CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

Chapter 5
Secondary Cache Interface

CONTENTS

CONTENTS

CONTENTS

Chapter 9
Error Handling

CONTENTS

ILLUSTRATIONS

ILLUSTRATIONS

ILLUSTRATIONS

ILLUSTRATIONS

TABLES

TABLES

About This Book

The primary objective of this user's manual is to describe the functionality of the MPC105 PCI bridge/memory controller (PCIB/MC) for use by systems designers and software developers. The MPC105 is the first device in a family of products that provides system-level support for industry-standard interfaces to be used with PowerPC™ microprocessors.

In this document, the term "60x" is used to denote a 32-bit microprocessor from the PowerPC Architecture™ family that conforms to the bus interface of the PowerPC 601™, PowerPC 603™, or PowerPC 604™ microprocessors. Note that this does not include the PowerPC 602™ microprocessor which has a multiplexed address/data bus. 60x processors implement the PowerPC architecture as it is specified for 32-bit addressing, which provides 32-bit effective (logical) addresses, integer data types of 8, 16, and 32 bits, and floating-point data types of 32 and 64 bits (single-precision and double-precision).

It must be kept in mind that each PowerPC processor is a unique PowerPC implementation. It is beyond the scope of the manual to provide a thorough description of the PowerPC architecture. Refer to *PowerPC Microprocessor Family: The Programming Environments* for more information about the architecture. It is also beyond the scope of the manual to provide a thorough description of the PCI local bus. Refer to *PCI Local Bus Specification* and *PCI System Design Guide* for more information about the PCI bus.

Audience

This manual is intended for system software and hardware developers who want to develop products incorporating PowerPC microprocessors and the PCI bus. It is assumed that the reader understands operating systems, microprocessor system design, and the basic principles of RISC processing.

Organization

Following is a summary and a brief description of the major sections of this manual:

- Chapter 1, “Overview,” is useful for readers who want a general understanding of the features and functions of the MPC105.
- Chapter 2, “Signal Descriptions,” provides descriptions of individual signals of the MPC105.
- Chapter 3, “Device Programming,” is useful for software engineers who need to understand the address space and functionality of the registers implemented in the MPC105.
- Chapter 4, “Processor Bus Interface,” describes the interaction between the MPC105 and the 60x processor or multiple 60x processors.
- Chapter 5, “Secondary Cache Interface,” describes the operation of the secondary or level 2 (L2) cache interface.
- Chapter 6, “Memory Interface,” provides details for interfacing the MPC105 to DRAM, SDRAM, ROM, and Flash ROM devices.
- Chapter 7, “PCI Bus Interface,”describes the MPC105 as a bridge from the 60x processor bus to the PCI bus and the MPC105 as a PCI agent.
- Chapter 8, “Internal Control,” describes the internal buffers between the interfaces of the MPC105.
- Chapter 9, “Error Handling,” describes how the MPC105 handles error detection and reporting on the three primary interfaces—processor interface, memory interface, and PCI interface.
- Appendix A, “Power Management,”provides information about power saving modes for the MPC105.
- Appendix B, “Bit and Byte Ordering,” describes big- and little-endian byte ordering and the implications on systems using the MPC105.
- Appendix C, “JTAG/Testing Support,” describes the IEEE 1149.1 functions used for facilitating board testing and chip debug.
- Appendix D, “Initialization Example,” provides sample initialization code in PowerPC assembly language.
- This manual also includes a glossary and an index.

In this document, the terms 601, 603, and 604 are used as abbreviations for the phrases, “PowerPC 601 microprocessor,” “PowerPC 603 microprocessor,” and “PowerPC 604 microprocessor,” respectively.

Additional Reading

Following is a list of additional reading that provides background for the information in this manual:

- *MPC105 PCI Bridge/Memory Controller Technical Summary*, MPC105/D
- *PowerPC 601 RISC Microprocessor User's Manual*, Rev 1 MPC601UM/AD (Motorola order number)
- *PowerPC 603 RISC Microprocessor User's Manual*, MPC603UM/AD (Motorola order number)
- *PowerPC 604 RISC Microprocessor User's Manual*, MPC604UM/AD (Motorola order number)
- *PCI Local Bus Specification* Rev 2.0, PCI Special Interest Group, Hillsboro, OR
- *PCI System Design Guide*, Rev 1.0, PCI Special Interest Group, Hillsboro, OR
- *PowerPC Microprocessor Family: The Programming Environments*, MPCFPE/AD (Motorola order number)
- *PowerPC Architecture: A Specification for a New Family of RISC Processors*, Morgan Kaufmann Publishers, Inc., San Francisco, CA
- John L. Hennessy and David A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers, Inc., San Mateo, CA

Conventions

This document uses the following notational conventions:

ACTIVE_HIGH	Names for signals that are active high are shown in uppercase text without an overbar. Active-high signals are referred to as asserted when they are high and negated when they are low.
$\overline{\text{ACTIVE\_LOW}}$	A bar over a signal name indicates that the signal is active low. Active-low signals are referred to as asserted (active) when they are low and negated when they are high.
0x0F	Hexadecimal numbers
0b0011	Binary numbers
REG[FIELD]	Abbreviations or acronyms for registers are shown in uppercase text. Specific bit fields or ranges are shown in brackets.
n	In certain contexts, such as a signal encoding, this indicates a variable. For example, if TT0–TT3 are binary encoded 0b*n*001, the state of TT0 could be 1 or 0.

Acronyms and Abbreviations

Table i contains acronyms and abbreviations that are used in this document.

Table i. Acronyms and Abbreviated Terms

Term	Meaning
BGA	Ball grid array package
BIST	Built-in self test
BIU	Bus interface unit
CAS	Column address strobe
CBR	CAS before RAS
DRAM	Dynamic random access memory
ErrDR	Error detection register
ErrEnR	Error enabling register
IEEE	Institute of Electrical and Electronics Engineers
JTAG	Joint test action group interface
L2	Secondary cache
MICR	Memory interface configuration register
MCCR	Memory control configuration register
PCI	Peripheral component interconnect
PCIB/MC	PCI bridge/memory controller
PICR	Processor interface configuration register
PLL	Phase-locked loop
PMC	Power management controller
RAS	Row address strobe
ROM	Read-only memory
RTC	Real-time clock
SDRAM	Synchronous dynamic random access memory
SIMM	Single in-line memory module
VCO	Voltage-controlled oscillator

Chapter 1
Overview

The MPC105 PCI bridge/memory controller (PCIB/MC) provides a PowerPC™ reference platform-compliant bridge between the PowerPC microprocessor family and the peripheral component interconnect (PCI) bus. PCI support allows system designers to rapidly design systems using peripherals already designed for PCI and the other standard interfaces available in the personal computer hardware environment. The MPC105 integrates secondary cache control and a high-performance memory controller that supports DRAM, SDRAM, ROM, and Flash ROM. The MPC105 uses an advanced, 3.3-V CMOS process technology and is fully compatible with TTL devices.

This chapter provides an overview of the MPC105. It includes the following:

- An overview of MPC105 features
- Details about the MPC105 device including descriptions of the MPC105's functional units and interfaces

1.1 MPC105 PCIB/MC Features

The MPC105 provides an integrated, high bandwidth, high performance, TTL-compatible interface between a 60x processor, a secondary (L2) cache or secondary 60x processor, the PCI bus, and main memory. This section summarizes the features of the MPC105 and provides a block diagram showing the major functional units.

Figure 1-1 shows the MPC105 in a typical system implementation. The major functional units within the MPC105 are also shown in Figure 1-1. Note that this is a conceptual block diagram intended to show the basic features rather than an attempt to show how these features are physically implemented on the device.

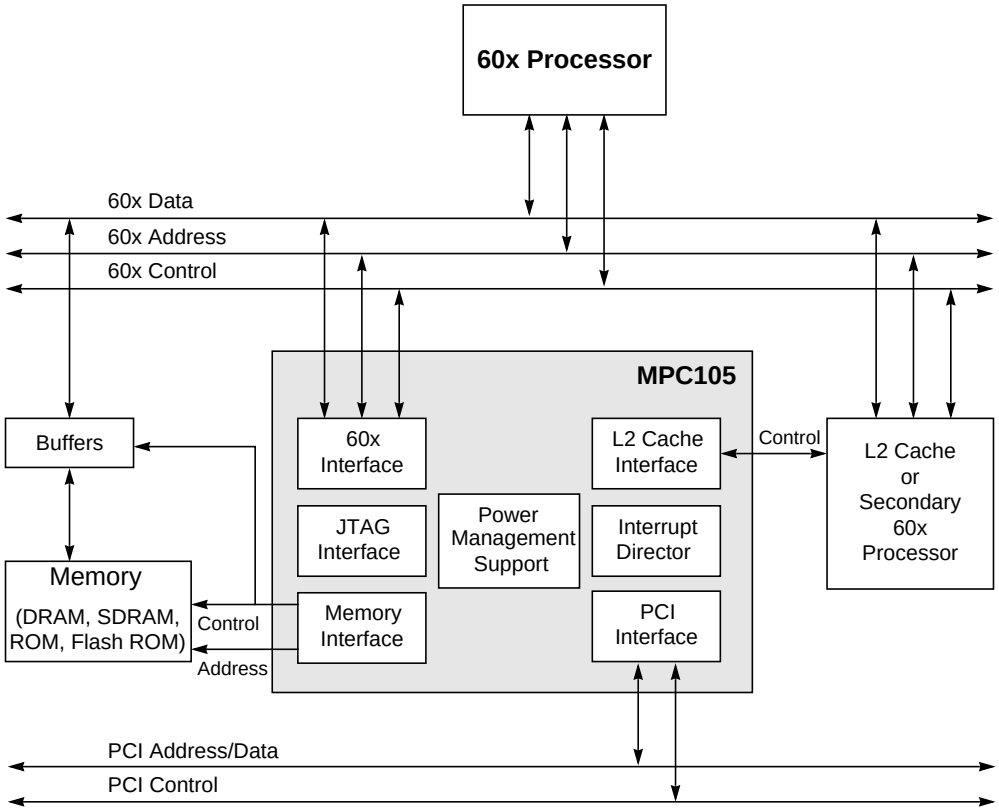


Figure 1-1. System Implementation and Block Diagram

Major features of the MPC105 are as follows:

- Processor interface
  - — 60x processors supported at a wide range of frequencies
  - — 32-bit address bus
  - — Configurable 64- or 32-bit data bus
  - — Accommodates either an external L2 cache or a secondary processor
  - — Arbitration for secondary processor on-chip
  - — Full memory coherency supported
  - — Pipelining of 60x accesses
  - — Store gathering on 60x to PCI writes

- Secondary (L2) cache interface
  - — Configurable for write-through or write-back operation
  - — 256-Kbyte, 512-Kbyte, 1-Mbyte sizes
  - — Up to 4 Gbytes of cacheable space
  - — Direct-mapped
  - — Parity supported
  - — Supports external byte decode or on-chip byte decode for write enables
  - — Programmable timing supported
  - — Synchronous burst or asynchronous SRAMs supported
- PCI interface
  - — Compliant with *PCI Local Bus Specification, Revision 2.0*
  - — Selectable big- or little-endian operation
  - — Store gathering on PCI writes to memory
  - — Selectable memory prefetching of PCI read accesses
  - — Only one external load presented by the MPC105 to the PCI bus
  - — PCI configuration registers
  - — Interface operates at 20MHz–33 MHz
  - — Data buffering (in/out)
  - — Word parity supported
  - — Supports PCI interlocked accesses to memory using $\overline{\text{LOCK}}$ signal and protocol
  - — 3.3 V/5.0 V-compatible
- Concurrent transactions on processor and PCI buses supported
- Memory interface
  - — Programmable timing supported
  - — Supports either DRAM or synchronous DRAM (SDRAM)
  - — High bandwidth (64-bit) data bus
  - — Configurable external buffer control logic
  - — Supports 1 to 8 banks built of x1, x4, x8, x9, x16, or x18 DRAMs
  - — 1 Gbyte of RAM space, 16 Mbytes of ROM space
  - — Supports self-refreshing DRAM in sleep and suspend mode
  - — Supports byte parity
  - — Supports PowerPC reference platform-compliant contiguous or discontiguous memory maps
  - — Supports 8-bit asynchronous ROM or 32-/64-bit burst-mode ROM
  - — Supports writing to Flash EPROMs
  - — TTL-compatible

- Power management
  - — Fully-static 3.3-V CMOS design
  - — Supports 60x nap, doze, and sleep power management modes, and suspend mode
- IEEE 1149.1-compliant, JTAG boundary-scan interface
- 304-pin ball grid array (BGA) package

1.2 MPC105 Major Functional Units

The MPC105 consists of the following major functional units:

- 60x processor interface
- Secondary (L2) cache/processor interface
- PCI interface
- Memory interface

This section describes each of these functional units.

1.2.1 60x Processor Interface

The MPC105 supports a programmable interface to a variety of PowerPC microprocessors operating at various bus speeds. The 60x processor interface uses a subset of the 60x bus protocol, which enables the interface between the processor and MPC105 to be optimized for performance.

Depending on the system implementation, the processor may operate at the PCI bus clock rate, or at two or three times the PCI bus clock rate. The bus is synchronous, with all timing relative to the rising edge of the bus clock. Inputs are sampled at, and outputs are driven from, this edge. The address bus is 32 bits wide and the data bus is 64 bits wide (or 32 bits in 32-bit mode). The MPC105 supports single-beat and burst data transfers. The processor interface has decoupled address and data buses to support pipelined transactions.

PCI bus accesses to the system memory space are passed to the 60x processor(s) and/or L2 cache for snooping purposes.

1.2.2 Secondary (L2) Cache/Processor Interface

The MPC105 allows for a variety of system configurations by providing support for either a direct-mapped, lookaside L2 cache or a secondary 60x processor. The MPC105 uses snoop operations to ensure data coherency between the caches (one or two L1 caches, or one L1 and one L2) and main memory.

The L2 cache interface generates the arbitration and support signals necessary to maintain a write-through or write-back L2 cache. The L2 cache interface supports either burst SRAMs or asynchronous SRAMs, and L2 data parity on a per-byte basis. The MPC105 features on-chip byte decoding for L2 data write enables or can be configured to use external logic for data write enable generation.

The L2 cache interface handles the following types of bus cycles:

- Normal 60x bus cycles
- 60x internal cache copy-back cycles
- L2 copy-back cycles
- Snoop cycles

When a secondary 60x processor is used instead of an L2 cache, three signals ($\overline{\text{DIRTY\_IN}}$/$\overline{\text{BR1}}$, $\overline{\text{DIRTY\_OUT}}$/$\overline{\text{BG1}}$, and $\overline{\text{TOE}}$/$\overline{\text{DBG1}}$) change their functions to allow for arbitration between two 60x processors. Excepting the bus request, bus grant, and data bus grant signals, all other 60x interface signals are shared by both 60x processors.

1.2.3 PCI Interface

The PCI interface connects the processor and memory buses to the PCI bus, to which I/O components are connected, without the need for "glue" logic. This interface acts as both a master and slave device. The PCI interface supports a 32-bit multiplexed, address/data bus that can operate from 20 MHz to 33 MHz. Buffers are provided for I/O operations between the PCI bus and the 60x processor or memory. Processor read and write operations each have a 32-byte buffer, and memory operations have one 32-byte read buffer and two 32-byte write buffers.

The PCI interface supports address and data parity with error checking and reporting. The interface also supports three physical address spaces—32-bit address memory, 32-bit address I/O, and some of the PCI 256-byte configuration space. Mode selectable big-endian to little-endian conversion is also supplied at the PCI interface.

The PCI interface is compliant with the *PCI Local Bus Specification, Revision 2.0*, and follows the guidelines in the *PCI System Design Guide, Revision 1.0* for host bridge architecture.

1.2.4 Memory Interface

The memory interface controls processor and PCI interactions to main memory. It is capable of supporting a variety of DRAM or SDRAM, and ROM or Flash ROM configurations as main memory. The maximum supported memory size is 1 Gbyte of DRAM or SDRAM, with 16 Mbytes of ROM or 1 Mbyte of Flash ROM. The MPC105 configures its memory control to support the various memory sizes through software initialization of on-chip configuration registers. Parity protection is provided for the DRAM or SDRAM. If SDRAM is used, it must comply with the JEDEC specification for SDRAM.

The MPC105 can control either a 64- or 32-bit data path to main memory; SDRAM systems support 64-bit data paths only. To reduce loading on the data bus, system designers may implement buffers between the 60x bus and memory. The MPC105 features configurable data buffer control logic to accommodate several buffer types. The MPC105 handles parity checking and generation, with eight parity bits checked or generated for a 64-bit data path, and four parity bits checked or generated for a 32-bit data path.

The MPC105 is capable of supporting a variety of DRAM or SDRAM configurations. Twelve multiplexed address signals provide for device depths up to 16 M (64 or 32 bits wide depending on the bus mode). Eight row address strobe/command select ($\overline{RAS}/\overline{CS}$) signals support up to eight banks of memory. Each bank can be 8 bytes wide. Eight column address strobe/data qualifier ($\overline{CAS}/\overline{DQM}$) signals are used to provide byte selection for memory access.

DRAM or SDRAM banks can be built of SIMMs or directly-attached memory chips. The data path to the memory banks must be either 32 or 64 bits wide (36 or 72 with parity). The banks can be constructed using x1, x4, x8, x9, x16, or x18 memory devices. Regardless of whether DRAMs or SDRAMs are used, the memory design must be byte-selectable for writes using the $\overline{CAS}/\overline{DQM}$ signals.

The MPC105 memory interface provides for doze, nap, sleep, and suspend power saving modes, defined in Section 1.3, “Power Management.” In the sleep and suspend power saving modes, the MPC105 can be configured to put the DRAM array into a self-refresh mode, (if supported by the DRAMs). The MPC105 may be configured to use the RTC input as its refresh time base in suspend mode. If self-refreshing DRAMs are not available or the RTC input is not used (in suspend mode), system software must preserve DRAM data (such as by copying the data to disk) in the sleep or suspend mode. In doze and nap power saving modes and in the full-on mode, the MPC105 supplies CAS before RAS (CBR) refresh to DRAM.

An MPC105 configuration signal (sampled at reset) determines whether the MPC105 accesses boot code from ROM or Flash ROM. If the MPC105 is configured to access boot code from ROM, the corresponding data path must be the same bit width as the DRAM or SDRAM data path (32 or 64 bits). Twenty address bits and two bank selects are provided for ROM systems. If the MPC105 is configured to access boot code from Flash ROM, the corresponding data path must be 8 bits wide and must be connected to the most significant byte of the data bus. Twenty address bits, one bank select signal, one output enable signal, and one write enable signal are provided for Flash ROM systems.

1.3 Power Management

The MPC105 provides hardware support for four levels of power reduction; the nap, doze, and sleep modes are invoked by register programming, and the suspend mode is invoked by assertion of an external signal. The design of the MPC105 is fully static, allowing internal logic states to be preserved during all power saving modes. The following sections describe the programmable power modes provided by the MPC105.

1.3.1 Full-On Mode

This is the default power state of the MPC105 following a hard reset, with all internal functional units fully powered and operating at full clock speed.

1.3.2 Doze Mode

In the doze power saving mode, all the MPC105 functional units are disabled except for PCI address decoding, system RAM refreshing, and the CPU bus request monitoring (through $\overline{\text{BR}}n$). Once the doze mode is entered, a hard reset, a PCI transaction referenced to system memory, or a bus request can bring the MPC105 out of the doze mode and into the full-on state. If the MPC105 is awakened for a processor or PCI bus access, the access is completed and the MPC105 returns to the doze mode. The doze mode is totally independent of the power saving mode of the processor.

1.3.3 Nap Mode

Further power savings can be achieved through the nap mode, when both the processor and the MPC105 are placed in a power reduction mode. In this mode, only the PCI address decoding, system RAM refreshing, and the processor bus request monitoring are still operating. Hard reset, a PCI bus transaction referenced to system memory, or a bus request can bring the MPC105 out of the nap mode. If the MPC105 is awakened by a PCI access, the access is completed, and the MPC105 returns to the nap mode. If the MPC105 is awakened by a processor access, the access is completed, but the MPC105 remains in the full-on state. When in the nap mode, the PLL (phase-locked loop) is required to be running and locked to the system clock (SYSCLK).

1.3.4 Sleep Mode

Sleep mode provides further power savings compared to the nap mode. As in nap mode, both the processor and the MPC105 are placed in a reduced power mode concurrently. In sleep mode, no functional units are operating except the system RAM refresh logic, which can continue (optionally) to perform the refresh cycles. A hard reset or a bus request wakes the MPC105 from the sleep mode. The PLL and SYSCLK inputs may be disabled by an external power management controller (PMC). For additional power savings, the PLL can be disabled by configuring the PLL0–PLL3 pins into the PLL bypass mode. When recovering from sleep mode, the external power management controller has to re-enable the PLL and SYSCLK first, and then wake up the system after allowing the PLL time to relock.

1.3.5 Suspend Mode

Suspend mode is activated through assertion of the $\overline{\text{SUSPEND}}$ signal. In suspend mode, the MPC105 may have its clock input and PLL shut down for additional power savings. Memory refresh can be accomplished in two ways—either by using self-refresh mode DRAMs or by using the RTC input. To exit the suspend mode, the system clock must be turned on in sufficient time to restart the PLL. After this time, $\overline{\text{SUSPEND}}$ may be negated. In suspend mode, all outputs (except memory refresh) are high-impedance and all inputs (including $\overline{\text{HRST}}$) are ignored.

Chapter 2
Signal Descriptions

This chapter provides descriptions of the MPC105's external signals. It describes each signal's behavior when the signal is asserted and negated and when the signal is an input or an output.

NOTE

A bar over a signal name indicates that the signal is active low—for example, $\overline{\text{ARTRY}}$ (address retry) and $\overline{\text{TS}}$ (transfer start). Active-low signals are referred to as asserted (active) when they are low and negated when they are high. Signals that are not active low, such as PAR (PCI bus parity signal) and TT0–TT4 (transfer type signals) are referred to as asserted when they are high and negated when they are low.

2.1 Signal Configuration

The MPC105's signals are grouped as follows:

- 60x processor interface signals
- Secondary (L2) cache/processor interface signals
- Memory interface signals
- PCI interface signals
- Interrupt, clock, and power management signals
- IEEE 1149.1 interface signals
- Configuration signals

Figure 2-1 illustrates the signals of the MPC105, showing how the signals are grouped. A pinout showing actual pin numbers is included in the MPC105 Hardware Specifications.

