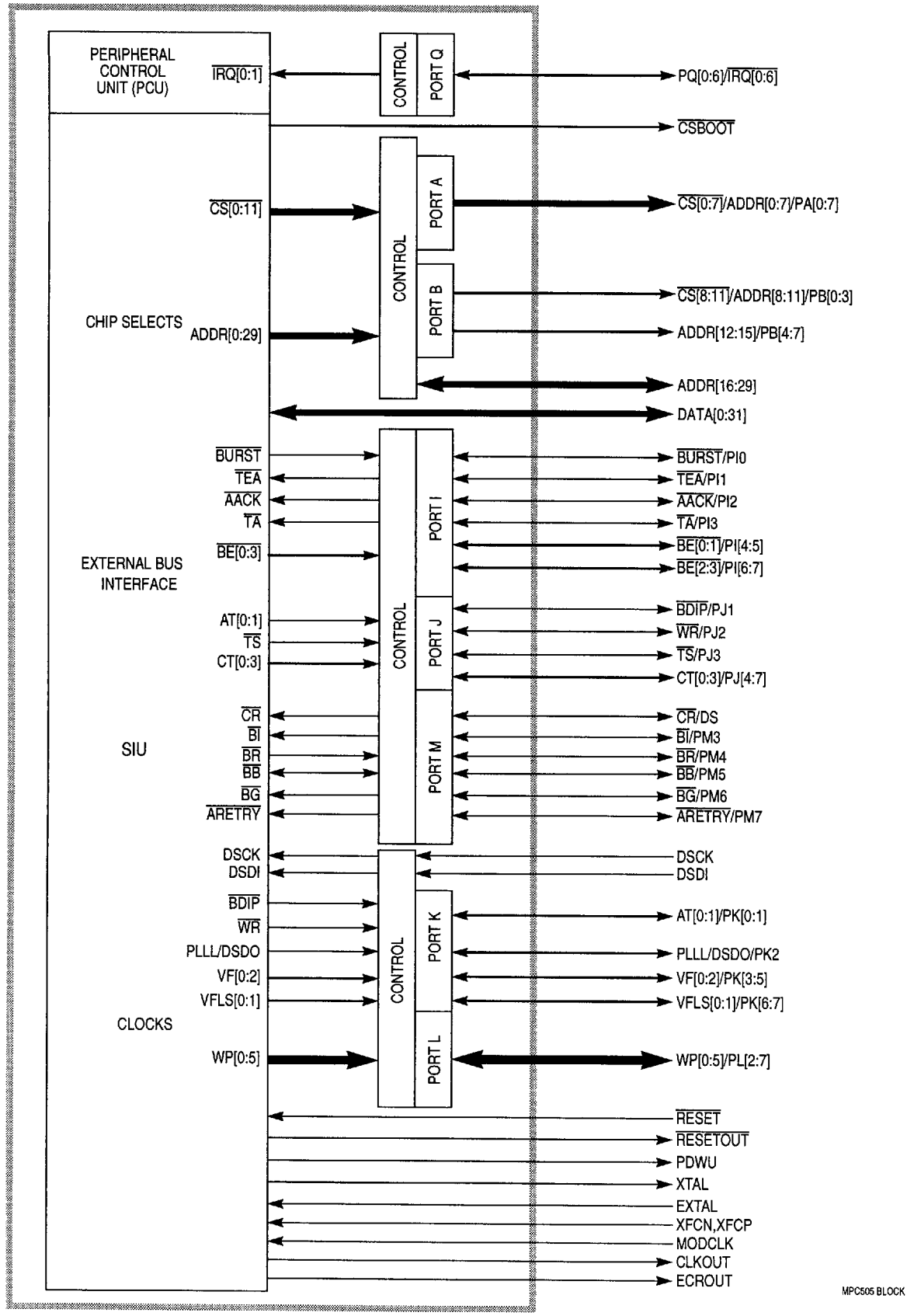


Figure 3 MPC505 Signals

1.4 Memory Map

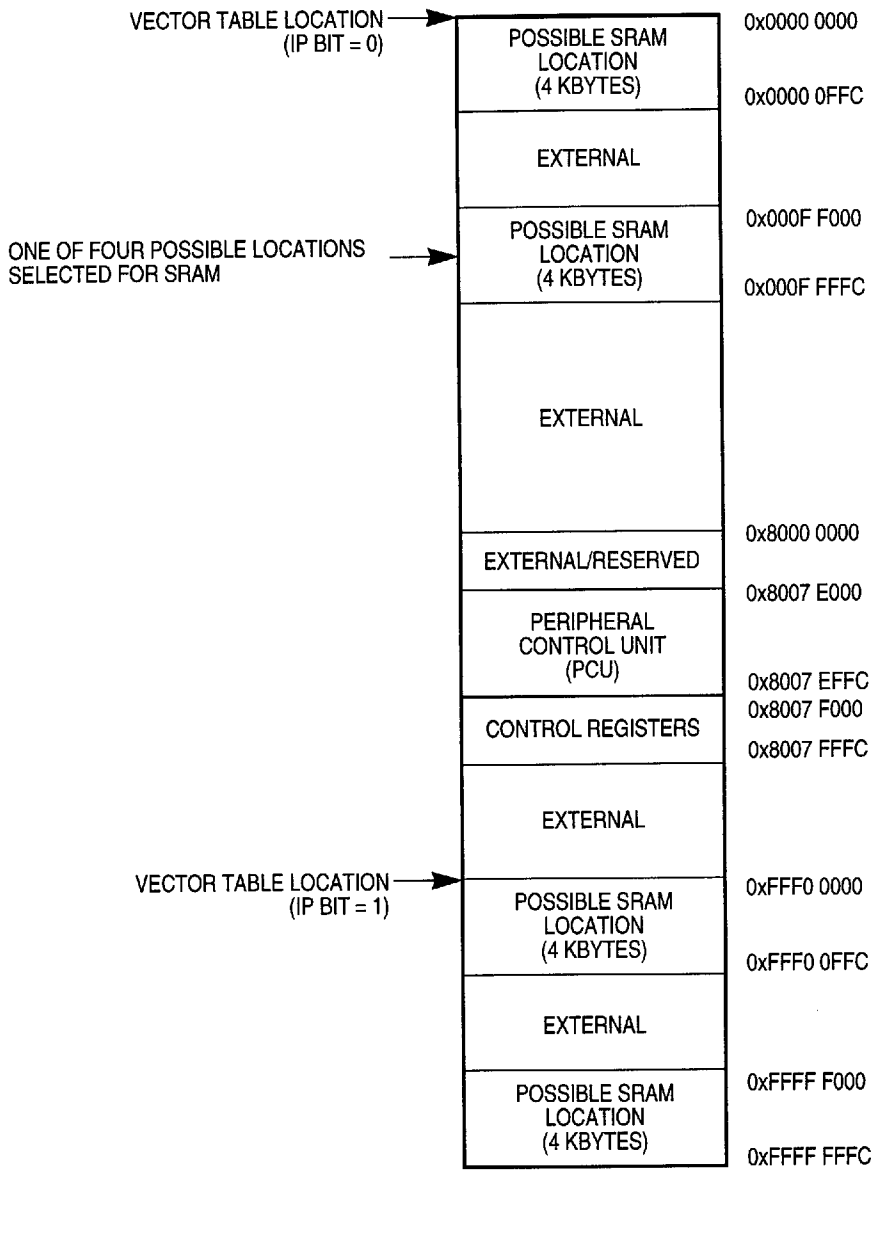


Figure 4 MPC505 Memory Map

The MPC family has a unified memory map including instruction memory (I-Mem), load/store memory (L-Mem), and all memory-mapped registers. I-Mem resides on the instruction bus; L-Mem resides on the load/store bus. The locations of I-Mem and L-Mem are selected in the MEMMAP register located in the SIU. In the MPC505, the SRAM module serves as L-Mem. The MPC505 has no I-Mem module.

2 SIGNAL DESCRIPTIONS

Table 1 MPC505 Pin List

Primary Function(s)	Port Function
Address Bus, Data Bus, Chip Selects	
ADDR[0:11]/$\overline{\text{CS}}$[0:11]	PA[0:7], PB[0:3]
ADDR[12:15]	PB[4:7]
ADDR[16:29]	—
DATA[0:31]	—
$\overline{\text{CSBOOT}}$	—
Bus Control, Clock, Development Support	
$\overline{\text{BURST}}$, $\overline{\text{TEA}}$, $\overline{\text{AACK}}$, $\overline{\text{TA}}$, $\overline{\text{BE}}$[0:1], $\overline{\text{BE2}}$/ADDR30, $\overline{\text{BE3}}$	PI[0:7]
AT[0:1], $\overline{\text{TS}}$, CT[0:3]	PJ[1:7]
$\overline{\text{BDIP}}$, $\overline{\text{WR}}$, PLLL/DSDO, VF[0:2], VFLS[0:1]	PK[0:7]
$\overline{\text{WP}}$[0:5]	PL[2:7]
$\overline{\text{BI}}$, $\overline{\text{BR}}$, $\overline{\text{BB}}$, $\overline{\text{BG}}$, $\overline{\text{ARETRY}}$	PM[3:7]
$\overline{\text{CR}}$/DS	—
DSCK, DSDI	—
XTAL, EXTAL, XFCN, XFCP, CLKOUT, ECROUT, PDWU, MODCLK	—
Reset, Interrupts	
$\overline{\text{RESET}}$, $\overline{\text{RESETOUT}}$	—
$\overline{\text{IRQ}}$[0:6]	PQ[0:6]
Test	
TDI, TDO, TCK, TMS, TRST	—
Power	
V_{DDSN}, V_{SSSN}	—
V_{DDI}, V_{SSI}	—
V_{DDE}, V_{SSE}	—
VDDKAP1, VDDKAP2	—

Table 2 Signal Descriptions

Mnemonic	Module	Direction	Description
ADDR[0:29]	EBI	Output	32-bit address bus. Driven by the bus master to index the bus slave. Low-order bit (ADDR31) not pinned out — use byte enables instead. BE2 functions as ADDR30 during accesses to 16-bit ports.
$\overline{\text{AACK}}$	EBI	Input	Address acknowledge. When asserted, indicates the slave has received the address from the bus master.
$\overline{\text{ARETRY}}$	EBI	Input	When asserted, indicates the master needs to retry its address phase.
AT[0:1]	EBI	Output	Address types. Define addressed space as user or supervisor, data or instruction.
$\overline{\text{BB}}$	EBI	Input/Output	Bus busy. Asserted by current bus master to indicate the bus is currently in use. Prospective new master should wait until the current master negates this signal.
$\overline{\text{BDIP}}$	EBI	Output	Burst data in progress. Asserted at the beginning of a burst cycle and negated prior to the last beat. This signal can be negated prior to the end of a burst to terminate the burst data phase early.
$\overline{\text{BE[0:3]}}$	EBI	Output	Byte enables. One byte enable per byte lane of the data bus.
$\overline{\text{BG}}$	EBI	Input	Bus grant. When asserted by bus arbiter, the bus is granted to the bus master. Each master has its own bus grant signal.
$\overline{\text{BI}}$	EBI	Input	Burst inhibit. When asserted, indicates the slave does not support burst mode.
$\overline{\text{BR}}$	EBI	Output	Bus request. When asserted, indicates the potential bus master is requesting the bus. Each master has its own bus request signal.
$\overline{\text{BURST}}$	EBI	Output	If asserted, indicates cycle is a burst cycle.
CLKOUT	EBI	Output	Continuously-running clock. All signals driven on the E-bus must be synchronized to the rising edge of this clock.
$\overline{\text{CR}}$	EBI	Input	Cancel reservation. Each RCPU has its own $\overline{\text{CR}}$ signal. When asserted, instructs the bus master to clear its reservation.
$\overline{\text{CSBOOT}}$	Chip Selects	Output	Chip select of system boot memory.
$\overline{\text{CS[0:11]}}$	Chip Selects	Output	Chip-select signals for external memory devices.
CT[0:3]	EBI	Output	Cycle type signals. Indicate what type of bus cycle the bus master is initiating.
DATA[0:31]	EBI	Input/Output	32-bit data bus.
$\overline{\text{DS}}$	EBI	Output	Data strobe. Asserted by EBI at the end of a chip-select-controlled bus cycle after the chip-select unit asserts the internal $\overline{\text{TA}}$ signal or the bus monitor timer asserts the internal $\overline{\text{TEA}}$ signal. Also asserted at the end of a show cycle. Used primarily by development tools.
DSCK	Dev. Support	Input	Development serial clock. Used to clock data shifted into or out of development serial port.

Table 2 Signal Descriptions (Continued)

Mnemonic	Module	Direction	Description
DSDI	Dev. Support	Input	Development serial data in. Used to shift development serial data into the development port shift register.
DSDO	Dev. Support	Output	Development serial data out. Used to shift development serial data out of the development port shift register.
ECROUT	Clocks	Output	Provides a clock reference output with a frequency equal to the crystal oscillator frequency divided by four (i.e., 1 MHz with a 4-MHz crystal oscillator frequency).
EXTAL	Clocks	Input	Connection for external crystal to the internal oscillator circuit, or clock input.
$\overline{\text{IRQ[0:6]}}$	PCU	Input	Interrupt request inputs.
MODCLK	Clocks	Input	Clock mode. The state of this signal and that of V_{DDSN} during reset determine the source of the system clock (normal operation, 1:1 mode, PLL bypass mode, or special test mode). Refer to Table 54 in **5 System Interface Unit** for details.
PA[0:7]	Ports	Output	Port A discrete output signals.
PB[0:7]	Ports	Output	Port B discrete output signals.
PDWU	Clocks	Output	Power-down wakeup to external power-on reset circuit.
PI[0:7]	Ports	Input/Output	Port I discrete input/output signals.
PJ[0:7]	Ports	Input/Output	Port J discrete input/output signals.
PK[0:7]	Ports	Input/Output	Port K discrete input/output signals.
PL[2:7]	Ports	Input/Output	Port L discrete input/output signals.
PM[3:7]	Ports	Input/Output	Port M discrete input/output signals.
PQ[0:6]	PCU	Input/Output	Port Q discrete input/output signals.
PLLL	Clock	Output	Indicates whether phase-locked loop is locked.
$\overline{\text{RESET}}$	EBI	Input	Hard reset. When asserted, devices on the bus must reset.
$\overline{\text{RESETOUT}}$	EBI	Output	Reset output signal. Asserted by MCU during reset. When asserted, instructs all devices monitoring this signal to reset all parts within themselves that can be reset by software.
$\overline{\text{TA}}$	EBI	Input	Transfer acknowledge. When asserted, indicates the slave has received the data during a write cycle or returned the data during a read cycle.
TCK	Test	Input	Test clock input with a pull-down resistor to synchronize the test logic.
TDI	Test	Input	Test data input with a pull-up resistor sampled on the rising edge of TCK.
TDO	Test	Output	Three-statable test data output that changes on the falling edge of TCK.
$\overline{\text{TEA}}$	EBI	Input	Transfer error acknowledge. Asserted by an external device to signal a bus error condition.

Table 2 Signal Descriptions (Continued)

Mnemonic	Module	Direction	Description
TMS	Test	Input	Test mode select input with a pull-up resistor. Sampled on the rising edge of TCK to sequence the test controller's state machine.
TRST	Test	Input	Asynchronous active-low test reset with a pull-up resistor that provides initialization of the TAP controller and other logic as required by the standard.
$\overline{TS}$	EBI	Output	Transfer start. When asserted, indicates the start of a bus cycle.
V_{DDSN}	Clocks	Input	Power supply input to the VCO. In addition, the state of this signal and that of MODCLK during reset determine the source of the system clock (normal operation, 1:1 mode, PLL bypass mode, or special test mode). Refer to Table 54 in **5 System Interface Unit** for details
VF[0:2]	Dev. Support	Output	Denotes the last fetched instruction or how many instructions were flushed from the instruction queue.
VFLS[0:1]	Dev. Support	Output	Denotes how many instructions are flushed from the history buffer during the current clock cycle. Also indicates freeze state.
V_{SSSN}	Clocks	Input	Power ground input to the VCO.
WP[0:5]	Dev. Support	Output	Output signals for I-bus watchpoints (WP[0:3]) and L-bus watchpoints (WP[4:5]).
XTAL	Clocks	Output	Connection for external crystal to the internal oscillator circuit.
$\overline{WR}$	EBI	Output	Asserted: write cycle. Negated: read cycle.
XFCN, XFCP	Clocks	Input	Used to add an external capacitor to the filter circuit of the phase-locked loop.

Table 3 MPC505 Power Connections

Pin	Description
V_{DDE}, V_{SSE}	External periphery power
V_{DDI}, V_{SSI}	Internal module power
V_{DDSN}, V_{SSSN}	Clock synthesizer power
VDDKAP1	Keep-alive power for the internal oscillator, time base, and decrementer
VDDKAP2	Keep-alive power for the SRAM array

3 CENTRAL PROCESSING UNIT

The PowerPC-based RISC processor (RCPU) used in the MPC500 family of microcontrollers integrates four execution units: an integer unit (IU), a load/store unit (LSU), a branch processing unit (BPU), and a floating-point unit (FPU). The use of simple instructions with rapid execution times yields high efficiency and throughput for MPC505-based systems.

Most integer instructions execute in one clock cycle. The FPU is designed to provide cost-effective solutions to most mathematical problems. It includes single- and double-precision multiply-add instructions. Instructions can complete out of order for increased performance; however, the processor makes execution appear sequential.

The MPC505 includes an on-chip, 4-Kbyte, two-way set-associative instruction cache. The cache uses a least recently used (LRU) replacement algorithm.

3.1 RCPU Features

Major features of the RCPU include the following:

- High-performance microprocessor
  - — Single clock-cycle execution for many instructions
- Four independent execution units and two register files
  - — Independent LSU for load and store operations
  - — BPU featuring static branch prediction
  - — A 32-bit IU
  - — Fully IEEE 754-compliant FPU for both single- and double-precision operations
  - — Thirty-two general-purpose registers (GPRs) for integer operands
  - — Thirty-two floating-point registers (FPRs) for single- or double-precision operands
- Facilities for enhanced system performance
  - — Programmable big- and little-endian byte ordering
  - — Atomic memory references
- In-system testability and debugging features through boundary-scan capability
- High instruction and data throughput
  - — Condition register (CR) look-ahead operations performed by BPU
  - — Branch-folding capability during execution (zero-cycle branch execution time)
  - — Programmable static branch prediction on unresolved conditional branches
  - — A prefetch queue that can hold up to four instructions, providing look-ahead capability
  - — Interlocked pipelines with feed-forwarding that control data dependencies in hardware
  - — 4-Kbyte instruction cache: two-way set-associative, LRU replacement algorithm

3.2 RCPU Block Diagram

Figure 5 provides a block diagram of the RCPU.

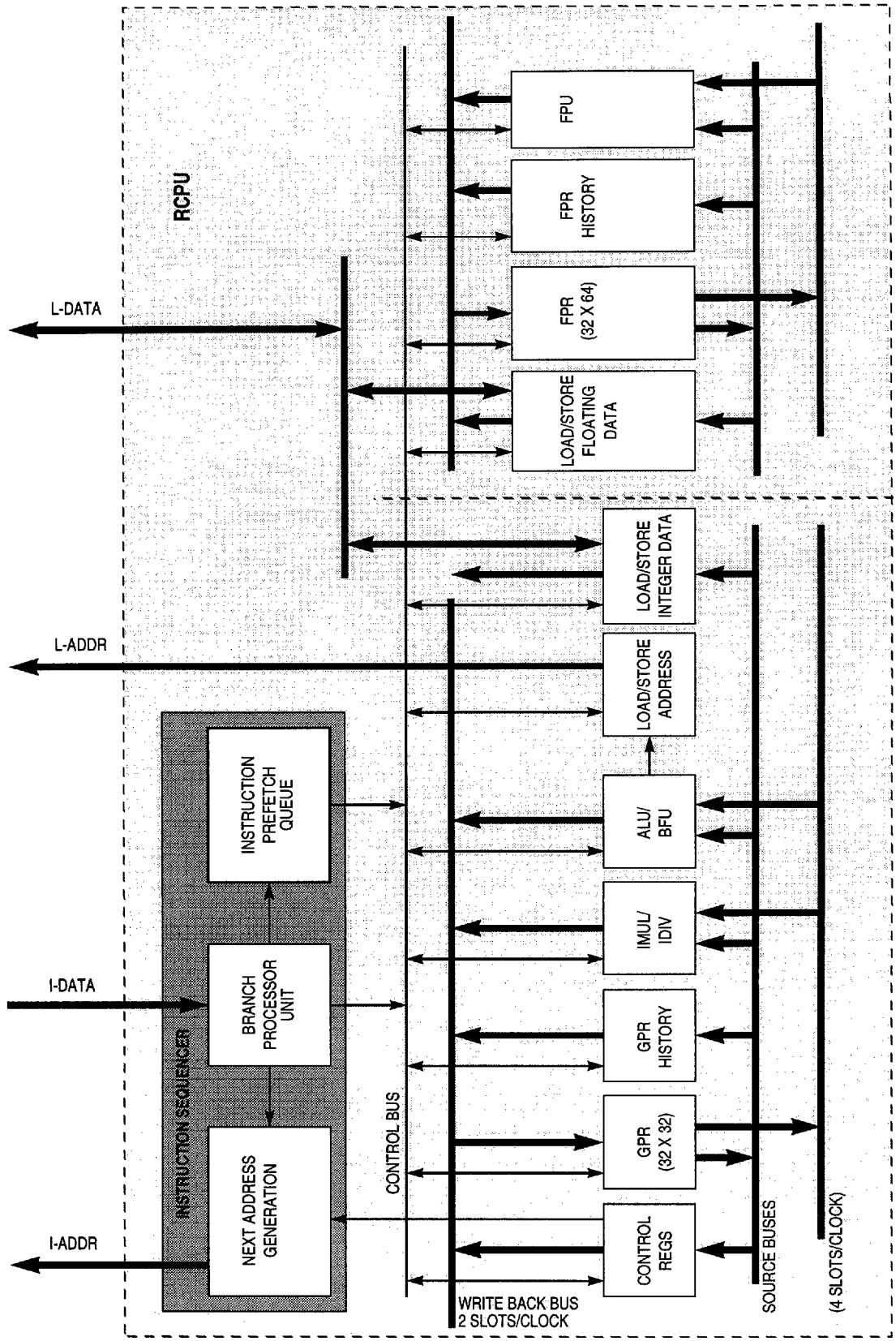


Figure 5 RCPU Block Diagram

3.3 Instruction Sequencer

The instruction sequencer provides centralized control over data flow between execution units and register files. The sequencer implements the basic instruction pipeline, fetches instructions from the memory system, issues them to available execution units, and maintains a state history so it can back the machine up in the event of an exception.

The instruction sequencer fetches the instructions from the instruction cache into the instruction pre-fetch queue. The BPU extracts branch instructions from the pre-fetch queue and uses static branch prediction on unresolved conditional branches to allow the instruction unit to fetch instructions from a predicted target instruction stream while a conditional branch is evaluated. The BPU folds out branch instructions for unconditional branches or conditional branches unaffected by instructions in the execution stage.

Instructions issued beyond a predicted branch do not complete execution until the branch is resolved, preserving the programming model of sequential execution. If branch prediction is incorrect, the instruction unit flushes all predicted path instructions, and instructions are issued from the correct path.

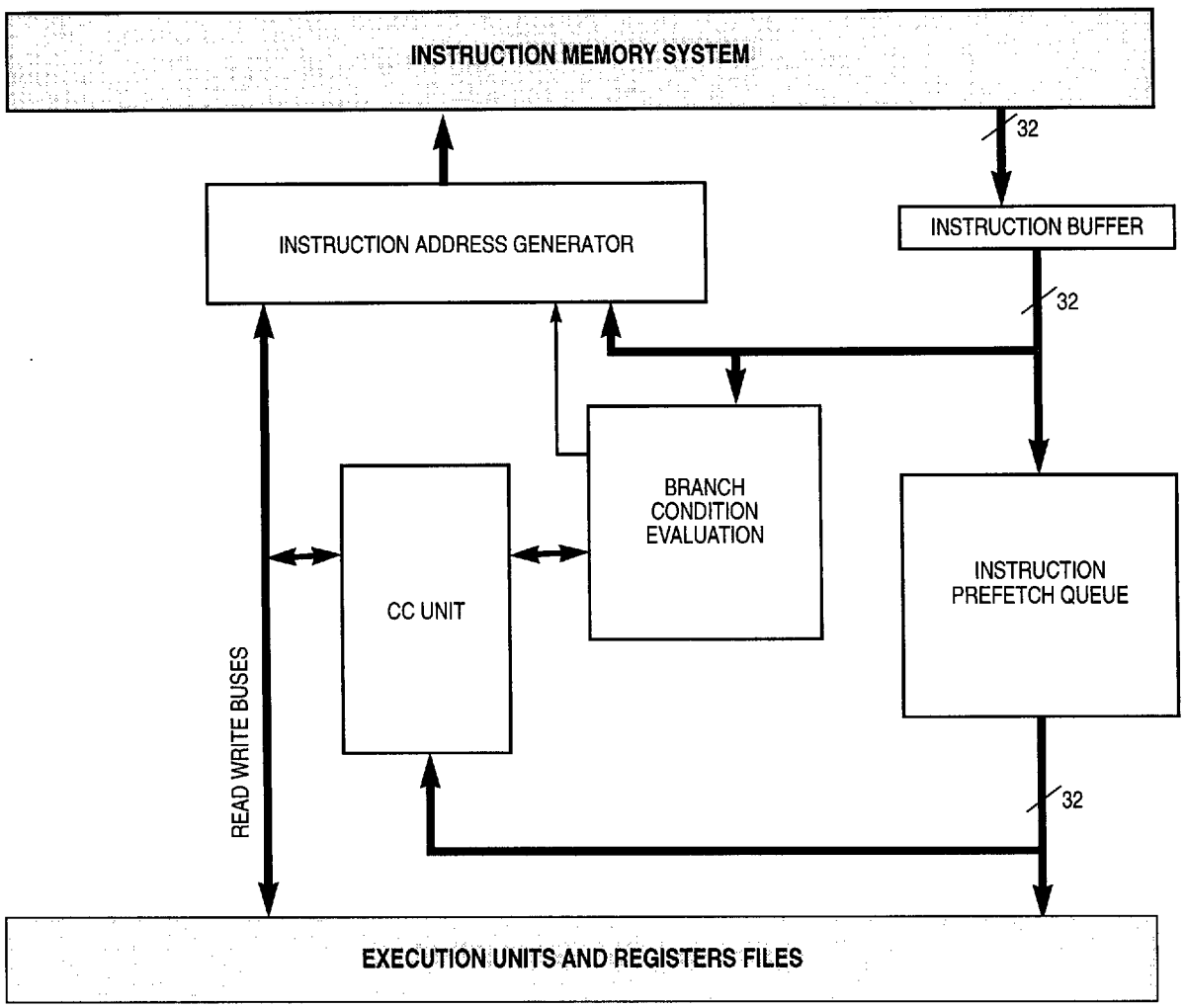


Figure 6 Sequencer Data Path

3.4 Independent Execution Units

The PowerPC architecture supports independent floating-point, integer, load-store, and branch processing execution units, making it possible to implement advanced features such as look-ahead operations. For example, since branch instructions do not depend on GPRs or FPRs, branches can often be resolved early, eliminating stalls caused by taken branches.

Table 4 summarizes the RCPU execution units.

Table 4 RCPU Execution Units

Unit	Description
Branch processing unit (BPU)	Includes the implementation of all branch instructions.
Load/store unit (LSU)	Includes implementation of all load and store instructions, whether defined as part of the integer processor or the floating-point processor.
Integer unit (IU)	Includes implementation of all integer instructions except load-store instructions. This module includes the GPRs (including GPR history and scoreboard) and the following subunits: The IMUL-IDIV includes the implementation of the integer multiply and divide instructions. The ALU-BFU includes implementation of all integer logic, add and subtract instructions, and bit field instructions.
Floating-point unit (FPU)	Includes the FPRs (including FPR history and scoreboard) and the implementation of all floating-point instructions except load and store floating-point instructions.

The following sections describe the execution units in greater detail.

3.4.1 Branch Processing Unit (BPU)

The BPU, located within the instruction sequencer, performs condition register look-ahead operations on conditional branches. The BPU looks through the instruction queue for a conditional branch instruction and attempts to resolve it early, achieving the effect of a zero-cycle branch in many cases.

The BPU uses a bit in the instruction encoding to predict the direction of the conditional branch. Therefore, when an unresolved conditional branch instruction is encountered, the processor prefetches instructions from the predicted target stream until the conditional branch is resolved.

The BPU contains an adder to compute branch target addresses and three special-purpose, user-accessible registers: the link register (LR), the count register (CTR), and the condition register (CR). The BPU calculates the return pointer for subroutine calls and saves it into the LR. The LR also contains the branch target address for the branch conditional to link register (**bclr*x***) instruction. The CTR contains the branch target address for the branch conditional to count register (**bcctr*x***) instruction. The contents of the LR and CTR can be copied to or from any GPR. Because the BPU uses dedicated registers rather than general-purpose or floating-point registers, execution of branch instructions is independent from execution of integer and floating-point instructions.

3.4.2 Integer Unit (IU)

The IU executes all integer processor instructions, except the integer storage access instructions, which are implemented by the load/store unit. The IU contains the following subunits:

- The IMUL–IDIV unit includes the implementation of the integer multiply and divide instructions.
- The ALU–BFU unit includes the implementation of all integer logic, add and subtract, and bit field instructions.

The IU also includes the integer exception register (XER) and the general-purpose register file.

IMUL–IDIV and ALU–BFU are implemented as separate execution units. The ALU–BFU unit can execute one instruction per clock cycle. IMUL–IDIV instructions require multiple clock cycles to execute. IMUL–IDIV is pipelined for multiply instructions, so that consecutive multiply instructions can be issued on consecutive clock cycles. Divide instructions are not pipelined; an integer divide instruction preceded or followed by an integer divide or multiply instruction results in a stall in the processor pipeline. Note that since IMUL–IDIV and ALU–BFU are implemented as separate execution units, an integer divide instruction preceded or followed by an ALU–BFU instruction does not cause a delay in the pipeline.

3.4.3 Load/Store Unit (LSU)

The load-store unit handles all data transfer between the general-purpose and floating-point register files and the internal load/store bus (L-bus). The load/store unit is implemented as an independent execution unit so that stalls in the memory pipeline do not cause the master instruction pipeline to stall (unless there is a data dependency). The unit is fully pipelined so that memory instructions of any size may be issued on back-to-back cycles.

There is a 32-bit wide data path between the load/store unit and the general-purpose register file and a 64-bit wide data path between the load/store unit and the floating-point register file. Single-word accesses to the internal on-chip data RAM require one clock, resulting in two clocks latency. Double-word accesses require two clocks, resulting in three clocks latency. Since the L-bus is 32 bits wide, double-word transfers require two bus accesses. The load/store unit performs zero-fill for byte and half-word transfers and sign extension for half-word transfers.