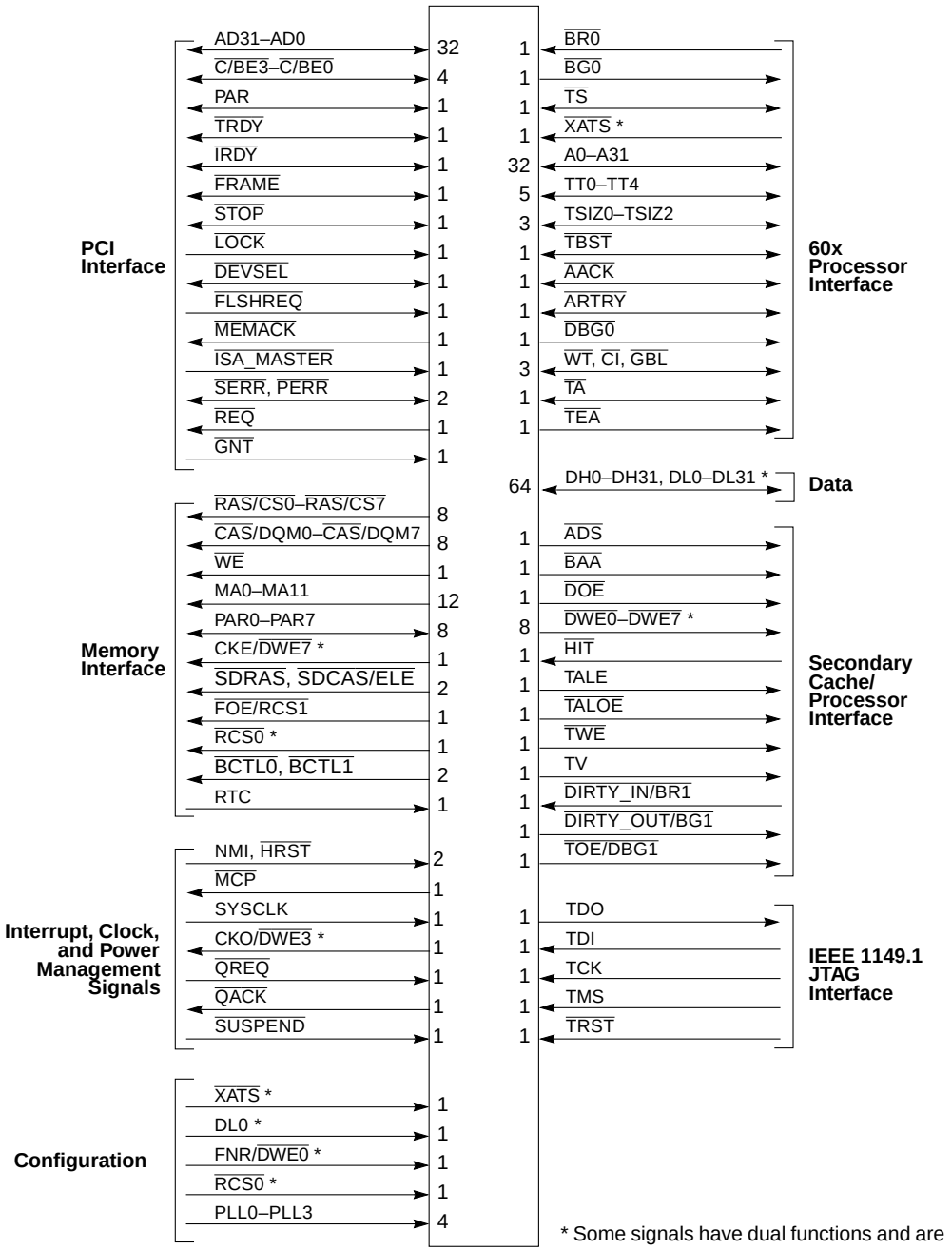


Figure 2-1. MPC105 Signal Groupings

2.2 Signal Descriptions

This section describes individual MPC105 signals, grouped according to Figure 2-1. The following sections are intended to provide a quick summary of signal functions.

2.2.1 60x Processor Interface Signals

This section provides descriptions of the 60x processor interface signals on the MPC105. Note that with the exception of $\overline{\text{BR0}}$, $\overline{\text{BG0}}$, and $\overline{\text{DBG0}}$, all of the 60x processor interface signals are connected to both processors in a multiprocessor system. See Section 4.1.2, "Multiprocessor System Configuration," for more information.

2.2.1.1 Bus Request 0 ($\overline{\text{BR0}}$)—Input

The bus request 0 ($\overline{\text{BR0}}$) signal is an input on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{BR0}}$ signal.

State Meaning Asserted—Indicates that the primary 60x requires mastership of the 60x bus for a transaction.

Negated—Indicates that the primary 60x does not require mastership of the 60x bus.

Timing Comments Assertion—May occur when bus grant 0 ($\overline{\text{BG0}}$) is negated and a bus transaction is needed by the 60x. This may occur even if the two possible pipeline accesses have already occurred.

Negation—Occurs for at least one bus cycle after an accepted, qualified bus grant, even if another transaction is pending on the 60x. It is also negated for at least one bus cycle when the assertion of $\overline{\text{ARTRY}}$ is detected on the 60x bus (except for assertions due to 60x snoop copybacks).

2.2.1.2 Bus Grant 0 ($\overline{\text{BG0}}$)—Output

The bus grant 0 ($\overline{\text{BG0}}$) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{BG0}}$ signal.

State Meaning Asserted—Indicates that the primary 60x may, with the proper qualification, begin a bus transaction and assume mastership of the address bus.

Negated—Indicates that the 60x is not granted mastership of the next address bus tenure.

Timing Comments Assertion—Occurs when $\overline{\text{BR0}}$ is the highest priority request that is asserted. Also occurs if the 60x is parked and no other request is pending.

Negation—Occurs when other higher priority transactions are pending.

2.2.1.3 Transfer Start ($\overline{TS}$)

The transfer start ($\overline{TS}$) signal is both an input and an output signal on the MPC105.

2.2.1.3.1 Transfer Start ($\overline{TS}$)—Output

Following are the state meaning and timing comments for the $\overline{TS}$ output signal.

State Meaning Asserted—Indicates that the MPC105 has started a bus transaction, and that the address and transfer attribute signals are valid. Note that the MPC105 only initiates a transaction to broadcast the address of a PCI access to memory for snooping purposes.

Negated—Has no special meaning.

Timing Comments Assertion—Occurs one cycle after $\overline{BG0}$ (or $\overline{DIRTY\_OUT/BG1}$ in a multiprocessor configuration) is negated if the address bus is idle. Otherwise, it occurs two clocks after the assertion of the address acknowledge ($\overline{AACK}$) signal.

Negation—Occurs one clock after assertion.

High-impedance—Occurs one clock after the assertion of $\overline{AACK}$.

2.2.1.3.2 Transfer Start ($\overline{TS}$)—Input

Following are the state meaning and timing comments for the $\overline{TS}$ input signal.

State Meaning Asserted—Indicates that a 60x bus master has begun a bus transaction, and that the address and transfer attribute signals are valid.

Negated—Has no special meaning.

Timing Comments Assertion—May occur one cycle after $\overline{BG0}$ (or $\overline{DIRTY\_OUT/BG1}$ in a multiprocessor configuration) is asserted.

Negation— Occurs one clock after assertion.

2.2.1.4 Extended Address Transfer Start ($\overline{XATS}$)—Input

The $\overline{XATS}$ signal is an input on the MPC105. If $\overline{XATS}$ is connected to a pull-down resistor to select address map B, it is ignored by the MPC105 and is not connected to the processor. Refer to Section 2.2.7.4, “Address Map (XATS)—Input” for more information.

Following are the state meaning and timing comments for the $\overline{XATS}$ signal.

State Meaning Asserted—Indicates that the 60x has started a direct-store access (using the extended transfer protocol). Since direct-store accesses are not supported by the MPC105, the MPC105 automatically asserts the transfer error acknowledge ($\overline{TEA}$) signal when $\overline{XATS}$ is asserted (provided $\overline{TEA}$ is enabled). If $\overline{TEA}$ is disabled, the MPC105 terminates the direct-store access by asserting $\overline{TA}$; however, no data is altered.

Negated—Has no special meaning.

Timing Comments Assertion—May occur one cycle after $\overline{\text{BG0}}$ (or $\overline{\text{DIRTY\_OUT/BG1}}$) is asserted. $\overline{\text{XATS}}$ can only be asserted by a processor (that is, the MPC105 cannot assert $\overline{\text{XATS}}$).

Negation—Occurs one clock after assertion.

2.2.1.5 Address Bus (A0–A31)

The address bus (A0–A31) consists of 32 signals that are both input and output signals.

2.2.1.5.1 Address Bus (A0–A31)—Output

Following are the state meaning and timing comments for A0–A31 as output signals.

State Meaning Asserted/Negated—Specifies the physical address for 60x bus snooping or for an L2 copy-back operation.

Timing Comments Assertion/Negation—Occurs coincident with $\overline{\text{TS}}$. Once driven, these signals remain stable for the entire address tenure.

High-impedance—Occurs on the cycle after the assertion of $\overline{\text{AACK}}$.

2.2.1.5.2 Address Bus (A0–A31)—Input

Following are the state meaning and timing comments for A0–A31 as input signals.

State Meaning Asserted/Negated—Specifies the physical address of the bus transaction. For burst reads, the address is aligned to the critical double-word address that missed in the instruction or data cache. For burst writes, the address is aligned to the double-word address of the cache line being pushed from the data cache.

Timing Comments Assertion/Negation—Must occur on the same bus cycle as the assertion of $\overline{\text{TS}}$. Once driven, these signals remain stable for the entire address tenure.

High-impedance—Occurs on the cycle after the assertion of $\overline{\text{AACK}}$.

2.2.1.6 Transfer Type (TT0–TT4)

The transfer type (TT0–TT4) signals consist of five input/output signals on the MPC105.

2.2.1.6.1 Transfer Type (TT0–TT4)—Output

Following are the state meaning and timing comments for TT0–TT4 as output signals.

State Meaning Asserted/Negated—Specifies the type of 60x bus transfer in progress for snooping. Refer to Section 4.3.2.1, “Transfer Type Signal Encodings,” for transfer type encodings.

Timing Comments Assertion/Negation—The same as A0–A31.

High-impedance—The same as A0–A31.

2.2.1.6.2 Transfer Type (TT0–TT4)—Input

Following are the state meaning and timing comments for TT0–TT4 as input signals.

State Meaning Asserted/Negated—Specifies the type of 60x bus transfer in progress. Refer to Section 4.3.2.1, "Transfer Type Signal Encodings," for transfer type encodings.

Timing Comments Assertion/Negation—The same as A0–A31.

High-impedance—The same as A0–A31.

2.2.1.7 Transfer Size (TSIZ0–TSIZ2)

The transfer size (TSIZ0–TSIZ2) signals consist of three input/output signals on the MPC105.

2.2.1.7.1 Transfer Size (TSIZ0–TSIZ2)—Output

Following are the state meaning and timing comments for TSIZ0–TSIZ2 as output signals. Note that all MPC105-generated snoop operations are eight-word bursts; therefore TSIZ0–TSIZ2 are always 0b010 for snoop operations.

State Meaning Asserted/Negated—In conjunction with the transfer burst ($\overline{\text{TBST}}$) signal, TSIZ0–TSIZ2 specify the data transfer size for the 60x bus transaction. Refer to Section 4.3.2.2, "$\overline{\text{TBST}}$ and TSIZ0–TSIZ2 Signals and Size of Transfer," for transfer size encodings.

Timing Comments Assertion/Negation—The same as A0–A31.

High-impedance—The same as A0–A31.

2.2.1.7.2 Transfer Size (TSIZ0–TSIZ2)—Input

Following are the state meaning and timing comments for TSIZ0–TSIZ2 as input signals.

State Meaning Asserted/Negated—In conjunction with the transfer burst ($\overline{\text{TBST}}$) signal, TSIZ0–TSIZ2 specify the data transfer size for the 60x bus transaction. Refer to Section 4.3.2.2, "$\overline{\text{TBST}}$ and TSIZ0–TSIZ2 Signals and Size of Transfer," for transfer size encodings.

Timing Comments Assertion/Negation—The same as A0–A31.

High-impedance—The same as A0–A31.

2.2.1.8 Transfer Burst ($\overline{\text{TBST}}$)

The transfer burst ($\overline{\text{TBST}}$) signal is an input/output signal on the MPC105.

2.2.1.8.1 Transfer Burst ($\overline{\text{TBST}}$)—Output

Following are the state meaning and timing comments for $\overline{\text{TBST}}$ as an output signal. Note that all MPC105 generated snoop operations are eight-word bursts; therefore, $\overline{\text{TBST}}$ is always asserted for snoop operations.

State Meaning Asserted—Indicates that a burst transfer is in progress.

Negated—Indicates that a burst transfer is not in progress.

Timing Comments Assertion/Negation—The same as A0–A31.

High-impedance—The same as A0–A31.

2.2.1.8.2 Transfer Burst ($\overline{\text{TBST}}$)—Input

Following are the state meaning and timing comments for $\overline{\text{TBST}}$ as an input signal.

State Meaning Asserted—Indicates that a burst transfer is in progress.

Negated—Indicates that a burst transfer is not in progress.

Timing Comments Assertion/Negation—The same as A0–A31.

High-impedance—The same as A0–A31.

2.2.1.9 Address Acknowledge ($\overline{\text{AACK}}$)

The address acknowledge ($\overline{\text{AACK}}$) signal is an input/output on the MPC105.

2.2.1.9.1 Address Acknowledge ($\overline{\text{AACK}}$)—Output

Following are the state meaning and timing comments for $\overline{\text{AACK}}$ as an output signal.

State Meaning Asserted—Indicates that the address tenure of a transaction is terminated. On the cycle following the assertion of $\overline{\text{AACK}}$, the bus master releases the address-tenure-related signals to the high-impedance state and samples $\overline{\text{ARTRY}}$.

Negated—Indicates that the address tenure must remain active, and the address tenure related signals driven.

Timing Comments Assertion—Occurs a programmable number of clocks after $\overline{\text{TS}}$ or whenever $\overline{\text{ARTRY}}$ conditions are resolved.

Negation—Occurs one clock after assertion.

2.2.1.9.2 Address Acknowledge ($\overline{\text{AACK}}$)—Input

Following are the state meaning and timing comments for $\overline{\text{AACK}}$ as an input signal.

State Meaning Asserted—Indicates that a 60x bus slave is terminating the address tenure. On the cycle following the assertion of $\overline{\text{AACK}}$, the bus master releases the address tenure related signals to the high-impedance state and samples $\overline{\text{ARTRY}}$.

Negated—Indicates that the address tenure must remain active, and the address tenure related signals driven.

Timing Comments Assertion—Occurs during the 60x bus slave access, at least two clocks after $\overline{\text{TS}}$.

Negation—Occurs one clock after assertion.

2.2.1.10 Address Retry ($\overline{\text{ARTRY}}$)

The address retry ($\overline{\text{ARTRY}}$) signal is both an input and output signal on the MPC105.

2.2.1.10.1 Address Retry ($\overline{\text{ARTRY}}$)—Output

Following are the state meaning and timing comments for $\overline{\text{ARTRY}}$ as an output signal.

State Meaning Asserted—Indicates that the initiating 60x bus master must retry the current address tenure.

Negated/High-impedance—Indicates that the MPC105 does not require the address tenure to be retried.

Timing Comments Assertion—Occurs one clock after the assertion of $\overline{\text{AACK}}$.

Negation—Occurs in the second clock cycle after the assertion of $\overline{\text{AACK}}$.

2.2.1.10.2 Address Retry ($\overline{\text{ARTRY}}$)—Input

Following are the state meaning and timing comments for $\overline{\text{ARTRY}}$ as an input signal.

State Meaning Asserted—During a snoop operation, indicates that the 60x either requires the current address tenure to be retried due to a pipeline collision or needs to perform a snoop copy-back.

During normal 60x bus cycles in a multiprocessor system, indicates that the other 60x requires the address tenure to be retried.

Negated/High-impedance—Indicates that the address tenure is not required to be retried.

Timing Comments Assertion—Occurs one clock after the assertion of $\overline{\text{AACK}}$. Note that $\overline{\text{ARTRY}}$ may be asserted early, but it is sampled one clock after the assertion of $\overline{\text{AACK}}$.

Negation—Occurs in the second clock cycle after the assertion of $\overline{\text{AACK}}$.

2.2.1.11 Data Bus Grant 0 ($\overline{\text{DBG0}}$)—Output

The data bus grant 0 ($\overline{\text{DBG0}}$) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{DBG0}}$ signal.

State Meaning Asserted—Indicates that the 60x may, with the proper qualification, assume mastership of the data bus. A qualified data bus grant is defined as the assertion of $\overline{\text{DBG0}}$, negation of $\overline{\text{DBB}}$, and negation of $\overline{\text{ARTRY}}$. The requirement for the $\overline{\text{ARTRY}}$ signal is only for the address bus tenure associated with the data bus tenure about to be granted (that is, not for another address tenure available because of address pipelining).

Negated—Indicates that the 60x is not granted mastership of the data bus.

Timing Comments Assertion—Occurs on the first clock in which the data bus is not busy and the processor has the highest priority outstanding data transaction.

Negation—Occurs one clock after assertion.

2.2.1.12 Data Bus (DH0–DH31, DL0–DL31)

The data bus (DH0–DH31, DL0–DL31) consists of 64 signals that are both input and output signals on the MPC105. Following are the state meanings for the DH0–DH31 and DL0–DL31 signals.

State Meaning The data bus is comprised of two halves—data bus high (DH) and data bus low (DL). See Table 2-1 for byte lane assignments.

Table 2-1. Data Bus Byte Lane Assignments

Data Bus Signals	Byte Lane
DH0–DH7	0
DH8–DH15	1
DH16–DH23	2
DH24–DH31	3
DL0–DL7	4
DL8–DL15	5
DL16–DL23	6
DL24–DL31	7

2.2.1.12.1 Data Bus (DH0–DH31, DL0–DL31)—Output

Following are the state meaning and timing comments for DH and DL as output signals.

State Meaning Asserted/Negated—Represents the value of data during the following transactions: a processor-read-from-PCI, a PCI-write-to-memory, or when the MPC105 flushes the L2 copy-back buffer.

Timing Comments Assertion/Negation—The DH and DL signals are valid one clock after the data bus is idle or any time thereafter.

High-impedance—Occurs on the clock cycle after the last assertion of $\overline{\text{TA}}$.

2.2.1.12.2 Data Bus (DH0–DH31, DL0–DL31)—Input

Following are the state meaning and timing comments for DH and DL as input signals.

State Meaning Asserted/Negated—Represents the state of data during the following transactions: a processor-write-to-PCI, a PCI-read-from-memory, or an L1 or L2 copy-back due to a snoop hit.

Timing Comments Assertion/Negation—For a processor cycle, DH and DL are valid one clock after the assertion of $\overline{DBG}$. For an L2 copy-back cycle, DH and DL are valid when $\overline{DOE}$ is active. For a PCI read from memory operation, DH and DL are valid at a time dependent on the memory interface configuration. Refer to Chapter 6, "Memory Interface," for more information.

High-impedance—Occurs on the clock cycle after the last assertion of $\overline{TA}$.

2.2.1.13 Write-Through ($\overline{WT}$)—Input/Output

The write-through ($\overline{WT}$) signal is both an input and output signal on the MPC105. Following are the state meaning and timing comments for the $\overline{WT}$ signal.

State Meaning Asserted—Indicates that an access is write-through.

Negated—Indicates that an access is write-back.

Timing Comments Assertion/Negation—The same as A0–A31.

High-impedance—The same as A0–A31.

2.2.1.14 Caching-Inhibited ($\overline{CI}$)—Input/Output

The caching-inhibited ($\overline{CI}$) signal is both an input and output signal on the MPC105. Following are the state meaning and timing comments for the $\overline{CI}$ signal.

State Meaning Asserted—Indicates that an access is caching-inhibited.

Negated—Indicates that an access is caching-allowed.

Timing Comments Assertion/Negation—The same as A0–A31.

High-impedance—The same as A0–A31.

2.2.1.15 Global ($\overline{GBL}$)—Input/Output

The global ($\overline{GBL}$) signal is both an input and output signal on the MPC105. Following are the state meaning and timing comments for the $\overline{GBL}$ signal.

State Meaning Asserted—Indicates that an access is global. Coherency needs to be enforced by hardware.

Negated—Indicates that an access is not global. Hardware-enforced coherency is not required.

Timing Comments Assertion/Negation—The same as A0–A31.

High-impedance—The same as A0–A31.

2.2.1.16 Transfer Acknowledge ($\overline{TA}$)

The transfer acknowledge ($\overline{TA}$) signal is both an input and output signal on the MPC105.

2.2.1.16.1 Transfer Acknowledge ($\overline{TA}$)—Output

Following are the state meaning and timing comments for $\overline{TA}$ as an output signal.

State Meaning Asserted—Indicates that the data has been latched for a write operation, or that the data is valid for a read operation, thus terminating the current data beat. If it is the last or only data beat, this also terminates the data tenure.

Negated—Indicates that the 60x must extend the current data beat (insert wait states) until data can be provided or accepted by the MPC105.

Timing Comments Assertion—Occurs on the clock in which the current data transfer can be completed.

Negation—Occurs after the clock cycle of the final (or only) data beat of the transfer. For a burst transfer, $\overline{TA}$ may be negated between beats to insert one or more wait states before the completion of the next beat.

2.2.1.16.2 Transfer Acknowledge ($\overline{TA}$)—Input

Following are the state meaning and timing comments for $\overline{TA}$ as an input signal.

State Meaning Asserted—Indicates that a 60x bus slave has latched data for a write operation, or is indicting the data is valid for a read operation. If it is the last (or only) data beat, the data tenure is terminated.

Negated—Indicates that the 60x bus master must extend the current data beat (insert wait states) until data can be provided or accepted by the 60x bus slave.

Timing Comments Assertion—Occurs during the 60x bus slave access, at least two clocks after $\overline{TS}$, when the data transfer can be completed.

Negation—Occurs after the clock cycle of the final (or only) data beat of the transfer. For a burst transfer, $\overline{TA}$ may be negated between beats to insert one or more wait states before the completion of the next beat.