Addresses are formed by adding the source one register operand specified by the instruction (or zero) to either a source two register operand or to a 16-bit, immediate value embedded in the instruction.

3.4.4 Floating-Point Unit (FPU)

The FPU contains a double-precision multiply array, the floating-point status and control register (FPSCR), and the FPRs. The multiply-add array allows the MPC505 to efficiently implement floating-point operations such as multiply, multiply-add, and divide.

The MPC505 depends on a software envelope to fully implement the IEEE floating-point specification. Overflows, underflows, NaNs, and denormalized numbers cause floating-point assist exceptions that invoke a software routine to deliver (with hardware assistance) the correct IEEE result.

To accelerate time-critical operations and make them more deterministic, the MPC505 provides a mode of operation that avoids invoking the software envelope and attempts to deliver results in hardware that are adequate for most applications, if not in strict conformance with IEEE standards. In this mode, denormalized numbers, NaNs, and IEEE invalid operations are treated as legitimate, returning default results rather than causing floating-point assist exceptions.

3.5 Levels of the PowerPC Architecture

The PowerPC architecture consists of three layers. Adherence to the PowerPC architecture can be measured in terms of which of the following levels of the architecture are implemented:

- PowerPC user instruction set architecture (UISA) — Defines the base user-level instruction set, user-level registers, data types, floating-point exception model, memory models for a uniprocessor environment, and programming model for a uniprocessor environment.
- PowerPC virtual environment architecture (VEA) — Describes the memory model for a multiprocessor environment, defines cache control instructions, and describes other aspects of virtual environments. Implementations that conform to the VEA also adhere to the UISA, but may not necessarily adhere to the OEA.
- PowerPC operating environment architecture (OEA) — Defines the memory management model, supervisor-level registers, synchronization requirements, and the exception model. Implementations that conform to the OEA also adhere to the UISA and the VEA.

3.6 RCPU Programming Model

The PowerPC architecture defines register-to-register operations for most computational instructions. Source operands for these instructions are accessed from the registers or are provided as immediate values embedded in the instruction opcode. The three-register instruction format allows specification of a target register distinct from the two source operands. Load and store instructions transfer data between memory and on-chip registers.

PowerPC processors have two levels of privilege: supervisor mode of operation (typically used by the operating environment) and user mode of operation (used by the application software). The programming models incorporate 32 GPRs, 32 FPRs, special-purpose registers (SPRs), and several miscellaneous registers.

Supervisor-level access is provided through the processor's exception mechanism. That is, when an exception is taken (either due to an error or problem that needs to be serviced, or deliberately through the use of a trap instruction), the processor begins operating in supervisor mode. The level of access is indicated by the privilege-level (PR) bit in the machine state register (MSR).

Figure 7 shows the user-level and supervisor-level RCPU programming models and also illustrates the three levels of the PowerPC architecture. The numbers to the left of the SPRs indicate the decimal number that is used in the syntax of the instruction operands to access the register.

Note that registers such as the general-purpose registers (GPRs) and floating-point registers (FPRs) are accessed through operands that are part of the instructions. Access to registers can be explicit (that is, through the use of specific instructions for that purpose such as Move to Special-Purpose Register (**mtspr**) and Move from Special-Purpose Register (**mfspr**) instructions) or implicitly as the part of the execution of an instruction. Some registers are accessed both explicitly and implicitly.

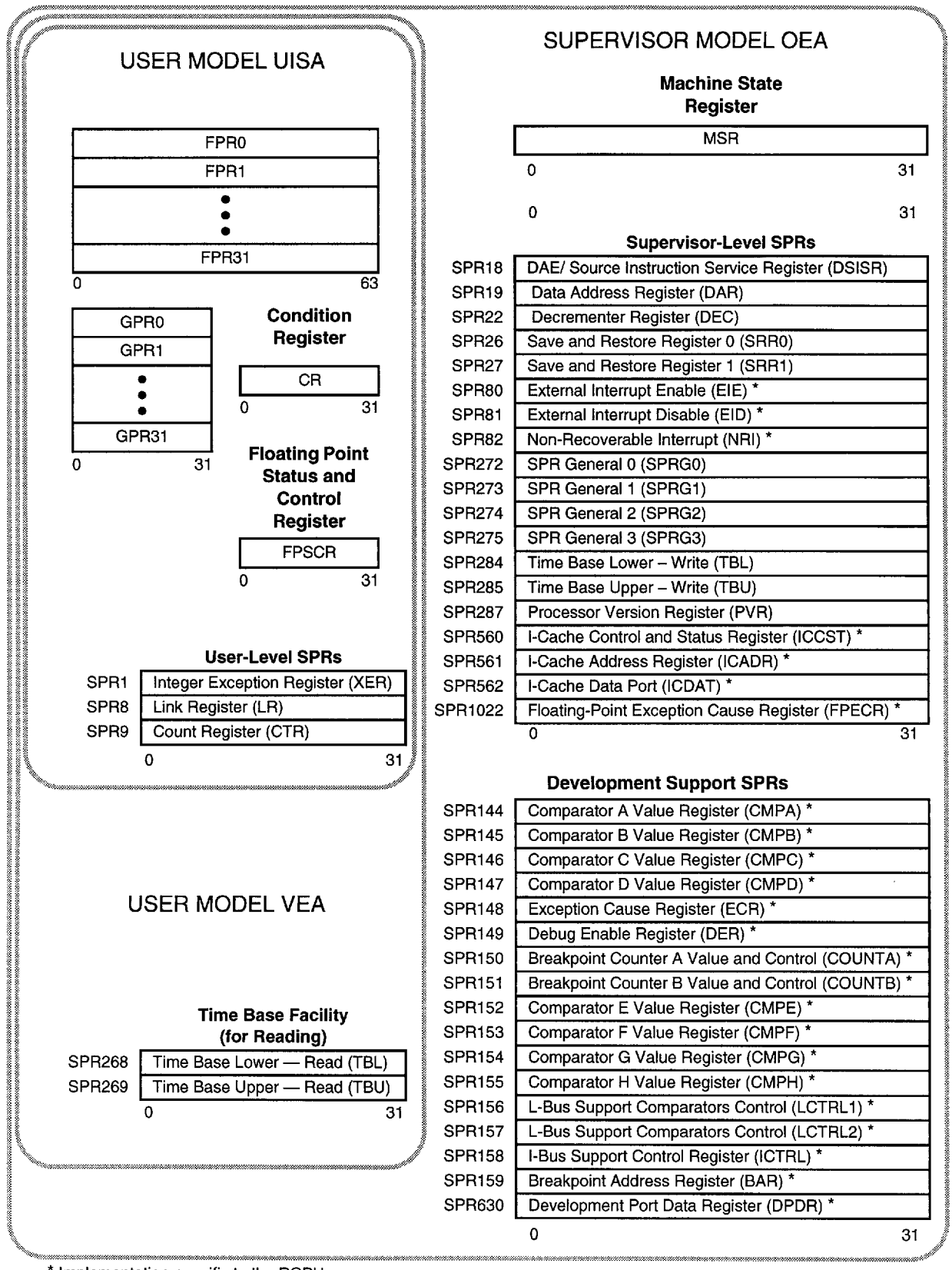


Figure 7 RCPU Programming Model

Where not otherwise noted, reserved fields in registers are ignored when written to and return zero when read. An exception to this rule is XER[16:23]. These bits are set to the value written to them and return that value when read.

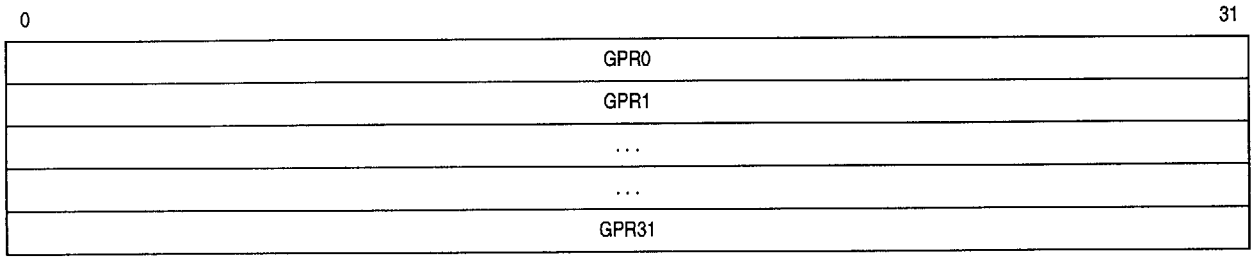
3.7 PowerPC UISA Register Set

The PowerPC UISA registers can be accessed by either user- or supervisor-level instructions. The general-purpose registers and floating-point registers are accessed through instruction operands.

3.7.1 General-Purpose Registers (GPRs)

Integer data is manipulated in the integer unit's thirty-two 32-bit GPRs, shown below. These registers are accessed as source and destination registers through operands in the instruction syntax.

GPRs — General-Purpose Registers



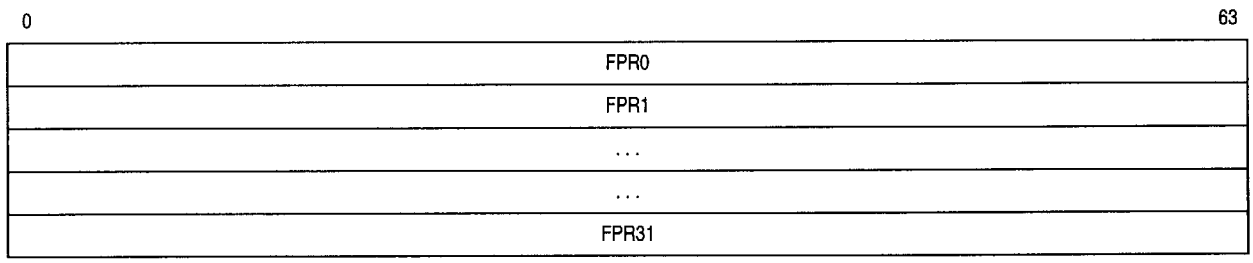
RESET: UNCHANGED

3.7.2 Floating-Point Registers (FPRs)

The PowerPC architecture provides thirty-two 64-bit FPRs. These registers are accessed as source and destination registers through operands in floating-point instructions. Each FPR supports the double-precision, floating-point format. Every instruction that interprets the contents of an FPR as a floating-point value uses the double-precision floating-point format for this interpretation. That is, all floating-point numbers are stored in double-precision format.

All floating-point arithmetic instructions operate on data located in FPRs and, with the exception of the compare instructions (which update the CR), place the result into an FPR. Information about the status of floating-point operations is placed into the floating-point status and control register (FPSCR) and in some cases, into the CR, after the completion of the operation's writeback stage. For information on how the CR is affected by floating-point operations, see **3.7.4 Condition Register (CR)**.

FPRs— Floating-Point Registers



RESET: UNCHANGED

3.7.3 Floating-Point Status and Control Register (FPSCR)

The FPSCR controls the handling of floating-point exceptions and records status resulting from the floating-point operations. FPSCR[0:23] are status bits. FPSCR[24:31] are control bits.

FPSCR[0:12] and FPSCR[21:23] are floating-point exception condition bits. These bits are sticky, except for the floating-point enabled exception summary (FEX) and floating-point invalid operation exception summary (VX). Once set, sticky bits remain set until they are cleared by an **mcrfs**, **mtfsfi**, **mtfsf**, or **mtfsb0** instruction.

Table 5 summarizes which bits in the FPSCR are sticky status bits, which are normal status bits, and which are control bits.

Table 5 FPSCR Bit Categories

Bits	Type
[0], [3:12], [21:23]	Status, sticky
[1:2], [13:20]	Status, not sticky
[24:31]	Control

FEX and VX are the logical ORs of other FPSCR bits. Therefore these two bits are not listed among the FPSCR bits directly affected by the various instructions.

FPSCR — Floating-Point Status and Control Register

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FX	FEX	VX	OX	UX	ZX	XX	VXS-NAN	VXISI	VXIDI	VXZDZ	VXIMZ	VXVC	FR	FI	FPRF[0]

RESET: UNCHANGED

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FPRF[1:4]				0	VX-SOFT	VX-SQRT	VXCVI	VE	OE	UE	ZE	XE	NI	RN	

RESET: UNCHANGED

A listing of FPSCR bit settings is shown in Table 6.

Table 6 FPSCR Bit Settings

Bit(s)	Name	Description
0	FX	Floating-point exception summary. Every floating-point instruction implicitly sets FPSCR[FX] if that instruction causes any of the floating-point exception bits in the FPSCR to change from 0 to 1. The **mcrfs** instruction implicitly clears FPSCR[FX] if the FPSCR field containing FPSCR[FX] is copied. The **mtfsf**, **mtfsfi**, **mtfsb0**, and **mtfsb1** instructions can set or clear FPSCR[FX] explicitly. This is a sticky bit.

Table 6 FPSCR Bit Settings (Continued)

Bit(s)	Name	Description
1	FEX	Floating-point enabled exception summary. This bit signals the occurrence of any of the enabled exception conditions. It is the logical OR of all the floating-point exception bits masked with their respective enable bits. The **mcrfs** instruction implicitly clears FPSCR[FEX] if the result of the logical OR described above becomes zero. The **mtfsf**, **mtfsfi**, **mtfsb0**, and **mtfsb1** instructions cannot set or clear FPSCR[FEX] explicitly. This is not a sticky bit.
2	VX	Floating-point invalid operation exception summary. This bit signals the occurrence of any invalid operation exception. It is the logical OR of all of the invalid operation exceptions. The **mcrfs** instruction implicitly clears FPSCR[VX] if the result of the logical OR described above becomes zero. The **mtfsf**, **mtfsfi**, **mtfsb0**, and **mtfsb1** instructions cannot set or clear FPSCR[VX] explicitly. This is not a sticky bit.
3	OX	Floating-point overflow exception. This is a sticky bit.
4	UX	Floating-point underflow exception. This is a sticky bit.
5	ZX	Floating-point zero divide exception. This is a sticky bit.
6	XX	Floating-point inexact exception. This is a sticky bit.
7	VXSNAN	Floating-point invalid operation exception for SNaN. This is a sticky bit.
8	VXISI	Floating-point invalid operation exception for ∞-∞. This is a sticky bit.
9	VXIDI	Floating-point invalid operation exception for ∞/∞. This is a sticky bit.
10	VXZDZ	Floating-point invalid operation exception for 0/0. This is a sticky bit.
11	VXIMZ	Floating-point invalid operation exception for ∞*0. This is a sticky bit.
12	VXVC	Floating-point invalid operation exception for invalid compare. This is a sticky bit.
13	FR	Floating-point fraction rounded. The last floating-point instruction that potentially rounded the intermediate result incremented the fraction. This bit is not sticky.
14	FI	Floating-point fraction inexact. The last floating-point instruction that potentially rounded the intermediate result produced an inexact fraction or a disabled exponent overflow. This bit is not sticky.
[15:19]	FPRF	Floating-point result flags. This field is based on the value placed into the target register even if that value is undefined. Refer to Table 7 for specific bit settings. 15 Floating-point result class descriptor (C). Floating-point instructions other than the compare instructions may set this bit with the FPCC bits, to indicate the class of the result. 16–19 Floating-point condition code (FPCC). Floating-point compare instructions always set one of the FPCC bits to one and the other three FPCC bits to zero. Other floating-point instructions may set the FPCC bits with the C bit, to indicate the class of the result. Note that in this case the high-order three bits of the FPCC retain their relational significance indicating that the value is less than, greater than, or equal to zero. 16 Floating-point less than or negative (FL or <) 17 Floating-point greater than or positive (FG or >) 18 Floating-point equal or zero (FE or =) 19 Floating-point unordered or NaN (FU or ?)
20	—	Reserved

Table 6 FPSCR Bit Settings (Continued)

Bit(s)	Name	Description
21	VXSOFT	Floating-point invalid operation exception for software request. This bit can be altered only by the **mcrfs**, **mtfsfi**, **mtfsf**, **mtfsb0**, or **mtfsb1** instructions. The purpose of VXSOFT is to allow software to cause an invalid operation condition for a condition that is not necessarily associated with the execution of a floating-point instruction. For example, it might be set by a program that computes a square root if the source operand is negative. This is a sticky bit.
22	VXSQRT	Floating-point invalid operation exception for invalid square root. This is a sticky bit. This guarantees that software can simulate **fsqrt** and **frsqrte**, and to provide a consistent interface to handle exceptions caused by square-root operations.
23	VXCVI	Floating-point invalid operation exception for invalid integer convert. This is a sticky bit.
24	VE	Floating-point invalid operation exception enable.
25	OE	Floating-point overflow exception enable.
26	UE	Floating-point underflow exception enable. This bit should not be used to determine whether denormalization should be performed on floating-point stores.
27	ZE	Floating-point zero divide exception enable.
28	XE	Floating-point inexact exception enable.
29	NI	Non-IEEE mode bit.
30–31	RN	Floating-point rounding control. 00 Round to nearest 01 Round toward zero 10 Round toward +infinity 11 Round toward -infinity

Table 7 illustrates the floating-point result flags that correspond to FPSCR[15:19].

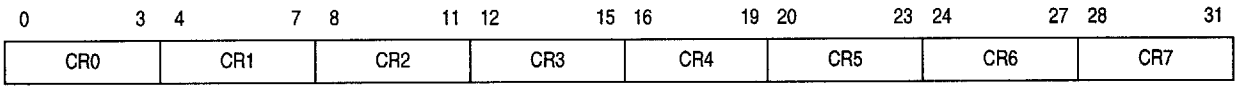
Table 7 Floating-Point Result Flags in FPSCR

Result Flags (Bits 15–19) C<>=?	Result value class
10001	Quiet NaN
01001	– Infinity
01000	– Normalized number
11000	– Denormalized number
10010	– Zero
00010	+ Zero
10100	+ Denormalized number
00100	+ Normalized number
00101	+ Infinity

3.7.4 Condition Register (CR)

The condition register (CR) is a 32-bit register that reflects the result of certain operations and provides a mechanism for testing and branching. The bits in the CR are grouped into eight 4-bit fields, CR0 to CR7.

CR — Condition Register



RESET: UNCHANGED

The CR fields can be set in the following ways:

- Specified fields of the CR can be set by a move instruction (**mtcrf**) to the CR from a GPR.
- Specified fields of the CR can be moved from one CR*x* field to another with the **mcrf** instruction.
- A specified field of the CR can be set by a move instruction (**mcrfs**) to the CR from the FPSCR.
- A specified field of the CR can be set by a move instruction (**mcrxr**) to the CR from the XER.
- Condition register logical instructions can be used to perform logical operations on specified bits in the condition register.
- CR0 can be the implicit result of an integer operation.
- CR1 can be the implicit result of a floating-point operation.
- A specified CR field can be the explicit result of either an integer or floating-point compare instruction.

Instructions are provided to test individual CR bits.

3.7.4.1 Condition Register CR0 Field Definition

In most integer instructions, when the CR is set to reflect the result of the operation (that is, when Rc = 1), and for **addic.**, **andi.**, and **andis.**, the first three bits of CR0 are set by an algebraic comparison of the result to zero; the fourth bit of CR0 is copied from XER[SO]. For integer instructions, CR[0:3] are set to reflect the result as a signed quantity. The result as an unsigned quantity or a bit string can be deduced from the EQ bit.

The CR0 bits are interpreted as shown in Table 8. If any portion of the result (the 32-bit value placed into the destination register) is undefined, the value placed in the first three bits of CR0 is undefined.

Table 8 Bit Settings for CR0 Field of CR

CR0 Bit	Description
0	Negative (LT) — This bit is set when the result is negative.
1	Positive (GT) — This bit is set when the result is positive (and not zero).
2	Zero (EQ) — This bit is set when the result is zero.
3	Summary overflow (SO) — This is a copy of the final state of XER[SO] at the completion of the instruction.

3.7.4.2 Condition Register CR1 Field Definition

In all floating-point instructions when the CR is set to reflect the result of the operation (that is, when Rc = 1), the CR1 field (bits 4 to 7 of the CR) is copied from FPSCR[0:3] to indicate the floating-point exception status. For more information about the FPSCR, see **3.7.3 Floating-Point Status and Control Register (FPSCR)**. The bit settings for the CR1 field are shown in Table 9.

Table 9 Bit Settings for CR1 Field of CR

CR1 Bit	Description
0	Floating-point exception (FX) — This is a copy of the final state of FPSCR[FX] at the completion of the instruction.
1	Floating-point enabled exception (FEX) — This is a copy of the final state of FPSCR[FEX] at the completion of the instruction.
2	Floating-point invalid exception (VX) — This is a copy of the final state of FPSCR[VX] at the completion of the instruction.
3	Floating-point overflow exception (OX) — This is a copy of the final state of FPSCR[OX] at the completion of the instruction.

3.7.4.3 Condition Register CR*n* Field — Compare Instruction

When a specified CR field is set by a compare instruction, the bits of the specified field are interpreted as shown in Table 10. A condition register field can also be accessed by the **mfcr**, **mcrf**, and **mtcrf** instructions.

Table 10 CR*n* Field Bit Settings for Compare Instructions

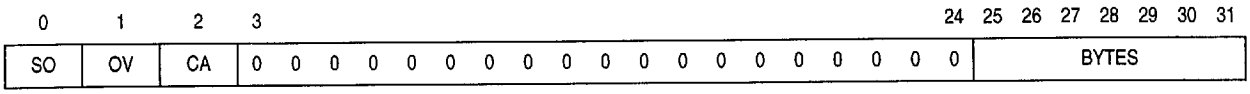
CR*n* Bit[1]	Description
0	Less than, floating-point less than (LT, FL). For integer compare instructions, (**r**A) < SIMM, UIMM, or (**r**B) (algebraic comparison) or (**r**A) SIMM, UIMM, or (**r**B) (logical comparison). For floating-point compare instructions, (**fr**A) < (**fr**B).
1	Greater than, floating-point greater than (GT, FG). For integer compare instructions, (**r**A) > SIMM, UIMM, or (**r**B) (algebraic comparison) or (**r**A) SIMM, UIMM, or (**r**B) (logical comparison). For floating-point compare instructions, (**fr**A) > (**fr**B).
2	Equal, floating-point equal (EQ, FE). For integer compare instructions, (**r**A) = SIMM, UIMM, or (**r**B). For floating-point compare instructions, (**fr**A) = (**fr**B).
3	Summary overflow, floating-point unordered (SO, FU). For integer compare instructions, this is a copy of the final state of XER[SO] at the completion of the instruction. For floating-point compare instructions, one or both of (**fr**A) and (**fr**B) is not a number (NaN).

1. Here, the bit indicates the bit number in any one of the four-bit subfields, CR0–CR7

3.7.5 Integer Exception Register (XER)

The integer exception register (XER) is a user-level, 32-bit register.

XER — Integer Exception Register **SPR 1**



RESET: UNCHANGED

The bit definitions for XER, shown in Table 12, are based on the operation of an instruction considered as a whole, not on intermediate results. For example, the result of the Subtract from Carrying (**subfc***x*) instruction is specified as the sum of three values. This instruction sets bits in the XER based on the entire operation, not on an intermediate sum.

In most cases, reserved fields in registers are ignored when written to and return zero when read. However, XER[16:23] are set to the value written to them and return that value when read.

Table 11 Integer Exception Register Bit Definitions

Bit(s)	Name	Description
0	SO	Summary Overflow (SO) — The summary overflow bit is set whenever an instruction sets the overflow bit (OV) to indicate overflow and remains set until software clears it. It is not altered by compare instructions or other instructions that cannot overflow.
1	OV	Overflow (OV) — The overflow bit is set to indicate that an overflow has occurred during execution of an instruction. Integer and subtract instructions having OE=1 set OV if the carry out of bit 0 is not equal to the carry out of bit 1, and clear it otherwise. The OV bit is not altered by compare instructions or other instructions that cannot overflow.
2	CA	Carry (CA) — In general, the carry bit is set to indicate that a carry out of bit 0 occurred during execution of an instruction. Add carrying, subtract from carrying, add extended, and subtract from extended instructions set CA to one if there is a carry out of bit 0, and clear it otherwise. The CA bit is not altered by compare instructions or other instructions that cannot carry, except that shift right algebraic instructions set the CA bit to indicate whether any '1' bits have been shifted out of a negative quantity.
3:24	—	Reserved
25:31	BYTES	This field specifies the number of bytes to be transferred by a Load String Word Indexed (**lswx**) or Store String Word Indexed (**stswx**) instruction.

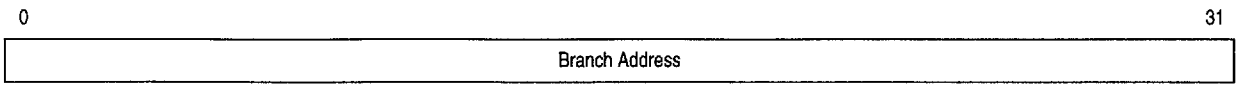
3.7.6 Link Register (LR)

The 32-bit link register supplies the branch target address for the Branch Conditional to Link Register (**bclr***x*) instruction, and can be used to hold the logical address of the instruction that follows a branch and link instruction.

Note that although the two least-significant bits can accept any values written to them, they are ignored when the LR is used as an address.

Both conditional and unconditional branch instructions include the option of placing the effective address of the instruction following the branch instruction in the LR. This is done regardless of whether the branch is taken.

LR — Link Register **SPR 8**

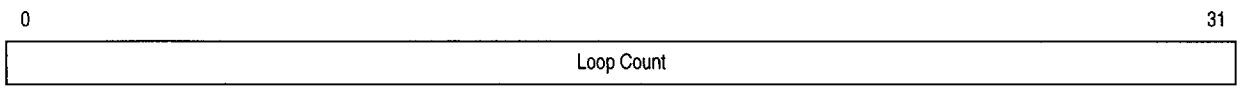


RESET: UNCHANGED

3.7.7 Count Register (CTR)

The count register (CTR) is a 32-bit register for holding a loop count that can be decremented during execution of branch instructions that contain an appropriately coded BO field. If the value in CTR is 0 before being decremented, it is –1 afterward. The count register provides the branch target address for the Branch Conditional to Count Register (**bcctr***x*) instruction.

CTR — Count Register **SPR 9**



RESET: UNCHANGED

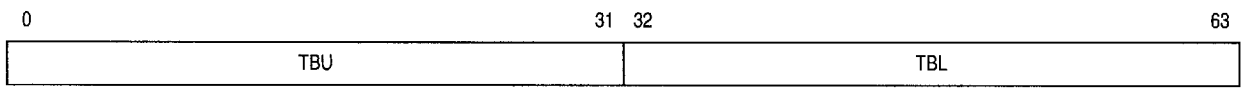
3.8 PowerPC VEA Register Set — Time Base

The PowerPC virtual environment architecture (VEA) defines registers in addition to those in the UISA register set. The PowerPC VEA register set can be accessed by all software with either user- or supervisor-level privileges.

The PowerPC VEA includes the time base facility (TB), a 64-bit structure that contains a 64-bit unsigned integer that is incremented periodically. The frequency at which the counter is updated is implementation-dependent.

The TB consists of two 32-bit registers: time base upper (TBU) and time base lower (TBL). In the context of the VEA, user-level applications are permitted read-only access to the TB. The OEA defines supervisor-level access to the TB for writing values to the TB. Different SPR encodings are provided for reading and writing the time base.

TB — Time Base (Reading) **SPR 268, 269**



RESET: UNCHANGED

Table 12 Time Base Field Definitions

Bits	Name	Description
0-31	TBU	Time Base (Upper) — The high-order 32 bits of the time base
32-63	TBL	Time Base (Lower) — The low-order 32 bits of the time base

In 32-bit PowerPC implementations such as the RCPU, it is not possible to read the entire 64-bit time base in a single instruction. The **mftb** simplified mnemonic copies the lower half of the time base register (TBL) to a GPR, and the **mftbu** simplified mnemonic copies the upper half of the time base (TBU) to a GPR.

3.9 PowerPC OEA Register Set

The PowerPC operating environment architecture (OEA) includes a number of SPRs and other registers that are accessible only by supervisor-level instructions. Some SPRs are RCPU-specific; some RCPU SPRs may not be implemented in other PowerPC processors, or may not be implemented in the same way.

3.9.1 Machine State Register (MSR)

The machine state register is a 32-bit register that defines the state of the processor. When an exception occurs, the current contents of the MSR are loaded into SRR1, and the MSR is updated to reflect the exception-processing machine state. The MSR can also be modified by the **mtmsr**, **sc**, and **rfi** instructions. It can be read by the **mfmsr** instruction.

MSR — Machine State Register

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED															ILE
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EE	PR	FP	ME	FE0	SE	BE	FE1	0	IP	RESERVED				RI	LE
RESET:															
0	0	0	U	0	0	0	0	0	*	0	0	0	0	0	0

*Reset value of this bit depends on the value of the data bus reset configuration word.

Table 13 shows the bit definitions for the MSR.

Table 13 Machine State Register Bit Settings

Bit(s)	Name	Description
0–14	—	Reserved
15	ILE	Exception little endian mode. When an exception occurs, this bit is copied into MSR[LE] to select the endian mode for the context established by the exception. 0 = Processor runs in big endian mode during exception processing. 1 = Processor runs in little endian mode during exception processing.
16	EE	External interrupt enable 0 = The processor delays recognition of external interrupts and decrementer exception conditions. 1 = The processor is enabled to take an external interrupt or the decrementer exception.
17	PR	Privilege level 0 = The processor can execute both user- and supervisor-level instructions. 1 = The processor can only execute user-level instructions.

Table 13 Machine State Register Bit Settings (Continued)

Bit(s)	Name	Description
18	FP	Floating-point available 0 = The processor prevents dispatch of floating-point instructions, including floating-point loads, stores and moves. Floating-point enabled program exceptions can still occur and the FPRs can still be accessed. 1 = The processor can execute floating-point instructions, and can take floating-point enabled exception type program exceptions.
19	ME	Machine check enable 0 = Machine check exceptions are disabled. 1 = Machine check exceptions are enabled.
20	FE0	Floating-point exception mode 0 (See Table 14.)
21	SE	Single-step trace enable 0 = The processor executes instructions normally. 1 = The processor generates a single-step trace exception upon the successful execution of the next instruction. When this bit is set, the processor dispatches instructions in strict program order. Successful execution means the instruction caused no other exception. Single-step tracing may not be present on all implementations.
22	BE	Branch trace enable 0 = No trace exception occurs when a branch instruction is completed 1 = Trace exception occurs when a branch instruction is completed
23	FE1	Floating-point exception mode 1 (See Table 14.)
24	—	Reserved.
25	IP	Exception prefix. The setting of this bit specifies the location of the exception vector table. 0 = Exception vector table starts at the physical address 0x0000 0000. 1 = Exception vector table starts at the physical address 0xFFF0 0000.
26–29	—	Reserved
30	RI	Recoverable exception (for machine check and non-maskable breakpoint exceptions) 0 = Machine state is not recoverable. 1 = Machine state is recoverable.
31	LE	Little endian mode 0 = Processor operates in big-endian mode during normal processing. 1 = Processor operates in little-endian mode during normal processing.

The floating-point exception mode bits are interpreted as shown in Table 14.

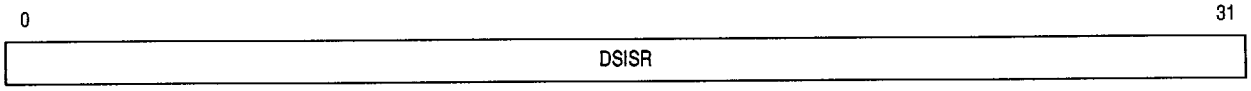
Table 14 Floating-Point Exception Mode Bits

FE[0:1]	Mode
00	Ignore exceptions mode — Floating-point exceptions do not cause the floating-point assist error handler to be invoked.
01, 10, 11	Floating-point precise mode — The system floating-point assist error handler is invoked precisely at the instruction that caused the enabled exception.

3.9.2 DAE/Source Instruction Service Register (DSISR)

The 32-bit DSISR identifies the cause of data access and alignment exceptions.

DSISR— DAE/Source Instruction Service Register **SPR 18**

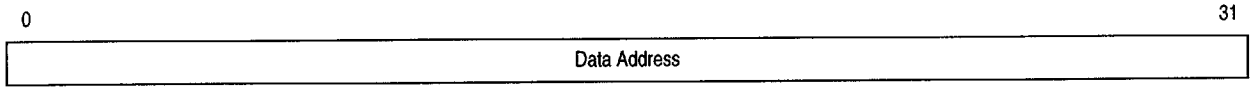


RESET: UNCHANGED

3.9.3 Data Address Register (DAR)

After an alignment exception, the DAR is set to the effective address of a load or store element.

DAR— Data Address Register **SPR 19**

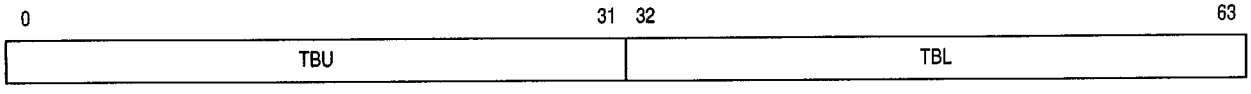


RESET: UNCHANGED

3.9.4 Time Base Facility (TB) — OEA

As described in **3.8 PowerPC VEA Register Set — Time Base**, the time base (TB) provides a 64-bit incrementing counter. The VEA defines user-level, read-only access to the TB. Writing to the TB is reserved for supervisor-level applications such as operating systems and bootstrap routines. The OEA defines supervisor-level, write access to the TB.

TB — Time Base (Writing) **SPR 284, 285**



RESET: UNCHANGED

Table 15 Time Base Field Definitions

Bits	Name	Description
0:31	TBU	Time Base (Upper) — The high-order 32 bits of the time base
32:63	TBL	Time Base (Lower) — The low-order 32 bits of the time base

The TB can be written to at the supervisor privilege level only. The **mttbl** and **mttbu** simplified mnemonics write the lower and upper halves of the TB, respectively. The **mtspr**, **mttbl**, and **mttbu** instructions treat TBL and TBU as separate 32-bit registers; setting one leaves the other unchanged. It is not possible to write the entire 64-bit time base in a single instruction.

For information about reading the time base, refer to **3.8 PowerPC VEA Register Set — Time Base**.

3.9.5 Decrementer Register (DEC)

The decrementer (DEC, SPR 22) is a 32-bit decrementing counter defined by the PowerPC architecture to provide a decrementer exception after a programmable delay. The DEC satisfies the following requirements:

- Loading a GPR from the DEC has no effect on the DEC.
- Storing a GPR to the DEC replaces the value in the DEC with the value in the GPR.
- Whenever bit 0 of the DEC changes from zero to one, a decrementer exception request (unless masked) is signaled. Multiple DEC exception requests may be received before the first exception occurs; however, any additional requests are canceled when the exception occurs for the first request.
- If the DEC is altered by software and the content of bit 0 is changed from zero to one, an exception request is signaled.

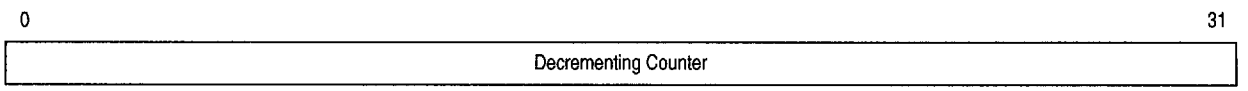
The decrementer frequency is based on a subdivision of the processor clock. In the MPC505, the default frequency is 1 MHz. A bit in the system clock control register (SCCR) in the SIU determines the clock source of both the decrementer and the time base.

With a 1-MHz input frequency, the decrementer period is 4295 seconds (approximately 71.6 minutes):

$$T_{DEC} = 2^{32} / 1\ MHz = 4295\ seconds$$

The state of DEC after standby power is restored is indeterminate. The DEC runs continuously after power-up (unless the clock module is programmed to turn off the clock). System software must perform any initialization. The decrementer is not affected by reset and continues counting while reset is asserted. A decrementer exception may be signaled to software prior to initialization.

DEC — Decrementer Register **SPR 22**

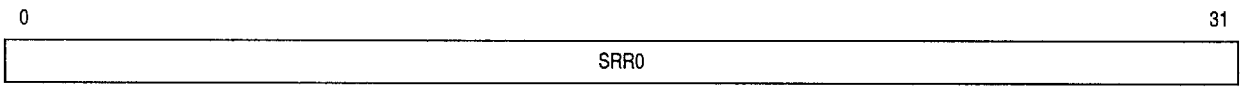


RESET: UNCHANGED

3.9.6 Machine Status Save/Restore Register 0 (SRR0)

The machine status save/restore register 0 (SRR0) is a 32-bit register that identifies where instruction execution should resume when an **rfi** instruction is executed following an exception. It also holds the effective address of the instruction that follows the System Call (**sc**) instruction.

SRR0 — Machine Status Save/Restore Register 0 **SPR 26**



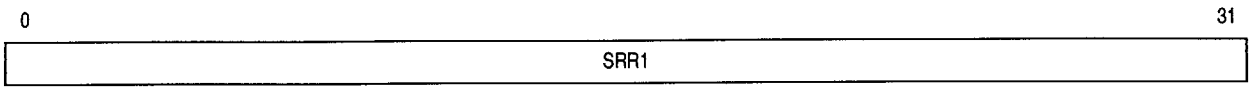
RESET: UNDEFINED

When an exception occurs, SRR0 is set to point to an instruction such that all prior instructions have completed execution and no subsequent instruction has begun execution. The instruction addressed by SRR0 may not have completed execution, depending on the exception type. SRR0 addresses either the instruction causing the exception or the immediately following instruction. The instruction addressed can be determined from the exception type and status bits.

3.9.7 Machine Status Save/Restore Register 1 (SRR1)

SRR1 is a 32-bit register used to save machine status on exceptions and to restore machine status when an **rfi** instruction is executed.

SRR1 — Machine Status Save/Restore Register 1 **SPR 27**



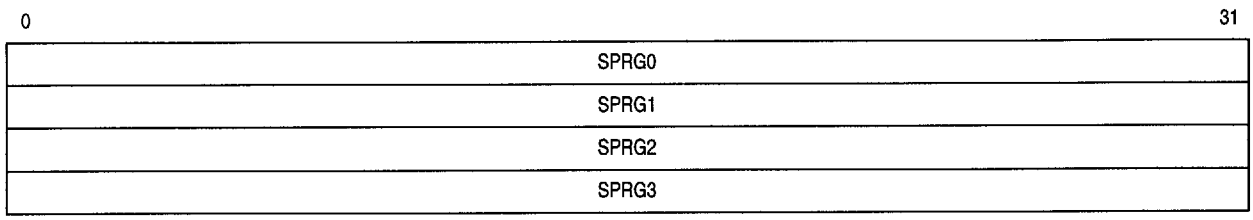
RESET: UNDEFINED

In general, when an exception occurs, SRR1[0:15] are loaded with exception-specific information, and MSR[16:31] are placed into SRR1[16:31].

3.9.8 General SPRs (SPRG0–SPRG3)

SPRG0–SPRG3 are 32-bit registers provided for general operating system use, such as performing a fast state save and for supporting multiprocessor implementations. SPRG0–SPRG3 are shown below.

SPRG0–SPRG3 — General Special-Purpose Registers 0–3 **SPR 272 – SPR 275**



RESET: UNCHANGED

Uses for SPRG0–SPRG3 are shown in Table 16.

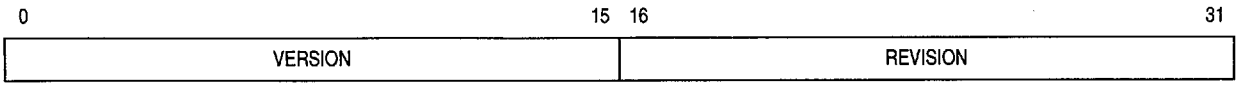
Table 16 Uses of SPRG0–SPRG3

Register	Description
SPRG0	Software may load a unique physical address in this register to identify an area of memory reserved for use by the exception handler. This area must be unique for each processor in the system.
SPRG1	This register may be used as a scratch register by the exception handler to save the content of a GPR. That GPR then can be loaded from SPRG0 and used as a base register to save other GPRs to memory.
SPRG2	This register may be used by the operating system as needed.
SPRG3	This register may be used by the operating system as needed.

3.9.9 Processor Version Register (PVR)

The PVR is a 32-bit, read-only register that identifies the version and revision level of the PowerPC processor. The contents of the PVR can be copied to a GPR by the **mfspr** instruction. Read access to the PVR is available in supervisor mode only; write access is not provided.

PVR — Processor Version Register **SPR 287**



RESET: UNCHANGED

Table 17 Processor Version Register Bit Settings

Bit(s)	Name	Description
0:15	VERSION	A 16-bit number that identifies the version of the processor and of the PowerPC architecture
16:31	REVISION	A 16-bit number that distinguishes between various releases of a particular version

3.9.10 Implementation-Specific SPRs

The RCPU includes several implementation-specific SPRs that are not defined by the PowerPC architecture. These registers can be accessed by supervisor-level instructions only.

3.9.10.1 EIE, EID, and NRI Special-Purpose Registers

The RCPU includes three implementation-specific SPRs to facilitate the software manipulation of the MSR[RI] and MSR[EE] bits. Issuing the **mtspr** instruction with one of these registers as an operand causes the RI and EE bits to be set or cleared as shown in Table 18.

A read (**mfspr**) of any of these locations is treated as an unimplemented instruction, resulting in a software emulation exception.

Table 18 EIE, EID, AND NRI Registers

SPR Number (Decimal)	Mnemonic	MSR[EE]	MSR[RI]
80	EIE	1	1
81	EID	0	1
82	NRI	0	0

3.9.10.2 Instruction-Cache Control Registers

The implementation-specific supervisor-level SPRs shown in Table 19 control the operation of the instruction cache.

Table 19 Instruction Cache Control Registers

SPR Number (Decimal)	Name	Description
560	ICCST	I-cache control and status register
561	ICADR	I-cache address register
562	ICDAT	I-cache data port (read only)

Refer to **4 Instruction Cache** for details on these registers.

3.9.10.3 Development Support Registers

Table 20 lists the implementation-specific RCPU registers provided for development support.

Table 20 Development Support Registers

SPR Number (Decimal)	Mnemonic	Name
144	CMPA	Comparator A Value Register
145	CMPB	Comparator B Value Register
146	CMPC	Comparator C Value Register
147	CMPD	Comparator D Value Register
148	ECR	Exception Cause Register
149	DER	Debug Enable Register
150	COUNTA	Breakpoint Counter A Value and Control Register
151	COUNTB	Breakpoint Counter B Value and Control Register
152	CMPE	Comparator E Value Register
153	CMPF	Comparator F Value Register
154	CMPG	Comparator G Value Register
155	CMPH	Comparator H Value Register
156	LCTRL1	L-Bus Support Control Register 1
157	LCTRL2	L-Bus Support Control Register 2
158	ICTRL	I-Bus Support Control Register
159	BAR	Breakpoint Address Register
630	DPDR	Development Port Data Register

Refer to the *RCPU Reference Manual* (RCPURM/AD) for detailed descriptions of these registers.

3.9.10.4 Floating-Point Exception Cause Register (FPECR)

The FPECR is a 32-bit supervisor-level internal status and control register used by the floating-point assist software envelope. Refer to the *RCPU Reference Manual* (RCPURM/AD) for more information on this register.

3.10 Instruction Set

All PowerPC instructions are encoded as single words (32 bits). Instruction formats are consistent among all instruction types, permitting efficient decoding to occur in parallel with operand accesses. This fixed instruction length and consistent format greatly simplifies instruction pipelining.

The PowerPC instructions are divided into the following categories:

- Integer instructions. These include computational and logical instructions.
  - — Integer arithmetic instructions
  - — Integer compare instructions

— Integer logical instructions
— Integer rotate and shift instructions

- Floating-point instructions. These include floating-point computational instructions, as well as instructions that affect the floating-point status and control register (FPSCR).
  — Floating-point arithmetic instructions
  — Floating-point multiply/add instructions
  — Floating-point rounding and conversion instructions
  — Floating-point compare instructions
  — Floating-point status and control instructions
- Load/store instructions. These include integer and floating-point load and store instructions.
  — Integer load and store instructions
  — Integer load and store multiple instructions
  — Floating-point load and store
  — Primitives used to construct atomic memory operations (**lwarx** and **stwcx.** instructions)
- Flow control instructions. These include branching instructions, condition register logical instructions, trap instructions, and other instructions that affect the instruction flow.
  — Branch and trap instructions
  — Condition register logical instructions
- Processor control instructions. These instructions are used for synchronizing memory accesses and cache management.
  — Move to/from SPR instructions
  — Move to/from MSR
  — Synchronize
  — Instruction synchronize
- Memory control instructions. These instructions provide control of the instruction cache.
  — Supervisor-level cache management instructions
  — User-level cache instructions

Note that this grouping of the instructions does not indicate which execution unit executes a particular instruction or group of instructions.

Integer instructions operate on byte, half-word, and word operands. Floating-point instructions operate on single-precision (one word) and double-precision (one double word) floating-point operands. The PowerPC architecture uses instructions that are four bytes long and word-aligned. It provides for byte, half-word, and word operand loads and stores between memory and a set of 32 GPRs. It also provides for word and double-word operand loads and stores between memory and a set of 32 floating-point registers (FPRs).

Computational instructions do not modify memory. To use a memory operand in a computation and then modify the same or another memory location, the memory contents must be loaded into a register, modified, and then written back to the target location with distinct instructions.

PowerPC processors follow the program flow when they are in the normal execution state. However, the flow of instructions can be interrupted directly by the execution of an instruction or by an asynchronous event. Either kind of exception may cause one of several components of the system software to be invoked.

3.10.1 Instruction Set Summary

Table 21 provides a summary of RCPU instructions. Refer to the *RCPU Reference Manual* (RCPURM/AD) for a detailed description of the instruction set.

Table 21 Instruction Set Summary

Mnemonic	Operand Syntax	Name
add (add. addo addo.)	rD,rA,rB	Add
addc (addc. addco addco.)	rD,rA,rB	Add Carrying
adde (adde. addeo addeo.)	rD,rA,rB	Add Extended
addi	rD,rA,SIMM	Add Immediate
addic	rD,rA,SIMM	Add Immediate Carrying
addic.	rD,rA,SIMM	Add Immediate Carrying and Record
addis	rD,rA,SIMM	Add Immediate Shifted
addme (addme. addmeo addmeo.)	rD,rA	Add to Minus One Extended
addze (addze. addzeo addzeo.)	rD,rA	Add to Zero Extended
and (and.)	rA,rS,rB	AND
andc (andc.)	rA,rS,rB	AND with Complement
andi.	rA,rS,UIMM	AND Immediate
andis.	rA,rS,UIMM	AND Immediate Shifted
b (ba bl bla)	target_addr	Branch
bc (bca bcl bcla)	BO,BI,target_addr	Branch Conditional
bcctr (bcctrl)	BO,BI	Branch Conditional to Count Register
bclr (bclrl)	BO,BI	Branch Conditional to Link Register
cmp	**crf**D,L,rA,rB	Compare
cmpi	**crf**D,L,rA,SIMM	Compare Immediate
cmpl	**crf**D,L,rA,rB	Compare Logical
cmpli	**crf**D,L,rA,UIMM	Compare Logical Immediate
cntlzw (cntlzw.)	rA,rS	Count Leading Zeros Word
crand	**crb**D,**crb**A,**crb**B	Condition Register AND
crandc	**crb**D,**crb**A, **crb**B	Condition Register AND with Complement
creqv	**crb**D,**crb**A, **crb**B	Condition Register Equivalent
crnand	**crb**D,**crb**A,**crb**B	Condition Register NAND
crnor	**crb**D,**crb**A,**crb**B	Condition Register NOR
cror	**crb**D,**crb**A,**crb**B	Condition Register OR
crorc	**crb**D,**crb**A, **crb**B	Condition Register OR with Complement
crxor	**crb**D,**crb**A,**crb**B	Condition Register XOR
divw (divw. divwo divwo.)	rD,rA,rB	Divide Word
divwu divwu. divwuo divwuo.	rD,rA,rB	Divide Word Unsigned

Table 21 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
eieio	—	Enforce In-Order Execution of I/O
eqv (eqv.)	rA,rS,rB	Equivalent
extsb (extsb.)	rA,rS	Extend Sign Byte
extsh (extsh.)	rA,rS	Extend Sign Half Word
fabs (fabs.)	frD,frB	Floating Absolute Value
fadd (fadd.)	frD,frA,frB	Floating Add (Double-Precision)
fadds (fadds.)	frD,frA,frB	Floating Add Single
fcmpo	crfD,frA,frB	Floating Compare Ordered
fcmpu	crfD,frA,frB	Floating Compare Unordered
fctiw (fctiw.)	frD,frB	Floating Convert to Integer Word
fctiwz (fctiwz.)	frD,frB	Floating Convert to Integer Word with Round toward Zero
fdiv (fdiv.)	frD,frA,frB	Floating Divide (Double-Precision)
fdivs (fdivs.)	frD,frA,frB	Floating Divide Single
fmadd (fmadd.)	frD,frA,frC,frB	Floating Multiply-Add (Double-Precision)
fmadds (fmadds.)	frD,frA,frC,frB	Floating Multiply-Add Single
fmr (fmr.)	frD,frB	Floating Move Register
fmsub (fmsub.)	frD,frA,frC,frB	Floating Multiply-Subtract (Double-Precision)
fmsubs (fmsubs.)	frD,frA,frC,frB	Floating Multiply-Subtract Single
fmul (fmul.)	frD,frA,frC	Floating Multiply (Double-Precision)
fmuls (fmuls.)	frD,frA,frC	Floating Multiply Single
fnabs (fnabs.)	frD,frB	Floating Negative Absolute Value
fneg (fneg.)	frD,frB	Floating Negate
fnmadd (fnmadd.)	frD,frA,frC,frB	Floating Negative Multiply-Add (Double-Precision)
fnmadds (fnmadds.)	frD,frA,frC,frB	Floating Negative Multiply-Add Single
fnmsub (fnmsub.)	frD,frA,frC,frB	Floating Negative Multiply-Subtract (Double-Precision)
fnmsubs (fnmsubs.)	frD,frA,frC,frB	Floating Negative Multiply-Subtract Single
frsp (frsp.)	frD,frB	Floating Round to Single
fsub (fsub.)	frD,frA,frB	Floating Subtract (Double-Precision)
fsubs (fsubs.)	frD,frA,frB	Floating Subtract Single
icbi	rA,rB	Instruction Cache Block Invalidate
isync	—	Instruction Synchronize

Table 21 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
lbz	rD,d(rA)	Load Byte and Zero
lbzu	rD,d(rA)	Load Byte and Zero with Update
lbzux	rD,rA,rB	Load Byte and Zero with Update Indexed
lbzx	rD,rA,rB	Load Byte and Zero Indexed
lfd	frD,d(rA)	Load Floating-Point Double
lfdu	frD,d(rA)	Load Floating-Point Double with Update
lfdux	frD,rA,rB	Load Floating-Point Double with Update Indexed
lfdx	frD,rA,rB	Load Floating-Point Double Indexed
lfs	frD,d(rA)	Load Floating-Point Single
lfsu	frD,d(rA)	Load Floating-Point Single with Update
lfsux	frD,rA,rB	Load Floating-Point Single with Update Indexed
lfsx	frD,rA,rB	Load Floating-Point Single Indexed
lha	rD,d(rA)	Load Half Word Algebraic
lhau	rD,d(rA)	Load Half Word Algebraic with Update
lhaux	rD,rA,rB	Load Half Word Algebraic with Update Indexed
lhax	rD,rA,rB	Load Half Word Algebraic Indexed
lhbrx	rD,rA,rB	Load Half Word Byte-Reverse Indexed
lhz	rD,d(rA)	Load Half Word and Zero
lhzu	rD,d(rA)	Load Half Word and Zero with Update
lhzux	rD,rA,rB	Load Half Word and Zero with Update Indexed
lhzx	rD,rA,rB	Load Half Word and Zero Indexed
lmw	rD,d(rA)	Load Multiple Word
lswi	rD,rA,NB	Load String Word Immediate
lswx	rD,rA,rB	Load String Word Indexed
lwarx	rD,rA,rB	Load Word and Reserve Indexed
lwbrx	rD,rA,rB	Load Word Byte-Reverse Indexed
lwz	rD,d(rA)	Load Word and Zero
lwzu	rD,d(rA)	Load Word and Zero with Update
lwzux	rD,rA,rB	Load Word and Zero with Update Indexed
lwzx	rD,rA,rB	Load Word and Zero Indexed
mcrf	**crf**D,**crf**S	Move Condition Register Field

Table 21 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
mcrfs	**crf**D,**crf**S	Move to Condition Register from FPSCR
mcrxr	**crf**D	Move to Condition Register from XER
mfcr	**r**D	Move from Condition Register
mffs (mffs.)	**fr**D	Move from FPSCR
mfmsr	**r**D	Move from Machine State Register
mfspr	**r**D,SPR	Move from Special Purpose Register
mftb	**r**D, TBR	Move from Time Base
mtcrf	CRM,**r**S	Move to Condition Register Fields
mtfsb0 (mtfsb0.)	**crb**D	Move to FPSCR Bit 0
mtfsb1 (mtfsb1.)	**crb**D	Move to FPSCR Bit 1
mtfsf (mtfsf.)	FM,**fr**B	Move to FPSCR Fields
mtfsfi (mtfsfi.)	**crf**D,IMM	Move to FPSCR Field Immediate
mtmsr	**r**S	Move to Machine State Register
mtspr	SPR,**r**S	Move to Special Purpose Register
mulhw (mulhw.)	**r**D,**r**A,**r**B	Multiply High Word
mulhwu (mulhwu.)	**r**D,**r**A,**r**B	Multiply High Word Unsigned
mulli	**r**D,**r**A,SIMM	Multiply Low Immediate
mullw (mullw. mullwo mullwo.)	**r**D,**r**A,**r**B	Multiply Low
nand (nand.)	**r**A,**r**S,**r**B	NAND
neg (neg. nego nego.)	**r**D,**r**A	Negate
nor (nor.)	**r**A,**r**S,**r**B	NOR
or (or.)	**r**A,**r**S,**r**B	OR
orc (orc.)	**r**A,**r**S,**r**B	OR with Complement
ori	**r**A,**r**S,UIMM	OR Immediate
oris	**r**A,**r**S,UIMM	OR Immediate Shifted
rfi	—	Return from Interrupt
rlwimi (rlwimi.)	**r**A,**r**S,SH,MB,ME	Rotate Left Word Immediate then Mask Insert
rlwinm (rlwinm.)	**r**A,**r**S,SH,MB,ME	Rotate Left Word Immediate then AND with Mask
rlwnm (rlwnm.)	**r**A,**r**S,**r**B,MB,ME	Rotate Left Word then AND with Mask
sc	—	System Call
slw (slw.)	**r**A,**r**S,**r**B	Shift Left Word
sraw (sraw.)	**r**A,**r**S,**r**B	Shift Right Algebraic Word

Table 21 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
srawi (srawi.)	rA,rS,SH	Shift Right Algebraic Word Immediate
srw (srw.)	rA,rS,rB	Shift Right Word
stb	rS,d(rA)	Store Byte
stbu	rS,d(rA)	Store Byte with Update
stbux	rS,rA,rB	Store Byte with Update Indexed
stbx	rS,rA,rB	Store Byte Indexed
stfd	frS,d(rA)	Store Floating-Point Double
stfdu	frS,d(rA)	Store Floating-Point Double with Update
stfdux	frS,rB	Store Floating-Point Double with Update Indexed
stfdx	frS,rB	Store Floating-Point Double Indexed
stfiwx	frS,rB	Store Floating-Point as Integer Word Indexed
stfs	frS,d(rA)	Store Floating-Point Single
stfsu	frS,d(rA)	Store Floating-Point Single with Update
stfsux	frS,rB	Store Floating-Point Single with Update Indexed
stfsx	frS,r B	Store Floating-Point Single Indexed
sth	rS,d(rA)	Store Half Word
sthbrx	rS,rA,rB	Store Half Word Byte-Reverse Indexed
sthu	rS,d(rA)	Store Half Word with Update
sthux	rS,rA,rB	Store Half Word with Update Indexed
sthx	rS,rA,rB	Store Half Word Indexed
stmw	rS,d(rA)	Store Multiple Word
stswi	rS,rA,NB	Store String Word Immediate
stswx	rS,rA,rB	Store String Word Indexed
stw	rS,d(rA)	Store Word
stwbrx	rS,rA,rB	Store Word Byte-Reverse Indexed
stwcx.	rS,rA,rB	Store Word Conditional Indexed
stwu	rS,d(rA)	Store Word with Update
stwux	rS,rA,rB	Store Word with Update Indexed
stwx	rS,rA,rB	Store Word Indexed
subf (subf. subfo subfo.)	rD,rA,rB	Subtract From
subfc (subfc. subfco subfco.)	rD,rA,rB	Subtract from Carrying
subfe (subfe. subfeo subfeo.)	rD,rA,rB	Subtract from Extended

Table 21 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
subfic	rD,rA,SIMM	Subtract from Immediate Carrying
subfme (subfme. subfmeo subfmeo.)	rD,rA	Subtract from Minus One Extended
subfze (subfze. subfzeo subfzeo.)	rD,rA	Subtract from Zero Extended
sync	—	Synchronize
tw	TO,rA,rB	Trap Word
twi	TO,rA,SIMM	Trap Word Immediate
xor (xor.)	rA,rS,rB	XOR
xori	rA,rS,UIMM	XOR Immediate
xoris	rA,rS,UIMM	XOR Immediate Shifted

3.10.2 Recommended Simplified Mnemonics

To simplify assembly language coding, a set of alternative mnemonics is provided for some frequently used operations (such as no-op, load immediate, load address, move register, and complement register).

For a complete list of simplified mnemonics, see the *RCPU Reference Manual* (RCPURM/AD). Programs written to be portable across the various assemblers for the PowerPC architecture should not assume the existence of mnemonics not described in that manual.

3.10.3 Calculating Effective Addresses

The effective address (EA) is the 32-bit address computed by the processor when executing a memory access or branch instruction or when fetching the next sequential instruction.

The PowerPC architecture supports two simple memory addressing modes:

- EA = (**r**A|0) + 16-bit offset (including offset = 0) (register indirect with immediate index)
- EA = (**r**A|0) + **r**B (register indirect with index)

These simple addressing modes allow efficient address generation for memory accesses. Calculation of the effective address for aligned transfers occurs in a single clock cycle.

For a memory access instruction, if the sum of the effective address and the operand length exceeds the maximum effective address, the storage operand is considered to wrap around from the maximum effective address to effective address 0.

Effective address computations for both data and instruction accesses use 32-bit unsigned binary arithmetic. A carry from bit 0 is ignored in 32-bit implementations.

3.11 Exception Model

The PowerPC exception mechanism allows the processor to change to supervisor state as a result of external signals, errors, or unusual conditions arising in the execution of instructions. When exceptions occur, information about the state of the processor is saved to certain registers, and the processor begins execution at an address (exception vector) predetermined for each exception. Processing of exceptions occurs in supervisor mode.

Although multiple exception conditions can map to a single exception vector, a more specific condition may be determined by examining a register associated with the exception — for example, the DAE/source instruction service register (DSISR) and the floating-point status and control register (FPSCR). Additionally, some exception conditions can be explicitly enabled or disabled by software.

3.11.1 Exception Classes

The MPC505 exception classes are shown in Table 22.

Table 22 MPC505 Exception Classes

Class	Exception Type
Asynchronous, unordered	Machine check System reset
Asynchronous, ordered	External interrupt Decrementer
Synchronous (ordered, precise)	Instruction-caused exceptions

3.11.2 Ordered Exceptions

In the MPC505, all exceptions except for reset, debug port nonmaskable interrupts, and machine check exceptions are ordered. Ordered exceptions satisfy the following criteria:

- Only one exception is reported at a time. If, for example, a single instruction encounters multiple exception conditions, those conditions are encountered sequentially. After the exception handler handles an exception, instruction execution continues until the next exception condition is encountered.
- When the exception is taken, no program state is lost.

3.11.3 Unordered Exceptions

Unordered exceptions may be reported at any time and are not guaranteed to preserve program state information. The processor can never recover from a reset exception. It can recover from other unordered exceptions in most cases. However, if a debug port nonmaskable interrupt or machine check exception occurs during the servicing of a previous exception, the machine state information in SRR0 and SRR1 (and, in some cases, the DAR and DSISR) may not be recoverable; the processor may be in the process of saving or restoring these registers.

To determine whether the machine state is recoverable, the user can read the RI (recoverable exception) bit in SRR1. During exception processing, the RI bit in the MSR is copied to SRR1 and then cleared. The operating system should set the RI bit in the MSR at the end of each exception handler's prologue (after saving the program state) and clear the bit at the start of each exception handler's epilogue (before restoring the program state). Then, if an unordered exception occurs during the servicing of an exception handler, the RI bit in SRR1 will contain the correct value.

3.11.4 Precise Exceptions

In the MPC505, all synchronous (instruction-caused) exceptions are precise. When a precise exception occurs, the processor backs the machine up to the instruction causing the exception. This ensures that the machine is in its correct architecturally-defined state. The following conditions exist at the point a precise exception occurs:

1. Architecturally, no instruction following the faulting instruction in the code stream has begun execution.

2. All instructions preceding the faulting instruction appear to have completed with respect to the executing processor.
3. SRR0 addresses either the instruction causing the exception or the immediately following instruction. Which instruction is addressed can be determined from the exception type and the status bits.
4. Depending on the type of exception, the instruction causing the exception may not have begun execution, may have partially completed, or may have completed execution.