2.2.1.17 Transfer Error Acknowledge ($\overline{TEA}$)—Output

The transfer error acknowledge ($\overline{TEA}$) signal is an output on the MPC105. Note that the $\overline{TEA}$ signal can be disabled by clearing the TEA_EN bit in processor interface configuration register 1 (PICR1). Following are the state meaning and timing comments for the $\overline{TEA}$ signal.

State Meaning Asserted—Indicates that a bus error has occurred. Assertion of $\overline{TEA}$ terminates the transaction in progress; that is, it is not necessary to assert $\overline{TA}$ because it will be ignored by the target processor. An unsupported memory transaction, such as a direct-store access or a graphics read or write, will cause the assertion of $\overline{TEA}$ (provided TEA is enabled).

Negated—Indicates that no bus error was detected.

Timing Comments Assertion—Occurs on the first clock after the bus error is detected.
Negation—Occurs one clock after assertion.

2.2.2 Secondary Cache/Processor Interface Signals

The MPC105 provides support for either a secondary lookaside L2 cache or a second 60x processor. The signals $\overline{\text{DIRTY\_IN/BR1}}$, $\overline{\text{DIRTY\_OUT/BG1}}$, and $\overline{\text{TOE/DBG1}}$ function differently depending on whether the MPC105 is controlling an L2 cache or a secondary 60x processor. Section 2.2.2.1, “Secondary (L2) Cache Signals,” describes the L2 cache configuration for these signals; Section 2.2.2.2, “Secondary Processor Signals,” describes the secondary 60x processor configuration for these signals.

2.2.2.1 Secondary (L2) Cache Signals

This section provides a brief description of the secondary (L2) cache interface signals. The L2 cache interface supports either burst SRAMs, or asynchronous SRAMs. Some of the L2 interface signals perform different functions depending on the SRAM configuration and whether the on-chip byte decode logic is enabled.

2.2.2.1.1 Address Strobe/Data Address Latch Enable ($\overline{\text{ADS}}$/DALE)—Output

The address strobe/data address latch enable ($\overline{\text{ADS}}$/DALE) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{ADS}}$/DALE signal.

State Meaning Asserted—For a burst SRAM configuration, causes the burst SRAM to latch the current address.
–or–
For an asynchronous SRAM configuration, keeps the external address latch transparent.

Negated—For a burst SRAM configuration, indicates that the burst SRAMs should use addresses from an internal counter.
–or–
For an asynchronous SRAM configuration, causes the external address latch to latch the current address.

Timing Comments Assertion—For a burst SRAM configuration, the MPC105 asserts $\overline{\text{ADS}}$/DALE during the 60x bus address phase. The MPC105 also asserts $\overline{\text{ADS}}$/DALE when a write cycle needs to be aborted.
–or–
For an asynchronous SRAM configuration, the MPC105 asserts $\overline{\text{ADS}}$/DALE when the data SRAM access starts.

Negation—For a burst SRAM configuration, the MPC105 negates $\overline{\text{ADS}}$/DALE until data access is completed.
–or–
For an asynchronous SRAM configuration, the MPC105 negates $\overline{\text{ADS}}$/DALE when the data SRAM access is completed.

2.2.2.1.2 Bus Address Advance/Burst Address 1 ($\overline{\text{BAA}}$/BA1)—Output

The bus address advance ($\overline{\text{BAA}}$/BA1) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{BAA}}$/BA1 signal.

State Meaning Asserted—For a burst SRAM configuration, indicates that the burst SRAMs should increment their internal addresses.
–or–
For an asynchronous SRAM configuration, indicates the least significant bit of the burst address.

Negated—Indicates no change to addresses.

Timing Comments Assertion/Negation—For a burst SRAM configuration, the MPC105 asserts $\overline{\text{BAA}}$/BA1 together with $\overline{\text{TA}}$ during a read access and one clock after $\overline{\text{TA}}$ during write cycles (to advance the burst address).
–or–
For an asynchronous SRAM configuration, the MPC105 may change the state of $\overline{\text{BAA}}$/BA1 after $\overline{\text{ADS}}$/DALE is negated.

2.2.2.1.3 Data RAM Output Enable ($\overline{\text{DOE}}$)—Output

The data RAM output enable ($\overline{\text{DOE}}$) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{DOE}}$ signal.

State Meaning Asserted—Indicates that the L2 data RAMs should drive the data bus.

Negated—Indicates that the L2 data RAM outputs should be released to the high-impedance state.

Timing Comments Assertion/Negation—See Chapter 5, “Secondary Cache Interface,” for more detailed timing information.

2.2.2.1.4 Data RAM Write Enable (FNR/$\overline{\text{DWE0}}$, $\overline{\text{DWE}}$/$\overline{\text{DWE1}}$, $\overline{\text{DWE2}}$, CKO/$\overline{\text{DWE3}}$, $\overline{\text{DWE4}}$–$\overline{\text{DWE6}}$, CKE/$\overline{\text{DWE7}}$)—Output

The data RAM write enable (FNR/$\overline{\text{DWE0}}$, $\overline{\text{DWE}}$/$\overline{\text{DWE1}}$, $\overline{\text{DWE2}}$, CKO/$\overline{\text{DWE3}}$, $\overline{\text{DWE4}}$–$\overline{\text{DWE6}}$, and CKE/$\overline{\text{DWE7}}$) signals are outputs on the MPC105. For brevity, when the MPC105 is in the on-chip byte decode mode, this manual will refer to the data RAM write enable signals as $\overline{\text{DWE0}}$–$\overline{\text{DWE7}}$, or simply $\overline{\text{DWE}}$*n*. Note that three of the $\overline{\text{DWE}}$*n* signals have multiple functions–FNR/$\overline{\text{DWE0}}$ also functions as the Flash/nonvolatile ROM configuration input signal, CKO/$\overline{\text{DWE3}}$ also functions as the test clock output, and CKE/$\overline{\text{DWE7}}$ also functions as the SDRAM clock enable output. Following are the state meaning and timing comments for the $\overline{\text{DWE}}$*n* signals.

State Meaning Asserted—For the external byte decode mode, $\overline{\text{DWE}}$/$\overline{\text{DWE1}}$ indicates that a write to the L2 data RAMs is in progress.
–or–
For the on-chip byte decode mode, $\overline{\text{DWE0}}$–$\overline{\text{DWE7}}$ function as the individual byte lane (0–7) write enables for the L2 data RAMs.

Negated—Indicates that no writes to the L2 data RAMs are in progress.

Timing Comments Assertion/Negation—See Chapter 5, “Secondary Cache Interface,” for more detailed timing information.

2.2.2.1.5 Hit ($\overline{\text{HIT}}$)—Input

The hit ($\overline{\text{HIT}}$) signal is an input on the MPC105. The polarity of the $\overline{\text{HIT}}$ signal is programmable by using the PICR2[CF_HIT_HIGH] parameter; see Section 3.2.7, “Processor Interface Configuration Registers,” for more information. Following are the state meaning and timing comments for the $\overline{\text{HIT}}$ signal.

State Meaning Asserted—Indicates that the L2 cache has detected a hit.

Negated—Indicates that the L2 cache has not detected a hit.

Note that the polarity of $\overline{\text{HIT}}$ is programmable.

Timing Comments Assertion/Negation—The $\overline{\text{HIT}}$ signal should be valid when the L2 hit delay after $\overline{\text{TS}}$ expires, and held valid until the end of the address phase. The L2 hit delay is programmable by using the PICR2[CF_L2_HIT_DELAY] parameter.

2.2.2.1.6 Tag Address Latch Enable/Burst Address 0 (TALE/BA0)—Output

The TALE/BA0 signal is an output on the MPC105. Following are the state meaning and timing comments for the TALE/BA0 signal.

State Meaning Asserted—For a burst SRAM configuration, deselects the L2 data RAM in early write mode (PICR2[CF_WMODE] = 0b11); see Section 5.3.2.4, “CF_WMODE,” for more information
–or–
For an asynchronous SRAM configuration, indicates the most significant bit of the burst address.

Negated—For a burst SRAM configuration, selects the L2 data RAM for early write mode.

Timing Comments Assertion/Negation—For a burst SRAM configuration, the MPC105 negates TALE/BA0 when $\overline{\text{TS}}$ goes active and asserts TALE/BA0 when $\overline{\text{AACK}}$ goes active. The MPC105 also asserts TALE/BA0 when aborting a write cycle in early write mode.
–or–
For an asynchronous SRAM configuration, the MPC105 may change the state of TALE/BA0 after $\overline{\text{ADS}}$/DALE is negated.

2.2.2.1.7 Tag Address Latch Output Enable ($\overline{\text{TALOE}}$)—Output

The $\overline{\text{TALOE}}$ signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{TALOE}}$ signal.

State Meaning Asserted—Indicates that the address latch should drive the high-order L2 local address bus for tag lookup or tag write.

Negated—Indicates that the address latch should be released to the high-impedance state.

Timing Comments Assertion/Negation—$\overline{\text{TALOE}}$ is negated one clock cycle before $\overline{\text{TOE/DBG1}}$ goes active and asserted one clock cycle after $\overline{\text{TOE/DBG1}}$ goes inactive after a tag read cycle. Otherwise, $\overline{\text{TALOE}}$ is normally asserted.

2.2.2.1.8 Tag Write Enable ($\overline{\text{TWE}}$)—Output

The tag write enable ($\overline{\text{TWE}}$) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{TWE}}$ signal.

State Meaning Asserted—Indicates that the L2 tag address, valid, and dirty bits should be updated.

Negated—Indicates that the L2 tag address, valid, and dirty bits do not need updating.

Timing Comments Assertion/Negation—The $\overline{\text{TWE}}$ signal is asserted for one clock cycle during tag write operations.

2.2.2.1.9 Tag Valid (TV)—Output

The tag valid (TV) signal is an output on the MPC105. The polarity of the TV signal is programmable by using the PICR2[CF_MOD_HIGH] parameter; see Section 3.2.7, "Processor Interface Configuration Registers," for more information. Following are the state meaning and timing comments for the TV signal.

State Meaning Asserted—Indicates that the current L2 cache line should be marked valid.

Negated—Indicates the current L2 cache line is invalid.

Note that the polarity of TV is programmable.

Timing Comments Assertion/Negation—The TV signal is valid when $\overline{\text{TWE}}$ is asserted to update the tag status. TV is held valid for one clock after $\overline{\text{TWE}}$ is negated. Otherwise, TV is normally driven active for tag lookup operations.

High-impedance—The TV signal is released to a high-impedance state when $\overline{\text{TALOE}}$ is negated.

2.2.2.1.10 Dirty In ($\overline{\text{DIRTY\_IN/BR1}}$)—Input

The dirty in ($\overline{\text{DIRTY\_IN/BR1}}$) signal is an input on the MPC105. The function of this signal when in the multiprocessor configuration is described in Section 2.2.2.1.1, "Bus Request 1 ($\overline{\text{DIRTY\_IN/BR1}}$)—Input." When used as an L2 cache signal, the polarity of the $\overline{\text{DIRTY\_IN/BR1}}$ signal is programmable by using the PICR2[CF_MOD_HIGH] parameter; see Section 3.2.7, "Processor Interface Configuration Registers," for more information. Following are the state meaning and timing comments for the $\overline{\text{DIRTY\_IN/BR1}}$ signal.

State Meaning Asserted—Indicates that the selected L2 cache line is modified.

Negated—Indicates that the selected L2 cache line is unmodified.

Timing Comments Assertion/Negation—The DIRTY_IN/BR1 signal is valid when the L2 hit delay after TS expires. The DIRTY_IN/BR1 signal is held valid until the end of the address phase.

2.2.2.1.11 Dirty Out (DIRTY_OUT/BG1)—Output

The dirty out (DIRTY_OUT/BG1) signal is an output on the MPC105. The function of this signal when in the multiprocessor configuration is described in Section 2.2.2.2.2, "Bus Grant 1 (DIRTY_OUT/BG1)—Output." When used as an L2 cache signal, the polarity of the DIRTY_OUT/BG1 signal is programmable by using the PICR2[CF_MOD_HIGH] parameter; see Section 3.2.7, "Processor Interface Configuration Registers," for more information. Following are the state meaning and timing comments for the DIRTY_OUT/BG1 signal.

State Meaning Asserted—Indicates that the L2 cache line should be marked modified.

Negated—Indicates that the L2 cache line should be marked unmodified.

Timing Comments Assertion/Negation—The DIRTY_OUT/BG1 signal is valid when TWE is asserted to indicate a new line status. The DIRTY_OUT/BG1 signal is held valid for one clock cycle after TWE is negated.

2.2.2.1.12 Tag Output Enable (TOE/DBG1)—Output

The tag output enable (TOE/DBG1) signal is an output on the MPC105. The function of this signal when in the multiprocessor configuration is described in Section 2.2.2.2.3, "Data Bus Grant 1 (TOE/DBG1)—Output." Following are the state meaning and timing comments for the TOE/DBG1 signal.

State Meaning Asserted—Indicates that the tag RAM should drive its indexed content onto the 60x address bus.

Negated—Indicates that the tag RAM output should be released to the high-impedance state.

Timing Comments Assertion/Negation—Asserted for two or three clock cycles for tag read operations during L2 copy-back cycles (depending on PICR2[CF_HOLD]); see Chapter 5, "Secondary Cache Interface," for more detailed timing information.

2.2.2.2 Secondary Processor Signals

When a secondary 60x processor is used instead of an L2 cache, three signals change their functions. This section provides a brief description of the secondary processor interface signals. Note that with the exception of bus request (BR*n*), bus grant (BG*n*), and data bus grant (DBG*n*), all of the 60x processor interface signals are connected to both processors in a multiprocessor system. See Section 4.1.2, "Multiprocessor System Configuration," for more information.

2.2.2.2.1 Bus Request 1 (DIRTY_IN/BR1)—Input

The bus request 1 (DIRTY_IN/BR1) signal is an input on the MPC105. The function of this signal when in the L2 cache configuration is described in Section 2.2.2.1.10, "Dirty In (DIRTY_IN/BR1)—Input." Following are the state meaning and timing comments for the DIRTY_IN/BR1 signal.

State Meaning Asserted—Indicates that the secondary processor requires mastership of the 60x bus for a transaction.

Negated—Indicates that the secondary processor does not require mastership of the bus.

Timing Comments Assertion—May occur when DIRTY_OUT/BG1 is negated and a bus transaction is needed by the secondary processor. This may occur even if the two possible pipeline accesses have already occurred.

Negation—Occurs for at least one bus cycle after an accepted, qualified bus grant, even if another transaction is pending on the secondary processor. It is also negated for at least one bus cycle when the assertion of ARTRY is detected on the 60x bus (except for assertions due to 60x snoop copy-backs).

2.2.2.2.2 Bus Grant 1 (DIRTY_OUT/BG1)—Output

The bus grant 1 (DIRTY_OUT/BG1) signal is an output on the MPC105. The function of this signal when in the L2 cache configuration is described in Section 2.2.2.1.11, "Dirty Out (DIRTY_OUT/BG1)—Output." Following are the state meaning and timing comments for the DIRTY_OUT/BG1 signal.

State Meaning Asserted—Indicates that the secondary processor may, with the proper qualification, begin a bus transaction and assume mastership of the address bus.

Negated—Indicates that the secondary processor is not granted mastership of the next address bus tenure.

Timing Comments Assertion—Occurs when DIRTY_IN/BR1 is the highest priority request that is asserted.

Negation—Occurs when other higher priority transactions are pending.

2.2.2.2.3 Data Bus Grant 1 (TOE/DBG1)—Output

The data bus grant 1 (TOE/DBG1) signal is an output on the MPC105. The function of this signal when in the L2 cache configuration is described in Section 2.2.2.1.12, "Tag Output Enable (TOE/DBG1)—Output." Following are the state meaning and timing comments for the TOE/DBG1 signal.

State Meaning Asserted—Indicates that the secondary processor may, with the proper qualification, assume mastership of the data bus. A qualified data bus grant is defined as the assertion of TOE/DBG1, negation of DBB, and negation of ARTRY. The ARTRY signal is only for the address bus tenure associated with the data bus tenure about to be granted (that is, not for another address tenure available because of address pipelining).

Negated— indicates that the secondary processor is not granted mastership of the data bus.

Timing Comments Assertion—Occurs one bus clock cycle before data bus is available, and when the secondary processor has the highest priority for an outstanding data transaction.

Negation—Occurs one clock after assertion.

2.2.3 Memory Interface Signals

This section provides a brief description of the memory interface signals on the MPC105. The memory interface supports either standard DRAMs or synchronous DRAMs (SDRAMs) and either standard ROMs or Flash ROMs. Some of the memory interface signals perform different functions depending on the RAM and ROM configurations.

2.2.3.1 Row Address Strobe/Command Select (RAS/CS0–RAS/CS7)—Output

The eight row address strobe/command select (RAS/CS0–RAS/CS7) signals are output on the MPC105. Following are the state meaning and timing comments for the RAS/CS*n* output signals.

State Meaning Asserted—Indicates that a DRAM row address is valid and selects one of the rows in the selected bank.
–or–
Selects an SDRAM bank to perform a memory operation.

Negated—Indicates DRAM precharge period.
–or–
Indicates no SDRAM action during the current cycle.

Timing Comments Assertion—The MPC105 asserts the RAS/CS*n* signal to begin a memory (DRAM or SDRAM) cycle. For DRAM, all other memory interface signal timings are referenced to RAS/CS*n*. For SDRAM, RAS/CS*n* must be valid on the rising edge of the 60x bus clock.

2.2.3.2 Column Address Strobe/Data Qualifier (CAS/DQM0–CAS/DQM7)—Output

The eight column address strobe/data qualifier (CAS/DQM0–CAS/DQM7) signals are outputs on the MPC105. Following are the state meaning and timing comments for the CAS/DQM*n* output signals.

State Meaning Asserted—Indicates that a DRAM column address is valid and selects one of the columns in the row. (Note that the $\overline{\text{CAS}}$/DQM*n* signals are active low for DRAM.)
–or–
Prevents writing to SDRAM. (Note that the $\overline{\text{CAS}}$/DQM*n* signals are active high for SDRAM.)

Negated—Indicates that the current DRAM data transfer has completed.
–or–
Allows a read or write operation to SDRAM.

$\overline{\text{CAS}}$/DQM0 connects to the most significant byte select.
$\overline{\text{CAS}}$/DQM7 connects to the least significant byte select.

Timing Comments Assertion—For DRAM, the MPC105 asserts $\overline{\text{CAS}}$/DQM*n* two to eight cycles after $\overline{\text{RAS}}$/$\overline{\text{CS}}$*n* (depending on the setting of the MCCR3[RCD_2] parameter). See Section 6.3.3, "DRAM Interface Timing," for more information. For SDRAM, $\overline{\text{CAS}}$/DQM*n* must be valid on the rising edge of the 60x bus clock during read or write cycles.

2.2.3.3 Write Enable ($\overline{\text{WE}}$)—Output

The write enable ($\overline{\text{WE}}$) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{WE}}$ output signal.

State Meaning Asserted—Enables writing to DRAM or Flash ROM.
–or–
Part of SDRAM command encoding. See Section 6.4, "SDRAM Interface Operation," for more information.

Negated—No DRAM or Flash ROM write operation is pending.
–or–
Part of SDRAM command encoding.

Timing Comments Assertion—For DRAM, the MPC105 asserts $\overline{\text{WE}}$ concurrent with the column address and prior to $\overline{\text{CAS}}$/DQM*n*. For SDRAM, the MPC105 asserts $\overline{\text{WE}}$ concurrent with $\overline{\text{SDCAS}}$/$\overline{\text{ELE}}$ for write operations.

2.2.3.4 Memory Address/ROM Address (MA0–MA11/AR8–AR19)—Output

The memory address/ROM address (MA0–MA11/AR8–AR19) signals consist of 12 output signals on the MPC105. Following are the state meaning and timing comments for the MA0–MA11/AR8–AR19 output signals.

State Meaning Asserted/Negated—Represents the row/column multiplexed physical address for DRAMs or SDRAMs (MA0 is the most significant address bit; MA11 is the least significant address bit).
–or–
Represents bits 8–19 of the ROM or Flash ROM address (the 12 lowest-order bits, with AR19 as the lsb). Bits 0–7 of the ROM address are provided by PAR0–PAR7/AR0–AR7.

Timing Comments Assertion—For DRAM, the row address is valid on assertion of $\overline{\text{RAS}}$/$\overline{\text{CS}}$*n*, and the column address is valid on assertion of $\overline{\text{CAS}}$/DQM*n*. For SDRAM, the row address is valid on the rising edge of the 60x bus clock when $\overline{\text{SDRAS}}$ is asserted, and the column address is valid on the rising edge of the 60x bus clock when $\overline{\text{SDCAS}}$/$\overline{\text{ELE}}$ is asserted. For ROM, the address is valid on assertion of either $\overline{\text{RSC0}}$ or $\overline{\text{FOE}}$/$\overline{\text{RSC1}}$. For Flash ROM the address is valid on assertion of $\overline{\text{RCS0}}$.

2.2.3.5 Memory Parity/ROM Address (PAR0–PAR7/AR0–AR7)

The eight memory parity/ROM address (PAR0–PAR7/AR0–AR7) signals are both input and output signals for the parity function, but are output signals only for the ROM address function.

2.2.3.5.1 Memory Parity/ROM Address (PAR0–PAR7/AR0–AR7)—Output

Following are the state meaning and timing comments for PAR0–PAR7 as output signals.

State Meaning Asserted/Negated—Represents the byte parity being written to memory (PAR0 is the most significant parity bit and corresponds to byte lane 0 which is selected by $\overline{\text{CAS}}$/DQM0). Asserted or negated as appropriate to provide odd parity (including the parity bit).
–or–
Represents bits 0–7 of the ROM or Flash ROM address (the eight highest-order bits, with AR0 as the msb). Bits 8–19 of the ROM address are provided by MA0-MA11/AR8–AR19.

Timing Comments Assertion/Negation—For DRAMs or SDRAMs, PAR0–PAR7 are valid concurrent with DH0–DH31 and DL0–DL31. For ROMs or Flash ROMs, AR0–AR7 are valid concurrent with AR8–AR19.

2.2.3.5.2 Memory Parity (PAR0–PAR7/AR0–AR7)—Input

Following are the state meaning and timing comments for PAR0–PAR7/AR0–AR7 as input signals.