3.11.5 Exception Vector Table

The setting of the exception prefix (IP) bit in the MSR determines how exceptions are vectored. If the bit is cleared, the exception vector table begins at the physical address 0x0000 0000; if IP is set, the exception vector table begins at the physical address 0xFFF0 0000. Table 23 shows the exception vector offset of the first instruction of the exception handler routine for each exception type.

Table 23 Exception Vector Offset Table

Vector Offset (Hexadecimal)	Exception Type
00000	Reserved
00100	System reset
00200	Machine check
00300	Data access
00400	Instruction access
00500	External interrupt
00600	Alignment
00700	Program
00800	Floating-point unavailable
00900	Decrementer
00A00	Reserved
00B00	Reserved
00C00	System call
00D00	Trace
00E00	Floating-point assist
01000	Software emulation
01C00	Data breakpoint
01D00	Instruction breakpoint
01E00	Maskable external breakpoint
01F00	Nonmaskable external breakpoint

3.12 Instruction Timing

The MPC505 processor is pipelined. Because the processing of an instruction is broken into a series of stages, an instruction does not require the entire resources of the processor.

The instruction pipeline in the MPC505 has four stages:

1. The dispatch stage is implemented using a distributed mechanism. The central dispatch unit broadcasts the instruction to all units. In addition, scoreboard information (regarding data dependencies) is broadcast to each execution unit. Each execution unit decodes the instruction. If the instruction is not implemented, a program exception is taken. If the instruction is legal and no data dependency is found, the instruction is accepted by the appropriate execution unit, and the data found in the destination register is copied to the history buffer. If a data dependency exists, the machine is stalled until the dependency is resolved.
2. In the execute stage, each execution unit that has an executable instruction executes the instruction. (For some instructions, this occurs over multiple cycles).
3. In the writeback stage, the execution unit writes the result to the destination register and reports to the history buffer that the instruction is completed.
4. In the retirement stage, the history buffer retires instructions in architectural order. An instruction retires from the machine if it completes execution with no exceptions and if all instructions preceding it in the instruction stream have finished execution with no exceptions. As many as six instructions can be retired in one clock.

The history buffer maintains the correct architectural machine state. An exception is taken only when the instruction is ready to be retired from the machine (i.e., after all previously-issued instructions have already been retired from the machine). When an exception is taken, all instructions following the excepting instruction are canceled, i.e., the values of the affected destination registers are restored using the values saved in the history buffer during the dispatch stage.

Figure 8 shows basic instruction pipeline timing.

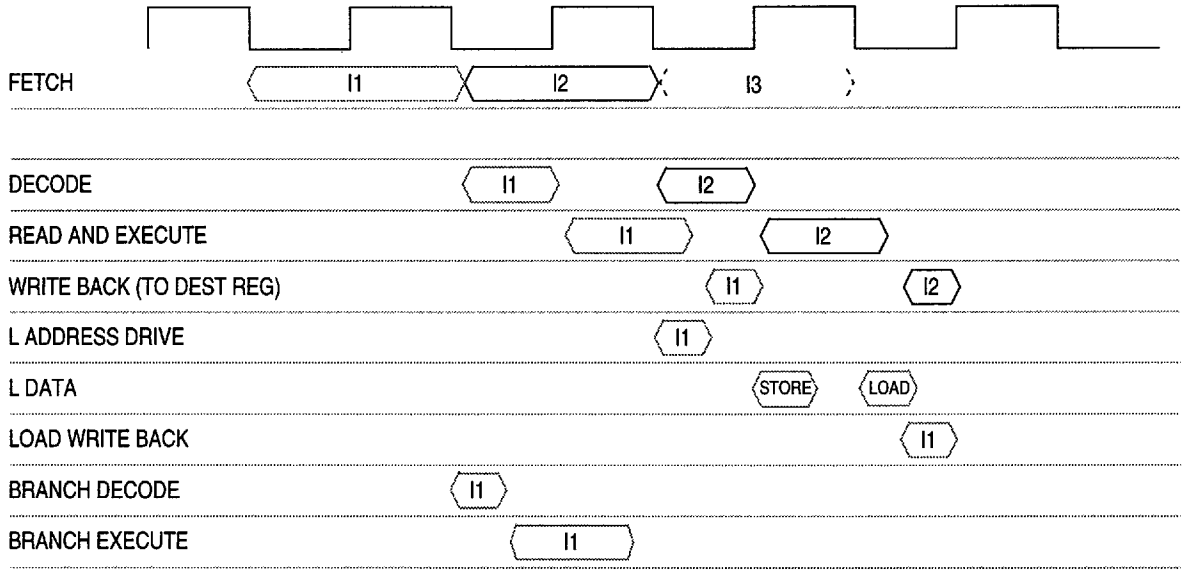


Figure 8 Basic Instruction Pipeline

Table 24 indicates the latency and blockage for each type of instruction. Latency refers to the interval from the time an instruction begins execution until it produces a result that is available for use by a subsequent instruction. Blockage refers to the interval from the time an instruction begins execution until its execution unit is available for a subsequent instruction. Note that when the blockage equals the latency, it is not possible to issue another instruction to the same unit in the same cycle in which the first instruction is being written back.

Table 24 Instruction Latency and Blockage

Instruction Type	Precision	Latency	Blockage
Floating-point multiply-add	Double Single	7 6	7 6
Floating-point add or subtract	Double Single	4 4	4 4
Floating-point multiply	Double Single	5 4	5 4
Floating-point divide	Double Single	17 10	17 10
Integer multiply	—	3	2
Integer divide	—	10	9
Other integer instructions	—	1	0

4 INSTRUCTION CACHE

The MPC505 instruction cache (I-cache) is a 4-Kbyte, two-way set associative cache. The cache is organized into 128 sets, with two lines per set and four words per line. Cache lines are aligned on four-word boundaries in memory.

A cache access cycle begins with an instruction request from the CPU instruction unit. In case of a cache hit, the instruction is delivered to the instruction unit. In case of a cache miss, the cache initiates a burst read cycle (four beats per burst, one word per beat) on the instruction bus (I-bus) with the address of the requested instruction. The first word received from the bus is the requested instruction. The cache forwards this instruction to the instruction unit as soon as it is received from the I-bus. A cache line is then selected to receive the data which will be coming from the bus. An LRU (least recently used) replacement algorithm is used to select a line when no empty lines are available.

Each cache line can be used as an SRAM, allowing the application to lock critical code segments that need fast and deterministic execution time.

Cache coherency in a multi-processor environment is maintained by software and supported by a fast hardware invalidation capability.

4.1 Instruction Cache Features

- Four Kbytes, two-way set associative, four words in a line
- LRU replacement policy
- Lockable SRAM (cache line granularity)
- Critical word first burst access
- Stream hit (allows fetch from the burst buffer and of the word currently on the I-bus)
- Efficiently utilizes the pipeline of the I-bus by initiating a new burst cycle (if miss is detected) while delivering the tail of the previous missed line to the instruction unit
- Cache control:
  - — Supports PowerPC invalidate instruction
  - — Supports load and lock (cache line granularity)
- Supports cache inhibit:
  - — As a cache mode of operation (cache disable)
  - — On memory regions (supported by the chip select logic)
- Miss latency is reduced by
  - — Sending address to the cache and to the I-bus simultaneously; and
  - — Aborting on cache hit before cycle goes external
- Minimum operational power consumption
- Supports reads of tags (including all attributes) and data arrays (for debugging purposes)

4.2 Instruction Cache Organization

Figure 9 illustrates the I-cache organization.

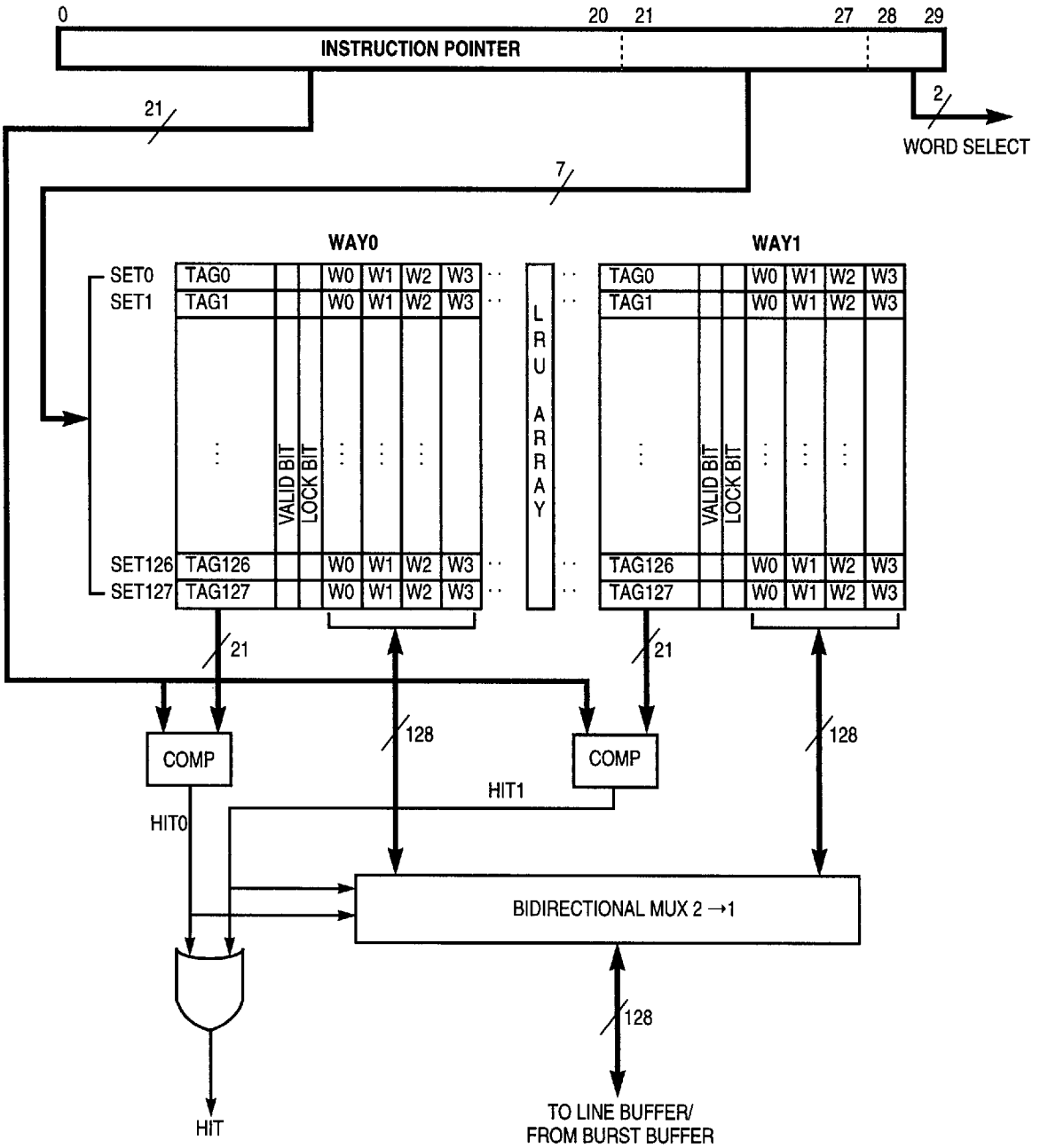


Figure 9 Instruction Cache Organization

Figure 10 illustrates the data path of the I-cache.

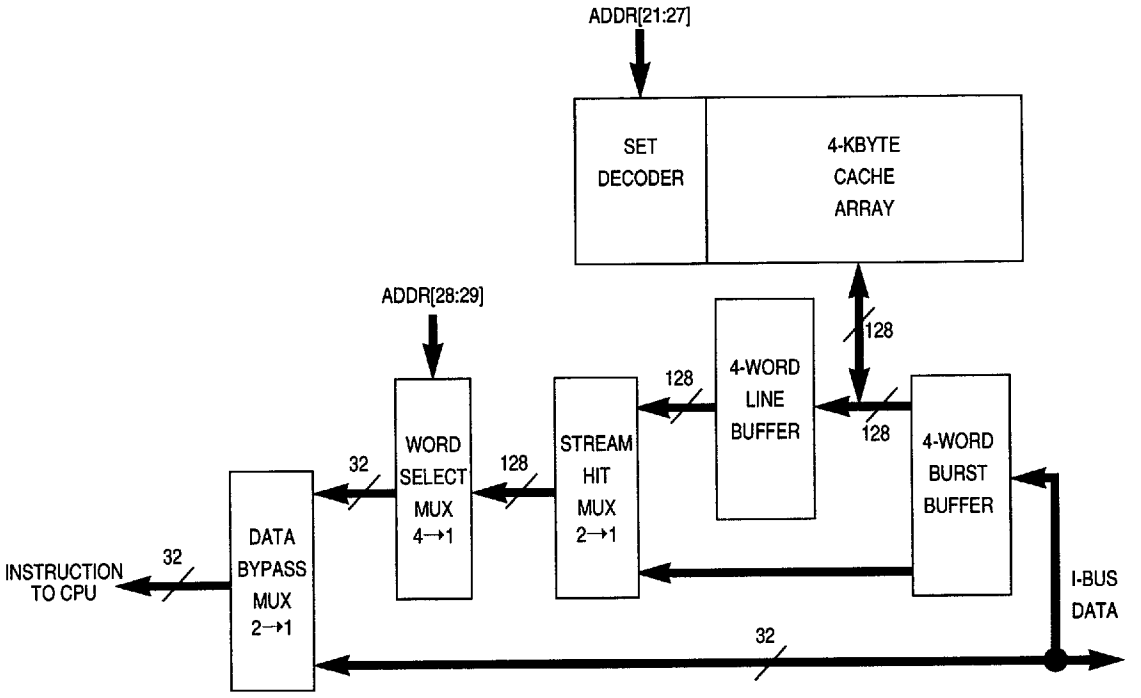


Figure 10 Instruction Cache Data Path

4.3 Programming Model

Three special purpose registers (SPRs) control the I-cache:

Table 25 Instruction Cache Programming Model

Name	SPR Number (Decimal)	Description
ICCST	560	I-cache control and status register
ICADR	561	I-cache address register
ICDAT	562	I-cache data port (read only)

These registers are privileged; attempting to access them when the CPU is operating at the user privilege level results in a program exception.

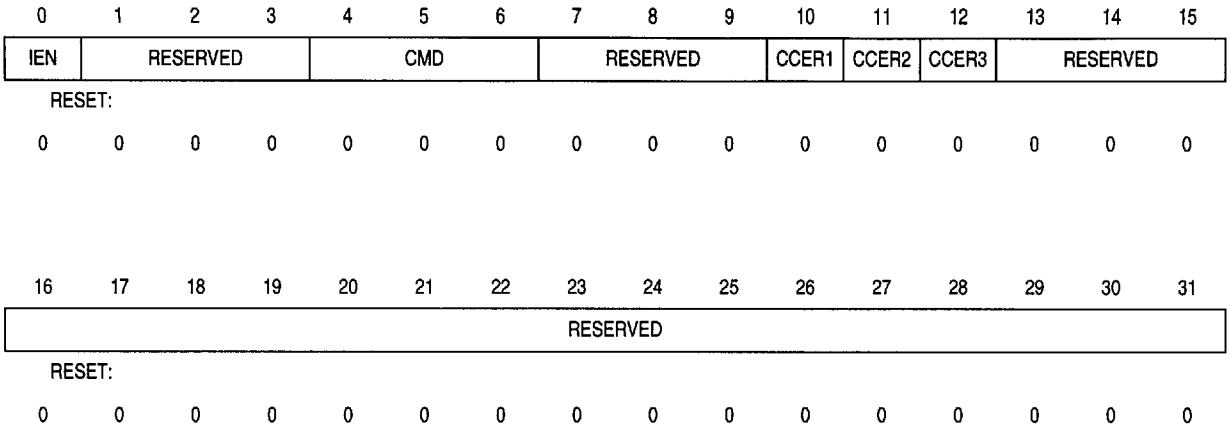


Table 26 ICCST Bit Settings

Bits	Mnemonic	Description
0	IEN	I-cache enable status bit. This bit is a read-only bit. Any attempt to write it is ignored. 0 = I-cache is disabled 1 = I-cache is enabled
[1:3]	—	Reserved
[4:6]	CMD	I-Cache Command 000 = No command 001 = **Cache enable** 010 = **Cache disable** 011 = **Load & lock** 100 = **Unlock line** 101 = **Unlock all** 110 = **Invalidate all** 111 = Reserved
[7:9]	—	Reserved
10	CCER1	I-Cache Error Type 1 (sticky bit) 0 = No error 1 = Error
11	CCER2	I-Cache Error Type 2 (sticky bit) 0 = No error 1 = Error
12	CCER3	I-Cache Error Type 3 (sticky bit) 0 = No error 1 = Error
[13:31]	—	Reserved

ICADR — I-Cache Address Register **SPR561**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADR																															
RESET:																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 27 I-Cache Address Register (ICADR)

Bits	Mnemonic	Description
[0:31]	ADR	The address to be used in the command programmed in the control and status register

ICSDAT — I-Cache Data Register **SPR 562**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DAT																															
RESET:																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28 I-Cache Data Register (ICDAT)

Bits	Mnemonic	Description
[0:31]	DAT	The data received when reading information from the I-cache

4.4 Cache Operation

On an instruction fetch, bits [21:27] of the instruction's address are used as an index into the cache to retrieve the tags and data of one set. The tags from both accessed lines are then compared to bits [0:20] of the instruction's address. If a match is found and the matched entry is valid, then the access is a cache hit.

If neither tag matches or if the matched tag is not valid, the access is a cache miss.

The I-cache includes one burst buffer that holds the last line received from the bus, and one line buffer that holds the last line received from the cache array. If the requested data is found in one of these buffers, the access is considered a cache hit.

To minimize power consumption, the I-cache attempts to make use of data stored in one of its internal buffers. Using a special indication from the CPU, it is also possible, in some cases, to detect that the requested data is in one of the buffers early enough so the cache array is not activated at all.

4.4.1 Cache Hit

On a cache hit, bits [28:29] of the instruction's address are used to select one word from the cache line whose tag matched. In the same clock cycle, the instruction is transferred to the instruction unit of the processor.

4.4.2 Cache Miss

On a cache miss, the address of the missed instruction is driven on the I-bus with a four-word burst transfer read request. A cache line is then selected to receive the data that will be coming from the bus. The selection algorithm gives first priority to invalid lines. If neither of the two candidate lines in the selected set are invalid, then the least recently used line is selected for replacement. Locked lines are never replaced.

The transfer begins with the word requested by the instruction unit (critical word first), followed by any remaining words of the line, then by any remaining words at the beginning of the line (wrap around). As the missed instruction is received from the bus, it is immediately delivered to the instruction unit and also written to the burst buffer.

As subsequent instructions are received from the bus they are also written into the burst buffer and, if needed, delivered to the instruction unit (stream hit) either directly from the bus or from the burst buffer. When the entire line resides in the burst buffer, it is written to the cache array if the cache array is not busy with an instruction unit request.

If a bus error is encountered on the access to the requested instruction, a machine check interrupt is taken. If a bus error occurs on any access to other words in the line, the burst buffer is marked invalid and the line is not written to the array. If no bus error is encountered, the burst buffer is marked valid and eventually is written to the array.

Together with the missed word, an indication may arrive from the I-bus that the memory device is non-cacheable. If such an indication is received, the line is not written to the cache, so that subsequent references to the same line will cause the line to be refetched.

4.5 Cache Commands

The MPC505 instruction cache supports the PowerPC invalidate instruction together with some additional commands that help control the cache and debug the information stored in it. The additional commands are implemented using the three special purpose control registers ICCST, ICADR, and ICDAT.

Most of the commands are executed immediately after the control register is written and cannot generate any errors. When these commands are executed, there is no need to check the error status in the ICCST.

Some commands may take longer and may generate errors. In the MPC505, only **load & lock** is such a command. When executing this command, the user needs to insert an **isync** instruction immediately after the I-cache command and check the error status in the ICCST after the **isync**. The error type bits in the ICCST are sticky, allowing the user to perform a series of I-cache commands before checking the termination status. These bits are set by hardware and cleared by software.

All cache commands except the **icbi** CPU instruction require setting the appropriate bits in the ICCST. Since the ICCST is a supervisor-level register, only the **icbi** instruction can be performed at the user privilege level.

4.5.1 Instruction Cache Block Invalidate

The MPC505 implements the PowerPC instruction cache block invalidate (**icbi**) as if it pertains only to the MPC505 instruction cache. This instruction does not broadcast on the external bus and the CPU does not snoop this instruction if broadcast by other masters.

This command is not privileged and has no error cases that the user needs to check.

4.5.2 Invalidate All

To invalidate the whole cache, set the **invalidate all** command in the ICCST. This command has no error cases that the user needs to check.

The command makes all valid lines in the cache invalid, except lines that are locked. After this command is executed, the LRU pointer of each set points to an unlocked way. If both lines in the set are unlocked, the LRU pointer points to way zero. This last feature is useful in order to initialize the I-cache out of reset.

4.5.3 Load and Lock

The **load & lock** operation is used to lock critical code segments in the cache. The **load & lock** operation is performed on a single cache line. After a line is locked it operates as a regular instruction SRAM; it will not be replaced during future misses and will not be affected by invalidate commands.

After the **load & lock** command is written to the ICCST, the cache checks if the line containing the byte addressed by the ICADR is in the cache. If it is, the line is locked and the command terminates with no exception. If the line is not in the cache a regular miss sequence is initiated. After the whole line is placed in the cache the line is locked.

The user needs to check the error type bits in the ICCST to determine if the operation completed properly or not. The **load & lock** command can generate two errors:

- Type 1 — bus error in one of the cycles that fetches the line.
- Type 2 — no place to lock. It is the responsibility of the user to make sure that there is at least one unlocked way in the appropriate set.

4.5.4 Unlock Line

The **unlock line** operation is used to unlock locked cache lines. The **unlock line** operation is performed on a single cache line. If the line is found in the cache (cache hit), it is unlocked and starts to operate as a regular valid cache line. If the line is not found in the cache (cache miss), no operation is performed, and the command terminates with no exception.

This command has no error cases that the user needs to check.

4.5.5 Unlock All

The **unlock all** operation is used to unlock the whole cache. This operation is performed on all cache lines. If a line is locked, it is unlocked and starts to operate as a regular valid cache line. If a line is not locked or if it is invalid, no operation is performed.

This command has no error cases that the user needs to check.

4.5.6 Cache Enable

To enable the cache, set the **cache enable** command in the ICCST. This operation can be performed only at the supervisor privilege level. The **cache enable** command has no error cases that the user needs to check.

Following reset, the **invalidate all** and **unlock all** commands must be performed before the **cache enable** command.

4.5.7 Cache Disable

To disable the cache, set the **cache disable** command in the ICCST. This operation can be performed only at the supervisor privilege level. The cache disable command has no error cases that the user needs to check.

4.5.8 Cache Inhibit

A memory region can be programmed in the chip select logic to be cache inhibit. Any reference to a cache inhibited memory region always results in cache miss.

4.5.9 Cache Read

The user can read all data stored in the instruction cache, including the data stored in the tags array.

To read the data that is stored in the I-cache, follow these steps:

1. Write to the ICADR the address of the data to be read. Note that it is also possible to read this register for debugging purposes.
2. Read the ICDAT.

So that all parts of the I-cache can be accessed, the ICADR is divided into several fields, shown in Table 29.

Table 29 ICADR Bits Function for the Cache Read Command

[0:17]	18	19	20	[21:27]	[28:29]	[30:31]
Reserved	0 = tag 1 = data	0 = way 0 1 = way 1	Reserved	Set select	Word select (used only for data array)	Reserved

When the data array is read from, the 32 bits of the word selected by the ICADR are placed in the target general-purpose register.

When the tag array is read, the 21 bits of the tag selected by the ICADR, along with additional information, are placed in the target general-purpose register. Table 30 illustrates the bits layout of the I-cache data register when a tag is read.

Table 30 ICDAT Layout During a Tag Read

[0:20]	21	22	23	24	[25:31]
Tag value	Reserved	0 = not valid 1 = valid	0 = not locked 1 = locked	LRU bit	Reserved

5 SYSTEM INTERFACE UNIT

The system interface unit (SIU) controls the buses of the chip and provides a system clock, chip selects, system protection, I/O ports, and reset control.

The MPC505 has an internal Harvard architecture and a single external bus. The internal buses are the instruction bus (I-bus) and the load/store bus (L-bus). The external bus interface (EBI) connects each of these internal buses with the external bus (E-bus). The chip select block provides user-programmable chip selects to select external memory or peripherals. The clock block controls the generation of the system clocks and such features as programmability of the clocks and low power modes. The reset control function interfaces to the reset pins and provides a reset status register. The I/O ports provide untimed I/O functions on pins that are not used for their primary function.

A block diagram of the SIU is shown in Figure 11.

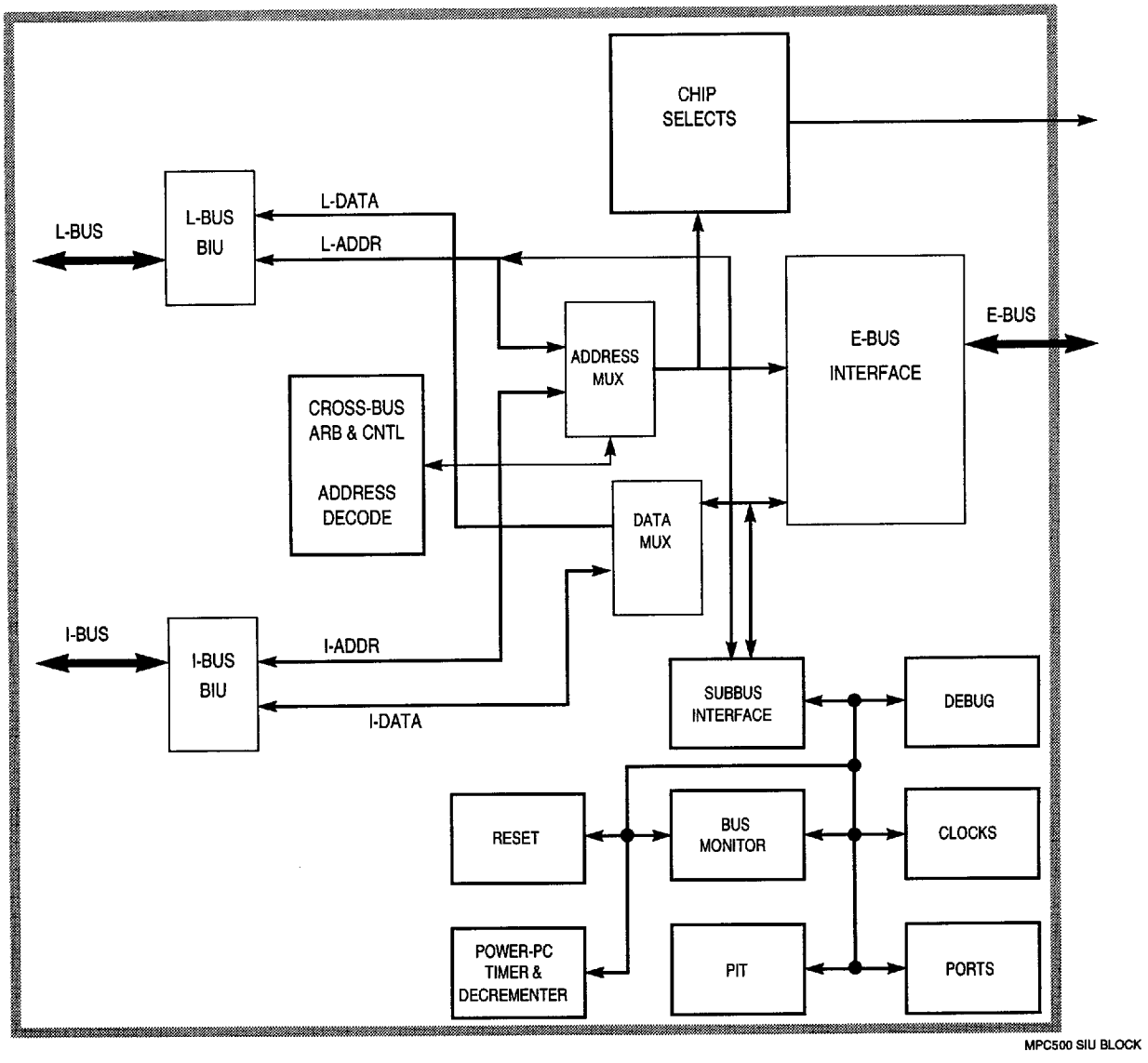


Figure 11 SIU Block Diagram

5.1 SIU Address Map

Table 31 is an address map of the SIU registers.

Table 31 SIU Address Map

Access	Address	Register
S	0x8007 FC00	SIU MODULE CONFIGURATION REGISTER (SIUMCR)
Test	0x8007 FC04	SIU TEST REGISTER 1 (SIUTEST1)
—	0x8007 FC08 – 0x8007 FC1C	RESERVED
S	0x8007 FC20	MEMORY MAPPING (MEMMAP)
S	0x8007 FC24	SPECULATIVE ADDRESS REGISTER (SPECADDR)
S	0x8007 FC28	SPECULATIVE MASK REGISTER (SPECMASK)
Test	0x8007 FC2C	TERMINATION STATUS REGISTER (TERMSTAT)
—	0x8007 FC30 – 0x8007 FC3C	RESERVED
S/U	0x8007 FC40	PERIODIC INTERRUPT CONTROL AND STATUS REGISTER (PICSR)
S/U	0x8007 FC44	PERIODIC INTERRUPT TIMER REGISTER (PIT)
S	0x8007 FC48	BUS MONITOR CONTROL REGISTER (BMCR)
S	0x8007 FC4C	RESET STATUS REGISTER (RSR)
S	0x8007 FC50	SYSTEM CLOCK CONTROL REGISTER (SCCR)
S	0x8007 FC54	SYSTEM CLOCK LOCK AND STATUS REGISTER (SCLSR)
—	0x8007 FC58 – 0x8007 FC5C	RESERVED
S	0x8007FC60	PORT M DATA DIRECTION (DDRM)
S	0x8007FC64	PORT M PIN ASSIGNMENT (PMPAR)
S/U	0x8007FC68	PORT M DATA (PORTM)
—	0x8007FC6C– 0x8007FC80	RESERVED
S	0x8007FC84	PORT A, B PIN ASSIGNMENT (PAPAR, PBPAR)
S/U	0x8007FC88	PORT A, B DATA (PORTA, PORTB)
—	0x8007FC8C– 0x8007FC94	RESERVED
S	0x8007FC98	PORT I, J, K, L DATA DIRECTION (DDRI, DDRJ, DDRK, DDRL)
S	0x8007FC9C	PORT I, J, K, L PIN ASSIGNMENT (PIPAR, PJPAR, PKPAR, PLPAR)
S/U	0x8007FCA0	PORT I, J, K, L DATA (PORTI, PORTJ, PORTK, PORTL)
—	0x8007 FCA4 – 0x8007 FD94	RESERVED

Table 31 SIU Address Map (Continued)

Access	Address	Register
S	0x8007 FD94	$\overline{CS11}$ OPTION REGISTER (CSOR11)
S	0x8007 FD98	RESERVED
S	0x8007 FD9C	$\overline{CS10}$ OPTION REGISTER (CSOR10)
S	0x8007 FDA0	RESERVED
S	0x8007 FDA4	$\overline{CS9}$ OPTION REGISTER (CSOR9)
S	0x8007 FDA8	RESERVED
S	0x8007 FDAC	$\overline{CS8}$ OPTION REGISTER (CSOR8)
S	0x8007 FDB0	RESERVED
S	0x8007 FDB4	$\overline{CS7}$ OPTION REGISTER (CSOR7)
S	0x8007 FDB8	RESERVED
S	0x8007 FDBC	$\overline{CS6}$ OPTION REGISTER (CSOR6)
S	0x8007 FDC0	$\overline{CS5}$ BASE ADDRESS REGISTER 5 (CSBAR5)
S	0x8007 FDC4	$\overline{CS5}$ OPTION REGISTER (CSOR5)
S	0x8007 FDC8	$\overline{CS4}$ BASE ADDRESS REGISTER (CSBAR4)
S	0x8007 FDCC	$\overline{CS4}$ OPTION REGISTER (CSOR4)
S	0x8007 FDD0	$\overline{CS3}$ BASE ADDRESS REGISTER (CSBAR3)
S	0x8007 FDD4	$\overline{CS3}$ OPTION REGISTER 3 (CSOR3)
S	0x8007 FDD8	$\overline{CS2}$ BASE ADDRESS REGISTER 2 (CSBAR2)
S	0x8007 FDDC	$\overline{CS2}$ OPTION REGISTER 2 (CSOR2)
S	0x8007 FDE0	$\overline{CS1}$ BASE ADDRESS REGISTER (CSBAR1)
S	0x8007 FDE4	$\overline{CS1}$ OPTION REGISTER (CSOR1)
S	0x8007 FDE8	RESERVED
S	0x8007 FDEC	$\overline{CS0}$ OPTION REGISTER (CSOR0)
S	0x8007 FDF0	$\overline{CSBOOT}$ SUB-BLOCK BASE ADDRESS REGISTER (CSBTSBBAR)
S	0x8007 FDF4	$\overline{CSBOOT}$ SUB-BLOCK OPTION REGISTER (CSBTSBOR)
S	0x8007 FDF8	$\overline{CSBOOT}$ BASE ADDRESS REGISTER (CSBTBAR)
S	0x8007 FDFC	$\overline{CSBOOT}$ OPTION REGISTER (CSBTOR)

5.2 SIU Module Configuration

The SIU module configuration register (SIUMCR) configures various aspects of SIU operation. The internal memory mapping register (MEMMAP) enables and sets the base address of the L-bus and I-bus internal memory blocks. These registers are accessible in supervisor mode only.

SIUMCR — SIU Module Configuration Register **0x8007 FC00**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SIUFRZ	RESERVED		CSR	LST	0	SUP		DLK	LOK	RESERVED				LSHOW	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PARTNUM								MASKNUM							
RESET:															
Read-Only Fixed Value								Read-Only Fixed Value							

Table 32 SIUMCR Bit Settings

Bit(s)	Name	Description
0	SIUFRZ	SIU Freeze 0 = Decrementer and time base registers and the periodic interrupt timer continue to run while internal freeze signal is asserted (reset value). 1 = Decrementer and time base registers and the periodic interrupt timer stop while the internal freeze signal is asserted.
[1:2]	—	Reserved
3	CSR	Checkstop reset enable 0 = No action taken when SIU receives the checkstop signal from the CPU and debug mode not enabled (reset value). 1 = SIU causes a reset upon receiving checkstop signal from CPU and debug mode not enabled. If debug mode is enabled, the MCU enters debug mode when the checkstop signal is received, regardless of CSR value.
4	LST	Burst style: $\overline{\text{BDIP}}$ or $\overline{\text{LAST}}$ 0 = $\overline{\text{BDIP}}$ pin uses $\overline{\text{BDIP}}$ timing (reset value): assert $\overline{\text{BDIP}}$ during burst, negate $\overline{\text{BDIP}}$ during last beat of burst 1 = $\overline{\text{BDIP}}$ pin uses $\overline{\text{LAST}}$ timing: assert $\overline{\text{LAST}}$ during last beat of burst
5	—	Reserved
[6:7]	SUP	Supervisor/unrestricted space. These bits control access to certain SIU registers. (Other registers are always supervisor access only.) 00 = Unrestricted access (reset value) 01 = Supervisor mode access only 10 = Supervisor mode write access only, unrestricted read access 11 = Supervisor mode access only

Table 32 SIUMCR Bit Settings (Continued)

Bit(s)	Name	Description
8	DLK	Debug register lock. This bit can be written only when internal freeze signal is asserted. DLK allows development software to configure show cycles and prevent normal software from subsequently changing this configuration. This bit overrides the LOK in controlling the LSHOW field. 0 = Writes to LSHOW field in SIUMCR are allowed (reset value). 1 = Writes to LSHOW field are not allowed.
9	LOK	Register lock. Once this bit is set, writes to the SIUMCR and chip-select registers have no effect and cause a data error to be generated in the internal bus. In normal operation, this is a set-only bit; once set, it cannot be cleared by software. When the internal freeze signal is asserted, the bit can be set or cleared by software. 0 = Normal operation (reset value) 1 = All bits in the SIUMCR and all of the chip-select registers are locked
[10:13]	—	Reserved
[14:15]	LSHOW	L-bus show cycles 00 = Disable show cycles for all internal L-bus cycles (reset value) 01 = Show address and data of all internal L-bus write cycles 10 = Reserved 11 = Show address and data of all internal L-bus cycles
[16:23]	PARTNUM	Part number. This read-only field is mask programmed with a code corresponding to the number of the MCU.
[24:31]	MASKNUM	Mask number. This read-only field is mask programmed with a code corresponding to the mask number of the MCU.

MEMMAP — Memory Mapping Register **0x8007 FC20**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LEN	RESERVED							LMEMBASE		RESERVED					
RESET:															
*	0	0	0	0	0	0	0	*	*	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
IEN	LIX	RESERVED						IMEMBASE		RESERVED					
RESET:															
*	1	0	0	0	0	0	0	*	*	0	0	0	0	0	0

* Reset value depends on the value of the reset configuration word.

Table 33 MEMMAP Bit Settings

Bit(s)	Name	Description
0	LEN	L-bus memory enable 0 = L-bus memory disabled 1 = L-bus memory enabled Reset state depends on the reset configuration word.
[1:7]	—	Reserved

Table 33 MEMMAP Bit Settings (Continued)

Bit(s)	Name	Description
[8:9]	LMEMBASE	Base address of the L-bus memory block (SRAM) 00 = Starting address is 0x0000 0000 01 = Ending address is 0x000F FFFF 10 = Starting address is 0xFFF0 0000 11 = Ending address is 0xFFFF FFFF Reset value depends on the reset configuration word.
[10:15]	—	Reserved
16	IEN	I-bus memory enable. This bit has no effect on the MPC505, which has no I-bus memory. 0 = I-bus memory disabled 1 = I-bus memory enabled
17	LIX	L-bus to I-bus cross bus access enable 0 = Disable data accesses to I-bus memory 1 = Enable data accesses to I-bus memory (reset value)
[18:23]	—	Reserved
[24:25]	IMEMBASE	Base address of the I-bus memory block. These bits have no effect on the MPC505, which has no I-bus memory. 00 = Starting address is 0x0000 0000 01 = Ending address is 0x000F FFFF 10 = Starting address is 0xFFF0 0000 11 = Ending address is 0xFFFF FFFF Reset state depends on the reset configuration word.
[26:31]	—	Reserved

5.3 External Bus Interface

The external bus interface (EBI) interfaces the external bus (E-bus) with the two internal buses (I-bus and L-bus). The E-bus can perform synchronous, pipeline, or burst transfers. Signals driven onto the E-bus are required to meet the set-up and hold times relative to the rising edge of the bus clock. The bus has the ability to support multiple masters, but its protocol is optimized for a single-processor environment.

5.3.1 Features

- No external glue logic required for a simple system.
- Supports different memory (SRAM, EEPROM) types: asynchronous, synchronous, pipelineable, burstable.
- Fast (one-clock) arbitration possible.
- Bus is synchronous — all signals are referenced to the rising edge of the bus clock.
- 32-bit data bus, 32-bit address bus with byte enables.
- Protocol allows wait states to be inserted during the data phase and supports early burst termination.
- Supports devices with small port sizes.
- Bus electrical specification minimizes system power consumption.

5.3.2 Basic Pipeline

The EBI supports a pipeline depth of one, meaning that two addresses can be active on a bus at the same time. Pipelining is simplified by using SIU chip selects, since chip-select registers can have the information about the characteristics of each external memory. It is also possible to have back-to-back address phases on the external bus, if a chip select returns $\overline{\text{AACK}}$ for the first address phase immediately. **5.4 Chip Selects** discusses which cycles can be pipelined.

5.3.3 Burst Transfers

The RCPU can initiate burst read cycles but not burst write cycles. At the start of a burst transfer, the master drives the address, the address attributes, transfer start, and the $\overline{\text{BURST}}$ signal to indicate a burst transfer. If the slave can perform burst transfers, it negates the burst inhibit signal ($\overline{\text{BI}}$). If the slave does not support burst transfers, it asserts $\overline{\text{BI}}$.

During the data phase of a burst read cycle, the master receives data from the addressed slave. If the master needs more than one data item, it asserts $\overline{\text{BDIP}}$. Upon receiving the second-to-last piece of data, the master negates $\overline{\text{BDIP}}$. The slave stops driving new data after it receives the negation of $\overline{\text{BDIP}}$ at the rising edge of the clock.

5.3.4 External Bus Signals

Table 34 summarizes the E-bus signals. The following abbreviations are used in this table:

M = Bus master
S = Slave device
A = Central bus arbiter
T = Bus watchdog timer
X = Any device on the system

Table 34 EBI Signal Descriptions

Mnemonic	Direction	Description
Address and Data Bus		
ADDR[0:29]	M → S	32-bit address bus. Driven by the bus master to index the bus slave.
DATA[0:31]	M ↔ S	32-bit data bus.
Transfer Attributes		
$\overline{\text{WR}}$	M → S	Asserted: write cycle. Negated: read cycle.
$\overline{\text{BURST}}$	M → S	If asserted, indicates cycle is a burst cycle.
$\overline{\text{BE}}$[0:3]	M → S	Byte enables. One byte enable per byte lane of the data bus. Refer to Table 36.
AT[0:1]	M → S	Address types. Define addressed space as user or supervisor, data or instruction. Refer to Table 35 for encodings.
CT[0:3]	M → S	Cycle type signals. Indicate what type of bus cycle the bus master is initiating. Used for development support. Refer to Table 37 for encodings.
Transfer Handshakes		
$\overline{\text{TS}}$	M → S	Transfer start. When asserted, indicates the start of a bus cycle.
$\overline{\text{AACK}}$	S → M	Address acknowledge. When asserted, indicates the slave has received the address from the bus master.
$\overline{\text{BDIP}}$	M → S	Burst data in progress. When asserted, indicates the data beat in front of the current one is needed by the master. This signal is negated prior to the end of a burst to terminate the burst data phase early.
$\overline{\text{BI}}$	S → M	Burst inhibit. When asserted, indicates the slave does not support burst mode. Sampled at same time as $\overline{\text{AACK}}$.

Table 34 EBI Signal Descriptions (Continued)

Mnemonic	Direction	Description
$\overline{TA}$	S → M	Transfer acknowledge. When asserted, indicates the slave has received the data during a write cycle or returned the data during a read cycle.
$\overline{TEA}$	T, S → M	Transfer error acknowledge. When asserted, indicates an error condition has occurred during the bus cycle.
DS	M → S	Data strobe. Asserted by the EBI at the end of a chip-select-controlled bus cycle after the chip-select unit asserts the internal $\overline{TA}$ signal or the bus monitor asserts the internal $\overline{TEA}$ signal. Also asserted at the end of a show cycle. Used for development support.
$\overline{ARETRY}$	S, A → M	Address retry. When asserted, indicates the master needs to retry its address phase.
Arbitration		
$\overline{BR}$	M → A	Bus request. When asserted, indicates the potential bus master is requesting the bus. Each master has its own bus request signal.
$\overline{BG}$	A → M	Bus grant. When asserted by bus arbiter, the bus is granted to the bus master. Each master has its own bus grant signal.
$\overline{BB}$	M → M, A	Bus busy. Asserted by current bus master to indicate the bus is currently in use. Prospective new master should wait until the current master negates this signal.
Miscellaneous		
$\overline{CR}$	X → M	Cancel reservation. Each RCPU has its own $\overline{CR}$ signal. When asserted (by external reservation logic), instructs the bus master to clear its reservation; some other master has touched its reserved space.
$\overline{IRQ[0:6]}$	X → M	Interrupt request inputs.
$\overline{RESET}$	Source → M	Hard reset. When asserted, devices on the bus must reset.
$\overline{RESETOUT}$	M → S	Reset output. When asserted, instructs all devices monitoring this signal to reset all parts within themselves that can be reset by software.
CLKOUT	Source → M, S	Continuously-running clock. All signals driven on the E-bus must be synchronized to the rising edge of this clock.

Table 35 Address Type Encodings

AT0	AT1	Address Space
0	0	User data space
0	1	User instruction space
1	0	Supervisor data space
1	1	Supervisor instruction space

Table 36 Byte Enable Encodings

Byte Enable	Use During 32-Bit Port Access	Use During 16-Bit Port Access
$\overline{BE0}$	Byte Enable for DATA[0:7]	Byte Enable for DATA[0:7]
$\overline{BE1}$	Byte Enable for DATA[8:15]	Byte Enable for DATA[8:15]
$\overline{BE2}$	Byte Enable for DATA[16:23]	ADDR30
$\overline{BE3}$	Byte Enable for DATA[24:31]	0 = Operand size is word 1 = Operand size is byte or half word

Table 37 shows the encodings for the cycle type pins. Refer to the *RCPU Reference Manual* (RCPURM/AD) for more information on using the cycle type pins in a development system.

Table 37 Cycle Type Encodings

CT[0:3]	Cycle Type	Description
0000	Normal bus cycle	This is a normal external bus cycle. Both the address and data phase are seen on the external bus. This cycle requires an $\overline{AACK}$ and a $\overline{TA}$ signal. This cycle type is used for sequential fetches and for prefetches of predicted branch targets where the branch condition has not been evaluated before the prefetch. It is also used for all non-reservation type load/store cycles.
0001	Reservation start if address type is data OR Instruction fetch marked as indirect change-of-flow if address type is instruction	If the address type is data (AT1 = 0), then this is a data access to the external bus. Both the address and the data phase are seen on the external bus. This cycle requires an $\overline{AACK}$ and a $\overline{TA}$ signal. When this cycle starts, external snooping logic should latch the address to track the reservation. If the address type is instruction (AT1 = 1), then this is an instruction fetch cycle marked as an indirect change-of-flow cycle. Both the address and the data phase are seen on the external bus. This cycle requires an $\overline{AACK}$ and a $\overline{TA}$ signal. This cycle type is used when an external address is the destination of a branch instruction or the destination of an exception or VSYNC cycle.
0010	Emulation memory select (not supported in MPC505)	This is a special external bus cycle to emulation memory replacing internal I-mem or L-mem (and not resulting in a cache hit). The MPC505 does not support this cycle type.
0011	PRU select (not supported in MPC505)	This is a normal external bus cycle access to a port replacement chip used for emulation support. Both the address and the data phase are seen on the external bus. This cycle requires an $\overline{AACK}$ and a $\overline{TA}$ signal. It indicates that an access was made which would have gone to an internal port control register if the chip were not operating in PRU mode.

Table 37 Cycle Type Encodings (Continued)

CT[0:3]	Cycle Type	Description
0100	I-Mem (not supported in MPC505)	These are internal visibility cycles. This cycle is self-terminating and does not require $\overline{AACK}$ and $\overline{TA}$ signals. These encodings indicate that an access or aborted fetch (resulting from either a cache hit or a speculative load that is subsequently discarded) was made to an address on the internal I-bus or L-bus. An instruction access (AT1=1) with an address which is an indirect branch target is indicated as a write on the $\overline{WR}$ signal. The I-Mem cycle type is not supported in the MPC505.
0101	L-Mem	
0110	E-Mem cache hit, not using a chip select	This is an internal visibility cycle. It always has an address phase and includes a data phase for data accesses. This cycle is self-terminating and does not require $\overline{AACK}$ and $\overline{TA}$ handshaking signals. It indicates that an access was made to an address on the external bus and that a cache hit or an aborted fetch (resulting from a speculative load that is subsequently discarded) occurred. An instruction access with an address that is an indirect branch target is indicated as a write on the $\overline{WR}$ signal.
0111	Internal register	This is an internal visibility cycle. It always has an address phase and a data phase. This cycle is self-terminating and does not require $\overline{AACK}$ and $\overline{TA}$ signals. It indicates that an access was made to a control register or internal IMB address. These accesses are always cache-inhibited.
1000	E-Mem cache hit to $\overline{CSBOOT}$ region	These are internal visibility cycles. They always have an address phase and include a data phase for data accesses. These cycles are self terminating and do not require $\overline{AACK}$ and $\overline{TA}$ handshaking signals. These encodings indicate that an access was made to an address on the external bus and that a cache hit or an aborted fetch (resulting from a speculative load that is subsequently discarded) occurred. An instruction access with an address that is an indirect branch target is indicated as a write on the $\overline{WR}$ signal. The region indicated is the main chip-select region, not the sub-region.
1001	E-Mem cache hit to $\overline{CS1}$ region	
1010	E-Mem cache hit to $\overline{CS2}$ region	
1011	E-Mem cache hit to $\overline{CS3}$ region	
1100	E-Mem cache hit to $\overline{CS4}$ region	
1101	E-Mem cache hit to $\overline{CS5}$ region	
1110	Reserved	—
1111	Reserved	—

5.3.5 Preventing Speculative Loads

The SIU can be programmed to prevent speculative loads to a selected external region. A speculative load is an out-of-order load that may be subsequently discarded, such as a load following an unresolved conditional branch. Preventing speculative loads may be necessary to ensure that certain devices operate correctly, such as a memory-mapped I/O device with a status register that is updated whenever it is read.

Two registers and their associated logic allow a block ranging in size from 1 Kbyte to 64 Kbytes, or parts of the block, to be protected from speculative accesses. The most significant 22 bits of the address of each L-bus cycle are bitwise compared to the non-speculative base address register (SPECADDR). Each result bit equal to one if the corresponding bits of the address bus and SPEC-

ADDR register match. A bitwise OR is performed on the lower six bits of the resulting word with the mask in the non-speculative mask register (SPECMASK). If all six of these result bits are one, and if all 16 of the higher result bits are also one (i.e., the upper 16 bits of the address match), then speculative accesses are prevented during the current cycle.

SPECADDR — Non-Speculative Base Address Register **0x8007 FC24**

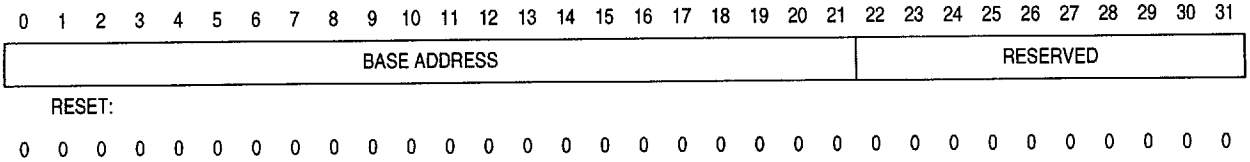


Table 38 SPECADDR Bit Settings

Bit(s)	Name	Description
[0:21]	BASE ADDRESS	22-bit base address of region protected from speculative loads.
[22:31]	—	Reserved

SPECMASK — Non-Speculative Mask Register **0x8007 FC28**

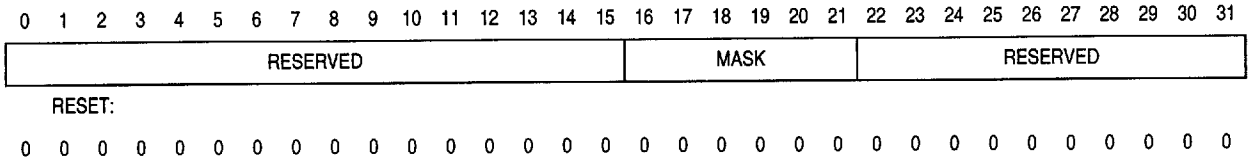


Table 39 SPECMASK Bit Settings

Bit(s)	Name	Description
[0:15]	—	Reserved
[16:21]	MASK	6-bit mask that specifies which block or blocks within region specified in SPECMASK register are actually protected from speculative accesses.
[22:31]	—	Reserved

Because the mask register can contain any 6-bit value, the mask can allow for blocks of up to 64 Kbytes, and it can provide for smaller blocks of memory that alternately allow and prevent speculative loads. Table 40 provides several examples. In these examples, the protected blocks are those that match the value in the SPECADDR register.

Table 40 Speculative Mask Values

Mask Value (Binary)	Protected Region
000000	1-Kbyte block
111111	64-Kbyte block
111110	Every second 1-Kbyte block within a 64-Kbyte block
111101	Every second 2-Kbyte block within a 64-Kbyte block
110011	Every fourth 4-Kbyte block within a 64-Kbyte block
100011	Every eighth 4-Kbyte block within a 64-Kbyte block
010000	Every sixteenth 1-Kbyte block within a 32-Kbyte block

Protection from speculative loads can be disabled by setting the SPECADDR register to an internal or unimplemented address range.

5.3.6 Accesses to 16-Bit Ports

The EBI supports accesses to 16-bit ports on the external bus. 16-bit port access is a chip-select option; the access must be initiated using one of the chip selects ($\overline{CS[0:11]}$). A 16-bit port must connect its data lines to the upper 16 bits of the external data bus (DATA[0:15]). During an access to a 16-bit port, byte enable signals $\overline{BE[0:1]}$ are used to indicate which bytes of the half-word are being accessed, and the $\overline{BE2}$/ADDR30 pin functions as ADDR30 (active high). $\overline{BE3}$ is asserted (low) if the operand is a word and negated (high) if the operand is a byte or half-word.

5.4 Chip Selects

Typical microcontrollers require additional hardware to provide external chip-select signals. On the MPC505, the chip select logic controls the slaves of typical uniprocessor systems. This allows the user to implement simple systems without the need to design any external glue logic.

Figure 12 is an example of a typical uniprocessor system. This kind of system usually consists of a CPU, some memories, and some peripherals. In single-master systems the CPU is the only device that can be a bus master on the E-bus; memories and peripherals are slaves.

The chip-select module provides the necessary control signals, such as the chip enable ($\overline{CE}$), write enable ($\overline{WE}$), and output enable ($\overline{OE}$), for the external memory and peripheral devices. In addition, the chip-select module provides some handshakes for the external bus and some limited protection mechanisms for the system.

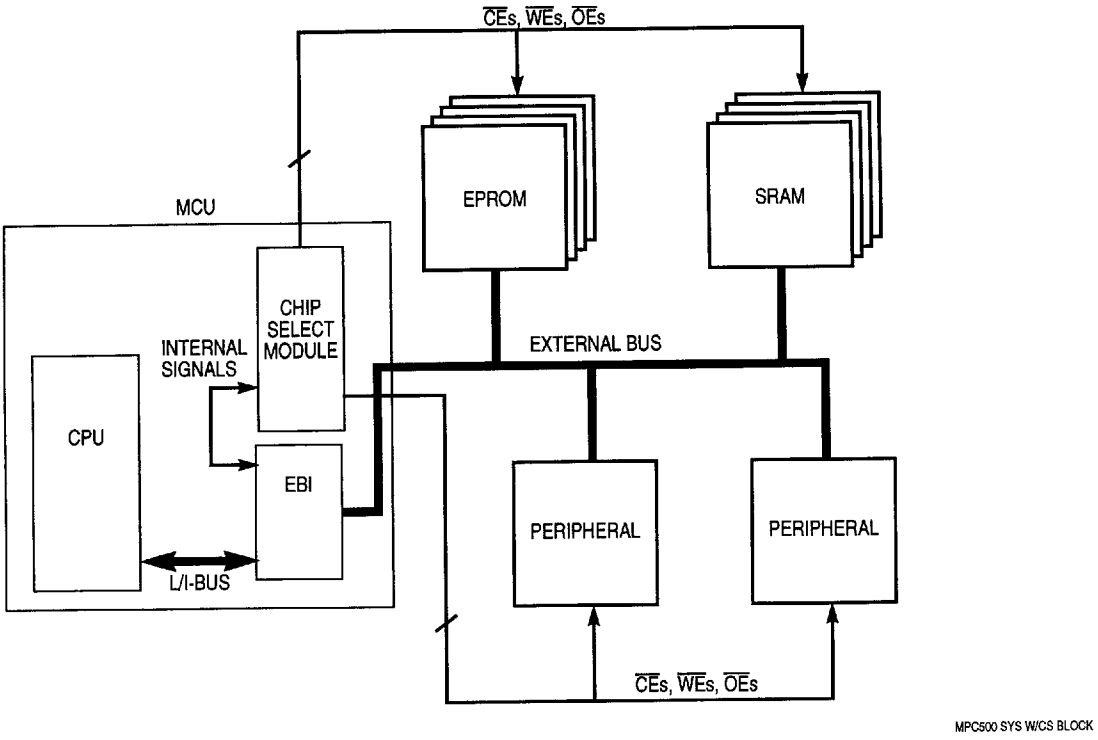


Figure 12 Simplified Uniprocessor System with Chip-Select Logic

5.4.1 Chip-Select Module Features

- No external glue logic required for simple systems if the chip-select module is used.
- Modular architecture for ease of expansion.
- 12 pins programmable as $\overline{\text{CE}}$s, $\overline{\text{OE}}$s, or $\overline{\text{WE}}$s plus one $\overline{\text{CSBOOT}}$ pin.
- Capable of supporting pipelineable, burstable devices.
- Returns bus handshake signals internally for the selected address regions.
- Supports up to six different regions.
- Up to seven programmable wait states for slave devices.
- Arbitrates the data path for slave devices
- Controls the clocking of data to the slaves during write cycles.
- Keeps slave sequentially consistent (data in the same order as addresses).
- Programmability for:
  - — Latching and non-latching device types.
  - — Burstable and non-burstable device types.
- Programmable address range, block size.
- Programmable burst features:
  - — Interruptible burst on any burstable device.
  - — Pipelineable with other devices during burst cycle.
  - — Supports two different burst protocols.
- Supports pipelineable accesses
  - — Up to two concurrent accesses can be outstanding to two different regions (one access to each region).
  - — Overlaps second address with first data phase of accesses to the same region.
- Allows multi-level protections within one region.

5.4.2 Chip-Selects Block Diagram

Figure 13 shows the functional block diagram of the chip-select module.

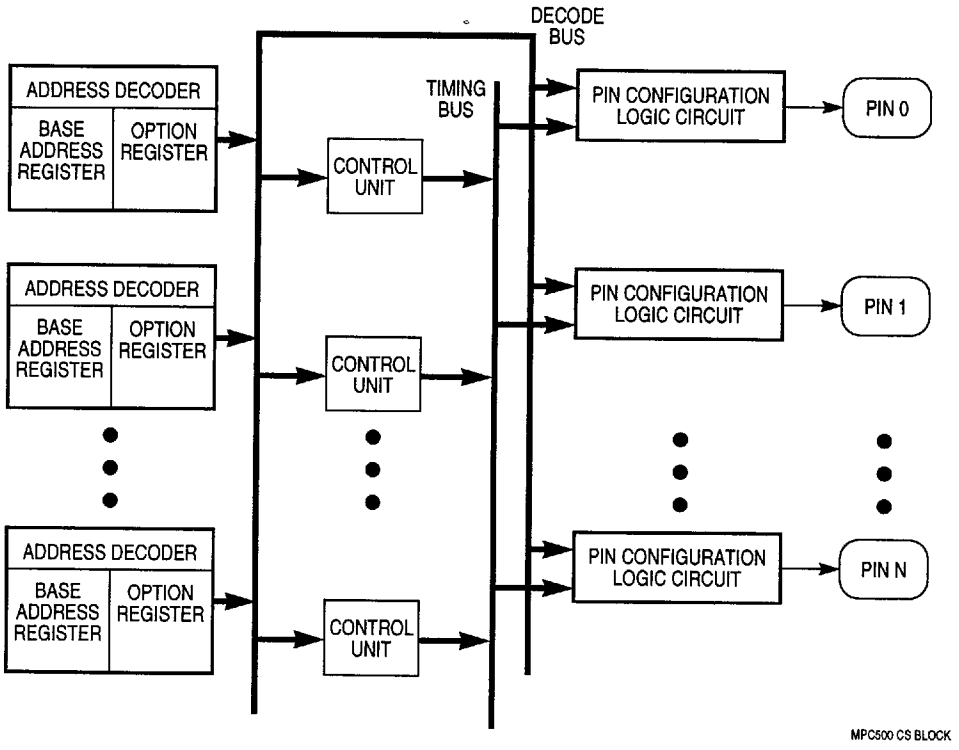


Figure 13 Chip-Select Functional Block Diagram

5.4.3 Chip-Select Pins

The pin configuration (PCON) field in each chip-select option register configures the associated pin to function as a chip enable ($\overline{CE}$), write enable ($\overline{WE}$), output enable ($\overline{OE}$), or alternate-function pin. For pins configured for their alternate function, the port A pin assignment register configures the pin as either an address bus signal (ADDR[0:11]) or a port A or B output signal (PA[0:7] and PB[0:3]). Notice that the $\overline{CSBOOT}$ pin has no alternate function.

Table 41 describes the chip-select pins.