State Meaning Asserted/Negated—Represents the byte parity being read from memory (PAR0 is the most significant parity bit and corresponds to byte lane 0 which is selected by $\overline{\text{CAS}}$/DQM0).

Timing Comments Assertion/Negation—For DRAMs or SDRAMs, PAR0–PAR7 are valid concurrent with DH0–DH31 and DL0–DL31.

2.2.3.6 Memory Clock Enable (CKE/$\overline{\text{DWE7}}$)—Output

The memory clock enable (CKE/$\overline{\text{DWE7}}$) signal is an output on the MPC105. Following are the state meaning and timing comments for the CKE/$\overline{\text{DWE7}}$ output signal.

State Meaning Asserted—Enables the internal clock circuit of the SDRAM memory chip. Also, CKE/$\overline{\text{DWE7}}$ is part of the SDRAM command encoding.

Negated—Disables the internal clock circuit of the SDRAM memory chip. Also, CKE/$\overline{\text{DWE7}}$ is part of the SDRAM command encoding. Note that the MPC105 negates CKE/$\overline{\text{DWE7}}$ during certain system power-down situations.

Timing Comments Assertion—CKE/$\overline{\text{DWE7}}$ is valid on the rising edge of the 60x bus clock. See Section 6.4, “SDRAM Interface Operation,” for more information.

2.2.3.7 SDRAM Row Address Strobe ($\overline{\text{SDRAS}}$)—Output

The SDRAM row address strobe ($\overline{\text{SDRAS}}$) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{SDRAS}}$ output signal.

State Meaning Asserted/Negated—$\overline{\text{SDRAS}}$ is part of the SDRAM command encoding and is used for SDRAM bank selection during read or write operations. See Section 6.4, “SDRAM Interface Operation,” for more information.

Timing Comments Assertion—$\overline{\text{SDRAS}}$ is valid on the rising edge of the 60x bus clock when a $\overline{\text{RAS/CS}}$*n* signal is asserted.

2.2.3.8 SDRAM Column Address Strobe/External Latch Enable ($\overline{\text{SDCAS/ELE}}$)—Output

The SDRAM column address strobe/external latch enable ($\overline{\text{SDCAS/ELE}}$) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{SDCAS/ELE}}$ output signal.

State Meaning Asserted—$\overline{\text{SDCAS/ELE}}$ is part of the SDRAM command encoding and is used for SDRAM column selection during read or write operations. See Section 6.4, “SDRAM Interface Operation,” for more information.

–or–

For DRAM, $\overline{\text{SDCAS/ELE}}$ enables an external latched data buffer for read operations, if such a buffer is used in the system.

Negated—$\overline{\text{SDCAS/ELE}}$ is part of SDRAM command encoding used for SDRAM column selection during read or write operations.

–or–

For DRAM, $\overline{\text{SDCAS/ELE}}$ disables the external latched data buffer, if such a buffer is used in the system.

Timing Comments Assertion—For SDRAM, $\overline{\text{SDCAS/ELE}}$ is valid on the rising edge of the 60x bus clock when a $\overline{\text{RAS/CS}}n$ signal is asserted. For systems that use an external data buffer, $\overline{\text{SDCAS/ELE}}$ follows CAS timing for read operations, and follows $\overline{\text{RCS0}}$ and $\overline{\text{FOE/RCS1}}$ for ROM/Flash ROM read operations.

2.2.3.9 ROM Bank 0 Select ($\overline{\text{RCS0}}$)—Output

The ROM bank 0 select ($\overline{\text{RCS0}}$) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{RCS0}}$ output signal.

State Meaning Asserted—Selects the first ROM bank for a read access or Flash ROM for a read or write access.

Negated—Deselects bank 0, indicating no pending memory access to ROM or Flash ROM.

Timing Comments Assertion—The MPC105 asserts $\overline{\text{RCS0}}$ at the start of a ROM/Flash ROM access cycle.

2.2.3.10 Flash ROM Output Enable/ROM Bank 1 Select ($\overline{\text{FOE/RCS1}}$)—Output

The Flash ROM output enable/ROM bank 1 select ($\overline{\text{FOE/RCS1}}$) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{FOE/RCS1}}$ output signal. Note that the $\overline{\text{FOE/RCS1}}$ signal provides no indication of any write operation(s) to Flash ROM.

State Meaning Asserted—Enables Flash ROM output for the current read access.
–or–
Selects the second ROM bank for a read access.

Negated—Indicates that there is currently no read access to Flash ROM.
–or–
Indicates that there is currently no read access to ROM.

Timing Comments Assertion—The MPC105 asserts $\overline{\text{FOE/RCS1}}$ at the start of a ROM/Flash ROM access cycle.

2.2.3.11 Buffer Control ($\overline{\text{BCTL0}}$–$\overline{\text{BCTL1}}$)—Output

The two buffer control ($\overline{\text{BCTL0}}$–$\overline{\text{BCTL1}}$) signals are outputs on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{BCTL0}}$ and $\overline{\text{BCTL1}}$ output signals.

State Meaning Asserted/Negated—Used to control external data bus buffers (directional control and high-impedance state) between the 60x bus and memory). See Section 6.2, “Memory Interface Signal Buffering,” for more information.

Note that data buffers may be optional for lightly loaded data buses, but buffers are required whenever an L2 cache and ROM/Flash ROM (on the 60x processor/memory bus) are both in the system.

Timing Comments Assertion/Negation—Valid during data transfers (write or read) to or from memory.

2.2.3.12 Real Time Clock (RTC)—Input

The real time clock (RTC) signal is an input on the MPC105. Following are the state meaning and timing comments for the RTC input signal.

State Meaning Asserted/Negated—RTC is an external clock source for the memory refresh logic when the MPC105 is in the suspend power-saving mode.

Timing Comments Assertion—The maximum period of RTC is 1/4 of the refresh interval of the DRAM. For example, the minimum frequency for RTC when using DRAMs with a 125 µs refresh interval would be 32 kHz.

2.2.4 PCI Interface Signals

This section provides descriptions of the PCI interface signals on the MPC105. Note that throughout this manual, signals and bits of the PCI interface are referenced in little-endian format.

2.2.4.1 PCI Address/Data Bus (AD31–AD0)

The PCI address/data bus (AD31–AD0) consists of 32 signals that are both input and output signals on the MPC105.

2.2.4.1.1 Address/Data (AD31–AD0)—Output

Following is the state meaning for AD31–AD0 as output signals.

State Meaning Asserted/Negated—Represents the physical address during the address phase of a PCI transaction. During the data phase(s) of a PCI transaction, AD31–AD0 contain data being written.

The AD7–AD0 signals define the least significant byte and AD31–AD24 the most significant byte.

2.2.4.1.2 Address/Data (AD31–AD0)—Input

Following is the state meaning for AD31–AD0 as input signals.

State Meaning Asserted/Negated—Represents the address to be decoded as check for device select during the address phase of a PCI transaction or data being received during the data phase(s) of a PCI transaction.

2.2.4.2 Command/Byte Enables ($\overline{\text{C/BE3}}$–$\overline{\text{C/BE0}}$)

The four command/byte enable ($\overline{\text{C/BE3}}$–$\overline{\text{C/BE0}}$) signals are both input and output signals on the MPC105.

2.2.4.2.1 Command/Byte Enables ($\overline{C/BE3}$–$\overline{C/BE0}$)—Output

Following is the state meaning for $\overline{C/BE3}$–$\overline{C/BE0}$ as output signals.

State Meaning Asserted—During the address phase, $\overline{C/BE3}$–$\overline{C/BE0}$ define the bus command. See Section 7.3.2, “PCI Bus Commands,” for more information. During the data phase, $\overline{C/BE3}$–$\overline{C/BE0}$ are used as byte enables. Byte enables determine which byte lanes carry meaningful data. The $\overline{BE0}$ signal applies to the least significant byte.

2.2.4.2.2 Command/Byte Enables ($\overline{C/BE3}$–$\overline{C/BE0}$)—Input

Following is the state meaning for $\overline{C/BE3}$–$\overline{C/BE0}$ as input signals.

State Meaning Asserted—Indicates the command that another master is running, or which byte lanes are valid.

2.2.4.3 Parity (PAR)

The PCI parity (PAR) signal is both an input and output signal on the MPC105.

2.2.4.3.1 Parity (PAR)—Output

Following is the state meaning for PAR as an output signal.

State Meaning Asserted—Indicates odd parity across the AD31–AD0 and $\overline{C/BE3}$–$\overline{C/BE0}$ signals during address and data phases.

Negated—Indicates even parity across the AD31–AD0 and $\overline{C/BE3}$–$\overline{C/BE0}$ signals during address and data phases.

2.2.4.3.2 Parity (PAR)—Input

Following is the state meaning for PAR as an input signal.

State Meaning Asserted—Indicates odd parity driven by another PCI master or the PCI target during read data phases.

Negated—Indicates even parity driven by another PCI master or the PCI target during read data phases.

2.2.4.4 Target Ready ($\overline{TRDY}$)

The target ready ($\overline{TRDY}$) signal is both an input and output signal on the MPC105.

2.2.4.4.1 Target Ready ($\overline{TRDY}$)—Output

Following is the state meaning for $\overline{TRDY}$ as an output signal.

State Meaning Asserted—Indicates that the MPC105, acting as a PCI target, can complete the current data phase of a PCI transaction. During a read, the MPC105 asserts $\overline{TRDY}$ to indicate that valid data is present on AD31–AD0. During a write, the MPC105 asserts $\overline{TRDY}$ to indicate that it is prepared to accept data.

Negated—Indicates that the PCI master needs to wait before the MPC105, acting as a PCI target, can complete the current data phase. During a read, the MPC105 negates $\overline{\text{TRDY}}$ to insert a wait cycle because it cannot provide valid data to the master. During a write, the MPC105 negates $\overline{\text{TRDY}}$ to insert a wait cycle because it cannot accept data from the master.

2.2.4.4.2 Target Ready ($\overline{\text{TRDY}}$)—Input

Following is the state meaning for $\overline{\text{TRDY}}$ as an input signal.

State Meaning Asserted—Indicates another PCI target is able to complete the current data phase of a transaction.

Negated—Indicates a wait cycle from another target.

2.2.4.5 Initializer Ready ($\overline{\text{IRDY}}$)

The initializer ready ($\overline{\text{IRDY}}$) signal is both an input and output signal on the MPC105.

2.2.4.5.1 Initializer Ready ($\overline{\text{IRDY}}$)—Output

Following is the state meaning for $\overline{\text{IRDY}}$ as an output signal.

State Meaning Asserted—Indicates that the MPC105, acting as a PCI master, can complete the current data phase of a PCI transaction. During a write, the MPC105 asserts $\overline{\text{IRDY}}$ to indicate that valid data is present on AD31–AD0. During a read, the MPC105 asserts $\overline{\text{IRDY}}$ to indicate that it is prepared to accept data.

Negated—Indicates that the PCI target needs to wait before the MPC105, acting as a PCI master, can complete the current data phase. During a write, the MPC105 negates $\overline{\text{IRDY}}$ to insert a wait cycle because it cannot provide valid data to the target. During a read, the MPC105 negates $\overline{\text{IRDY}}$ to insert a wait cycle because it cannot accept data from the target.

2.2.4.5.2 Initializer Ready ($\overline{\text{IRDY}}$)—Input

Following is the state meaning for $\overline{\text{IRDY}}$ as an input signal.

State Meaning Asserted—Indicates another PCI master is able to complete the current data phase of a transaction.

Negated—Indicates a wait cycle from another master.

2.2.4.6 Frame ($\overline{\text{FRAME}}$)

The frame ($\overline{\text{FRAME}}$) signal is both an input and output signal on the MPC105.

2.2.4.6.1 Frame ($\overline{\text{FRAME}}$)—Output

Following is the state meaning for $\overline{\text{FRAME}}$ as an output signal.

State Meaning Asserted—Indicates that the MPC105, acting as a PCI master, is initiating a bus transaction. While $\overline{\text{FRAME}}$ is asserted, data transfers may continue.

Negated—If $\overline{\text{IRDY}}$ is asserted, indicates that the PCI transaction is in the final data phase.

2.2.4.6.2 Frame ($\overline{\text{FRAME}}$)—Input

Following is the state meaning for $\overline{\text{FRAME}}$ as an input signal.

State Meaning Asserted—Indicates that another PCI master is initiating a bus transaction.

Negated—Indicates that the transaction is in the final data phase or that the bus is idle.

2.2.4.7 Stop ($\overline{\text{STOP}}$)

The stop ($\overline{\text{STOP}}$) signal is both an input and output signal on the MPC105.

2.2.4.7.1 Stop ($\overline{\text{STOP}}$)—Output

Following is the state meaning for $\overline{\text{STOP}}$ as an output signal.

State Meaning Asserted—Indicates that the MPC105, acting as a PCI target, is requesting that the bus master stop the current transaction.

Negated—Indicates that the current transaction can continue.

2.2.4.7.2 Stop ($\overline{\text{STOP}}$)—Input

Following is the state meaning for $\overline{\text{STOP}}$ as an input signal.

State Meaning Asserted—Indicates that a target is requesting that the MPC105, acting as the PCI master, stop the current transaction.

Negated—Indicates that the current transaction can continue.

2.2.4.8 Lock ($\overline{\text{LOCK}}$)—Input

The lock ($\overline{\text{LOCK}}$) signal is an input on the MPC105. See Section 7.2.1, “Exclusive Access,” for more information. Following is the state meaning for the $\overline{\text{LOCK}}$ input signal.

State Meaning Asserted—Indicates that a master is requesting exclusive access to memory, which may require multiple transactions to complete.

Negated—Indicates that a normal operation is occurring on the bus or an access to a locked target is occurring.

2.2.4.9 Device Select ($\overline{\text{DEVSEL}}$)

The device select ($\overline{\text{DEVSEL}}$) signal is both an input and output signal on the MPC105.

2.2.4.9.1 Device Select ($\overline{\text{DEVSEL}}$)—Output

Following is the state meaning for $\overline{\text{DEVSEL}}$ as an output signal.

State Meaning Asserted—Indicates that the MPC105 has decoded the address and is the target of the current access.

Negated—Indicates that the MPC105 has decoded the address and is not the target of the current access.

2.2.4.9.2 Device Select ($\overline{\text{DEVSEL}}$)—Input

Following is the state meaning for $\overline{\text{DEVSEL}}$ as an input signal.

State Meaning Asserted—Indicates that some PCI agent (other than the MPC105) has decoded its address as the target of the current access.

Negated—Indicates that no PCI agent has been selected.

2.2.4.10 PCI Bus Request ($\overline{\text{REQ}}$)—Output

The PCI bus request ($\overline{\text{REQ}}$) signal is an output signal on the MPC105. Following is the state meaning for the $\overline{\text{REQ}}$ output signal.

State Meaning Asserted—Indicates that the MPC105 is requesting control of the PCI bus. Note that $\overline{\text{REQ}}$ is a point-to-point signal. Every master has its own $\overline{\text{REQ}}$ signal.

Negated—Indicates that the MPC105 does not require use of the PCI bus. If the PCI bus grant ($\overline{\text{GNT}}$) signal is asserted before the MPC105 has a transaction to perform (that is, the MPC105 is parked), $\overline{\text{REQ}}$ will not be asserted when the MPC105 needs control of the PCI bus.

2.2.4.11 PCI Bus Grant ($\overline{\text{GNT}}$)—Input

The PCI bus grant ($\overline{\text{GNT}}$) signal is an input signal on the MPC105. Following is the state meaning for the $\overline{\text{GNT}}$ input signal.

State Meaning Asserted—Indicates that the MPC105 has been granted control of the PCI bus. Note that $\overline{\text{GNT}}$ is a point-to-point signal. Every master has its own $\overline{\text{GNT}}$ signal. If $\overline{\text{GNT}}$ is asserted before the MPC105 has a transaction to perform (that is, the MPC105 is parked), the MPC105 will drive AD31–AD0, $\overline{\text{C/BE3}}$–$\overline{\text{C/BE0}}$, and PAR to stable (but meaningless) states until they are needed for a legitimate transaction.

Negated—Indicates that the MPC105 has not been granted control of the PCI bus, and cannot run a PCI transaction.

2.2.4.12 Parity Error ($\overline{\text{PERR}}$)

The PCI parity error ($\overline{\text{PERR}}$) signal is both an input and output signal on the MPC105.

2.2.4.12.1 Parity Error ($\overline{\text{PERR}}$)—Output

Following is the state meaning for $\overline{\text{PERR}}$ as an output signal.

State Meaning Asserted—Indicates that the MPC105, acting as a PCI agent, detected a data parity error. (The PCI master drives $\overline{\text{PERR}}$ on reads; the PCI target drives $\overline{\text{PERR}}$ on writes)

Negated—Indicates no error.

2.2.4.12.2 Parity Error ($\overline{\text{PERR}}$)—Input

Following is the state meaning for $\overline{\text{PERR}}$ as an input signal.

State Meaning Asserted—Indicates that another PCI agent detected a data parity error while the MPC105 was sourcing data (the MPC105 was acting as the PCI master during a write, or was acting as the PCI target during a read).

Negated—Indicates no error.

2.2.4.13 System Error ($\overline{\text{SERR}}$)

The PCI system error ($\overline{\text{SERR}}$) signal is both an input and output signal on the MPC105.

2.2.4.13.1 System Error ($\overline{\text{SERR}}$)—Output

Following is the state meaning for $\overline{\text{SERR}}$ as an output signal.

State Meaning Asserted—Indicates that an address parity error or some other system error (where the result will be a catastrophic error) was detected.

Negated—Indicates no error.

2.2.4.13.2 System Error ($\overline{\text{SERR}}$)—Input

Following is the state meaning for $\overline{\text{SERR}}$ as an input signal.

State Meaning Asserted—Indicates that another target has determined a catastrophic error.

Negated—Indicates no error.

2.2.4.14 PCI Sideband Signals

The PCI specification loosely defines a sideband signal as any signal not part of the PCI specification that connects two or more PCI-compliant agents, and has meaning only to those agents. The MPC105 implements three PCI sideband signals—$\overline{\text{ISA\_MASTER}}$, $\overline{\text{FLSHREQ}}$, and $\overline{\text{MEMACK}}$.

2.2.4.14.1 ISA Master ($\overline{\text{ISA\_MASTER}}$)—Input

The ISA master ($\overline{\text{ISA\_MASTER}}$) signal is an input signal on the MPC105. Following is the state meaning for the $\overline{\text{ISA\_MASTER}}$ input signal.

State Meaning Asserted—Indicates that an ISA master is requesting system memory. This signal is an implied address bit 31 for ISA devices that cannot drive a full 32-bit address. Accordingly, when the MPC105 detects $\overline{\text{ISA\_MASTER}}$ asserted, it automatically asserts $\overline{\text{DEVSEL}}$. Note that due to the automatic assertion of $\overline{\text{DEVSEL}}$ when $\overline{\text{ISA\_MASTER}}$ is asserted, possible bus contention can occur if the current transaction is not truly intended for the MPC105 (or system memory behind it).

Negated—Indicates that no ISA master requires system memory.

2.2.4.14.2 Flush Request ($\overline{\text{FLSHREQ}}$)—Input

The flush request ($\overline{\text{FLSHREQ}}$) signal is an input signal on the MPC105. Following is the state meaning for the $\overline{\text{FLSHREQ}}$ input signal.

State Meaning Asserted—Indicates that a device needs to have the MPC105 flush all of its current operations.

Negated—Indicates normal operation for the MPC105.

2.2.4.14.3 Flush Acknowledge ($\overline{\text{MEMACK}}$)—Output

The flush acknowledge ($\overline{\text{MEMACK}}$) signal is an output signal on the MPC105. Following is the state meaning for the $\overline{\text{MEMACK}}$ output signal.

State Meaning Asserted—Indicates that the MPC105 has flushed all of its current operations and has blocked all 60x transfers except snoop copy-back operations. The MPC105 asserts $\overline{\text{MEMACK}}$ after the flush is complete, if $\overline{\text{FLSHREQ}}$ is asserted.

Negated—Indicates the MPC105 may still have operations in its queues. The MPC105 negates $\overline{\text{MEMACK}}$ two cycles after $\overline{\text{FLSHREQ}}$ is negated.

2.2.5 Interrupt, Clock, and Power Management Signals

The MPC105 coordinates a few miscellaneous signals across the memory bus, the PCI bus, and the 60x processor bus. These include interrupt, clocking, and power management signals. This section provides a brief description of these signals.

2.2.5.1 Nonmaskable Interrupt (NMI)—Input

The nonmaskable interrupt (NMI) signal is an input on the MPC105. Following are the state meaning and timing comments for the NMI input signal.

State Meaning Asserted—Indicates that the MPC105 should signal a machine check interrupt to the 60x processor.

Negated—No special meaning.

Timing Comments Assertion—NMI may occur at any time, asynchronous to SYSCLK.

Negation—Should not occur until after the interrupt is taken.

2.2.5.2 Hard Reset ($\overline{\text{HRST}}$)—Input

The hard reset ($\overline{\text{HRST}}$) signal is an input on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{HRST}}$ input signal.

State Meaning Asserted/Negated—Indicates that a complete hard reset must be initiated by the MPC105 (perform circuit initialization followed by a system reset interrupt). During assertion, all bidirectional signals are released to the high-impedance state and all output signals are either in a high-impedance or inactive state.

Negated—Indicates that normal operation should proceed.

Timing Comments Assertion—May occur at any time, asynchronous to SYSCLK.

Negation—May occur at any time after the minimum hard reset pulse width has been met.

2.2.5.3 Machine Check ($\overline{\text{MCP}}$)—Output

The machine check ($\overline{\text{MCP}}$) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{MCP}}$ output signal.

State Meaning Asserted—Indicates that the MPC105 detected an illegal transaction, a memory select error, or a parity error on a memory read cycle. Assertion of $\overline{\text{SERR}}$, $\overline{\text{PERR}}$, or NMI may also trigger $\overline{\text{MCP}}$.

Negated—Indicates that normal operation should proceed.

Timing Comments Assertion—Occurs synchronous to SYSCLK.

Negation—Occurs after all error flags have been cleared by software.

2.2.5.4 System Clock (SYSCLK)—Input

The system clock (SYSCLK) signal is an input on the MPC105. The SYSCLK signal sets the frequency of operation for the PCI bus, and provides a reference clock for the phase-locked loops in the MPC105. SYSCLK is used to synchronize bus operations. See Section 2.3, “Clocking,” for more information.