Table 41 Chip-Select Pin Functions

Chip-Select Function	Alternate Function	Pin Function in Chip-Select Mode
$\overline{CSBOOT}$	—	Can be the $\overline{CE}$ of the system boot memory (power-on default). In systems with no external boot device, this pin can be configured as $\overline{WE}$ or $\overline{OE}$ of EPROMs or SRAMs.
$\overline{CS0}$/ $\overline{CSBTOE}$	ADDR0/PA0	Can be $\overline{WE}$ or $\overline{OE}$ of EPROMs or SRAMs. When configured as a chip select, this pin is assigned to be the $\overline{OE}$ of the $\overline{CSBOOT}$ pin following reset.
$\overline{CS1}$	ADDR1/PA1	Can be $\overline{CE}$, $\overline{WE}$, or $\overline{OE}$ of EPROMs or SRAMs.
$\overline{CS2}$	ADDR2/PA2	Can be $\overline{CE}$, $\overline{WE}$, or $\overline{OE}$ of EPROMs or SRAMs.
$\overline{CS3}$	ADDR3/PA3	Can be $\overline{CE}$, $\overline{WE}$, or $\overline{OE}$ of EPROMs or SRAMs.
$\overline{CS4}$	ADDR4/PA4	Can be $\overline{CE}$, $\overline{WE}$, or $\overline{OE}$ of EPROMs or SRAMs.
$\overline{CS5}$	ADDR5/PA5	Can be $\overline{CE}$, $\overline{WE}$, or $\overline{OE}$ of EPROMs or SRAMs.
$\overline{CS6}$	ADDR6/PA6	Can be $\overline{WE}$ or $\overline{OE}$ of EPROMs or SRAMs.
$\overline{CS7}$	ADDR7/PA7	Can be $\overline{WE}$ or $\overline{OE}$ of EPROMs or SRAMs.
$\overline{CS8}$	ADDR8/PB0	Can be $\overline{WE}$ or $\overline{OE}$ of EPROMs or SRAMs.
$\overline{CS9}$	ADDR9/PB1	Can be $\overline{WE}$ or $\overline{OE}$ of EPROMs or SRAMs.
$\overline{CS10}$	ADDR10/PB2	Can be $\overline{WE}$ or $\overline{OE}$ of EPROMs or SRAMs.
$\overline{CS11}$	ADDR11/PB3	Can be $\overline{WE}$ or $\overline{OE}$ of EPROMs or SRAMs.

5.4.4 Chip-Select Registers and Address Map

Chip-select registers are 32 bits wide. Reads of unimplemented bits in these registers return zero, and writes have no effect.

One base address register and one option register are associated with each chip-select pin that can act as a chip enable. The $\overline{\text{CSBOOT}}$ pin has a dedicated sub-block for multi-level protection. It has two base address registers and two option registers. One option register is associated with each pin that cannot act as a chip enable.

Table 42 is an address map of the chip-select module. As the entries in the Access column indicate, all chip-select registers are accessible at the supervisor privilege level only.

Table 42 Chip-Select Module Address Map

Access	Address	Register
—	0x8007 FD00 – 0x8007 FD94	RESERVED
S	0x8007 FD94	$\overline{\text{CS11}}$ OPTION REGISTER (CSOR11)
S	0x8007 FD98	RESERVED
S	0x8007 FD9C	$\overline{\text{CS10}}$ OPTION REGISTER (CSOR10)
S	0x8007 FDA0	RESERVED
S	0x8007 FDA4	$\overline{\text{CS9}}$ OPTION REGISTER (CSOR9)
S	0x8007 FDA8	RESERVED
S	0x8007 FDAC	$\overline{\text{CS8}}$ OPTION REGISTER (CSOR8)
S	0x8007 FDB0	RESERVED
S	0x8007 FDB4	$\overline{\text{CS7}}$ OPTION REGISTER (CSOR7)
S	0x8007 FDB8	RESERVED
S	0x8007 FDBC	$\overline{\text{CS6}}$ OPTION REGISTER (CSOR6)
S	0x8007 FDC0	$\overline{\text{CS5}}$ BASE ADDRESS REGISTER 5 (CSBAR5)
S	0x8007 FDC4	$\overline{\text{CS5}}$ OPTION REGISTER (CSOR5)
S	0x8007 FDC8	$\overline{\text{CS4}}$ BASE ADDRESS REGISTER (CSBAR4)
S	0x8007 FDCC	$\overline{\text{CS4}}$ OPTION REGISTER (CSOR4)
S	0x8007 FDD0	$\overline{\text{CS3}}$ BASE ADDRESS REGISTER (CSBAR3)
S	0x8007 FDD4	$\overline{\text{CS3}}$ OPTION REGISTER 3 (CSOR3)
S	0x8007 FDD8	$\overline{\text{CS2}}$ BASE ADDRESS REGISTER 2 (CSBAR2)
S	0x8007 FDDC	$\overline{\text{CS2}}$ OPTION REGISTER 2 (CSOR2)
S	0x8007 FDE0	$\overline{\text{CS1}}$ BASE ADDRESS REGISTER (CSBAR1)
S	0x8007 FDE4	$\overline{\text{CS1}}$ OPTION REGISTER (CSOR1)
S	0x8007 FDE8	RESERVED
S	0x8007 FDEC	$\overline{\text{CS0}}$ OPTION REGISTER (CSOR0)

Table 42 Chip-Select Module Address Map (Continued)

Access	Address	Register
S	0x8007 FDF0	$\overline{\text{CSBOOT}}$ SUB-BLOCK BASE ADDRESS REGISTER (CSBTSBBAR)
S	0x8007 FDF4	$\overline{\text{CSBOOT}}$ SUB-BLOCK OPTION REGISTER (CSBTSBOR)
S	0x8007 FDF8	$\overline{\text{CSBOOT}}$ BASE ADDRESS REGISTER (CSBTBAR)
S	0x8007 FDFC	$\overline{\text{CSBOOT}}$ OPTION REGISTER (CSBTOR)

5.4.4.1 Chip-Select Base Address Registers

Base address registers contain the base address of the range of memory to which the chip select circuit responds. All base address registers contain the same fields but have different reset values.

CSBTBAR — $\overline{\text{CSBOOT}}$ Base Address Register **0x8007 FDF8**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BA																				RESERVED											
RESET:																															
IP	IP	IP	IP	IP	IP	IP	IP	IP	IP	IP	IP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The reset value of the BA field in the CSBTBAR equals 0x00000 if the exception prefix (IP) bit in the MSR is zero (default), and 0xFFF00 if IP equals one.

CSBTSBBAR — $\overline{\text{CSBOOT}}$ Sub-Block Base Address Register **0x8007 FDF0**
CSBAR1 — $\overline{\text{CS1}}$ Base Address Register **0x8007 FDE0**
CSBAR2 — $\overline{\text{CS2}}$ Base Address Register **0x8007 FDD8**
CSBAR3 — $\overline{\text{CS3}}$ Base Address Register **0x8007 FDD0**
CSBAR4 — $\overline{\text{CS4}}$ Base Address Register **0x8007 FDC8**
CSBAR5 — $\overline{\text{CS5}}$ Base Address Register **0x8007 FDC0**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BA																				RESERVED											
RESET:																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43 Chip-Select Base Address Registers Bit Settings

Bit(s)	Name	Description
[0:19]	BA	Base Address. Bits 0 through 19 of the base address of the block to which the chip select responds. Register bit 0 corresponds to address bit 0; register bit 19 corresponds to address bit 19.
1	—	Reserved.

5.4.4.2 Chip-Select Option Registers

The option register for $\overline{\text{CSBOOT}}$ has the same field definitions as the option registers for $\overline{\text{CS[1:5]}}$, but the $\overline{\text{CSBOOT}}$ option register (CSBTOR) has different reset values. The $\overline{\text{CS0}}$ and $\overline{\text{CS[6:10]}}$ option registers contain a subset of the fields in the CSBTOR. The CSBOOT sub-block option register contains a different subset of the fields in the CSBTOR.

CSBTOR — $\overline{\text{CSBOOT}}$ Option Register **0x8007 FDFC**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BSIZE				SBLK	SUPV	DSP	WP	CI	RESERVED				ACKEN	TADLY	
RESET:															
1	0	0	1	0	1	0	1	0	0	0	0	0	1	*	*

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TADLY	PS		PCON		BYTE		REGION			RESERVED		ITYPE			
RESET:															
*	*	*	0	0	0	0	0	0	0	0	0	*	*	*	*

*From data bus reset configuration word

CSBTSBOR — $\overline{\text{CSBOOT}}$ Sub-Block Option Register **0x8007 FDF4**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BSIZE				SBLK	SUPV	DSP	WP	CI	RESERVED						
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSOR0 — $\overline{\text{CS0}}$ Option Register **0x8007 FDEC**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED			PCON		BYTE		REGION			RESERVED					
RESET:															
0	0	0	*	*	0	0	0	0	0	0	0	0	0	0	0

*0b10 if pin is configured as a chip select, otherwise 0b11

CSOR1 — $\overline{CS1}$ Option Register
CSOR2 — $\overline{CS2}$ Option Register
CSOR3 — $\overline{CS3}$ Option Register
CSOR4 — $\overline{CS4}$ Option Register
CSOR5 — $\overline{CS5}$ Option Register

0x8007 FDE4
0x8007 FDDC
0x8007 FDD4
0x8007 FDCC
0x8007 FDC4

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BSIZE				SBLK	SUPV	DSP	WP	CI	RESERVED				ACKEN	TADLY	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TADLY	PS		PCON		BYTE		REGION			RESERVED		ITYPE			
RESET:															
0	0	0	*	*	0	0	0	0	0	0	0	0	0	0	0

* 0b10 if pin is configured as a chip select, otherwise 0b11

CSOR6 — $\overline{CS6}$ Option Register
CSOR7 — $\overline{CS7}$ Option Register
CSOR8 — $\overline{CS8}$ Option Register
CSOR9 — $\overline{CS9}$ Option Register
CSOR10 — $\overline{CS10}$ Option Register
CSOR11 — $\overline{CS11}$ Option Register

0x8007 FDBC
0x8007 FDB4
0x8007 FDAC
0x8007 FDA4
0x8007 FD9C
0x8007 FD94

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED			PCON		BYTE		REGION			RESERVED					
RESET:															
0	0	0	*	*	0	0	0	0	0	0	0	0	0	0	0

* 0b10 if pin is configured as a chip select, otherwise 0b11

Table 32 describes the fields in the chip-select option registers.

Table 44 Chip-Select Option Register Bit Settings

Bit(s)	Name	Description
[0:3]	BSIZE	Block size. This field determines the size of the block associated with the base address. Block size encodings are shown in Table 45.

Table 44 Chip-Select Option Register Bit Settings (Continued)

Bit(s)	Name	Description
4	SBLK	Sub-block 0 = Address space is a main block 1 = Address space specified by the BA and BSIZE fields of the corresponding base address and option registers, respectively, is a sub-block within a larger main block. Pairing of main blocks and sub-blocks is shown in Table 46.
5	SUPV	Supervisor mode 0 = Data access is permitted in supervisor or user mode 1 = Data access is permitted in supervisor mode only
6	DSP	Data space only 0 = Address block may contain both instructions and data. 1 = Address block contains data only.
7	WP	Write protect 0 = Block is available for both read and write operations 1 = Block is read only
8	CI	Cache inhibit 0 = Information in this block can be cached. 1 = Information in this block should not be cached.
[9:12]	—	Reserved
13	ACKEN	Acknowledge enable. 0 = Chip-select logic will not return $\overline{TA}$ and $\overline{AACK}$ signals 1 = Chip-select logic will return $\overline{TA}$ and $\overline{AACK}$ signals
[14:16]	TADLY	$\overline{TA}$ delay. Indicates the latency of the device for the first $\overline{TA}$ returned. Up to seven wait states are allowed. 000 = 0 wait states 001 = 1 wait state 010 = 2 wait states 011 = 3 wait states 100 = 4 wait states 101 = 5 wait states 110 = 6 wait states 111 = 7 wait states
[17:18]	PS	Port size 00 = Reserved 01 = 16-bit port 10 = 32-bit port 11 = Reserved
[19:20]	PCON	Pin configuration. Note that only pins $\overline{CSBOOT}$ and $\overline{CS}$[1:5] can be $\overline{CE}$ pins. 00 = Chip enable ($\overline{CE}$) 01 = Write enable ($\overline{WE}$) 10 = Output enable ($\overline{OE}$) 11 = Alternate function (address bus or discrete output)
[21:22]	BYTE	Byte enable. This field applies to pins configured as $\overline{WE}$s only. Specifies for which of the four bytes in a word the $\overline{WE}$ is asserted. If the region can always be written in 32-bit quantity, this field can be programmed to any value. 00 = Byte enable 0 01 = Byte enable 1 10 = Byte enable 2 11 = Byte enable 3

Table 44 Chip-Select Option Register Bit Settings (Continued)

Bit(s)	Name	Description
[23:25]	REGION	Memory region (only applicable when pin is configured to be a $\overline{WE}$ or $\overline{OE}$ pin). These bits indicate the memory region with which the pin is associated. 000 = $\overline{CSBOOT}$ 001 = $\overline{CS1}$ 010 = $\overline{CS2}$ 011 = $\overline{CS3}$ 100 = $\overline{CS4}$ 101 = $\overline{CS5}$ 110 = Reserved 111 = Reserved
[26:27]	—	Reserved
[28:31]	ITYPE	Interface type. Indicates the type of memory or peripheral device being controlled. Refer to Table 47 for details.

Table 45 Block Size Encoding

BSIZE Field (Binary)	Block Size (Bytes)	Address Lines Compared
0000	Invalid	Chip select is not asserted until BSIZE field is assigned a nonzero value.
0001	4 K	ADDR[0:19]
0010	8 K	ADDR[0:18]
0011	16 K	ADDR[0:17]
0100	32 K	ADDR[0:16]
0101	64 K	ADDR[0:15]
0110	128 K	ADDR[0:14]
0111	256 K	ADDR[0:13]
1000	512 K	ADDR[0:12]
1001	1 M	ADDR[0:11]
1010	2 M	ADDR[0:10]
1011	4 M	ADDR[0:9]
1100	8 M	ADDR[0:8]
1101	16 M	ADDR[0:7]
1110	32 M	ADDR[0:6]
1111	64 M	ADDR[0:5]

Table 46 Main Block and Sub-Block Pairings

Main Block	Sub-Block
$\overline{\text{CSBOOT}}$	$\overline{\text{CS1}}$
$\overline{\text{CS2}}$	$\overline{\text{CS3}}$
$\overline{\text{CS4}}$	$\overline{\text{CS5}}$

Table 47 Interface Types

ITYPE (Binary)	Interface Type
0000	Generic asynchronous region with output buffer turn-off time of less than or equal to one clock period. Cannot be pipelined.
0001	Generic asynchronous region with output buffer turn-off time of two clock periods. Cannot be pipelined.
0010	Synchronous region (no burst) with asynchronous $\overline{\text{OE}}$.
0011	Synchronous region (no burst) with synchronous $\overline{\text{OE}}$.
0100	Reserved.
0101	Region with fixed burst access capability (burst type 1 — uses $\overline{\text{BDIP}}$ timing) and asynchronous $\overline{\text{OE}}$.
0110	Reserved.
0111	Region with fixed burst access capability (burst type 1 — uses $\overline{\text{BDIP}}$ timing) and synchronous $\overline{\text{OE}}$.
1000	Region with fixed burst access capability (burst type 2 — uses $\overline{\text{LAST}}$ timing) but may not have $\overline{\text{OE}}$. Chip-select unit will not generate an $\overline{\text{OE}}$ signal for this interface type.
1001	Synchronous region (no burst) with synchronous $\overline{\text{OE}}$ (same as ITYPE 3) with early overlapping of accesses to the region.
1010–1111	Reserved.

5.5 System Protection

SIU system protection features include a periodic interrupt timer and a bus monitor.

Additional MPC505 system protection features include the PowerPC decrementer and time base, described in **3 Central Processing Unit**, and a software watchdog, described in **6 Peripheral Control Unit**.

5.5.1 System Protection Features

- The bus monitor monitors any internal-to-external bus accesses. Four selectable response time periods are available, ranging from 16 to 256 system clock cycles. An internal bus error signal is generated if a bus time-out occurs.
- The periodic interrupt timer generates an interrupt after a period specified by the user.

5.5.2 System Protection Registers

Table 48 shows the system protection registers.

Table 48 System Protection Address Map

Access	Address	Register
S/U	0x8007 FC40	Periodic Interrupt Control and Select Register (PICSR)
S/U	0x8007 FC44	Periodic Interrupt Timer Register (PIT)
S	0x8007 FC48	Bus Monitor Control Register (BMCR)

The reset status register is described in **5.7 Reset**. The remaining system protection registers are described in this section.

PICSR — Periodic Interrupt Control and Select Register **0x8007 FC40**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	PTE	PIE	RESERVED		PCFS			RESERVED							PS
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PITC															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49 PICSR Bit Settings

Bit(s)	Name	Description
0	—	Reserved
1	PTE	Periodic timer enable 0 = Disable decrementer counter 1 = Enable decrementer counter
2	PIE	Periodic interrupt enable 0 = Disable periodic interrupt 1 = Enable periodic interrupt
[3:4]	—	Reserved
[5:7]	PCFS	PIT clock frequency select. Refer to Table 50.
[8:14]	—	Reserved
15	PS	PIT status (sticky bit) 0 = No PIT interrupt asserted 1 = Periodic interrupt asserted

Table 49 PICSR Bit Settings (Continued)

Bit(s)	Name	Description
[16:31]	PITC	Periodic interrupt timing count. Number of counts to load into the PIT.

To achieve a PIT setting of approximately 1 MHz, assign the PCFS field the value indicated in Table 50.

Table 50 Recommended Settings for PCFS[0:2]

Input Frequency Range	PCFS[0:2]	Divide Input Frequency (EXTAL) by
1 MHz < FREQ ≤ 4 MHz	0b000	4
4 MHz < FREQ ≤ 8 MHz	0b001	8
8 MHz < FREQ ≤ 16 MHz	0b010	16
16 MHz < FREQ ≤ 32 MHz	0b011	32
32 MHz < FREQ ≤ 64 MHz	0b100	64
Reserved	0b101	NA
Reserved	0b110	NA
Reserved	0b111	NA

PIT — Periodic Interrupt Timer Register **0x8007 FC40**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED																PIT															

RESET: UNDEFINED

The periodic interrupt timer register is a read-only register that shows the current value in the periodic interrupt down counter. Writes to this register have no effect. Reads of the register do not affect the counter.

BMCR — Bus Monitor Control Register **0x8007 FC48**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED				BMLK	BME	BMT		RESERVED							
RESET:															
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The bus monitor control register enables or disables the bus monitor and selects its time-out period.

Table 51 BMCR Bit Settings

Bit(s)	Name	Description
[0:3]	—	Reserved
4	BMLK	Bus monitor lock 0 = Enable changes to BMLK, BME, BMT 1 = Ignore writes to BMLK, BME, BMT BMLK is a set-only bit; writing a zero to BMLK after it has been set has no effect.
5	BME	Bus monitor enable 0 = Disable bus monitor 1 = Enable bus monitor
[6:7]	BMT	Bus monitor timing. These bits select the time-out period, in system clocks, for the bus monitor. 00 = 256 system clocks 01 = 64 system clocks 10 = 32 system clocks 11 = 16 system clocks
[8:31]	—	Reserved

5.5.3 Hardware Bus Monitor

Typical bus systems require a bus monitor to detect excessively long data and address acknowledge response times. The MPC505 provides a bus monitor to monitor internal-to-external bus accesses on the E-bus. If the external bus pipeline depth is zero (all previous external bus cycles are complete), the monitor counts from transfer start to transfer acknowledge. Otherwise, the monitor counts from transfer acknowledge to transfer acknowledge. If the monitor times out, transfer error acknowledge ($\overline{\text{TEA}}$) is internally asserted.

Four selectable response time periods are available, ranging from 16 to 256 system clock cycles. The programmability of the time-out allows for a variation in system peripheral response time. The timing mechanism is derived from taps off a divider chain which is clocked by the system clock.

The bus monitor is always enabled while freeze is asserted and debug mode is enabled or while debug mode is enabled and the debug non-maskable breakpoint is asserted, even if the bus monitor enable (BME) bit in the bus monitor control register is cleared.

5.5.4 Periodic Interrupt Timer (PIT)

The periodic interrupt timer consists of a 16-bit counter clocked by the input clock signal divided by four. A 4-MHz system clock frequency results in a one-microsecond count interval. The input clock signal is supplied by the clock module. In order to ensure adequate range for the PIT, the input clock signal to the PIT must not exceed 4 MHz, even if a full frequency input is used instead of an oscillator-PLL clocking scheme.

The 16-bit counter counts down to zero when loaded with a value from the PITC. After the timer reaches zero, the PS bit is set and an interrupt is generated if the PIE bit is set to one. The software service routine should read the PS bit and then write it to zero to terminate the interrupt request. At the next input clock edge, the value in the PITC is loaded into the counter, and the process starts over again.

5.6 Clock Submodule

The clock submodule consists of the crystal oscillator (OSC), a phase-locked loop (PLL), the engineering clock reference output signal (ECROUT), the reduced frequency divider (RFD), the clock generator blocks, the system clock control register (SCCR), and the system clock control and status register (SCCSR). The clock submodule also provides a clock source for the PowerPC time base and decrementer.

The recommended timing reference for the MPC505 is a 4-MHz crystal. The system operating frequency is generated through a programmable phase-locked loop. The PLL is programmable in integer multiples of 4 MHz to generate operating frequencies of 16 MHz to 44 MHz.

If the crystal ceases to function, the loss of oscillator (LOO) bit is set and the PLL is forced to operate in the self-clocked mode (SCM). This mode provides a system clock frequency of approximately 4 MHz. The exact frequency depends on the voltage and temperature of the chip.

The PLL can be bypassed by grounding the V_{DDSN} pin. Note that in this case, the input frequency needs to be twice the desired operating system frequency. With V_{DDSN} grounded, the multiplication factor (MF) bits no longer have any effect on the system frequency, but the reduced frequency (RFD) bits and the low power mode (LPM) bits do have an effect.

ECROUT is an output signal with a frequency equal to the reference frequency divided by four (1 MHz with a 4-MHz crystal). It is provided as a timing reference for external devices.

Three different low-power modes are available to minimize standby power usage. Normal operation or one of the three low-power modes is selected by programming the LPM bits in the SCCR.

5.6.1 Clock Submodule Signal Descriptions

Table 52 describes the signals used by the clock module.

Table 52 Clock Module Signal Descriptions

Mnemonic	Name	Direction	Description
CLKOUT	System clock out	Output	System clock output. Used as the bus timing reference by external devices.
EXTAL, XTAL	Crystal oscillator	Input, Output	Connections for external crystal to the internal oscillator circuit. An external oscillator should serve as input to the EXTAL pin, when used.
XFCN, XFCP	External filter capacitor	Input	These pins are used to add an external capacitor to the filter circuit of the phase-locked loop.
MODCLK	Clock mode select	Input	The state of this input signals during reset selects the source of the system clock. Refer to **5.6.3 System Clock Sources.**
V_{DDSN}, V_{SSSN}	Synthesizer power	Input	These pins supply a quiet power source to the VCO.
ECROUT	Engineering clock reference output	Output	Buffered output of the crystal oscillator. The ECROUT output frequency is equal to the crystal oscillator frequency divided by four.
PLLL/DSDO	PLL lock status or debug output	Output	Phase-locked loop status output or debug output.

Table 52 Clock Module Signal Descriptions

Mnemonic	Name	Direction	Description
PDWU	Power-down wakeup	Output	Asserted or negated, respectively, by software setting or clearing the WUR bit in the SCLSR. Also asserted when decrementer counts down to zero. Can be used as power-down wakeup to external power-on reset circuit.

5.6.2 Clock Power Supplies

The power supply for each block of the clock submodule is shown in Table 53.

Table 53 Clock Module Power Supplies

Power Supply	Blocks
V_{DDI}	CLKOUT ECR PIT Clock RFD PLL (Digital)
VDDKAP1	Decrementer/Time Base Clock Oscillator SCCR SCCSR
V_{DDSN}	PLL (Analog)

To improve noise immunity, the PLL has its own set of power supply pins, V_{DDSYN} and V_{SSSN}. Only the charge pump and the VCO are powered by these pins.

The oscillator, system clock control register, and system clock control and status register are powered from the keep alive power supply (VDDKAP1). In addition, VDDKAP1 powers the time base and decrementer. This allows the time base to continue incrementing even when the main power to the MCU is off. While the main power is off, the decrementer also continues to count and may be used to signal the external power supply to enable power to the system at specific intervals. This is the power down wakeup feature.

5.6.3 System Clock Sources

The V_{DDSN} and MODCLK pins are used to configure the clock source for the MCU. The configuration modes are shown in Table 54. When both pins are high, the CPU clocks are configured for normal operation and the SPLL is fully programmable. If MODCLK = 0 and V_{DDSN} = 1, then the PLL enters 1:1 frequency mode (i.e. $freq_{clkout(max)} = freq_{osc}$, where osc is the oscillator driven by either an external crystal or an external clock source). If V_{DDSN} = 0 and MODCLK = 1, then the PLL is disabled and bypassed. In this case, $freq_{clkout(max)} = freq_{osc}/2$. However, the oscillator source must have a 50% duty cycle for both 1:1 and bypass modes. In each of these three cases, $freq_{clkout}$ can be reduced by the RFD[0:2] bits. Note that $freq_{clkout(max)}$ corresponds to the RFD bits being cleared to zero.

If V_{DDSN} = 0 and MODCLK = 0, then the CPU clocks are configured in special test mode. In this mode, the PLL and most of the clock generation circuitry are bypassed. This mode is provided for factory test only.

Table 54 System Clock Sources

VDDSN	MODCLK	SPLL Options
1	1	Normal Operation
1	0	1:1 Mode ($freq_{clkout(max)}$= $freq_{osc}$)
0	1	SPLL Bypass Mode ($freq_{clkout(max)}$ = $freq_{osc}$/2)
0	0	Special Test Mode

5.6.4 Low-Power Modes

The clock module provides one normal operating mode and three low-power modes. The low-power mode (LPM) bits in the SCCR select one of these four modes. When one of the three low-power modes is selected, the EBI prevents the CPU from starting any more bus cycles, but allows the current bus cycle to terminate. At the end of the current bus cycle, the appropriate clocks are stopped and the EBI continues operation as defined for the low power mode selected.

The normal operating mode, state 0x0, is the state out of clock reset. This is also the state the bits go to when the low power mode exit signal arrives.

Mode 1 is single-chip mode. In this mode, CLKOUT is turned off. This mode can be selected when the MCU is used by itself and does not need to provide a system clock.

Mode 2 is doze mode. In this state, not only is CLKOUT turned off, but also all internal clocks are turned off. However, the oscillator and the PLL continue to operate normally.

Mode 3 is sleep mode. In this state, all clocks are turned off, including the oscillator and the PLL.

Table 55 summarizes the events that cause the MCU to exit from each of the three low-power modes.

Table 55 Exiting Low-Power Mode

Event Causing Exit from Low-Power Mode	LPM 01	LPM 10	LPM 11
$\overline{\text{RESETOUT}}$ assertion	Yes	Yes	Yes
$\overline{\text{IRQ}}$ pin assertion (if LPMM=1)	Yes	Yes	Yes
Decrementer interrupt	Yes	Yes	No
PIT Interrupt	Yes	Yes	No

5.6.5 System Clock Control Register (SCCR)

The SCCR controls the operation of the SPLL. It is powered by VDDKAP1.

SCCR — System Clock Control Register **0x8007 FC50**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED			LPMM	TBS	DCE	LOLRE	LOORE	RES'D	MF				RESERVED		
RESET:															
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED						LPM		RESERVED				RFD			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Table 56 SCCR Bit Settings

Bit(s)	Name	Description
[0:2]	—	Reserved
3	LPMM	Low power mode mask 0 = IRQ[0:1] pins cannot be used to wake up from LPM 1 = IRQ[0:1] pins can be used to wake up from LPM
4	TBS	Time base/decrementer source 0 = Source is crystal oscillator ÷ 4 1 = Source is system clock ÷ 4
5	DCE	Decrementer clock enable 0 = Clock to decrementer is disabled 1 = Clock to decrementer is enabled
6	LOLRE	Loss-of-lock reset enable 0 = Loss of lock does not cause reset 1 = Loss of lock causes reset
7	LOORE	Loss-of-oscillator reset enable 0 = Loss of oscillator does not cause reset 1 = Loss of oscillator causes reset
8	—	Reserved
[9:12]	MF	Multiplication factor. The output of the VCO is divided down to generate the feedback signal to the phase comparator. The MF field controls the value of the divider in the PLL feedback loop. The MF and RFD fields determine the CLKOUT frequency. Refer to Table 57. X000 = x 4 X001 = x 5 X010 = x 6 X011 = x 7 X100 = x 8 X101 = x 9 X110 = x 10 X111 = x 11
[13:21]	—	Reserved

Table 56 SCCR Bit Settings (Continued)

Bit(s)	Name	Description
[22:23]	LPM	Low-power mode select bits. Refer to **5.6.4 Low-Power Modes**. 00 = Normal operating mode 01 = Low-power mode 1 (single chip) 10 = Low-power mode 2 (doze) 11 = Low-power mode 3 (sleep) Since all clocks are stopped in sleep mode, exiting this mode requires the normal crystal start-up time plus the PLL lock time. Minimum length of the exit signal is two clocks in single-chip mode, three clocks in doze mode, and until the PLL is stable in sleep mode.
[24:27]	—	Reserved
[28:31]	RFD	Reduced-frequency divider. The RFD field controls a prescaler at the output of the PLL. The MF and RFD fields determine the CLKOUT frequency. Refer to Table 57. 0000 = ÷ 1 0001 = ÷ 2 0010 = ÷ 4 0011 = ÷ 8 0100 = ÷ 16 0101 = ÷ 32 0110 = ÷ 64 0111 = ÷ 128 1000 = ÷ 256 1001 = ÷ 512 1010 = ÷ 1024 1011 = ÷ 1024 1100 = ÷ 1024 1101 = ÷ 1024 1110 = ÷ 1024 1111 = ÷ 1024

Table 57 CLKOUT Frequencies with a 4-MHz Crystal[1]

RFD[0:3]	CLKOUT (Hz)							
	MF = X000 (x4)	MF = X001 (x5)	MF = X010 (x6)	MF = X011 (x7)	MF = X100 (x8)	MF = X101 (x9)	MF = X110 (x10)	MF = X111 (x11)
0 = 0000 (÷ 1)	16.000 M	20.000 M	24.000M	28.000 M	32.000 M	36.000 M	40.000 M	44.000 M
1 = 0001 (÷ 2)	8.000 M	10.000 M	12.000 M	14.000 M	16.000 M	18.000 M	20.000 M	22.000 M
2 = 0010 (÷ 4)	4.000 M	5.000 M	6.000 M	7.000 M	8.000 M	9.000 M	10.000 M	11.000 M
3 = 0011 (÷ 8)	2.000 M	2.500 M	3.000 M[2]	3.500 M	4.000 M	4.500 M	5.000 M	5.500 M
4 =0100 (÷ 16)	1.000 M	1.250 M	1.500 M	1.750 M	2.000 M	2.250 M	2.500 M	2.750 M
5 = 0101 (÷ 32)	0.500 M	0.625 M	0.750 M	0.875 M	1.000 M	1.125 M	1.250 M	1.3750 M
6 = 0110 (÷ 64)	0.250 M	0.313 M	0.375 M	0.438 M	0.500 M	0.563 M	0.625 M	0.688 M
7 = 0111 (÷ 128)	0.125 M	0.156 M	0.188 M	0.219 M	0.250M	0.281 M	0.313 M	0.344 M
8 = 1000 (÷ 256)	62.500 K	78.125 K	93.750 K	0.109 M	0.125 M	0.141 M	0.156 M	0.172 M
9 = 1001 (÷ 512)	31.250 K	39.063 K	46.875 K	54.688 K	62.500 K	70.313 K	78.125 K	85.938 K
10 =1010 (÷ 1024)	15.625 K	19.531 K	23.438 K	27.344 K	31.250 K	35.156 K	39.063 K	42.969 K
11 =1010 (÷ 1024)	15.625 K	19.531 K	23.438 K	27.344 K	31.250 K	35.156 K	39.063 K	42.969 K
12 =1010 (÷ 1024)	15.625 K	19.531 K	23.438 K	27.344 K	31.250 K	35.156 K	39.063 K	42.969 K
13 =1010 (÷ 1024)	15.625 K	19.531 K	23.438 K	27.344 K	31.250 K	35.156 K	39.063 K	42.969 K
14 =1010 (÷ 1024)	15.625 K	19.531 K	23.438 K	27.344 K	31.250 K	35.156 K	39.063 K	42.969 K
15 =1010 (÷ 1024)	15.625 K	19.531 K	23.438 K	27.344 K	31.250 K	35.156 K	39.063 K	42.969 K

1. Settings resulting in CLKOUT frequencies in the shaded areas should not be used in the MPC505.
2. Default setting.

5.6.6 System Clock Lock and Status Register (SCLSR)

The SCLSR contains control and status information for the PLL.

SCLSR — System Clock Lock and Status Register **0x8007 FC54**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED				STME	MPL	LPML	RFDL	RESERVED						STMS	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED							WUR	RESERVED					SPLSS	SPLS	LOO
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	U	U	U

U = Unaffected by reset

Table 58 SCLSR Bit Settings

Bit(s)	Name	Description
[0:3]	—	Reserved
4	STME	System PLL test mode enable. Used for factory testing only. 0 = Test mode disabled 1 = Test mode enabled
5	MPL	MF lock (set-once bit) 0 = Writes to MF field (in SCCR) allowed 1 = Writes to MF field have no effect
6	LPML	Low-power mode lock (set-once bit) 0 = Writes to LPM bits allowed 1 = Writes to LPM bits have no effect
7	RFDL	Reduced-frequency divide lock 0 = Writes to RF bits allowed 1 = Writes to RF bits have no effect
[8:13]	—	Reserved
[14:15]	STMS	System PLL test mode select. Used for factory testing only. 00 = Normal operation 00, 01, 11 = Special test modes (for factory test only)
[16:22]	—	Reserved
23	WUR	Wake-up request 0 = PDWU pin forced low (request power off) 1 = PDWU pin forced high (request power on)
[24:28]	—	Reserved
29	SPLSS	System PLL lock status sticky bit 0 = PLL has gone out of lock since software last set this bit 1 = PLL has remained in lock since software last set this bit
30	SPLS	System PLL lock status 0 = PLL has not locked 1 = PLL has locked
31	LOO	Loss-of-oscillator status 0 = Clock detected 1 = Crystal not detected

5.7 Reset

Reset procedures handle system initialization and recovery from catastrophic failure. The MPC505 performs reset with a combination of hardware and software. The SIU determines whether a reset is valid, asserts control signals, performs basic system configuration based on hardware mode-select inputs, and then passes control to the CPU.

Reset is the highest-priority CPU exception. Any processing in progress is aborted by the reset exception and cannot be restarted. Only essential tasks are performed during reset exception processing. Other initialization tasks must be accomplished by the exception handler routine.

5.7.1 Reset Sources

The following sources can cause reset:

- External reset pin ($\overline{\text{RESET}}$)
- Loss of oscillator
- Loss of PLL lock
- Software watchdog reset
- Checkstop reset
- JTAG reset (external TRST pin)

All of these reset sources are fed into the reset controller. The reset status register (RSR) reflects the most recent source, or sources, of reset. (Simultaneous reset requests can cause more than one bit to be set at the same time.) This register contains one bit for each reset source. A bit set to logic one indicates the type of reset that last occurred.

Individual bits in the RSR can be cleared by writing them as zeros after reading them as ones. (Writing individual bits as ones has no effect.) The register can be read at all times. Assertion of the $\overline{\text{RESET}}$ pin clears all bits except the RESET bit.

RSR — Reset Status Register **0x8007 FC4C**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESET	LOO	LOL	SW	CR	JTAG	RESERVED									

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED															

Table 59 Reset Status Register Bit Settings

Bit(s)	Name	Description
0	RESET	If set, source of reset is the external $\overline{\text{RESET}}$ input pin. This pin should be asserted whenever VDD is below VDD_{min}.
1	LOO	If set, source of reset is a loss of oscillator. The clock module asserts loss-of-oscillator reset when the MCU is in low power mode 3 or no clock signal is present on the EXTAL pin. If the clock module detects a loss-of-oscillator condition, erroneous external bus operation will occur if synchronous external devices use the MCU input clock. Erroneous operation can also occur if devices with a PLL use the MCU CLKOUT signal. This source of reset is masked by the loss-of-oscillator reset enable (LOORE) bit in the system clock control register (SCCR).

Table 59 Reset Status Register Bit Settings (Continued)

Bit(s)	Name	Description
2	LOL	If set, the cause of reset is the loss of PLL lock. The clock module asserts loss-of-lock reset when the PLL detects a loss of lock and the loss-of-lock reset enable bit is set in the system clock control register (SCCR). If the PLL detects a loss of lock condition, erroneous external bus operation will occur if synchronous external devices use the MCU input clock. Erroneous operation can also occur if devices with a PLL use the MCU CLKOUT signal. This source of reset is masked by the loss-of-lock reset enable (LOLRE) bit in the system clock control register.
3	SW	If set, source of reset is a software watchdog time-out. This occurs when the software watchdog counter reaches zero.
4	CR	If set, the source of reset is a checkstop. This occurs when the processor enters the checkstop state and the checkstop reset is enabled.
5	JTAG	If set, the source of reset is the JTAG module (external TRST pin asserted). This reset occurs only during production testing.
[6:31]	—	Reserved

5.7.2 Power-On Reset

The MPC505 does not have a power-on reset circuit. This function must be provided externally.

5.7.3 Reset Flow

The reset pin is synchronized to the system clock (CLKOUT). If the system clock is not running (during low-power stop), reset is synchronized to the crystal. This synchronized reset input causes the MCU to be placed in reset. While the chip is in reset, the $\overline{\text{RESETOUT}}$ output signal is asserted. $\overline{\text{RESETOUT}}$ is released after the VCO is locked, at a minimum of 15 clock cycles (for internal reset sources) or 17 clock cycles (for external reset sources) after the reset source is detected as negated.

During this time, activity on the internal buses is halted. Prior to the release of $\overline{\text{RESETOUT}}$, any mode-select pins are sampled (if necessary). Internal reset is released at the same time as $\overline{\text{RESETOUT}}$; however, internal buses are not released until 15 clock cycles (for internal reset sources) or 17 clock cycles (for external reset sources) after $\overline{\text{RESETOUT}}$ is detected as negated. After that time, instruction fetches may occur.

5.7.4 Configuration During Reset

Many SIU pins can have more than one function. The logic state of certain mode-select pins during reset determines which functions are assigned to certain pins with multiple functions. These mode-select pins determine other aspects of operating configuration as well.

Basic operating configuration is determined by the DSDI pin, as shown in Table 60.

Table 60 Pin Configuration During Reset

Pin During Reset	Function Affected
DSDI Asserted (1)	Data Bus Configuration Mode
DSDI Negated (0)	Internal Default Mode

Refer to **8.6 Development Port and Debug Mode Configuration** for information on setting debug options during reset.

Data bus configuration mode is used if DSDI is asserted (high) during the assertion of $\overline{\text{RESETOUT}}$. In this case, chip configuration is determined by the levels driven on the external data bus (DATA[0:5] at a minimum) while $\overline{\text{RESETOUT}}$ is asserted.

This scheme allows users of the internal default mode to limit their required external configuration hardware to two pulldown resistors. It also allows many options to be configured with a single three-state octal buffer.

5.7.4.1 Data Bus Configuration Mode

If data bus configuration mode is selected (DSDI is asserted), then the part is configured according to the values latched off of the data bus pins DATA[0:31]. The external data bus is divided into four groups:

- DSDI, DATA[0:5]
- DATA[6:13]
- DATA[14:21]
- DATA[22:31]

This grouping allows the user to use three-state octal buffers to only drive valid data on the pins for those reset configuration options that the user would want to change. The state of the last pin in each group (pins 5, 13, and 21) determines whether the following reset configuration options use the internal default values or are configured from the data bus. The user is required to drive to a valid level all the pins in any of the groups that are to be changed. The functions selected by these pins are shown in Table 61.

5.7.4.2 Internal Default Mode

The internal default mode allows MCUs with on-board non-volatile memory modules to provide a pin configuration word on the I-bus or L-bus data bus during reset. For MCUs without such a memory module, such as the MPC505, the SIU provides a mask-programmed default value.

5.7.4.3 Data Bus Reset Configuration Word

In either reset configuration mode (data bus configuration mode or internal default mode), the configuration is accomplished within the MCU by driving a configuration word on the internal data bus before the internal $\overline{\text{RESET}}$ signal (reflected on $\overline{\text{RESETOUT}}$) is negated. When the internal $\overline{\text{RESET}}$ signal is negated, those functions that are configured at reset latch their configuration values from the assigned bits of the internal data bus. The format of the data bus reset configuration word is the same regardless of which configuration mode is selected, except that data bus bits 5, 13, and 21 have no meaning in internal default mode. Table 61 describes the configuration options. The column labeled "Internal Default Mode" shows the default reset configuration word if no word is provided by an internal non-volatile memory.

Table 61 Data Bus Reset Configuration Word

Data Bus Bit	Configuration Function Effected	Effect of Mode Select = 1 During Reset	Effect of Mode Select = 0 During Reset	Internal Default Mode
0	Address Bus	Minimum Bus Mode ADDR[0:11]/$\overline{\text{CS}}$[0:11] configured as chip selects	Maximum Bus Mode ADDR[0:11]/$\overline{\text{CS}}$[0:11] configured as address pins	1
1	Vector Table Location (IP Bit)	Vector Table 0xFFF0 0000	Vector Table 0x0000 0000	1
2	Burst Type/Indication	Type 2 ($\overline{\text{LAST}}$ Timing)	Type 1 ($\overline{\text{BDIP}}$ Timing)	0

Table 61 Data Bus Reset Configuration Word (Continued)

Data Bus Bit	Configuration Function Effected	Effect of Mode Select = 1 During Reset	Effect of Mode Select = 0 During Reset	Internal Default Mode
3	Interface Type for $\overline{\text{CSBOOT}}$	ITYPE = 001 Asynchronous (Time to Hi-Z = 2Clk)	ITYPE = 1000 Synchronous Burst	1
4	$\overline{\text{CSBOOT}}$ Port Size	32-Bit	16-Bit	0
5	Reset Configuration Source For DATA[6:13]	Latch Configuration from external pins.	Latch Configuration from internal defaults.	0
[6:8]	$\overline{\text{TA}}$ Delay For $\overline{\text{CSBOOT}}$	$\overline{\text{TA}}$ Delay Encoding 000 001 010 011 100 101 110 111	Number of Wait States 0 1 2 3 4 5 6 7	111
[9:10]	IMEMBASE[0:1]	IMEMBASE 00 01 10 11	I-Mem Block Placement Start Addr: 0x0000 0000 End Addr: 0x000F FFFF Start Addr: 0xFFF0 0000 End Addr: 0xFFFF FFFF Note: MPC505 does not contain I-Mem	01
[11:12]	LMEMBASE[0:1]	LMEMBASE 00 01 10 11	L-Mem (SRAM) Block Placement Start address: 0x0000 0000 End address: 0x000F FFFF Start address: 0xFFF0 0000 End address: 0xFFFF FFFF	11
13	Reset configuration source for DATA[14:21]	Latch configuration from external pins	Latch configuration from internal defaults.	0
14	CT[0:3], AT[0:1], $\overline{\text{TS}}$	CT[0:3], AT[0:1], $\overline{\text{TS}}$	PJ[1:7]	1
15	$\overline{\text{WR}}$, $\overline{\text{BDIP}}$	$\overline{\text{WR}}$, $\overline{\text{BDIP}}$	PK[0:1]	1
16	PLLL/DSDO, VF[0:2], VFLS[0:1], WP[1:5]	DSDO, Pipe Tracking, Watchpoints	PK[2:7], PL[2:7]	1
17	$\overline{\text{BURST}}$, $\overline{\text{TEA}}$, $\overline{\text{AACK}}$, $\overline{\text{TA}}$, $\overline{\text{BE}}$[0:3]	Handshake Pins	PORTI[0:7]	1
18	$\overline{\text{CR}}$, $\overline{\text{BI}}$, $\overline{\text{BR}}$, $\overline{\text{BB}}$, $\overline{\text{BG}}$, $\overline{\text{ARETRY}}$	Bus Arbitration Pins	PM[2:7]	1
19	Release reset when PLL locked (refer to Table 57)	Release internal reset 15 clock cycles (for internal reset sources) or 17 clock cycles (for external reset sources) after $\overline{\text{RESET}}$ negated (except during 1:1 mode)	Release internal reset when PLL locked and 15 clocks (for internal reset sources) or 17 clocks (for external reset sources) after $\overline{\text{RESET}}$ negated OR when time-out value in the timebase register has expired, whichever occurs first (during normal mode only)	1

Table 61 Data Bus Reset Configuration Word (Continued)

Data Bus Bit	Configuration Function Effected	Effect of Mode Select = 1 During Reset	Effect of Mode Select = 0 During Reset	Internal Default Mode
20	Reserved			0
21	Reset configuration source For DATA[22:31]	Latch configuration from external pins.	Latch configuration from internal defaults.	0
22	Reserved			0
23	Reserved			0
24	LEN	L-bus Memory modules are enabled.	L-bus Memory modules are disabled and emulated externally.	1
25	PRUMODE	Forces accesses to Ports A, B, I, J, K, and L to go external.	No effect	0
26	ADDR[12:15]	ADDR[12:15]	PB[4:7]	1
27	Reserved			0
28	Reserved			0
29	Reserved			0
30	Test Slave Mode Enable	Test Slave Mode Disabled	Test Slave Mode Enabled	1
31	Test Transparent Mode Enable	Test Transparent Mode Disabled	Test Transparent Mode Enabled	1

Table 62 Reset Behavior for Different Clock Modes

Clock Mode	V_{DDSN}	MODCLK	Internal DATA19 = 1 at Reset	Internal DATA19 = 0 at Reset
Normal operation	1	1	Release internal reset 15 (for an internal reset source) or 17 (for an external reset) clocks after $\overline{\text{RESET}}$ is negated	Release internal reset when PLL is locked and 15 (for an internal reset source) or 17 (for an external reset) clocks after $\overline{\text{RESET}}$ is negated OR when timeout value in the time base register has expired (whichever occurs first)
1:1 mode	1	0	Release internal reset when PLL is locked and 15 (for an internal reset source) or 17 (for an external reset) clocks after $\overline{\text{RESET}}$ is negated	
SPLL bypass mode	0	1	Release internal reset 15 (for an internal reset source) or 17 (for an external reset) clocks after $\overline{\text{RESET}}$ is negated	
Special test mode	0	0	Release internal reset 15 (for an internal reset source) or 17 (for an external reset) clocks after $\overline{\text{RESET}}$ is negated	

5.8 General-Purpose I/O

Many of the pins associated with the SIU can be used for more than one function. The primary function of these pins is to provide an external bus interface. When not used for their primary function, many of these pins can be used as digital I/O pins.

SIU digital I/O pins are grouped into eight-bit ports. The following registers are associated with each I/O port. (Output-only ports do not have a data direction register.)

- Pin assignment register — allows the user to configure a pin for its primary function or digital I/O.
- Data direction register — configures individual pins as input or output pins.
- Data register — monitors or controls the state of its pins, depending on the state of the data direction register for that pin.

If a pin is not configured as an I/O port pin in the pin assignment register, the data direction and data registers have no effect on the pin.

Ports A through L can be used with a port replacement unit (PRU). These ports provide three-clock-cycle access. If PRU mode is enabled at reset, access to these registers is disabled, and an external bus cycle is initiated. Other (non-PRU) ports provide two-clock-cycle access.

In addition to the SIU ports described in this section, port Q in the peripherals controller unit provides edge- or level-sensitive I/O. Refer to **6.5 Port Q** for information on port Q.

Table 63 is an address map of the SIU port registers.

Table 63 SIU Port Registers Address Map

Access	Address	Register
S	0x8007 FC60	PORT M DATA DIRECTION (DDRM)
S	0x8007 FC64	PORT M PIN ASSIGNMENT (PMPAR)
S/U	0x8007 FC68	PORT M DATA (PORTM)
—	0x8007 FC6C– 0x8007 FC80	RESERVED
S	0x8007 FC84	PORT A, B PIN ASSIGNMENT (PAPAR, PBPAR)
S/U	0x8007 FC88	PORT A, B DATA (PORTA, PORTB)
—	0x8007 FC8C– 0x8007 FC94	RESERVED
S	0x8007 FC98	PORT I, J, K, L DATA DIRECTION (DDRI, DDRJ, DDRK, DDRL)
S	0x8007 FC9C	PORT I, J, K, L PIN ASSIGNMENT (PIPAR, PJPAR, PKPAR, PLPAR)
S/U	0x8007 FCA0	PORT I, J, K, L DATA (PORTI, PORTJ, PORTK, PORTL)
—	0x8007 FCA4– 0x8007 FCFF	RESERVED

5.8.1 Port M

PORTM — Port M Data Register **0x8007 FC68**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	PM3	PM4	PM5	PM6	PM7	RESERVED							
RESET:															
0	0	0	U	U	U	U	U	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Writes to PORTM are stored in internal data latches. If any bit of the port is configured as an output, the value latched for that bit is driven onto the pin. A read of PORTM returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read will be the value stored in the internal data latch. PORTM can be read or written at any time. This register is unaffected by reset.

DDRM — Port M Data Direction Register **0x8007 FC60**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	DDM3	DDM4	DDM5	DDM6	DDM7	RESERVED							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The bits in this register control the direction of the port M pin drivers when the pins are configured as I/O pins. Setting a bit in this register to one configures the corresponding pin as an output; clearing the bit configures the pin as an input.

PMPAR — Port M Pin Assignment Register **0x8007 FC64**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	PMPA2	PMPA3	PMPA4	PMPA5	PMPA6	PMPA7	RESERVED							
RESET:															
0	0	*	*	*	*	*	*	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Reset setting depends on the value of the configuration word at reset.

The bits in this register control the function of the associated pins. Setting a bit in this register to one configures the corresponding pin as a bus control signal; clearing a bit configures the pin as an I/O pin (or as the DS signal, in the case of PMPA2).

Table 64 Port M Pin Assignments

PMPAR Bit	Port M Signal	Bus Control Signal
PMPA2	DS	CR
PMPA3	PM3	BI
PMPA4	PM4	BR
PMPA5	PM5	BB
PMPA6	PM6	BG
PMPA7	PM7	ARETRY

5.8.2 Ports A and B

Ports A and B are 8-bit output ports. Associated with each port is a data register and a pin assignment register; data direction registers are not needed.

PORTA, PORTB — Port A, B Data Registers **0x8007 FC88**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PA0	PA1	PA2	PA3	PA4	PA5	PA6	PA7	PB0	PB1	PB2	PB3	PB4	PB5	PB6	PB7
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

When a port A or port B pin is configured as a general-purpose output, the value in the port A or port B data register is driven onto the pin. PORTA and PORTB are unaffected by reset.

PAPAR, PBPAR — Port A, B Pin Assignment Register **0x8007 FC84**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PAPA0	PAPA1	PAPA2	PAPA3	PAPA4	PAPA5	PAPA6	PAPA7	PBPA0	PBPA1	PBPA2	PBPA3	PBPA4	PBPA5	PBPA6	PBPA7
RESET:															
1	1	1	1	1	1	1	1	1	1	1	1	*	*	*	*

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Reset setting depends on the value of the configuration word at reset.

Each bit in this register controls the function of the associated pin, provided the pin is configured for non-chip-select function in the corresponding chip-select options register. Setting a bit in the PAPAR or PBPAR configures the corresponding pin as an address bus pin; clearing the bit configures the pin as an I/O pin.

Table 65 Port A Pin Assignments

PMPAR Bit	Port A Signal	Bus Control Signal
PAPA0	PA0	ADDR0
PAPA1	PA1	ADDR1
PAPA2	PA2	ADDR2
PAPA3	PA3	ADDR3
PAPA4	PA4	ADDR4
PAPA5	PA5	ADDR5
PAPA6	PA6	ADDR6
PAPA7	PA7	ADDR7

Table 66 Port B Pin Assignments

PMPAR Bit	Port B Signal	Bus Control Signal
PBPA0	PB0	ADDR8
PBPA1	PB1	ADDR9
PBPA2	PB2	ADDR10
PBPA3	PB3	ADDR11
PBPA4	PB4	ADDR12

Table 66 Port B Pin Assignments (Continued)

PMPAR Bit	Port B Signal	Bus Control Signal
PBPA5	PB5	ADDR13
PBPA6	PB6	ADDR14
PBPA7	PB7	ADDR15

5.8.3 Ports I, J, K, and L

PORTI, PORTJ, PORTK, PORTL — Port I, J, K, L Data Registers **0x8007 FCA0**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PI0	PI1	PI2	PI3	PI4	PI5	PI6	PI7	0	PJ1	PJ2	PJ3	PJ4	PJ5	PJ6	PJ7
RESET:															
U	U	U	U	U	U	U	U	0	U	U	U	U	U	U	U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PK0	PK1	PK2	PK3	PK4	PK5	PK6	PK7	0	0	PL2	PL3	PL4	PL5	PL6	PL7
RESET:															
U	U	U	U	U	U	U	U	0	0	U	U	U	U	U	U

Writes to port I, J, K, and L data registers are stored in internal data latches. If any pin in one of these ports is configured as an output, the value latched for the corresponding data register bit is driven onto the pin. A read of one of these data registers returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the internal data latch. Port I, J, K, and L data registers can be read or written at any time. These registers are unaffected by reset.

DDRI, DDRJ, DDRK, DDRL — Port I, J, K, L Data Direction Registers **0x8007 FC98**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DDI0	DDI1	DDI2	DDI3	DDI4	DDI5	DDI6	DDI7	0	DDJ1	DDJ2	DDJ3	DDJ4	DDJ5	DDJ6	DDJ7
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DDK0	DDK1	DDK2	DDK3	DDK4	DDK5	DDK6	DDK7	0	0	DDL2	DDL3	DDL4	DDL5	DDL6	DDL7
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The bits in these registers control the direction of the associated pin drivers when the pins are configured as I/O pins. Setting a bit in these registers configures the corresponding pin as an output; clearing the bit configures the pin as an input.

PIPAR, PJPAR, PKPAR, PLPAR — Port I, J, K, L Pin Assignment Registers **0x8007 FC9C**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PIPA0	PIPA1	PIPA2	PIPA3	PIPA4	PIPA5	PIPA6	PIPA7	0	PJPA1	PJPA2	PJPA3	PJPA4	PJPA5	PJPA6	PJPA7
RESET:															
*	*	*	*	*	*	*	*	0	*	*	*	*	*	*	*

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PKPA0	PKPA1	PKPA2	PKPA3	PKPA4	PKPA5	PKPA6	PKPA7	0	0	PLPA2	PLPA3	PLPA4	PLPA5	PLPA6	PLPA7
RESET:															
*	*	*	*	*	*	*	*	0	0	*	*	*	*	*	*

* Reset setting depends on the value of the configuration word at reset.

The bits in these registers control the function of the associated pins. Setting a bit configures the corresponding pin as a bus control signal; clearing the bit configures the pin as an I/O pin.

Table 67 Port I Pin Assignments

PIPAR Bit	Port I Signal	Bus Control Signal
PIPA0	PI0	$\overline{\text{BURST}}$
PIPA1	PI1	$\overline{\text{TEA}}$
PIPA2	PI2	$\overline{\text{AACK}}$
PIPA3	PI3	$\overline{\text{TA}}$
PIPA4	PI4	$\overline{\text{BE0}}$
PIPA5	PI5	$\overline{\text{BE1}}$
PIPA6	PI6	$\overline{\text{BE2}}$
PIPA7	PI7	$\overline{\text{BE3}}$

Table 68 Port J Pin Assignments

PJPAR Bit	Port J Signal	Bus Control Signal
PJPA1	PJ1	AT0
PJPA2	PJ2	AT1
PJPA3	PJ3	$\overline{\text{TS}}$
PJPA4	PJ4	CT0
PJPA5	PJ5	CT1
PJPA6	PJ6	CT2
PJPA7	PJ7	CT3

Table 69 Port K Pin Assignments

PKPAR Bit	Port K Signal	Bus Control Signal
PKPA0	PK0	$\overline{\text{BDIP}}$
PKPA1	PK1	$\overline{\text{WR}}$
PKPA2	PK2	PLLL/DSDO
PKPA3	PK3	VF0
PKPA4	PK4	VF1
PKPA5	PK5	VF2
PKPA6	PK6	VFLS0
PKPA7	PK7	VFLS1

Table 70 Port L Pin Assignments

PLPAR Bit	Port L Signal	Bus Control Signal
PLPA2	PL2	WP0
PLPA3	PL3	WP1
PLPA4	PL4	WP2
PLPA5	PL5	WP3
PLPA6	PL6	WP4
PLPA7	PL7	WP5

5.8.4 Port Replacement Unit (PRU) Mode

The entire external bus interface must be supported in order to build an emulator for an MCU. The SIU contains support for external port replacement logic which can be used to faithfully replicate on-chip ports externally. This PRU mode allows system development of a single-chip application in expanded mode. Access (including access time) to the port replacement logic can be made transparent to the application software.

In PRU mode, all data, data direction, and pin assignment registers for ports A, B, I, J, K, and L are mapped externally. The SIU does not respond to these accesses, allowing external logic, such as a PRU, to respond.

PRU mode is invoked by pulling DATA25 high during reset. Other pins should be configured as bus control pins.

6 PERIPHERAL CONTROL UNIT

The peripheral control unit (PCU) consists of the following submodules:

- Software watchdog — provides system protection.
- Interrupt controller — controls the interrupts that external peripherals and internal modules send to the CPU.
- Port Q — provides for digital I/O on pins that are not being used as interrupt inputs.
- Test submodule — allows factory testing of the MCU.
- L-bus/IMB2 interface (LIMB) — provides an interface between the load/store bus and the second generation intermodule bus (IMB2). The IMB2, which is comparable to the IMB on modular M68300- and M68HC16-family MCUs, connects on-chip peripherals to the processor via the LIMB.

6.1 PCU Address Map

Table 71 shows the address map for the PCU. An entry of "S" in the Access column indicates that the register is accessible in supervisor mode only. "S/U" indicates that the register can be programmed to the desired privilege level. "Test" indicates that the register is accessible in test mode only.

Table 71 PCU Address Map

Access	Address	Register	
S	0x8007 EF80	PERIPHERAL CONTROL UNIT MODULE CONFIGURATION REGISTER (PCUMCR)	
—	0x8007 EF84 – 0x8007EF9C	RESERVED	
S	0x8007 EFA0	PENDING INTERRUPT REQUEST REGISTER (IRQPEND)	
S	0x8007 EFA4	ENABLED ACTIVE INTERRUPT REQUEST REGISTER (IRQAND)	
S	0x8007 EFA8	INTERRUPT ENABLE REGISTER (IRQENABLE)	
S	0x8007 EFAC	PIT/PORT Q INTERRUPT LEVEL REGISTER (PITQIL)	
—	0x8007 EFB0 – 0x8007 EFBC	RESERVED	
S	0x8007 EFC0	SOFTWARE SERVICE REGISTER (SWSR)	RESERVED
S	0x8007 EFC4	SOFTWARE WATCHDOG CONTROL FIELD/TIMING COUNT (SWCR/SWTC)	
S/U	0x8007 EFC8	SOFTWARE WATCHDOG REGISTER	
—	0x8007 EFCC	RESERVED	
S/U	0x8007 EFD0	PORT Q EDGE DETECT/DATA (PQEDGDAT)	RESERVED
S	0x8007 EFD4	PORT Q PIN ASSIGNMENT REGISTER (PQPAR)	
—	0x8007 EFD8 – 0x8007 EFFC	RESERVED	

CAUTION

Avoid writing to reserved location 0x8007 EF98. Setting the high-order bit in this reserved register causes the MCU to enter test mode.

6.2 Peripheral Control Unit Configuration

PCUMCR — Peripheral Control Unit Module Configuration Register **0x8007 EF80**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STOP	IRQMUX		RESERVED					SUPV		RESERVED					
RESET:															
0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 72 PCUMCR Bit Settings

Bit(s)	Name	Description
0	STOP	Stop system clock to peripherals controller 0 = Enable system clock to IMB2 modules 1 = Disable system clock to IMB2 modules
[1:2]	IRQMUX	Interrupt request multiplexer control 00 = Disable multiplexing scheme (8 possible interrupt sources) 01 = 2-to-1 multiplexing (16 possible interrupt sources) 10 = 3-to-1 multiplexing (24 possible interrupt sources) 11 = 4-to-1 multiplexing (32 possible interrupt sources)
[3:7]	—	Reserved
[8:9]	SUPV	Supervisor access for PCU registers 00 = Supervisor/unrestricted registers respond to accesses in supervisor or user data space. 01 = Supervisor/unrestricted registers respond to accesses in supervisor space only. 10 = Supervisor/unrestricted registers respond to read accesses in either data space, but write accesses can only be performed in supervisor data space. 10 = Undefined
[10:31]	—	Reserved

6.3 Software Watchdog

The software watchdog monitors the software interfaces of the system and requires the software to take periodic action in order to ensure that the program is executing properly. To protect against software error, the following service must be executed on a regular basis:

1. Write 0x556C to the SWSR
2. Write 0xAA39 to the SWSR

This sequence clears the watchdog timer, and the timing process begins again. If this periodic servicing does not occur, the software watchdog issues a reset. If any value other than 0x556C or 0xAA39 is written to the SWSR, the entire sequence must start over.

SWSR — Software Watchdog Service Register **0x8007 EFC8**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SWSR																RESERVED															
RESET:																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A write of 0x556C followed by a write of 0xAA39 to the SWSR causes the software watchdog register to be reloaded from the SWTC field of the SWCR.