2.2.5.5 Test Clock (CK0/$\overline{\text{DWE3}}$)—Output

The test clock (CK0/$\overline{\text{DWE7}}$) signal is an output on the MPC105. This signal provides a means to test or monitor the internal PLL output or the bus clock frequency. The CK0/$\overline{\text{DWE7}}$ clock should be used for testing purposes only. It is not intended as a reference clock signal.

2.2.5.6 Quiesce Request ($\overline{\text{QREQ}}$)—Input

The quiesce request ($\overline{\text{QREQ}}$) signal is an input on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{QREQ}}$ input signal.

State Meaning Asserted—Indicates that a 60x processor is requesting that all bus activity involving snoop operations pause or terminate so that the 60x processor may enter a low-power state.

Negated—Indicates that a 60x processor is in the full-on state.

Timing Comments Assertion/Negation—A 60x processor can assert $\overline{\text{QREQ}}$ at any time, asynchronous to the 60x bus clock. The MPC105 synchronizes $\overline{\text{QREQ}}$ internally.

2.2.5.7 Quiesce Acknowledge ($\overline{\text{QACK}}$)—Output

The quiesce acknowledge ($\overline{\text{QACK}}$) signal is an output on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{QACK}}$ output signal.

State Meaning Asserted—Indicates that the MPC105 is in a low-power state. All bus activity that requires snooping has terminated, and that the 60x processor may enter a low-power state.

Negated—Indicates that the 60x processor should not enter a low-power state. The MPC105 is in full-on state with normal bus activity.

Timing Comments Assertion—The MPC105 can assert $\overline{\text{QACK}}$ any time, synchronous to the 60x bus clock when $\overline{\text{QREQ}}$ is asserted.

Negation—The MPC105 can negate $\overline{\text{QACK}}$ any time, synchronous to the 60x bus clock.

2.2.5.8 Suspend ($\overline{\text{SUSPEND}}$)—Input

The suspend ($\overline{\text{SUSPEND}}$) signal is an input on the MPC105. Following are the state meaning and timing comments for the $\overline{\text{SUSPEND}}$ input signal.

State Meaning Asserted—Activates the suspend power-saving mode.

Negated—Deactivates the suspend power-saving mode.

Timing Comments Assertion— The $\overline{\text{SUSPEND}}$ signal can be asserted at any time, asynchronous to the 60x bus clock. The MPC105 synchronizes $\overline{\text{SUSPEND}}$ internally.

Negation— The $\overline{\text{SUSPEND}}$ signal can be negated at any time, asynchronous to the 60x bus clock, as long as it meets the timing requirements for turning the PLL and external clock on and off when entering and exiting suspend mode.

2.2.6 IEEE 1149.1 Interface Signals

To facilitate system testing, the MPC105 provides a JTAG test port that complies with the IEEE 1149.1 boundary-scan specification. This section describes the JTAG test port signals.

2.2.6.1 JTAG Test Data Output (TDO)—Output

Following is the state meaning for the TDO output signal.

State Meaning Asserted/Negated—The contents of the selected internal instruction or data register are shifted out onto this signal on the falling edge of TCK. The TDO signal will remain in a high-impedance state except when scanning of data is in progress.

2.2.6.2 JTAG Test Data Input (TDI)—Input

Following is the state meaning for the TDI input signal.

State Meaning Asserted/Negated—The value presented on this signal on the rising edge of TCK is clocked into the selected JTAG test instruction or data register.

Note that this input contains an internal pull-up resistor to ensure that an unterminated input appears as a high signal level to the test logic.

2.2.6.3 JTAG Test Clock (TCK)—Input

The JTAG test clock (TCK) signal is an input on the MPC105. Following is the state meaning for the TCK input signal.

State Meaning Asserted/Negated—This input should be driven by a free-running clock signal with a 50% duty cycle. Input signals to the test access port (TAP) are clocked in on the rising edge of TCK. Changes to the TAP output signals occur on the falling edge of TCK. The test logic allows TCK to be stopped.

Note that this input contains an internal pull-up resistor to ensure that an unterminated input appears as a high signal level to the test logic.

2.2.6.4 JTAG Test Mode Select (TMS)—Input

The test mode select (TMS) signal is an input on the MPC105. Following is the state meaning for the TMS input signal.

State Meaning Asserted/Negated—This signal is decoded by the internal JTAG TAP controller to distinguish the primary operation of the test support circuitry.

Note that this input contains an internal pull-up resistor to ensure that an unterminated input appears as a high signal level to the test logic.

2.2.6.5 JTAG Test Reset ($\overline{\text{TRST}}$)—Input

The test reset ($\overline{\text{TRST}}$) signal is an input on the MPC105. Following is the state meaning for the $\overline{\text{TRST}}$ input signal.

State Meaning Asserted—This input causes asynchronous initialization of the internal JTAG test access port controller. During power-on reset, the system should assert $\overline{\text{TRST}}$ to reset the JTAG control logic.

Negated—Indicates normal operation.

Note that this input contains an internal pull-up resistor to ensure that an unterminated input appears as a high signal level to the test logic.

2.2.7 Configuration Signals

The MPC105 has several signals that are sampled during power-on reset to determine the configuration of the ROM, Flash ROM, and dynamic memory, the data-bus width, and the phased-locked loop clock mode. This section describes the signals sampled during power-on reset, and how they are configured. Weak pull-up or pull-down resistors should be used so as to not interfere with the normal operation of the signals.

2.2.7.1 Flash/Nonvolatile ROM (FNR/$\overline{\text{DWE0}}$)—Input

The Flash/nonvolatile ROM configuration signal uses FNR/$\overline{\text{DWE0}}$ as a configuration input. Following is the state meaning for the FNR/$\overline{\text{DWE0}}$ configuration signal.

State Meaning High—Configures the MPC105 for Flash (8-bit interface) ROM memory.

Low—Configures the MPC105 for (32- or 64- bit interface) ROM memory.

2.2.7.2 ROM Location ($\overline{\text{RCS0}}$)—Input

The ROM location configuration signal uses $\overline{\text{RCS0}}$ as a configuration input. Following is the state meaning for the $\overline{\text{RCS0}}$ configuration signal.

State Meaning High—Indicates that ROM is located on the 60x processor/memory data bus.

Low—Indicates that ROM is located on the PCI bus.

2.2.7.3 60x Data Bus Width (DL0)—Input

The 60x data bus width configuration signal uses DL0 as a configuration input. Following is the state meaning for the DL0 configuration signal.

State Meaning High—Configures the MPC105 for 64-bit processor/memory data bus width.

Low—Configures the MPC105 for 32-bit processor/memory data bus width.

2.2.7.4 Address Map ($\overline{\text{XATS}}$)—Input

The address map configuration signal uses $\overline{\text{XATS}}$ as a configuration input. Following is the state meaning for the $\overline{\text{XATS}}$ configuration signal.

State Meaning High—Configures the MPC105 for address map A.

Low—Configures the MPC105 for address map B.

See Section 3.1, “Address Maps,” for more information.

2.2.7.5 Clock Mode (PLL0–PLL3)—Input

The clock mode (PLL0–PLL3) configuration signals are dedicated inputs on the MPC105. Following is the state meaning for the PLL0–PLL3 configuration signals.

State Meaning High/Low—Configures the operation of the PLL and the internal clock (core) frequency. Settings are based on the desired PCI bus and core frequency of operation. See Section 2.3, “Clocking,” for more information.

2.3 Clocking

The MPC105 requires a single system clock input, SYSCLK. The SYSCLK frequency dictates the frequency of operation for the PCI bus. An internal PLL on the MPC105 generates a master clock that is used for all of the internal (core) logic. The master clock provides the core frequency reference and is phase-locked to the SYSCLK input. The 60x processor, L2 cache, and memory interfaces operate at the core frequency.

The internal PLL on the MPC105 generates a core frequency either equal to the SYSCLK frequency (x1) or twice the frequency (x2) of SYSCLK (see Figure 2-2) depending on the clock mode configuration signals (PLL0–PLL3). The core frequency is phase-locked to the rising edge of SYSCLK. Note that SYSCLK is not required to have a 50% duty cycle.

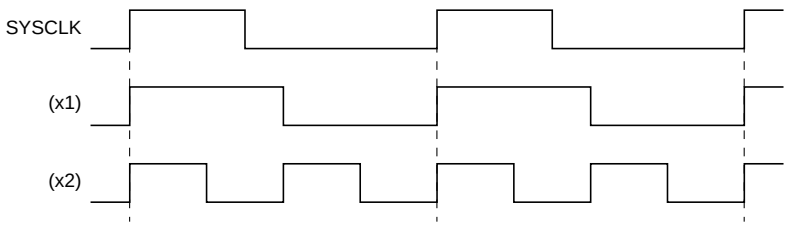


Figure 2-2. SYSCLK Input with Internal Multiples

The PLL is configured by the PLL0–PLL3 signals. For a given SYSCLK (PCI bus) frequency, the clock mode configuration signals (PLL0–PLL3) set the core frequency (and the frequency of the VCO controlling the PLL lock).

Table 2-2 summarizes the encodings for the PLL configuration signals.

Table 2-2. PLL Configuration

PLL0–PLL1 [1]	PLL2–PLL3 [2]	Core:SYSCLK Ratio [3]	VCO Multiplier [3]
0b00	—	1:1	—
0b01	—	2:1	—
—	0b00	—	x2
—	0b01	—	x4
—	0b10	—	x8
0b00	0b11	PLL Bypass[4]	
0b11	0b11	Clock Off[5]	

Notes: 1. PLL0–PLL1 select the core/SYSCLK ratio (1:1 or 2:1).
2. PLL2–PLL3 select the core/VCO multiplier (x2, x4, or x8).
3. Some PLL configurations may select bus, core, or VCO frequencies which are not useful, not supported, or not tested for the MPC105. For complete information, see the MPC105 Hardware Specifications for timing comments.
4. In PLL-bypass mode, the SYSCLK input signal clocks the internal logic directly, and the bus is set for 1:1 mode operation. The PLL-bypass mode is for test only, and is not intended for functional use.
5. In clock-off mode, no clocking occurs inside the MPC105 regardless of the SYSCLK input.

Chapter 3
Device Programming

The programmable aspects of the MPC105 are primarily for use by initialization and error handling software. This section details the selectable address maps and the configuration registers on the MPC105.

3.1 Address Maps

The MPC105 provides for two address maps designated map A and map B. The memory map is selected by the $\overline{\text{XATS}}$ input at power-up. Map A is selected by using a pull-up resistor ($\overline{\text{XATS}}$ = 1); map B is selected by using a pull-down resistor ($\overline{\text{XATS}}$ = 0). Note that $\overline{\text{XATS}}$ is not connected to the processor for map B. Bit 16 of the processor interface configuration register 1 (PICR1) indicates the state of $\overline{\text{XATS}}$ at power-up.

3.1.1 Address Map A

Address map A complies with the PowerPC reference platform specification. The address space of map A is divided into four areas—a system memory portion, a PCI I/O portion, a PCI memory portion, and a system ROM space. When configured for map A, the MPC105 translates addresses across the 60x and PCI buses as shown in Figure 3-1 through Figure 3-4.

Map A can be configured as contiguous or discontiguous by PICR1[XIO_MODE]. The contiguous map as seen from the processor is shown in Figure 3-1. The discontiguous map as seen from the processor is shown in Figure 3-2. The discontiguous map reserves a 32-byte block for each byte addressed on the PCI bus to aid in accessing PC-compatible I/O devices in the ISA/PCI I/O space address 0 to 64KB – 1.

Figure 3-3 and Figure 3-4 show the I/O and memory maps as seen from a PCI master. Table 3-1 shows an alternate view of map A.

Table 3-1. Address Map A—Alternate View

60x Processor Address Range				PCI Address Range	Definition
Hex		Decimal			
00000000	7FFFFFFF	0	2G – 1	No PCI cycle	System memory space
80000000	807FFFFF	2G	2G + 8M – 1	00000000–007FFFFF	ISA/PCI I/O space [1,2]
80800000	80FFFFFF	2G + 8M	2G + 16M – 1	00800000–00FFFFFF	Direct map PCI config[3]
81000000	BF7FFFFF	2G + 16M	3G – 8M – 1	01000000–3F7FFFFF	PCI I/O
BF800000	BFFFFFEF	3G – 8M	3G – 16 – 1	3F800000–3FFFFFEF	Reserved
BFFFFFF0	BFFFFFFF	3G – 16	3G – 1	3FFFFFF0–3FFFFFFF	PCI/ISA INTACK space[6]
C0000000	FEFFFFFF	3G	4G – 16M – 1	00000000–3EFFFFFF	PCI memory space
FF000000	FFFFFFFF	4G – 16M	4G – 1	No PCI cycle[4]	ROM space[4]

PCI I/O Transactions Address Range				60x Address Range	Definition
Hex		Decimal			
00000000	0000FFFF	0	64K – 1	No local memory cycle	ISA/PCI I/O
00010000	007FFFFF	64K	8M – 1	No local memory cycle	Reserved
00800000	3F7FFFFF	8M	1G – 8M – 1	No local memory cycle	PCI I/O space
3F800000	3FFFFFFF	1G - 8M	1G – 1	No local memory cycle	Reserved
40000000	FFFFFFF	1G	4G – 1	No local memory cycle	Reserved

PCI Memory Transactions Address Range				60x Address Range	Definition
Hex		Decimal			
00000000	00FFFFFF	0	16M – 1	No local memory cycle or ISA memory cycle[5]	ISA/PCI memory
01000000	7FFFFFFF	16M	2G – 1	No local memory cycle[5]	PCI memory
80000000	FFFFFFFF	2G	4G – 1	00000000–7FFFFFFF	System memory space

1. PCI configuration accesses to CF8 and CFC–CFF are handled as specified in the *PCI Local Bus Specification*. See Section 7.4.5, “Configuration Cycles,” for more information.
2. Processor addresses are translated to PCI addresses as follows:
   In contiguous mode:
   PCI address = 0x0 || proc_addr(1–31), PCI configuration accesses use processor addresses 80000CF8 and 80000CFC-80000CFF.
   In discontiguous mode:
   PCI address = 0x0000 || proc_addr(9–19) || proc_addr(27–31), PCI configuration accesses use processor addresses 80067018 and 8006701C-8006701F.
3. IDSEL for direct-access method: 11=0x808008xx, 12=0x808010xx, ..., 18=0x808400xx.
4. If the ROM is located on the PCI bus, these addresses are not reserved and PCI cycles will be

generated.

5. If the ISA_MASTER signal is asserted, the PCI memory cycle is forwarded to system memory and the 60x bus address becomes 0x00 || AD(29–0).
6. Reads to this address generate PCI interrupt-acknoledge cycles; writes to this address generate TEA (if enabled).

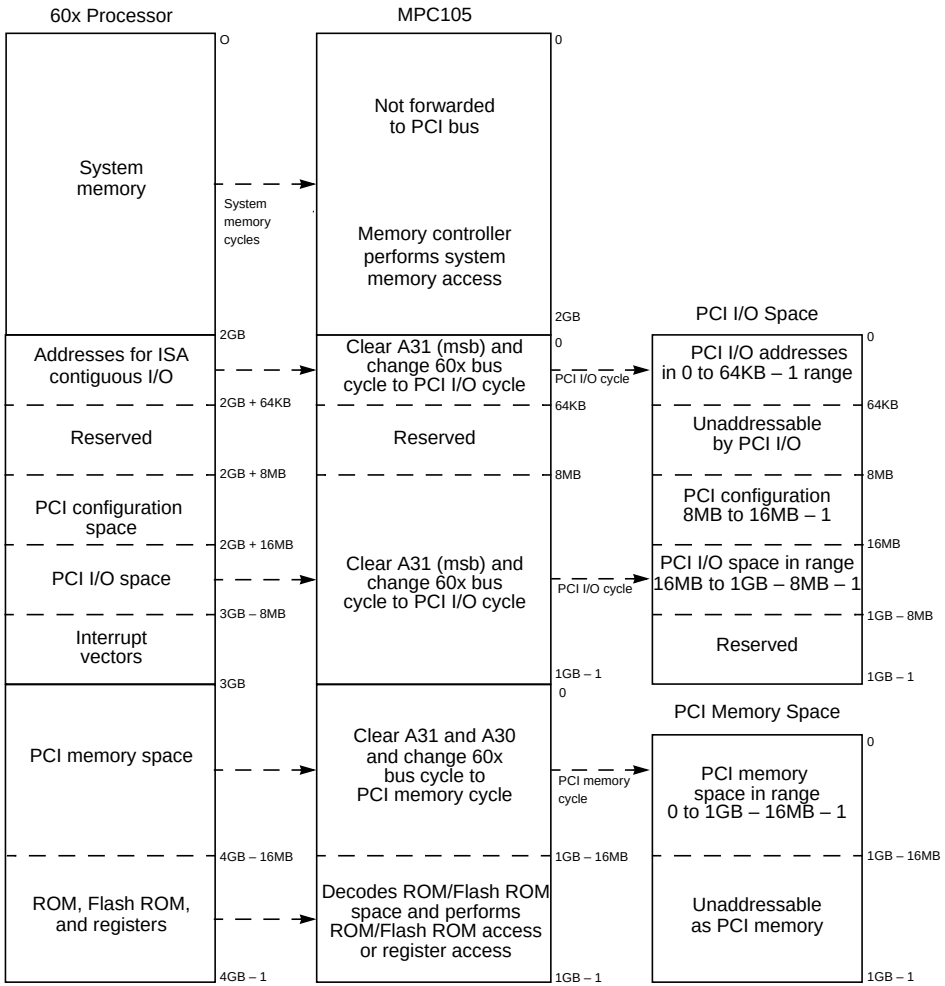


Figure 3-1. Address Map A (Contiguous Map)

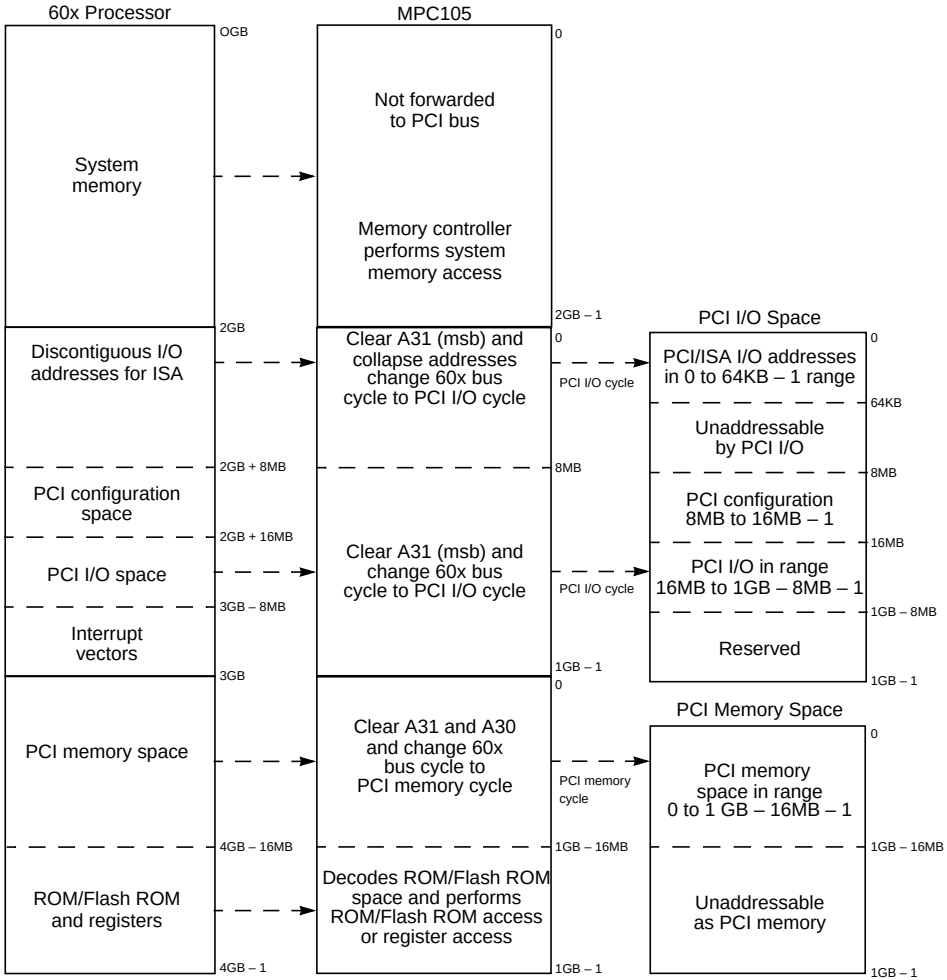


Figure 3-2. Address Map A (Discontiguous Map)

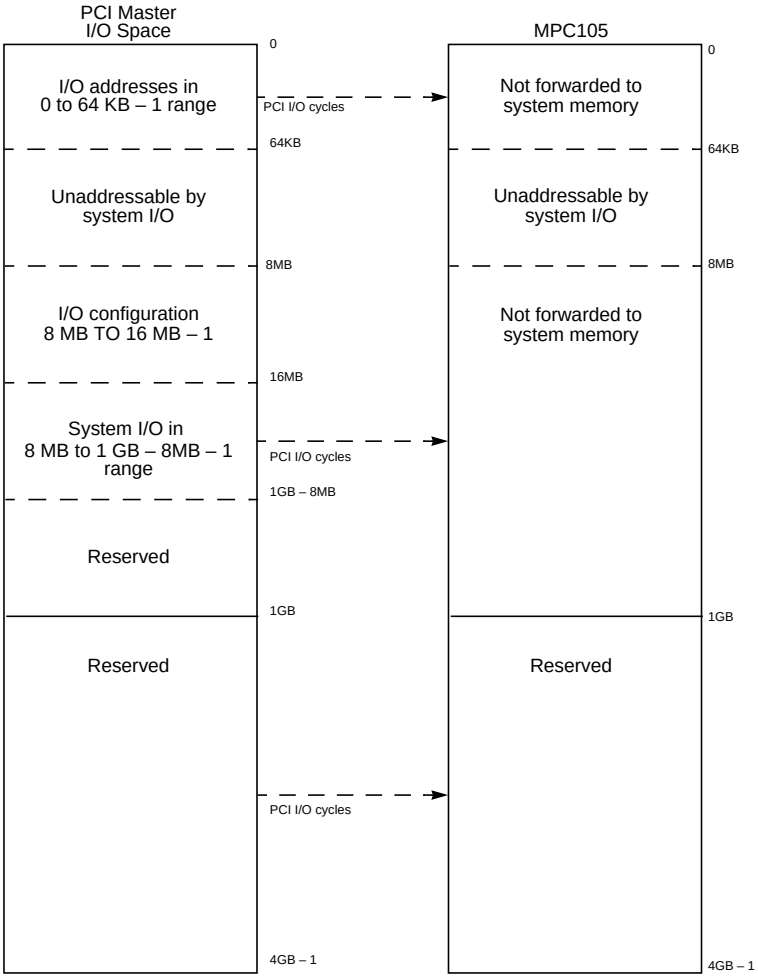


Figure 3-3. PCI I/O Map (Address Map A)

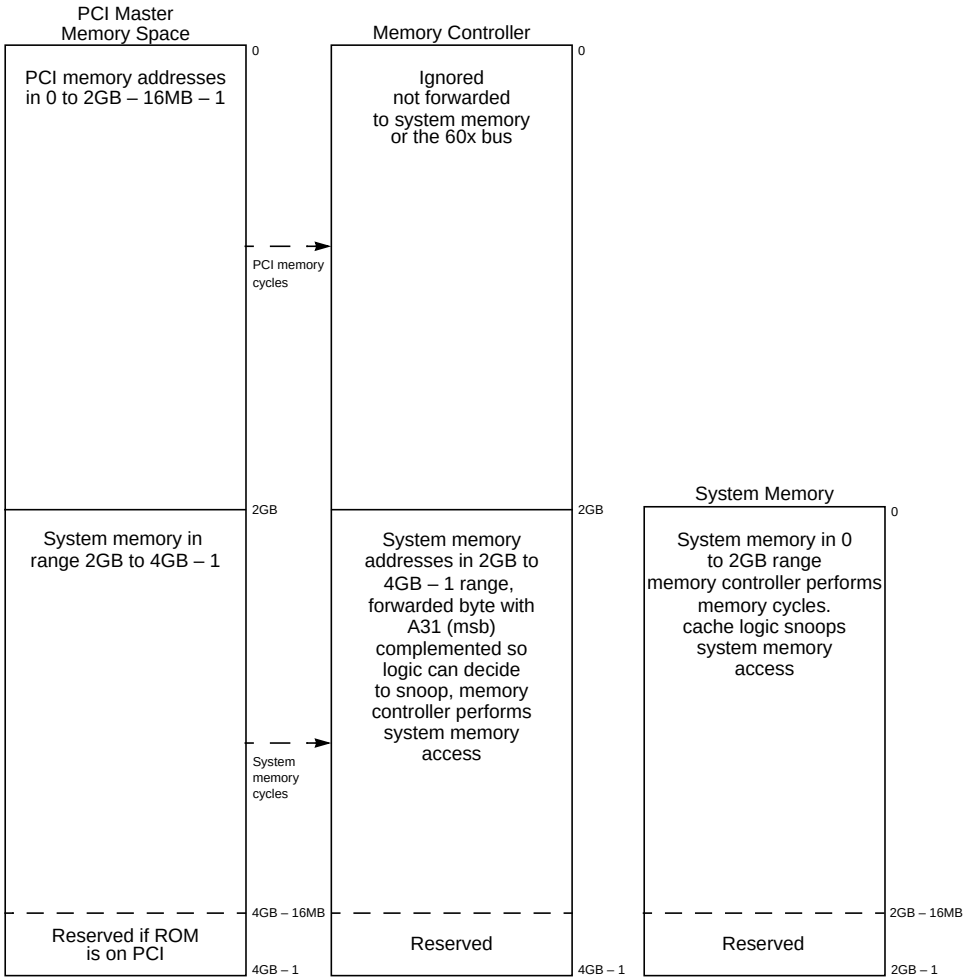


Figure 3-4. PCI Memory Map (Map A)

3.1.2 Address Map B

Address map B, shown in Figure 3-5, is the same as seen from the processor as it is from a PCI agent. That is, there is no translation by the MPC105 for the map B configuration (except for indirect configuration cycles). Address map B does not comply with the PowerPC reference platform specification. System memory occupies the space from 0 to 2G; PCI memory, I/O, and configuration space occupy the area from 2G to 4G-16M (with a few exceptions); the last 16M is the ROM space. Note that when using map B, the MPC105 will claim any PCI cycle with an address in the range from 0 to 2G by asserting $\overline{\text{DEVSEL}}$. Possible contention could occur if another PCI agent responds to addresses between 0 and 2G. Table 3-2 shows an alternate view of map B.

Table 3-2. Address Map B—Alternate View

Address Range	Definition
00000000–7FFFFFFF	Memory space
80000000–EFFFFFFF	PCI memory space
F0000000–F07FFFFF	PCI I/O space
F0800000–F0BFFFFF	PCI configuration space—address
F0C00000–F0DFFFFF	PCI configuration space—data
F0E00000–F0FFFFFF	PCI special-cycle/interrupt acknowledge[1]
F1000000–F1FFFFFF	PCI memory space
F2000000–F7FFFFFF	Reserved
F8000000–F8FFEFFF	PCI I/O space
F8FFF000–F8FFF0FF	MPC105 internal configuration space
F8FFF100–F8FFFFFF	PCI I/O space
F9000000–FEFFFFFF	PCI memory space
FF000000–FFFFFFFF	ROM space

[1] Reads from this address range generate PCI interrupt-acknowledge transactions Writes to this address range generate PCI special-cycle transactions.

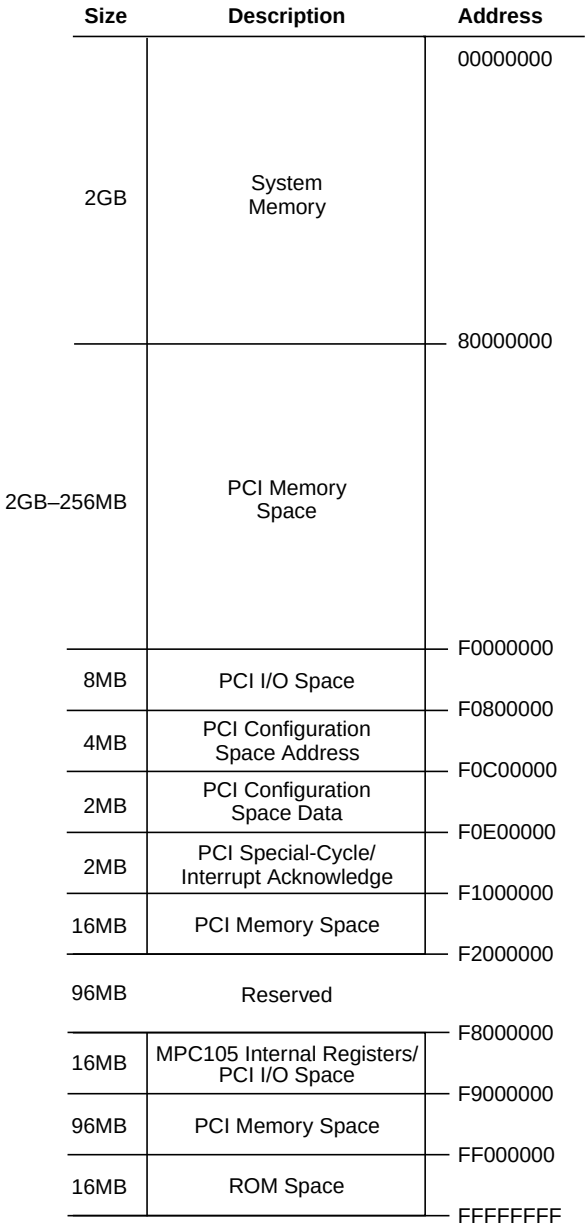


Figure 3-5. Address Map B

3.2 Configuration Registers

This section describes the programmable configuration registers of the MPC105. These registers are generally setup by initialization software following a power-on reset or hard reset, or by error handling routines. All the internal registers are intrinsically little-endian. In the following register descriptions, bit 0 is the least significant bit of the register.

Any reserved bits in the following register descriptions are not guaranteed to have predictable values. Software must preserve the values of reserved bits when writing to a configuration register. Also, when reading from a configuration register, software should not rely on the value of any reserved bit remaining consistent.

3.2.1 Configuration Register Access

When using address map A, the MPC105 configuration registers are accessed by an indirect method. The 32-bit register address (0x8000_00*nn*, where *nn* is the address offset of the desired configuration register—see Table 3-3 and Figure 3-6) is written to 0x8000_0CF8 (0x8006_7018 in discontiguous mode). Then, the data is accessed at addresses 0x8000_0CFC–0x8000_0CFF (0x8006_701C–0x8006_701F in discontiguous mode).

When using map B, the MPC105 configuration registers are accessed directly at address 0xF8FF_F0*nn*, where *nn* is the address offset of the desired configuration register (see Table 3-3 and Figure 3-6).

Certain configuration bits for the MPC105 can also be accessed at the addresses 0x8000_0092, 0x8000_081C and 0x8000_0850. These are compatible with the example system described by the *PowerPC Reference Platform Specification*. See Section 3.2.9, “External Configuration Registers,” for more information.

3.2.1.1 Configuration Register Access in Little-Endian Mode

In little-endian mode (both processor and the MPC105), the program should access the configuration registers using the above methods. The data appears in the 60x processor register in descending significance byte order (MSB to LSB) at the time it is stored to the MPC105. For the indirect-access method, the configuration register address in the processor register should appear (as data appears) in descending significance byte order (MSB to LSB) at the time it is stored to the MPC105.

Example: Map A configuration sequence, 4-byte data write to register at address offset 0xA8

```
Initial values:r0 contains 0x8000_00A8
               r1 contains 0x8000_0CF8
               r2 contains 0xAABB_CCDD
               Register at 0xA8 contains 0xFFFF_FFFF (AB to A8)
Code sequence: stw   r0,0(r1)
               stw   r2,4(r1)
```

```
Results:        Address 0x8000_0CF8 contains 0x8000_00A8 (MSB to LSB)
                Register at 0xA8 contains 0xAABB_CCDD (AB to A8)
```

Example: Map A configuration sequence, 2-byte data write to register at address offset 0xAA

```
Initial values:r0 contains 0x8000_00A8
               r1 contains 0x8000_0CF8
               r2 contains 0xAABB_CCDD
               Register at 0xA8 contains 0xFFFF_FFFF (AB to A8)
Code sequence: stw   r0,0(r1)
               sth   r2,6(r1)
Results:       Address 0x8000_0CF8 contains 0x8000_00A8 (MSB to LSB)
               Register at 0xA8 contains 0xCCDD_FFFF (AB to A8)
```

Note that in this example, the value 0x8000_00A8 is the configuration address register, not 0x8000_00AA. The address offset 0xAA is generated by using 0x8000_0CFE for the data access.

Example: Map A configuration sequence, 1-byte data read from register at address offset 0xA9

```
Initial values:r0 contains 0x8000_00A8
               r1 contains 0x8000_0CF8
               r2 contains 0xAABB_CCDD
               Register at 0xA8 contains 0xFFFF_FFFF (AB to A8)
Code sequence: stw   r0,0(r1)
               lbz   r2,5(r1)
Results:       Address 0x8000_0CF8 contains 0x8000_00A8 (MSB to LSB)
               r2 contains 0x0000_00FF
```

Example: Map B configuration sequence, 4-byte data write to register at address offset 0xA8

```
Initial values:r0 contains 0xAABB_CCDD
               r1 contains 0xF8FF_F0A8
               Register at 0xA8 contains 0xFFFF_FFFF (AB to A8)
Code sequence: stw   r0,0(r1)
Results:       Register at 0xA8 contains 0xAABB_CCDD (AB to A8)
```

Example: Map B configuration sequence, 1-byte data write to register at address offset 0xAB

```
Initial values:r0 contains 0xAABB_CCDD
               r1 contains 0xF8FF_F0A8
               Register at 0xA8 contains 0xFFFF_FFFF (AB to A8)
```

```
Code sequence: stb   r0,3(r1)
Results:       Register at 0xA8 contains 0xDDFF_FFFF (AB to A8)
```

3.2.1.2 Configuration Register Access in Big-Endian Mode

In big-endian mode (both the processor and the MPC105), software must byte-swap the data of the configuration register before performing an access. That is, the data appears in the processor register in ascending significance byte order (LSB to MSB). When using address map A (the indirect-access method), software loads the configuration register address and the configuration register data into the processor register in ascending significance byte order (LSB to MSB). When using address map B, software loads the configuration register address into the processor register in normal order (MSB to LSB); however, software loads the configuration register data in ascending significance byte order (LSB to MSB).

Note that in the following examples, the data in the configuration register (at 0xA8) is shown in little-endian order. This is because all the internal registers are intrinsically little-endian.

Example: Map A configuration sequence, 4-byte data write to register at address offset 0xA8

```
Initial values:r0 contains 0xA800_0080
               r1 contains 0x8000_0CF8
               r2 contains 0xDDCC_BBAA
               Register at 0xA8 contains 0xFFFF_FFFF (AB to A8)
Code sequence: stw   r0,0(r1)
               stw   r2,4(r1)
Results:       Address 0x8000_0CF8 contains 0x8000_00A8 (MSB to LSB)
               Register at 0xA8 contains 0xAABB_CCDD (AB to A8)
```

Example: Map B configuration sequence, 2-byte data write to register at address offset 0xAA

```
Initial values:r0 contains 0xDDCC_BBAA
               r1 contains 0xF8FF_F0A8
               Register at 0xA8 contains 0xFFFF_FFFF (AB to A8)
Code sequence: sth   r0,2(r1)
Results:       Register at 0xA8 contains 0xAABB_FFFF (AB to A8)
```

3.2.2 Configuration Register Summary

Table 3-3 summarizes the configuration registers provided by the MPC105. Figure 3-6 shows a map of the internal configuration space of the MPC105. Note that any configuration addresses not defined in Table 3-3 are reserved.

Table 3-3. MPC105 Configuration Registers

Address Offset	Register Size	Program Accessible Size	Register	Register Access Allowed	Reset Value
00	2 bytes	2 bytes	Vendor ID =1057h	Read	0x1057
02	2 bytes	2 bytes	Device ID = 0001h	Read	0x0001
04	2 bytes	2 bytes	PCI command	Read/write	0x0006
06	2 bytes	2 bytes	PCI status	Read/bit-reset	0x0080
08	1 byte	1 byte	Revision ID	Read	0x*nn*
09	1 byte	1 byte	Standard programming interface	Read	0x00
0A	1 byte	1 byte	Subclass code	Read	0x00
0B	1 byte	1 byte	Class code	Read	0x06
0C	1 byte	1 byte	Cache line size	Read	0x00
0D	1 byte	1 byte	Latency timer	Read	0x00
0E	1 byte	1 byte	Header type	Read	0x00
0F	1 byte	1 byte	BIST control	Read	0x00
3C	1 byte	1 byte	Interrupt line	Read	0x00
3D	1 byte	1 byte	Interrupt pin	Read	0x00
3E	1 byte	1 byte	MIN GNT	Read	0x00
3F	1 byte	1 byte	MAX LAT	Read	0x00
40	1 byte	1 byte	Bus number	Read	0x00
41	1 byte	1 byte	Subordinate bus number	Read	0x00
42	1 byte	1 byte	Disconnect counter	Read	0x00
44	2 bytes	2 bytes	Special-cycle address	Read	0x0000
70	2 bytes	1 or 2 bytes	Power management configuration	Read/write	0x00
80–83	4 bytes	1, 2, or 4 bytes	Memory starting address 1	Read/write	0x0000_0000
84–87	4 bytes	1, 2, or 4 bytes	Memory starting address 2	Read/write	0x0000_0000
88–8B	4 bytes	1, 2, or 4 bytes	Extended memory starting address 1	Read/write	0x0000_0000
8C–8F	4 bytes	1, 2, or 4 bytes	Extended memory starting address 2	Read/write	0x0000_0000
90–93	4 bytes	1, 2, or 4 bytes	Memory ending address 1	Read/write	0x0000_0000
94–97	4 bytes	1, 2, or 4 bytes	Memory ending address 2	Read/write	0x0000_0000

Table 3-3. MPC105 Configuration Registers (Continued)

Address Offset	Register Size	Program Accessible Size	Register	Register Access Allowed	Reset Value
98–9B	4 bytes	1, 2, or 4 bytes	Extended memory ending address 1	Read/write	0x0000_0000
9C–9F	4 bytes	1, 2, or 4 bytes	Extended memory ending address 2	Read/write	0x0000_0000
A0	1 byte	1 byte	Memory enable	Read/write	0x00
A8	4 bytes	1, 2, or 4 bytes	Processor interface configuration 1	Read/write	0xFF00_0010
AC	4 bytes	1, 2, or 4 bytes	Processor interface configuration 2	Read/write	0x000C_060C
BA	1 byte	1 byte	Alternate OS-visible parameters 1	Read/write	0x04
BB	1 byte	1 byte	Alternate OS-visible parameters 2	Read/write	0x00
C0	1 byte	1 byte	Error enabling 1	Read/write	0x01
C1	1 byte	1 byte	Error detection 1	Read/bit-reset	0x00
C3	1 byte	1 byte	60x bus error status	Read/bit-reset	0x00
C4	1 byte	1 byte	Error enabling 2	Read/write	0x00
C5	1 byte	1 byte	Error detection 2	Read/bit-reset	0x00
C7	1 byte	1 byte	PCI bus error status	Read/bit-reset	0x00
C8–CB	4 byte	4 bytes	60x/PCI error address	Read	0x00
F0	4 bytes	1, 2, or 4 bytes	Memory control configuration 1	Read/write	0xFF*n*2_0000
F4	4 bytes	1, 2, or 4 bytes	Memory control configuration 2	Read/write	0x0000_0003
F8	4 bytes	1, 2, or 4 bytes	Memory control configuration 3	Read/write	0x0000_0000
FC	4 bytes	1, 2, or 4 bytes	Memory control configuration 4	Read/write	0x0010_0000

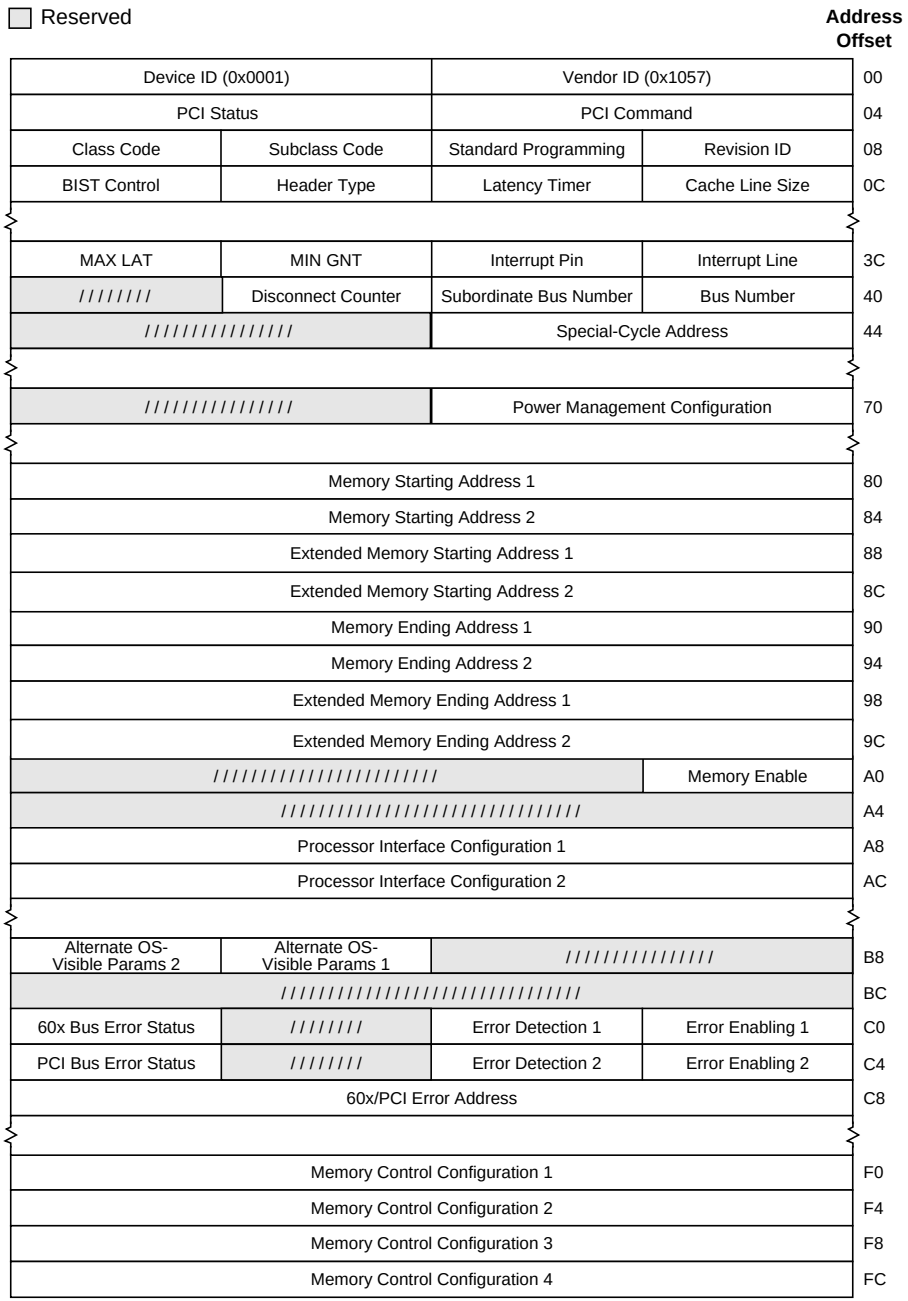


Figure 3-6. MPC105 Configuration Registers

3.2.3 PCI Registers

The *PCI Local Bus Specification* defines the configuration registers from 0x00 through 0x3F. Additionally, the host bridge architecture section of the *PCI System Design Guide* defines the bus number register (0x40), the subordinate bus number register (0x41), and the disconnect counter register (0x42).

With the exception of the PCI command, PCI status, and subordinate bus number registers, all of the PCI registers are read-only on the MPC105. Table 3-4 summarizes the PCI configuration registers of the MPC105. Detailed descriptions of these registers are provided in the *PCI Local Bus Specification*.