SWCR/SWTC — Software Watchdog Control Field/Timing Count **0x8007 EFC4**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	SWE	SWLK	SWTC							
RESET:															
0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SWTC															
RESET:															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The software watchdog control field consists of the software watchdog enable (SWE) and lock (SWLK) bits. Bits 8 through 31 of the SWCR/SWTC constitute the software watchdog timing count, a 24-bit register that controls the software watchdog time-out period.

Table 73 SWCR/SWTC Bit Settings

Bit(s)	Name	Description
[0:5]	—	Reserved
6	SWE	Software watchdog enable 0 = Disable watchdog counter 1 = Enable watchdog counter
7	SWLK	Software watchdog lock 0 = Enable changes to SWLK, SWE, SWTC 1 = Ignore writes to SWLK, SWE, SWTC
[8:31]	SWTC	Software watchdog timing count. This 24-bit register contains the count for the software watchdog timer, which counts at system clock frequency. If this register is loaded with zero, the maximum time-out is programmed.

SWR — Software Watchdog Register **0x8007 EFC8**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED								SWR																							
RESET:																															
X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The software watchdog register is a read-only register that shows the current value of the software watchdog down counter. Writes to this register have no effect.

6.4 Interrupt Controller

Interrupts provide a mechanism for asynchronous, real-time communication between I/O devices and the CPU. The MPC505 interrupt controller joins the simple interrupt structure of the CPU with the complex structure of interrupt sources in the system. The CPU has a single maskable external interrupt. A complete MPC505-based system can have multiple interrupting modules, each with multiple interrupt sources.

A block diagram of the MPC505 interrupt structure is shown in Figure 14.

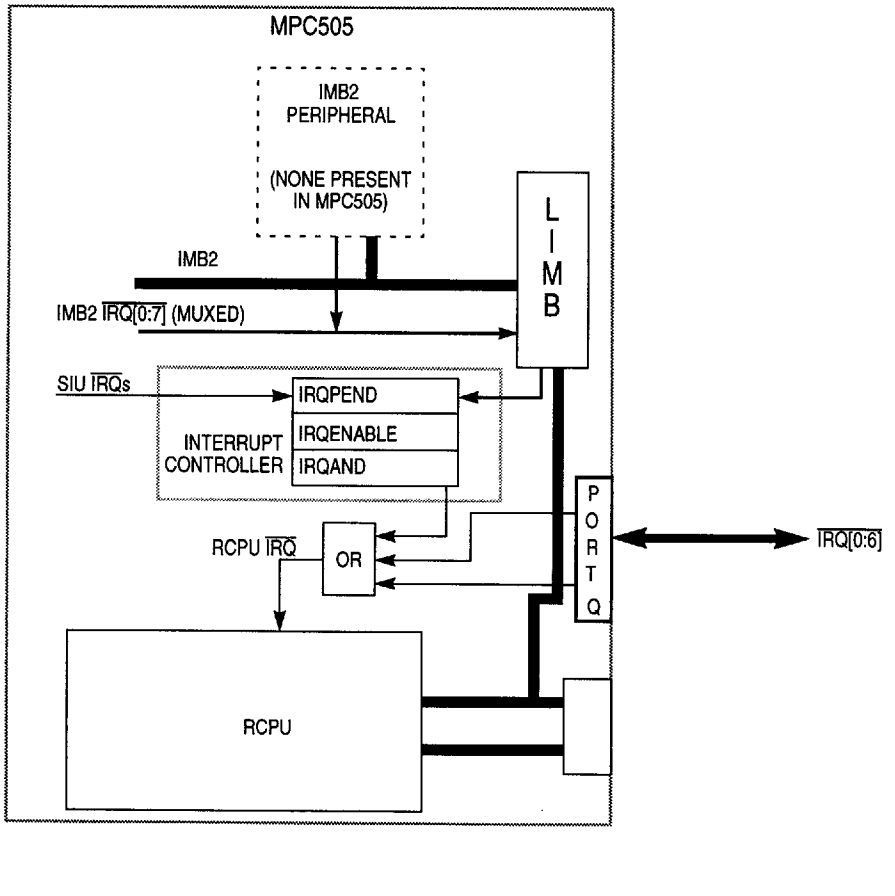


Figure 14 Interrupt Structure Block Diagram

The interrupt controller consolidates all the interrupt sources into a single interrupt signal to the processor. Since the MPC505 has no on-chip (IMB2) peripherals, the only interrupt sources are the periodic interrupt timer and external interrupt pins. Control and status registers associated with the interrupt controller indicate which of 32 possible interrupt levels are pending and control which interrupt sources are passed on to the CPU. Table 74 lists these registers.

Table 74 Interrupt Controller Registers

Name	Mnemonic	Description
Pending Interrupt Request Register	IRQPEND	Contains a status bit for each of the 32 interrupt levels. Each bit of IRQPEND is a read-only status bit that reflects the current state of the corresponding interrupt signal.
Interrupt Enable Register	IRQENABLE	Contains an enable bit for each of the 32 interrupt levels.
Enabled Active Interrupt Requests Register	IRQAND	Logical AND of the IRQPEND and IRQENABLE registers. This register reflects which levels are actually causing the $\overline{IRQ}$ input to the CPU to be asserted.
Interrupt Request Levels Register	PITQIL	Contains six 5-bit fields that determine the interrupt request levels of the PIT, the $\overline{IRQ}$[0:2] pins, and the $\overline{IRQ}$[0:1] signals from on-chip peripherals on the L-bus.

If a bit in the IRQPEND register is asserted and the corresponding bit in the IRQENABLE register is asserted, then the interrupt request line to the CPU will be asserted. In other words, if an interrupt request is pending on a certain level and that level is enabled in the IRQENABLE register, then the CPU $\overline{IRQ}$ line is asserted.

The interrupt controller does not enforce a priority scheme. All interrupt priority is determined by the software. In addition, the interrupt controller does not automatically update the interrupt mask upon entering or leaving interrupt processing. Any updates to the interrupt mask are the responsibility of software. In this way, the system is not limited to a particular interrupt priority updating scheme.

In addition to the interrupt controller on the MPC505, external peripheral chips in an MPC505-based system may contain their own interrupt controllers. Each interrupt input from an external peripheral chip to the MPC505 can be routed through the MPC505 interrupt controller, providing prioritization and enabling control on the MPC505 MCU, or can be routed directly to the CPU $\overline{IRQ}$ input. This allows the system interrupts to be structured in a cascade, where an interrupt controller on an external chip is read only if a certain interrupt on the CPU chip is serviced, or in parallel, where all interrupt controllers in the system are read and combined before it is determined which interrupt in the system needs servicing.

Any interrupts generated by an on-chip (IMB2) peripheral are passed down the IMB2 on eight time-multiplexed lines. This means that the IMB2 interrupts update the interrupt controller with a maximum latency of four clocks.

IRQPEND — Pending Interrupt Request Register **0x8007 EFA0**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
L16	L17	L18	L19	L20	L21	L22	L23	L24	L25	L26	L27	L28	L29	L30	L31
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The IRQPEND is a read-only status register that reflects the state of the 32 interrupt levels.

IRQAND — Enabled Active Interrupt Requests Register **0x8007 EFA4**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
L16	L17	L18	L19	L20	L21	L22	L23	L24	L25	L26	L27	L28	L29	L30	L31
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IRQAND is a read-only status register that is defined by the following equation:

IRQAND = IRQPEND & IRQENABLE

where & is a bitwise operation.

IRQENABLE — Interrupt Enable Register **0x8007 EFA8**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
L16	L17	L18	L19	L20	L21	L22	L23	L24	L25	L26	L27	L28	L29	L30	L31
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IRQENABLE is a read/write register. The bits in this register are affected only by writes from the CPU (or other bus master) and by reset.

PITQIL — PIT/Port Q Interrupt Levels Register **0x8007 EFAC**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	$\overline{IRQ0}$ LEVEL					$\overline{IRQ1}$ LEVEL					$\overline{IRQ2}$ LEVEL				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED											PIT $\overline{IRQ}$ LEVEL				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: bits 17 through 26 can be written to but have no effect.

This register is used to assign the interrupt request level of the periodic interrupt timer (PIT) and of the $\overline{IRQ}$[0:2] interrupt request pins. The remaining interrupt request pins have fixed values. $\overline{IRQ3}$ always generates a level 6 interrupt request; $\overline{IRQ4}$ generates a level 8 interrupt request; $\overline{IRQ5}$ generates a level 10 interrupt request; and $\overline{IRQ6}$ always generates a level 12 interrupt request.

6.5 Port Q

When not used as interrupt inputs, the $\overline{IRQ}$[0:6]/PQ[0:6] pins can be used for digital I/O. The following registers control port Q operation:

- Port Q Pin Assignment Register (PQPAR) — allows the user to configure each pin as a digital input, digital output, edge- or level-sensitive interrupt request to the CPU, or edge- or level-sensitive interrupt request to the interrupt controller.
- Port Q Edge Detect/Data Register (PQEDGDAT) — contains the following fields:
  - — Port Q Data Field (PQ[0:6]) — Monitors or controls the state of port Q pins, depending on the encoding for each pin in the PQPAR.
  - — Port Q Edge-Detect Status Field (PQE[0:6]) — Monitors when the proper transition occurs on a port Q or interrupt request pin.

PQEDGDAT — Port Q Edge Detect/Data Register **0x8007 EFD0**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PQE0	PQE1	PQE2	PQE3	PQE4	PQE5	PQE6	0	PQ0	PQ1	PQ2	PQ3	PQ4	PQ5	PQ6	0
RESET:															
U	U	0	0	0	0	0	0	U	U	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Port Q edge status (PQE) bits indicate when the proper transition has occurred on a port Q pin. Each pin can be configured as an interrupt input or as general-purpose I/O. If the pin is configured in the PQPAR as an edge-sensitive interrupt request pin, then the PQE bit acts as a status bit that

indicates whether the corresponding interrupt request line is asserted. The bit also acts as a status bit if the pins are configured as general-purpose inputs or outputs. When the pin is configured in edge-detect mode, the status bit is cleared by reading the bit as a one and then writing it to zero. In level-sensitive mode, the bit remains cleared.

A write to the port Q data register is stored in the internal data latch, and if any PQ bit is configured as an output, the value latched for that bit is driven onto the pin. A read of this port returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the internal data latch. The port Q data register can be read or written at any time.

PQPAR — Port Q Pin Assignment Register **0x8007 EFD4**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PQPA0		PQEDGE0		PQPA1		PQEDGE1		PQPA2		PQEDGE2		PQPA3		PQEDGE3	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PQPA4		PQEDGE4		PAPA5		PQEDGE5		PAPA6		PQEDGE6		RESERVED			
RESET:															
0	0	0	0	0	00	0	0	0	0	0	0	0	0	0	0

The PQPA fields select the basic function of each port Q pin, as shown in Table 75.

Table 75 Port Q Pin Assignments

PQPA Value	Pin Function	PORTQ (Port Q Data Field)	Interrupt Request Source
00	General-Purpose Input	A read returns the state of the pin. A write has no effect.	None
01	General-Purpose Output	A read returns the value in the latch. A write drives the value in the latch onto the pin.	None
10	$\overline{\text{IRQ}}$ to CPU	A read returns the state of the pin. A write has no effect.	From pin if PQEDG field is set to "Level," otherwise from port Q edge detect logic
11	$\overline{\text{IRQ}}$ to Interrupt Controller	A read returns the state of the pin. A write has no effect.	From pin if PQEDG field is set to "Level," otherwise from port Q edge detect logic

The PQEDGE fields select whether the port/interrupt pin is edge sensitive or level sensitive. When the selected transition occurs on a port Q pin, a corresponding status bit is set in the PQEDGDAT register.

Table 76 Port Q Edge Select Field Encoding

PQEDGE Value	Edge Select	PORTQE (Port Q Edge Detect Field)
00	Level sensitive	Returns zero when read
01	Falling-edge sensitive	Set on rising edge
10	Rising-edge sensitive	Set on falling edge
11	Either-edge sensitive	Set on either edge

7 STATIC RAM MODULE

The static RAM (SRAM) module consists of a 4-Kbyte block of static RAM. The primary function of this module is to serve as fast (one-cycle access), general-purpose RAM for the MCU. The SRAM can be read or written as either bytes, half-words or words.

The bus interface and control logic for the SRAM module are powered by V_{DD}. A separate pin, VDDKAP2, supplies power to the memory arrays. If main power is shut off, VDDKAP2 can be maintained in order to retain the data in the SRAM array. When the main power is off, access to the SRAM array is blocked.

7.1 Features

- Fast, One-Cycle Access
- Low-Power Mode
  - — Two-Cycle Access
  - — Pipelined for Back-to-Back Accesses
- Programmable Attributes (Supervisor Only, Data Only, Read Only)

7.2 Placement of SRAM in Memory Map

The SRAM module consists of two separately addressable sections. The first is the array itself. The second section is a set of registers used for configuration and testing of the SRAM array.

The SRAM array is assigned to one of four locations in the MCU address map by programming the LMEMBASE field in the SIU internal memory mapping register (MEMMAP). Table 77 indicates the location of the SRAM array for each value of LMEMBASE.

Table 77 MPC505 SRAM Module Addresses

LMEMBASE	SRAM Location
00	0x0000 0000 – 0x0000 0FFC
01	0x000F F000 – 0x000F FFFC
10	0xFFF0 0000 – 0xFFF0 0FFC
11	0xFFFF F000 – 0xFFFF FFFC

Refer to **5.2 SIU Module Configuration** for a diagram of the MEMMAP register.

Figure 15, a memory map of the MPC505, shows the possible locations of the SRAM array.

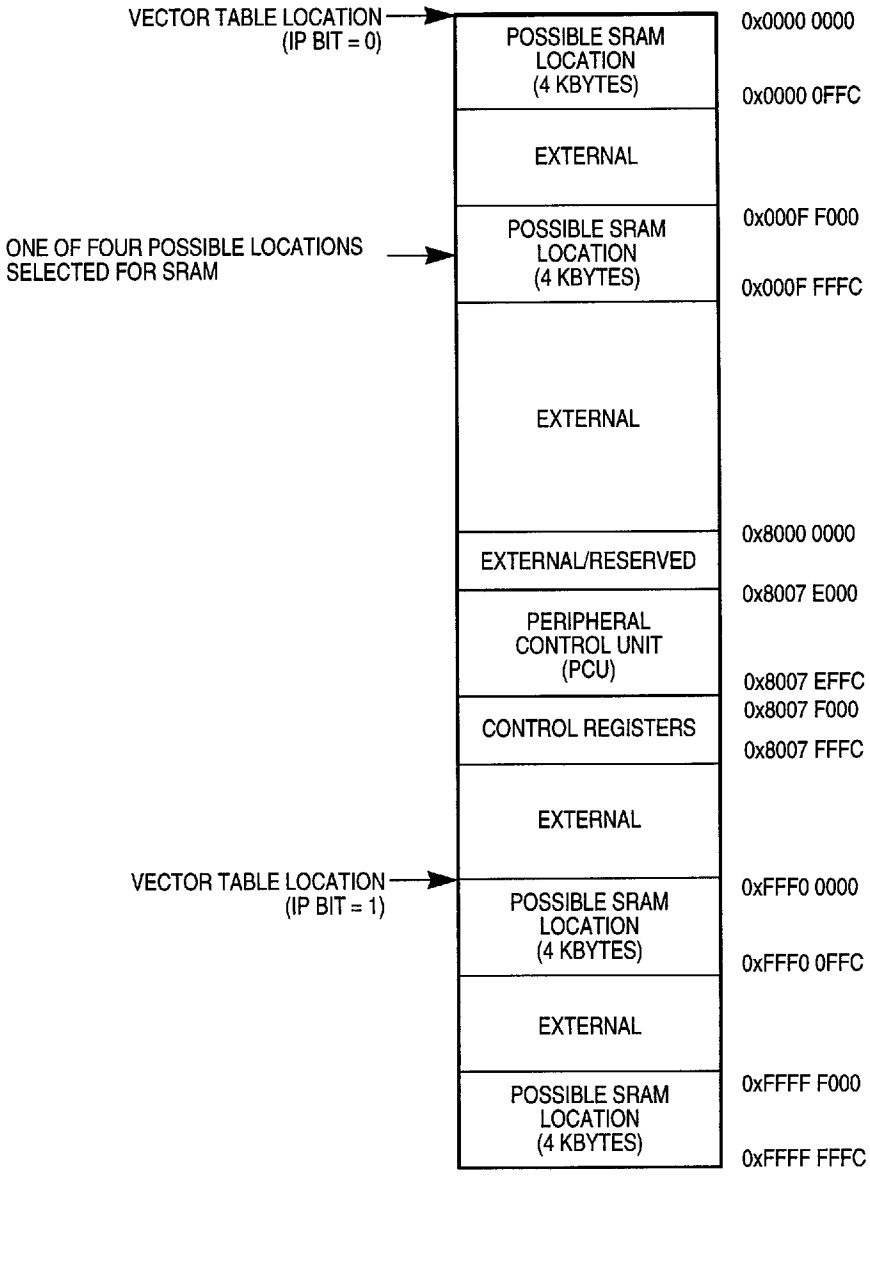


Figure 15 Placement of Internal SRAM in Memory Map

7.3 SRAM Module Registers

The control block for the SRAM module contains one control register for configuring the array and one control register for use in testing.

SRAMMCR — SRAM Module Configuration Register **0x8007 F000**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LCK	DIS	2CY	RESERVED												
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED				R0	D0	S0	RESERVED								
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each SRAM module configuration register contains bits for setting access rights to the array. Table 78 provides definitions for the bits.

Table 78 SRAMMCR Bit Settings

Bit(s)	Name	Description
0	LCK	Lock bit 0 = Writes to the SRAMMCR are accepted. 1 = Writes to the SRAMMCR are ignored.
1	DIS	Module disable 0 = SRAM module is enabled. 1 = SRAM module is disabled. Module can be subsequently re-enabled by software setting this bit or by reset. Attempts to read SRAM array when it is disabled result in internal $\overline{\text{TEA}}$ assertion.
2	2CY	Two-cycle mode 0 = SRAM module is in single-cycle mode (normal operation). 1 = SRAM module is in two-cycle mode. In this mode, the first cycle is used for decoding the address, and the second cycle is used for accepting or providing data. This mode provides some power savings while keeping the memory active.
[3:19]	—	Reserved
20	R0	Read only 0 = 4-Kbyte block is readable and writable. 1 = 4-Kbyte block is read only. Attempts to write to this space result in internal $\overline{\text{TEA}}$ assertion.
21	D0	Data only 0 = 4-Kbyte block can contain data or instructions. 1 = 4-Kbyte block contains data only. Attempts to load instructions from this space result in internal $\overline{\text{TEA}}$ assertion.
22	S0	Supervisor only 0 = 4-Kbyte block is placed in unrestricted space. 1 = 4-Kbyte block is placed in supervisor space. Attempts to access this space from the user privilege level result in internal $\overline{\text{TEA}}$ assertion.
[23:31]	—	Reserved

SRAMTST — SRAM Test Register **0x8007 F004**

The SRAM test register is used for factory testing only.

8 DEVELOPMENT SUPPORT

Development tools are used by a microcomputer system developer to debug the hardware and software of a target system. These tools are used to give the developer some control over the execution of the target program and to allow the user to debug the program by observing its execution. In-circuit emulators, bus state analyzers, and software monitors are the most frequently used debugging tools. This chip supports development and development tools by including internal breakpoint comparators, internal bus visibility, program flow tracking, and a development port.

8.1 Breakpoint Features

- Four instruction address bus (I-address), two load/store address bus (L-address) and two load/store data bus (L-data) magnitude comparators.
- L-data comparators operate on either signed or unsigned numbers and on byte, half word, and word operands.
- Comparator pairs can detect in-range and out-of-range conditions.
- Programmable AND-OR logic on comparator outputs generates six watchpoints.
- Two 16-bit counters are available to condition selected watchpoints.
- Breakpoints can be generated from watchpoints using dynamically controllable trap enables.
- L-address and L-data watchpoints and breakpoints are detected during the load/store bus cycles and are handled and reported when the instruction retires.
- Instructions that cause L-address and L-data breakpoints are executed before the processor branches to the breakpoint exception handler or debug mode.
- I-address watchpoints are detected during instruction fetch and are reported on instruction retirement.
- Instructions that cause I-address breakpoints are not executed before the processor branches to the breakpoint exception handler or debug mode.
- An external breakpoint may be entered using the development port or by any peripheral.
- Breakpoints can be used by an in-circuit emulator and bus analyzer or by a software monitor.

8.2 Internal Bus Visibility Features

- Visibility options for internal fetch cycles:
  - — Show all internal fetch cycles (default at reset).
  - — Show only internal change of program flow destination cycles.
  - — Show only internal indirect change of program flow destination cycles.
  - — Disable visibility of internal fetch cycles.
- Visibility options for internal L-bus (load/store) cycles:
  - — Show all internal L-bus cycles (default at reset).
  - — Show only internal write L-bus cycles.
  - — Disable the visibility of internal L-bus cycles.

8.3 Program Flow Tracking Features

- Exact program trace can be obtained with virtually no decrease in performance.
- Instruction execution tracking output signals:
  - — Flag all change of flow instructions and exceptions.
  - — Flag whether branches are taken or not taken.
  - — Allow tracking of which instructions have retired and which have been flushed from the pipeline.
  - — Allow the history buffer and instruction queue to be tracked by indicating how many instructions are flushed from the instruction queue or the history buffer each clock cycle.
- VSYNC control input causes processor synchronization and automatically enables internal indirect change of program flow destination show cycles.

8.4 Debug Modes

- Hardware debug mode supports debugging with an in-circuit emulator.
  - — All development support features are protected for use by an in-circuit emulator.
  - — Instructions are shifted in and data is shifted in and out through the development port in this mode.
- Software debug mode supports debugging with a software monitor.
  - — All development support features are protected for use by an exception handler (i.e., a software monitor program).
- Internal FREEZE signal may be used to stop operation of selected functions (such as timers) while the processor is in either debug mode.
- Entry into software or hardware debug mode is controlled by the debug enable register. Any interrupt (including breakpoints) can be enabled to enter debug mode instead of taking the normal exception.
- Hardware debug mode is enabled at reset.

8.5 Development Port Features

- Allows external development system full control over the CPU during debug without using regular bus pins and without placing restrictions on user software. This allows users to debug their target systems without making major changes to target system design.
- Supports communications using a three-pin serial interface for both software debug mode and hardware debug mode.
- Allows dynamic input of breakpoint trap enable bits, external breakpoints, and the VSYNC control in either debug mode.
- Development port pins are used at reset to select the debug mode and to configure the development port.
- Supports both asynchronous and synchronous serial transmissions.

8.6 Development Port and Debug Mode Configuration

Development support configuration options are shown in Table 79.

Table 79 Development Port and Debug Mode Configuration

Configuration Option	Select By
Enable hardware debug mode	Drive DSCK pin high while $\overline{\text{RESETOUT}}$ is asserted (low)[1]
Enter software debug mode	Drive DSCK pin low while $\overline{\text{RESETOUT}}$ is asserted (low)
Enable internal clock mode (uses CLKOUT signal to clock serial transmissions)	Drive DSDI pin high for first 16 clocks after $\overline{\text{RESETOUT}}$ is negated
Enable external clock mode (uses DSCK pin to clock serial transmissions)	Drive DSDI pin low for first 16 clocks after $\overline{\text{RESETOUT}}$ is negated
Force debug mode entry following reset	Drive DSCK pin high for first 16 clocks after $\overline{\text{RESETOUT}}$ is negated
Start normal execution following reset	Drive DSCK pin low for first 16 clocks after $\overline{\text{RESETOUT}}$ is negated

1. If DSCK continues to be asserted following reset (after debug mode is enabled), the processor takes a breakpoint exception and enters debug mode after fetching (but not executing) the reset vector.

8.7 Signal Descriptions

Table 80 describes the signals that are used for development support.

Table 80 Development Support Signal Descriptions

Mnemonic	Pin Name	Description
DSCK	Development Serial Clock	Used to clock data being shifted into and out of the development port shift register.
DSDI	Development Serial Data In	Development serial data is shifted into the development port shift register on this pin.
DSDO	Development Serial Data Out	Development serial data is shifted out of the development port shift register on this pin.
VFLS[0:1]	History Buffer Flush Status	Denotes how many instructions are flushed form the history buffer during current clock cycle.
VF[0:2]	Instruction Queue Status	Denotes the type of the last fetched instruction or how many instructions were flushed from the instruction queue.
WP[0:5]	Watchpoint [0:5]	Watchpoint output pins: four for I-bus watchpoints (WP[0:3]) and two for L-bus watchpoints (WP[4:5]).

8.8 Programming Model

Table 81 lists the registers used for development support. The registers are accessed with the **mtspr** and **mfspr** instructions.

Table 81 Development Support Programming Model

SPR Number (Decimal)	Mnemonic	Name
144	CMPA	Comparator A Value Register
145	CMPB	Comparator B Value Register
146	CMPC	Comparator C Value Register
147	CMPD	Comparator D Value Register
148	ICR	Interrupt Cause Register
149	DER	Debug Enable Register
150	COUNTA	Breakpoint Counter A Value and Control Register
151	COUNTB	Breakpoint Counter B Value and Control Register
152	CMPE	Comparator E Value Register
153	CMPF	Comparator F Value Register
154	CMPG	Comparator G Value Register
155	CMPH	Comparator H Value Register
156	LCTRL1	L-Bus Support Control Register 1
157	LCTRL2	L-Bus Support Control Register 2
158	ICTRL	I-Bus Support Control Register
159	BAR	Breakpoint Address Register
630	DPDR	Development Port Data Register

9 IEEE 1149.1-COMPLIANT INTERFACE

The MPC505 boundary-scan interface is a fully-compliant implementation of the IEEE 1149.1 standard. This section describes the MPC505 IEEE 1149.1 Joint Test Action Group (JTAG) interface.

The MPC505 has five dedicated JTAG pins, which are described in Table 82. The TDI and TDO scan ports are used to scan instructions as well as data into the various scan registers for JTAG operations. The scan operation is controlled by the test access port (TAP) controller, which in turn is controlled by the TMS input sequence.

Table 82 JTAG Interface Pin Descriptions

Signal Name	Input/Output	Internal Pull-Up/Pulldown Provided	IEEE 1149.1 Function
TDI	Input	Pull-up	Serial scan input pin. Sampled on the rising edge of TCK.
TDO	Output	No	Serial scan output pin. Actively driven during the Shift-IR and Shift-DR controller states. Changes on the falling edge of TCK.
TMS	Input	Pull-up	TAP controller mode pin. Sampled on the rising edge of TCK to sequence the test controller's state machine.
TCK	Input	Pulldown	Scan clock.
$\overline{\text{TRST}}$	Input	Pull-up	TAP controller asynchronous reset. Provides initialization of the TAP controller and other logic as required by the standard.

9.1 TAP Controller

$\overline{\text{TRST}}$ is a JTAG optional pin that is used to reset the TAP controller asynchronously. The $\overline{\text{TRST}}$ pin ensures that the JTAG logic does not interfere with the normal operation of the chip.

The TAP controller changes state either on the rising edge of TCK or when $\overline{\text{TRST}}$ is asserted.

The TDO signal remains in a high-impedance state except during the Shift-DR or Shift-IR controller states. During these controller states, TDO is updated on the falling edge of TCK.

The TAP controller states are designed as specified in the IEEE 1149.1 standard.

9.2 Instruction Register

The MPC505 implementation of IEEE 1149.1 interface includes the three mandatory public instructions (BYPASS, SAMPLE/PRELOAD, and EXTEST) and five public instructions (CLAMP, HIGHZ, EXTEST_PULLUP, TMSCAN, and IDCODE). The MPC505 contains a four-bit instruction register without parity consisting of a shift register with four parallel outputs. Data is transferred from the shift register to the parallel outputs during Update-IR controller state. The four bits are used to decode eight unique instructions as shown in Table 83.

Table 83 Instruction Register Encoding

Code				Instruction
B3	B2	B1	B0	
1	1	1	1	BYPASS
1	1	1	0	SAMPLE/PRELOAD
1	1	0	1	IDCODE
1	1	0	0	TMSCAN
1	0	x	x	BYPASS
0	1	x	x	BYPASS
0	0	1	1	CLAMP
0	0	1	0	HIGHZ
0	0	0	1	EXTEST_PULLUP
0	0	0	0	EXTEST

Summary of Changes

This is a complete revision, with complete reprint. All known errors in the publication have been corrected. The following summary lists significant changes.

Section 2 Signal Descriptions

Page 10	Added description of DS signal.

Section 3 Central Processing Unit

Page 18	Added Section 3.5 Levels of the PowerPC Architecture.
Pages 18 – 35	Revised and reorganized Section 3.6 RCPU Program Model, Section 3.7 PowerPC UISA Register Set, Section 3.8 PowerPC VEA Register Set, and Section 3.9 PowerPC OEA Register Set. Revised Figure 7 RCPU Programming Model. Corrected reset settings for the following registers: CR, FPSCR, MSR, XER, and TB. Changed the following bit names in the machine state register: ELE to ILE, RE to RI, EP to IP. Included implementation-specific SPRs.
Pages 36 – 41	Added Table 21 Instruction Set Summary.
Page 45	Added Table 24 Instruction Latency and Blockage.

Section 4 Instruction Cache

Page 53	Added Table 29 ICADR Bits Function for the Cache Read Command and Table 30 ICDAT Layout During a Tag Read.

Section 5 System Interface Unit

Page 54	Updated Figure 11 SIU Block Diagram.
Pages 55 – 56	Expanded Table 31 SIU Address Map.
Pages 57 – 58	Expanded description of LST bit in Table 32 SIUMCR Bit Settings.
Page 58	Corrected reset settings in MEMMAP register diagram.
Page 60	Added description of DS signal to Table 34 EBI Signal Descriptions.
Page 62	Updated Table 37 Cycle Type Encodings.
Page 63	Revised first paragraph of Section 5.3.5 Preventing Speculative Loads.
Page 67	Replaced the chip-select block diagram.
Page 75	Revised Table 47 Interface Types.
Page 79	Revised Table 52 Clock Module Signal Descriptions.
Page 81	Added Section 5.6.5 Low-Power Modes.
Page 84	Changed register name from system clock control and status register to system clock lock and status register.
Pages 86 – 87	Revised organization of Section 5.7 Reset. Revised first two paragraphs of Section 5.7.3 Reset Flow.
Pages 88 – 90	Revised Table 61 Data Bus Reset Configuration Word.
Page 90	Added Table 62 Reset Behavior for Different Clock Modes.
Pages 92 – 94	Modified Section 5.8.1 Port M to replace PM2 signal with DS.
Page 97	Added Section 5.8.4 Port Replacement Unit Mode.