Table 3-4. PCI Configuration Space Header Summary

Address Offset	Register Name	Description
00	Vendor ID	Identifies the manufacturer of the device (0x1057 = Motorola)
02	Device ID	Identifies the particular device (0x0001 = MPC105)
04	PCI Command	Provides coarse control over a device's ability to generate and respond to PCI bus cycles (see Section 3.2.3.1, "PCI Command Register," for more information)
06	PCI Status	Records status information for PCI bus-related events (see Section 3.2.3.2, "PCI Status Register," for more information)
08	Revision ID	Specifies a device-specific revision code (assigned by Motorola)
09	Standard programming interface	Identifies the register-level programming interface of the MPC105 (0x00)
0A	Subclass code	Identifies more specifically the function of the MPC105 (0x00 = host bridge)
0B	Base class code	Broadly classifies the type of function the MPC105 performs (0x06 = bridge device)
0C	Cache line size	Specifies the system cache line size
0D	Latency timer	Specifies the value of the latency timer for this bus master in PCI bus clock units
0E	Header type	Bits 0–6 identify the layout of bytes 10-3F; bit 7 indicates a multifunction device. The MPC105 uses the most common header type (0x00)
0F	BIST control	Optional register for control and status of built-in self test (BIST)
10–27	—	Reserved on the MPC105
28–2F	—	Reserved for future use by PCI
30–33	—	Reserved on the MPC105
34–3B	—	Reserved for future use by PCI
3C	Interrupt line	Contains interrupt line routing information
3D	Interrupt pin	Indicates which interrupt pin the device (or function) uses (0x00 = no interrupt pin)

Table 3-4. PCI Configuration Space Header Summary (Continued)

Address Offset	Register Name	Description
3E	MIN GNT	Specifies the length of the device's burst period (0x00 indicates that the MPC105 has no major requirements for the settings of latency timers)
3F	MAX LAT	Specifies how often the device needs to gain access to the PCI bus (0x00 indicates that the MPC105 has no major requirements for the settings of latency timers)
40	Bus number	Identifies the assigned bus number of the MPC105 (0x00 = host bridge)
41	Subordinate bus number	Identifies the bus number beneath the MPC105 bridge
42	Disconnect counter	Specifies the timer for target-disconnect timeout (0x00 = the timer is disabled)
43	—	Reserved on the MPC105

3.2.3.1 PCI Command Register

The two-byte PCI command register, shown in Figure 3-7, provides control over the ability to generate and respond to PCI cycles. Table 3-5 describes the bits of the PCI command register.

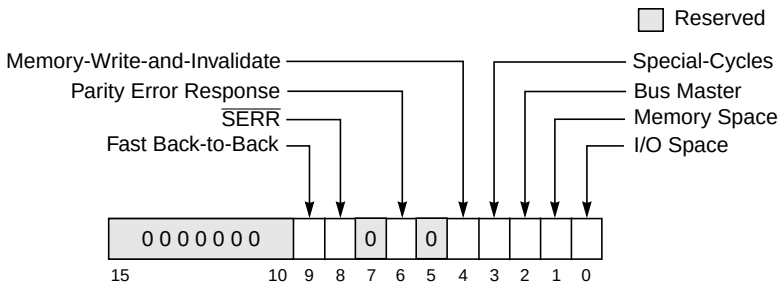


Figure 3-7. PCI Command Register

Table 3-5. Bit Settings for PCI Command Register—0x04

Bit	Description	Reset Value
15–10	These bits are reserved.	All 0s
9	This bit is hardwired to 0, indicating that the MPC105 (as a master) cannot run fast back-to-back transactions to different targets.	0

Table 3-5. Bit Settings for PCI Command Register—0x04 (Continued)

Bit	Description	Reset Value
8	This bit controls the $\overline{\text{SERR}}$ driver of the MPC105. This bit (and bit 6) must be set to report address parity errors. 0 Disables the $\overline{\text{SERR}}$ driver 1 Enables the $\overline{\text{SERR}}$ driver	0
7	These bits are reserved.	0
6	This bit controls whether the MPC105 responds to parity errors. 0 Parity errors are ignored and normal operation continues. 1 Action is taken on a parity error. See Chapter 9, “Error Handling,” for more information.	0
5	These bits are reserved.	0
4	This bit is hardwired to 0, indicating that the MPC105, acting as a master does not generate the memory-write-and-invalidate command. The MPC105 generates a memory-write command instead.	0
3	This bit is hardwired to 0, indicating that the MPC105 (as a target) ignores all special-cycle operations.	0
2	This bit controls whether the MPC105 can act as a master on the PCI bus. Note that if this bit is cleared, 60x to PCI writes will cause the data to be lost and 60x to PCI reads will assert $\overline{\text{TEA}}$ (provided the TEA_EN bit in PICR1 is set). 0 Disables the ability to generate PCI accesses 1 Enables the MPC105 to behave as a bus master	1
1	This bit controls whether the MPC105 (as a target) responds to memory accesses. 0 The MPC105 will not respond to PCI memory space accesses. 1 The MPC105 will respond to PCI memory space accesses.	1
0	This bit is hardwired to 0, indicating that the MPC105 (as a target) will not respond to PCI I/O space accesses.	0

3.2.3.2 PCI Status Register

The two-byte PCI status register, shown in Figure 3-8, is used to record status information for PCI bus-related events. The definition of each bit is given in Table 3-6. Only 2-byte accesses to address 0x06 are allowed.

Reads to this register behave normally. Writes are slightly different in that bits can be cleared, but not set. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear bit 14 and not affect any other bits in the register, write the value 0b0100_0000_0000_0000 to the register.

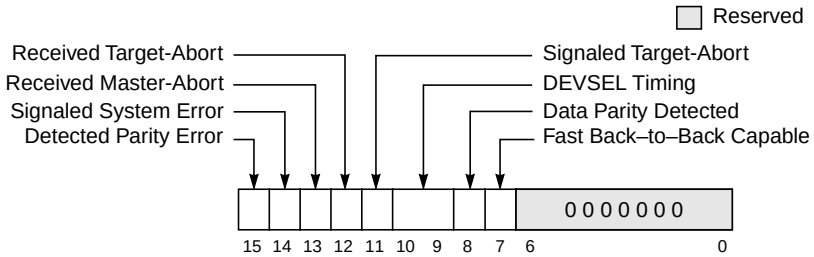


Figure 3-8. PCI Status Register

Table 3-6. Bit Settings for PCI Status Register—0x06

Bit	Description	Reset Value
15	This bit is set whenever the MPC105 detects a parity error, even if parity error handling is disabled (as controlled by bit 6 in the PCI command register).	0
14	This bit is set whenever the MPC105 asserts $\overline{\text{SERR}}$.	0
13	This bit is set whenever the MPC105, acting as the PCI master, terminates a transaction (except for a special-cycle) using master-abort.	0
12	This bit is set whenever an MPC105-initiated transaction is terminated by a target-abort.	0
11	This bit is set whenever the MPC105, acting as the PCI target, issues a target-abort to a PCI master.	0
10–9	These bits are hardwired to 0b00, indicating that the MPC105 uses fast device select timing.	All 0s
8	This bit is set upon detecting a data parity error. Three conditions must be met for this bit to be set: 1.The MPC105 detected a parity error. 2.MPC105 was acting as the bus master for the operation in which the error occurred. 3.Bit 6 in the PCI command register was set.	0
7	This bit is hardwired to 1, indicating that the MPC105 (as a target) is capable of accepting fast back-to-back transactions.	1
6–0	These bits are reserved.	All 0s

3.2.4 Power Management Configuration Register (PMCR)

The 2-byte power management configuration register (PMCR), shown in Figure 3-9, controls the power management functions of the MPC105. Also, some of the bits in this register configure the MPC105 to use the distinct power management features of different 60x processors. Table 3-7 describes the bits of the power management configuration register.

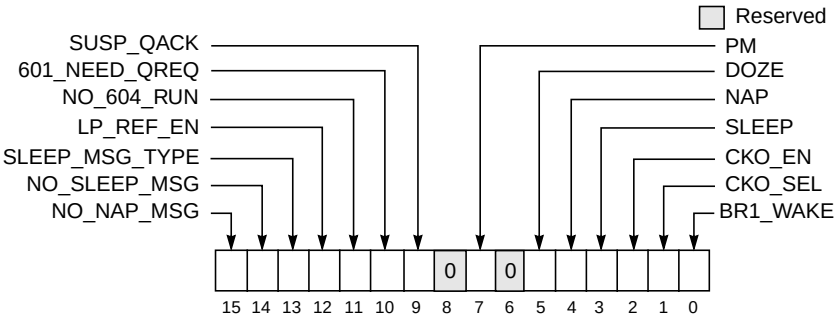


Figure 3-9. Power Management Configuration Register (PMCR)

Table 3-7. Bit Settings for Power Management Configuration Register—0x70

Bit	Name	Reset Value	Description
15	NO_NAP_MSG	0	HALT command broadcast 0 Indicates that the MPC105 will broadcast a HALT command on the PCI bus prior to entering the nap mode 1 Indicates that the MPC105 will not broadcast a HALT command on the PCI bus prior to entering the nap mode
14	NO_SLEEP_MSG	0	Sleep message broadcast 0 Indicates that the MPC105 will broadcast a sleep message command on the PCI bus prior to entering the sleep mode 1 Indicates that the MPC105 will not broadcast a sleep message command on the PCI bus prior to entering the sleep mode The sleep message will be either a SHUTDOWN or HALT command as determined by PMCR[SLEEP_MSG_TYPE].
13	SLEEP_MSG_TYPE	0	Sleep message type 0 Indicates that the MPC105 will broadcast a HALT command onto the PCI bus prior to entering the sleep mode 1 Indicates that the MPC105 will broadcast a SHUTDOWN command onto the PCI bus prior to entering the sleep mode Note that the sleep message will be broadcast only if PMCR[NO_SLEEP_MSG] = 0.
12	LP_REF_EN	0	Low-power refresh 0 Indicates that the MPC105 will not perform memory refresh cycles when it is in sleep or suspend mode 1 Indicates that the MPC105 will continue to perform memory refresh cycles when in sleep or suspend mode

Table 3-7. Bit Settings for Power Management Configuration Register—0x70 (Continued)

Bit	Name	Reset Value	Description
11	NO_604_RUN	0	When both a Power PC 604™ microprocessor and the MPC105 are in nap mode and the MPC105 is woken up by a PCI transaction which will access system memory, this bit controls whether the MPC105 asserts the $\overline{\text{QACK}}$ signal so the 604 can respond to the snoop ($\overline{\text{QACK}}$ is connected to the RUN signal on the 604). Note that the MPC105 ignores NO_604_RUN unless PICR1[PROC_TYPE] = 0b11, indicating a 604. 0 Indicates that the MPC105 will assert the $\overline{\text{QACK}}$ signal. 1 Indicates that the MPC105 will not assert the $\overline{\text{QACK}}$ signal.
10	601_NEED_QREQ	0	Indicates whether the MPC105 should use the $\overline{\text{QREQ}}$ signal as one of the conditions for entering the nap/sleep state when a PowerPC 601™ microprocessor is used in the system.Note that the MPC105 ignores 601_NEED_QREQ unless PICR1[PROC_TYPE] = 0b00, indicating a 601 processor. 0 Indicates that the $\overline{\text{QREQ}}$ signal is not required. 1 Indicates that the $\overline{\text{QREQ}}$ signal is required.
9	SUSP_QACK	0	Indicates whether the MPC105 asserts the $\overline{\text{QACK}}$ signal when entering the suspend power saving mode. 0 Indicates that the MPC105 will not assert the $\overline{\text{QACK}}$ signal when entering the suspend power saving mode. 1 Indicates that the MPC105 will assert $\overline{\text{QACK}}$ when entering the suspend power saving mode.
8	—	0	This bit is reserved.
7	PM	0	Power management enable 0 Disables the power management logic within the MPC105. 1 Enables the power management logic within the MPC105.
6	—	0	This bit is reserved.
5	DOZE	0	Enables/disables the doze mode capability of the MPC105. Note that this bit is only valid if MPC105 power management is enabled (PMCR[PM] = 1). 0 Disables the doze mode 1 Enables the doze mode
4	NAP	0	Enables/disables the nap mode capability of the MPC105. Note that this bit is only valid if MPC105 power management is enabled (PMCR[PM] = 1). 0 Disables the nap mode 1 Enables the nap mode
3	SLEEP	0	Enables/disables the sleep mode capability of the MPC105. Note that this bit is only valid if MPC105 power management is enabled (PMCR[PM] = 1). 0 Disables the sleep mode 1 Enables the sleep mode
2	CKO_EN	0	Enables/disables the test clock output driver. 0 Disables the test clock output driver 1 Enables the test clock output driver

Table 3-7. Bit Settings for Power Management Configuration Register—0x70 (Continued)

Bit	Name	Reset Value	Description
1	CKO_SEL	0	Selects the clock source for the test clock output. 0 Selects the internal clock as the test clock output source 1 Selects SYSCLK as the test clock output source
0	BR1_WAKE	0	Enables/disables awareness of a second processor for nap and sleep modes. 0 Indicates the MPC105 will not awaken from nap or sleep mode when $\overline{\text{DIRTY\_IN/BR1}}$ is asserted. 1 Indicates the MPC105 will awaken from nap or sleep mode when $\overline{\text{DIRTY\_IN/BR1}}$ is asserted in a multiprocessor configuration.

3.2.5 Error Handling Registers

Chapter 9, “Error Handling,” describes specific error conditions and how the MPC105 responds to them. The registers from 0xC0 through 0xCB control the error handling and reporting for the MPC105. The following sections provide descriptions of these registers.

3.2.5.1 Error Enabling Registers

Error enabling registers 1 and 2 (ErrEnR1 and ErrEnR2), shown in Figure 3-10 and Figure 3-11, control whether the MPC105 recognizes and reports specific error conditions. Table 3-8 describes the bits of ErrEnR1 and Table 3-9 describes the bits of ErrEnR2.

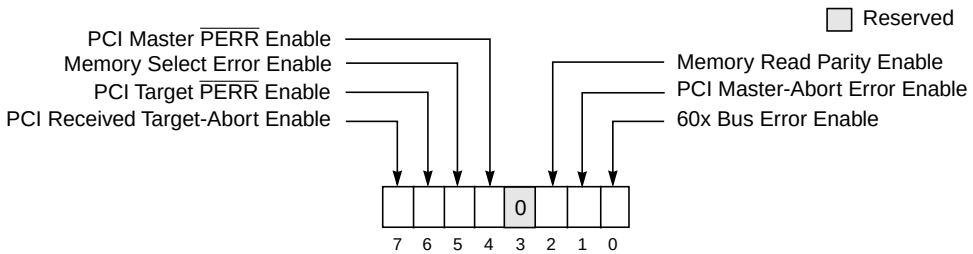


Figure 3-10. Error Enabling Register 1 (ErrEnR1)

Table 3-8. Bit Settings for Error Enabling Register 1 (ErrEnR1)—0xC0

Bit	Reset Value	Description
7	0	Received PCI target-abort error enable. This bit enables the reporting of target-abort errors that occur on the PCI bus for transactions involving the MPC105 as a master. 0 Received PCI target-abort error disabled 1 Received PCI target-abort error enabled
6	0	PCI target $\overline{\text{PERR}}$ enable. This bit enables the reporting of data parity errors on the PCI bus for transactions involving the MPC105 as a target. 0 Target $\overline{\text{PERR}}$ disabled 1 Target $\overline{\text{PERR}}$ enabled
5	0	Memory select error enable. This bit enables the reporting of memory select errors that occur on (attempted) accesses to system memory. 0 Memory select error disabled 1 Memory select error enabled
4	0	PCI master $\overline{\text{PERR}}$ enable. This bit enables the reporting of data parity errors on the PCI bus for transactions involving the MPC105 as a master. 0 Master $\overline{\text{PERR}}$ disabled 1 Master $\overline{\text{PERR}}$ enabled
3	0	Reserved
2	0	Memory read parity enable. This bit enables the reporting of system memory read parity errors that occur on accesses to system memory. 0 Memory read parity disabled 1 Memory read parity enabled
1	0	PCI master-abort error enable. This bit enables the reporting of master-abort errors that occurred on the PCI bus for transactions involving the MPC105 as a master. 0 PCI master-abort error disabled 1 PCI master-abort error enabled
0	1	60x bus error enable. This bit enables the reporting of 60x bus errors. 0 60x bus error disabled 1 60x bus error enabled

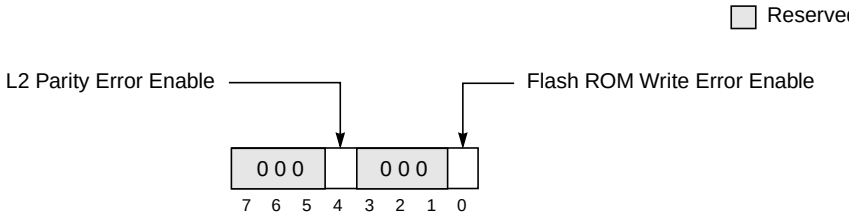


Figure 3-11. Error Enabling Register 2 (ErrEnR2)

Table 3-9. Bit Settings for Error Enabling Register 2 (ErrEnR2)—0xC4

Bit	Reset Value	Description
7–5	All 0s	Reserved
4	0	L2 Parity Error Enable 0 L2 read parity disabled 1 L2 read parity enabled
3–1	All 0s	Reserved
0	0	FLASH ROM write error enable 0 Disabled 1 Enabled

3.2.5.2 Error Detection Registers

Error detection registers 1 and 2 (ErrDR1 and ErrDR2), shown in Figure 3-12 and Figure 3-13, contain error flags that report when the MPC105 detects a specific error condition.

The error detection registers are bit-reset type registers. That is, reading from these registers occurs normally; however, write operations are different in that bits (error flags) can be cleared, but not set. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear bit 6 and not affect any other bits in the register, write the value 0b0100_0000 to the register. When the MPC105 detects an error, the appropriate error flag is set. Subsequent errors will set the appropriate error flags in the error detection registers, but the bus error status and error address are not recorded until the previous error flags are cleared.

Table 3-10 describes the bits of error detection register 1 and Table 3-11 describes the bits of error detection register 2.

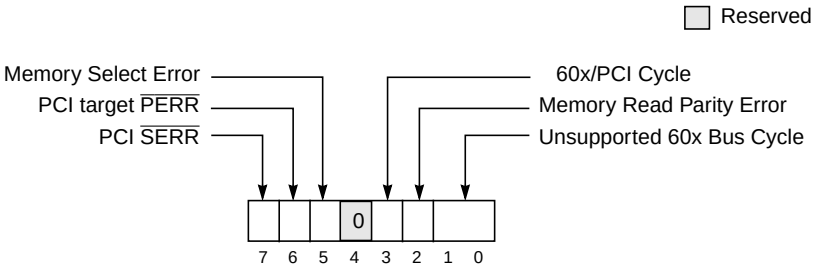


Figure 3-12. Error Detection Register 1 (ErrDR1)—0xC1

Table 3-10. Bit Settings for Error Detection Register 1 (ErrDR1)—0xC1

Bit	Reset Value	Description
7	0	PCI SERR. Note that for the MPC105 to recognize the assertion of SERR by another PCI agent, bit 5 (RX_SERR_EN) of the alternate OS-visible parameters register 1 must be set. See Table 3-32 for a description of the RX_SERR_EN bit. 0 No error signaled 1 Error signaled
6	0	PCI target PERR 0 The MPC105, as a PCI target, has not signaled PERR 1 The MPC105, as a PCI target, signaled PERR
5	0	Memory select error 0 No error detected 1 Memory select error detected
4	0	Reserved
3	0	60x/PCI cycle 0 Error occurred on a 60x-initiated cycle 1 Error occurred on a PCI-initiated cycle
2	0	Memory read parity error 0 No error detected 1 Parity error detected
1, 0	00	Unsupported 60x bus cycle 00 No error detected 01 Unsupported transfer attributes 10 XATS detected 11 Reserved

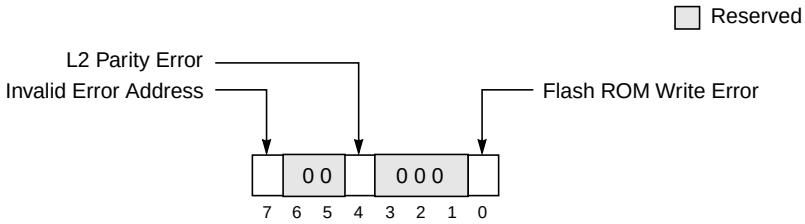


Figure 3-13. Error Detection register 2 (ErrDR2)—0xC5

Table 3-11. Bit Settings for Error Detection Register 2 (ErrDR2)—0xC5

Bit	Reset Value	Description
7	0	Invalid error address. This bit indicates whether the address stored in the 60x/PCI error address register is valid. 0 The address in the 60x/PCI error address register is valid. 1 The address in the 60x/PCI error address register is not valid.
6–5	All 0s	Reserved
4	0	L2 parity error 0 No error detected 1 L2 parity error detected
3–1	All 0s	Reserved
0	0	Flash ROM write error 0 No error detected 1 The MPC105 detected a write to Flash ROM when writes to Flash ROM are disabled.

3.2.5.3 Error Status Registers

The error status registers latch the state of the 60x or PCI address bus when an error is detected; see Figure 3-14, Figure 3-15, and Figure 3-16. These registers provide system status for error handling software.

Table 3-12 describes the bits of the 60x bus error status register, Table 3-13 describes the bits of the PCI bus error status register, and Table 3-14 describes the bits of 60x/PCI error address register.

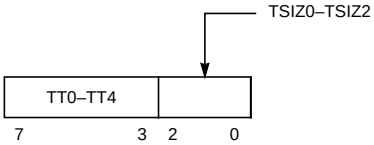


Figure 3-14. 60x Bus Error Status Register—0xC3

Table 3-12. Bit Settings for 60x Bus Error Status Register—0xC3

Bit	Reset Value	Description
7–3	00000	These bits maintain a copy of TT0–TT4. When a 60x bus error is detected, these bits are latched until all error flags are cleared.
2–0	000	These bits maintain a copy of TSIZ0–TSIZ2. When a 60x bus error is detected, these bits are latched until all error flags are cleared.

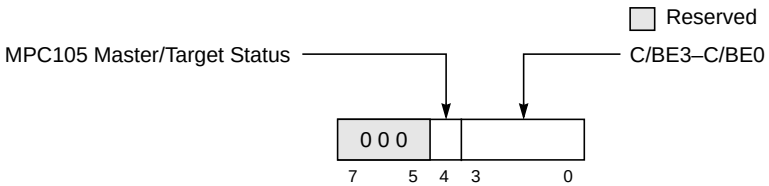


Figure 3-15. PCI Bus Error Status Register—0xC7

Table 3-13. Bit Settings for PCI Bus Error Status Register—0xC7

Bit	Reset Value	Description
7–5	000	Reserved
4	0	MPC105 master/target status 0 MPC105 is the PCI master 1 MPC105 is the PCI target
3–0	0000	These bits maintain a copy of $\overline{\text{C/BE3}}$–$\overline{\text{C/BE0}}$. When a PCI bus error is detected, these bits are latched until all error flags are cleared.

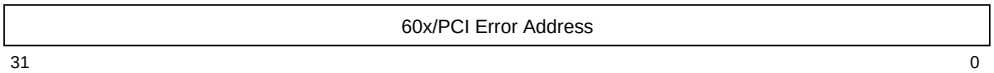


Figure 3-16. 60x/PCI Error Address Register—0xC8

Table 3-14. Bit Settings for 60x/PCI Error Address Register—0xC8

Bit	Description
31–24	A24–A31 or AD7–AD0 (dependent upon whether the error is a 60x bus error or a PCI bus error). When an error is detected, these bits are latched until all error flags are cleared.
23–16	A16–A23 or AD15–AD8 (dependent upon whether the error is a 60x bus error or a PCI bus error). When an error is detected, these bits are latched until all error flags are cleared.
15–8	A8–A15 or AD23–AD16 (dependent upon whether the error is a 60x bus error or a PCI bus error). When an error is detected, these bits are latched until all error flags are cleared.
7–0	A0–A7 or AD31–AD24 (dependent upon whether the error is a 60x bus error or a PCI bus error). When an error is detected, these bits are latched until all error flags are cleared.

3.2.6 Memory Interface Configuration Registers

The memory interface configuration registers (MICRs) control memory boundaries (starting and ending addresses), memory bank enables, memory timing, and external memory buffers. Initialization software must program the MICRs at power-on reset and then enable the memory interface on the MPC105 by setting the MEMGO bit in memory control configuration register 1 (MCCR1).

3.2.6.1 Memory Boundary Registers

The extended starting address (shown in Table 3-17 and Table 3-18) and the starting address (shown in Table 3-15 and Table 3-16) registers are used to define the lower address boundary for each memory bank. The lower boundary is determined by the following formula:

Lower boundary for bank *n* = 0b00 || <extended starting address *n*> || <starting address *n*> || 0x00000.

The extended ending address (shown in Table 3-21 and Table 3-22) and the ending address (shown in Table 3-19 and Table 3-20) are used to define the upper address boundary for each memory bank. The upper boundary is determined by the following formula:

Upper boundary for bank *n* = 0b00 || <extended ending address *n*> || <ending address *n*> || 0xFFFFF.

See Figure 3-17 and Table 3-15 for memory starting address register 1 bit settings.

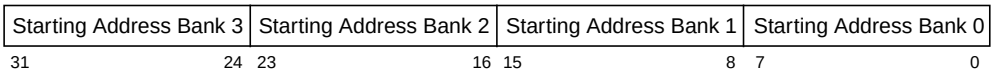


Figure 3-17. Memory Starting Address Register 1—0x80

Table 3-15. Bit settings for Memory Starting Address Register 1—0x80

Bit	Description	Byte Address
31–24	Starting address for bank 3	0x83
23–16	Starting address for bank 2	0x82
15–8	Starting address for bank 1	0x81
7–0	Starting address for bank 0	0x80

See Figure 3-18 and Table 3-16 for memory starting address register 2 bit settings.

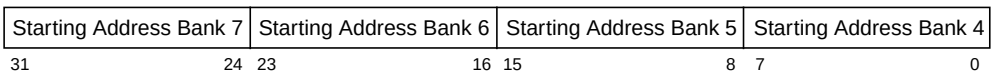


Figure 3-18. Memory Starting Address Register 2—0x84

Table 3-16. Bit Settings for Memory Starting Address Register 2—0x84

Bit	Description	Byte Address
31–24	Starting address for bank 7	0x87
23–16	Starting address for bank 6	0x86
15–8	Starting address for bank 5	0x85
7–0	Starting address for bank 4	0x84

See Figure 3-19 and Table 3-17 for extended memory starting address register 1 bit settings.

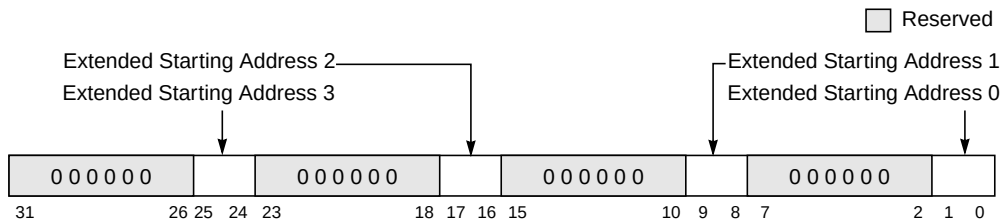


Figure 3-19. Extended Memory Starting Address Register 1—0x88

Table 3-17. Bit Settings for Extended Memory Starting Address Register 1—0x88

Bit	Description	Byte Address
31–26	Reserved	0x8B
25–24	Extended starting address for bank 3	
23–18	Reserved	0x8A
17–16	Extended starting address for bank 2	
15–10	Reserved	0x89
9–8	Extended starting address for bank 1	
7–2	Reserved	0x88
1–0	Extended starting address for bank 0	

See Figure 3-20 and Table 3-18 for extended memory starting address register 2 bit settings.

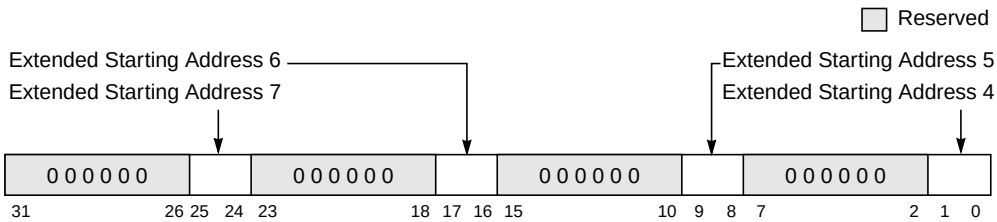


Figure 3-20. Extended Memory Starting Address Register 2—0x8C

Table 3-18. Bit Settings for Extended Memory Starting Address Register 2—0x8C

Bit	Description	Byte Address
31–26	Reserved	0x8F
25–24	Extended starting address for bank 7	
23–18	Reserved	0x8E
17–16	Extended starting address for bank 6	
15–10	Reserved	0x8D
9–8	Extended starting address for bank 5	
7–2	Reserved	0x8C
1–0	Extended starting address for bank 4	

See Figure 3-21 and Table 3-19 for memory ending address register 1 bit settings.

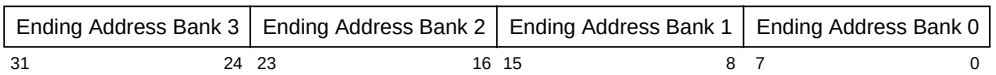


Figure 3-21. Memory Ending Address Register 1—0x90

Table 3-19. Bit Settings for Memory Ending Address Register 1—0x90

Bit	Description	Byte Address
31–24	Ending address for bank 3	0x93
23–16	Ending address for bank 2	0x92
15–8	Ending address for bank 1	0x91
7–0	Ending address for bank 0	0x90

See Figure 3-22 and Table 3-20 for memory ending address register 2 bit settings.

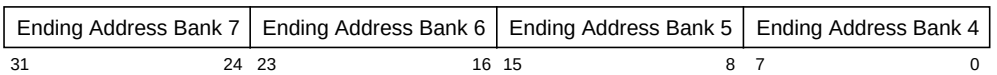


Figure 3-22. Memory Ending Address Register 2—0x94

Table 3-20. Bit Settings for Memory Ending Address Register 2—0x94

Bit	Description	Byte Address
31–24	Ending address for bank 7	0x97
23–16	Ending address for bank 6	0x96
15–8	Ending address for bank 5	0x95
7–0	Ending address for bank 4	0x94

See Figure 3-23 and Table 3-21 for extended memory ending address register 1 bit settings.

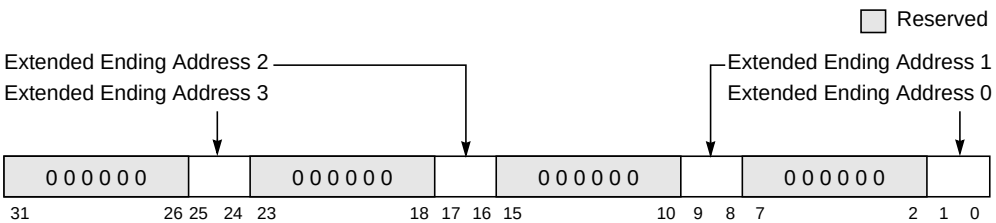


Figure 3-23. Bit Settings for Extended Memory Ending Address Register 1

Table 3-21. Bit Settings for Extended Memory Ending Address Register 1—0x98

Bit	Description	Byte Address
31–26	Reserved	0x9B
25–24	Extended ending address for bank 3	
23–18	Reserved	0x9A
17–16	Extended ending address for bank 2	
15–10	Reserved	0x99
9–8	Extended ending address for bank 1	
7–2	Reserved	0x98
1–0	Extended ending address for bank 0	

See Figure 3-24 and Table 3-22 for extended memory ending address register 2 bit settings.

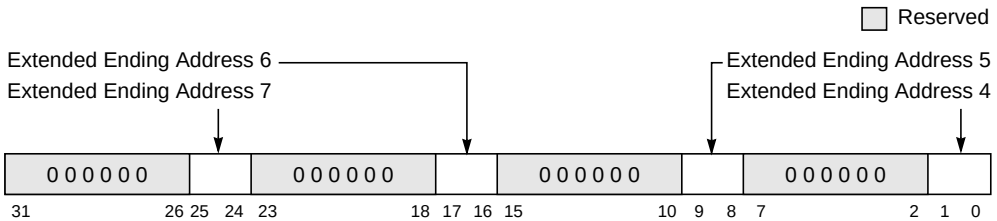


Figure 3-24. Extended Memory Ending Address Register 2—0x9C

Table 3-22. Bit Settings for Extended Memory Ending Address Register 2—0x9C

Bit	Description	Byte Address
31–26	Reserved	0x9F
25–24	Extended ending address for bank 7	
23–18	Reserved	0x9E
17–16	Extended ending address for bank 6	
15–10	Reserved	0x9D
9–8	Extended ending address for bank 5	
7–2	Reserved	0x9C
1–0	Extended ending address for bank 4	

3.2.6.2 Memory Bank Enable Register

Individual banks are enabled or disabled by using the 1-byte memory bank enable register, shown in Figure 3-25 and Table 3-23. If a bank is enabled, the ending address of that bank must be greater than or equal to its starting address. If a bank is disabled, no memory transactions will access that bank, regardless of its starting and ending addresses.

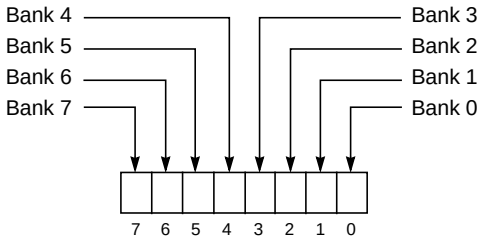


Figure 3-25. Memory Bank Enable Register—0xA0

Table 3-23. Bit Settings for Memory Bank Enable Register—0xA0

Bit	Description
7	Bank 7 0 Disabled 1 Enabled
6	Bank 6 0 Disabled 1 Enabled
5	Bank 5 0 Disabled 1 Enabled
4	Bank 4. 0 Disabled 1 Enabled
3	Bank 3 0 Disabled 1 Enabled
2	Bank 2 0 Disabled 1 Enabled
1	Bank 1 0 Disabled 1 Enabled
0	Bank 0 0 Disabled 1 Enabled

3.2.6.3 Memory Control Configuration Registers

The four 32-bit memory control configuration registers (MCCRs) set all RAM and ROM parameters. These registers are programmed by initialization software to adapt the MPC105 to the specific memory organization used in the system. After all the memory configuration parameters have been properly configured, the initialization software turns on the memory interface using the MEMGO bit in MCCR1. See Table 3-26 and Figure 3-23 for memory control configuration register 1 bit settings.

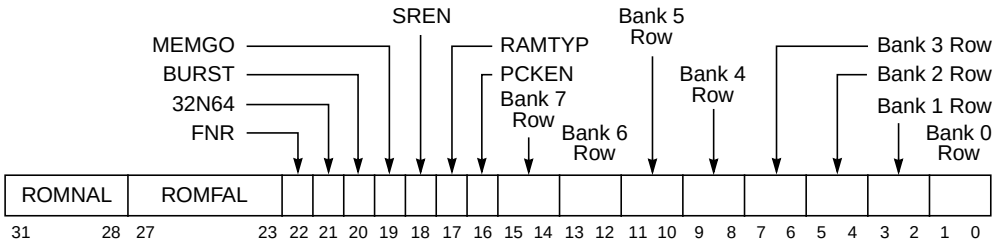


Figure 3-26. Memory Control Configuration Register 1 (MCCR1)—0xF0

Table 3-24. Bit Settings for Memory Control Configuration Register 1—0xF0

Bit	Name	Reset Value	Description
31–28	ROMNAL	All 1s	For burst-mode ROMs, ROMNAL controls the next nibble access time. For Flash ROMs, ROMNAL measures the write pulse recovery (high) time. The maximum value is 0b1111 (15).The actual cycle count will be three cycles more than the binary value of ROMNAL.
27–23	ROMFAL	All 1s	For nonburst ROMs, ROMNAL controls the access time. For burst mode ROMs, ROMNAL controls the first access time. For Flash ROMs, ROMNAL measures the write pulse low time. The maximum value is 0b11111 (31). The actual cycle count will be three cycles more than the binary value of ROMFAL.
22	FNR		Read only. FNR is sampled from the FNR/$\overline{\text{DWE0}}$ configuration signal at reset. 0 Indicates that the MPC105 has been configured for (32- or 64-bit interface) ROM memory 1 Indicates that the MPC105 has been configured for Flash (8-bit only) ROM memory
21	32N64		Read only. 32N64 is the inverse of the 60x data bus width configuration signal (DL0) sampled at reset. 0 Indicates that the MPC105 has been configured for 64-bit processor/memory data bus width 1 Indicates that the MPC105 has been configured for 32-bit processor/memory data bus width
20	BURST	0	Burst mode ROM timing enable. 0 Indicates standard (nonburst) ROM access timing 1 Indicates burst-mode ROM access timing

Table 3-24. Bit Settings for Memory Control Configuration Register 1—0xF0 (Continued)

Bit	Name	Reset Value	Description
19	MEMGO	0	RAM interface logic enable. Note that this bit must not be set until all other memory configuration parameters have been appropriately configured by boot code. 0 MPC105 RAM interface logic disabled 1 MPC105 RAM interface logic enabled
18	SREN	0	Self-refresh enable. Note that if self refresh is disabled the user is responsible for preserving the integrity of DRAM during sleep or suspend mode. 0 Disables the DRAM self refresh during sleep or suspend mode 1 Enables the DRAM self refresh during sleep or suspend mode
17	RAMTYP	1	RAM type 0 Indicates synchronous DRAM (SDRAM) 1 Indicates standard DRAM
16	PCKEN	0	Memory interface parity checking/generation enable 0 Disables parity checking and parity generation for transactions to DRAM/SDRAM memory 1 Enables parity checking and generation for all memory transactions to DRAM/SDRAM
15–14	Bank 7 row	00	RAM bank 7 row address bit count. These bits indicate the number of row address bits that are required by the RAM devices in bank 7. 00 9 row bits 01 10 row bits 10 11 row bits 11 12 row bits
13–12	Bank 6 row	00	RAM bank 6 row address bit count. These bits indicate the number of row address bits that are required by the RAM devices in bank 6. 00 9 row bits 01 10 row bits 10 11 row bits 11 12 row bits
11–10	Bank 5 row	00	RAM bank 5 row address bit count. These bits indicate the number of row address bits that are required by the RAM devices in bank 5. 00 9 row bits 01 10 row bits 10 11 row bits 11 12 row bits
9–8	Bank 4 row	00	RAM bank 4 row address bit count. These bits indicate the number of row address bits that are required by the RAM devices in bank 4. 00 9 row bits 01 10 row bits 10 11 row bits 11 12 row bits

Table 3-24. Bit Settings for Memory Control Configuration Register 1—0xF0 (Continued)

Bit	Name	Reset Value	Description
7–6	Bank 3 row	00	RAM bank 3 row address bit count. These bits indicate the number of row address bits that are required by the RAM devices in bank 3. 00 9 row bits 01 10 row bits 10 11 row bits 11 12 row bits
5–4	Bank 2 row	00	RAM bank 2 row address bit count. These bits indicate the number of row address bits that are required by the RAM devices in bank 2. 00 9 row bits 01 10 row bits 10 11 row bits 11 12 row bits
3–2	Bank 1 row	00	RAM bank 1 row address bit count. These bits indicate the number of row address bits that are required by the RAM devices in bank 1. 00 9 row bits 01 10 row bits 10 11 row bits 11 12 row bits
1–0	Bank 0 row	00	RAM bank 0 row address bit count. These bits indicate the number of row address bits that are required by the RAM devices in bank 0. 00 9 row bits 01 10 row bits 10 11 row bits 11 12 row bits

See Figure 3-27 and Table 3-25 for memory control configuration register 2 (MCCR2) bit settings.

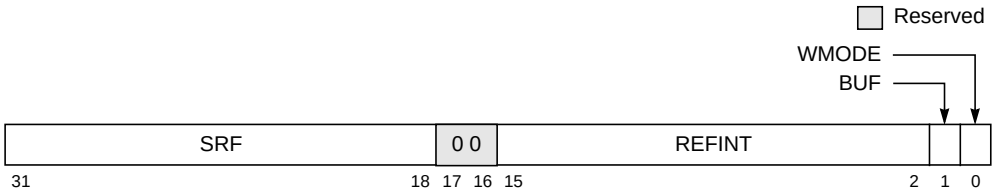


Figure 3-27. Memory Control Configuration Register 2 (MCCR2) (RAM Access Time)

Table 3-25. Memory Control Configuration Register 2 (RAM Access Time)—0xF4

Bit	Name	Reset Value	Description
31–18	SRF	All 0s	Self-refresh entry delay. During the sleep and suspend power saving modes, the MPC105 supports self refreshing DRAMs. If MCCR1[SREN] is set, SRF represents the number of clock cycles from the beginning of a CBR refresh cycle to the self refresh entry. The value for SRF depends on the specific DRAMs used. SDRAM systems do not use the SRF parameter.
17–16	—	00	Reserved
15–2	REFINT	All 0s	Refresh interval. These bits directly represent the number of clock cycles between CBR refresh cycles. One row is refreshed in each RAM bank during each CBR refresh cycle. The value for REFINT depends on the specific RAMs used and the operating frequency of the MPC105. See Section 6.3.4, “DRAM Refresh,” or Section 6.4.5, “SDRAM Refresh,” for more information. Note that the period of the refresh interval must be greater than the read/write access time to insure that read/write operations complete successfully.
1	BUF	1	Buffer mode. This bit controls how $\overline{\text{BCTL0}}$ and $\overline{\text{BCTL1}}$ operate. See Section 6.2, “Memory Interface Signal Buffering,” for more information. 0 $\overline{\text{BCTL0}}$ enables the buffer for write operations; $\overline{\text{BCTL1}}$ enables the buffer for read operations. 1 $\overline{\text{BCTL0}}$ controls the buffer direction; $\overline{\text{BCTL1}}$ acts as buffer enable.
0	WMODE	0	Applies to 32-bit data path mode only. Determines whether the burst ROMs can accept eight beats in a burst or only four. In 32-bit data path mode, burst transactions require eight data beats. If the burst ROM can only accept four beat per burst, the memory controller must perform two transactions to the ROM. 0 Four (4) beats per burst (default) 1 Eight (8) beats per burst

See Figure 3-28 and Table 3-26 for memory control configuration register 3 (MCCR3) bit settings.

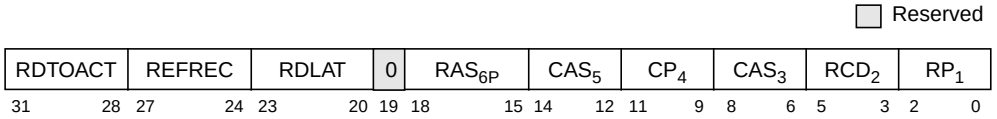


Figure 3-28. Memory Control Configuration Register 3 (MCCR3)

Table 3-26. Bit Settings for Memory Control Configuration Register 3—0xF8

Bit	Name	Reset Value	Description
31–28	RDTOACT	0000	Read to activate interval. For SDRAM only. These bits control the number of clock cycles from an SDRAM-read (with autoprecharge) command until an SDRAM-activate command is allowed. See Section 6.4.4, “SDRAM Interface Timing,” for more information. 0001 1 clock 0010 2 clocks 0011 3 clocks ... ... 1111 15 clocks 0000 16 clocks
27–24	REFREC	0000	Refresh to activate interval. For SDRAM only. These bits control the number of clock cycles from an SDRAM-refresh command until an SDRAM-activate command is allowed. See Section 6.4.4, “SDRAM Interface Timing,” for more information. 0001 1 clock 0010 2 clocks 0011 3 clocks ... ... 1111 15 clocks 0000 16 clocks
23–20	RDLAT	0000	Data latency from read command. For SDRAM only. These bits control the number of clock cycles from an SDRAM-read (with autoprecharge) command until the first data beat is available on the 60x data bus. See Section 6.4.4, “SDRAM Interface Timing,” for more information. 0001 1 clock 0010 2 clocks 0011 3 clocks ... ... 1111 15 clocks 0000 16 clocks
19	—	0	Reserved

Table 3-26. Bit Settings for Memory Control Configuration Register 3—0xF8 (Continued)

Bit	Name	Reset Value	Description
18–15	RAS_{6P}	0000	$\overline{RAS}$ assertion interval for CBR refresh. For DRAM only. These bits control the number of clock cycles $\overline{RAS}$ is held asserted during CBR refresh. The value for RAS_{6P} depends on the specific DRAMs used and the 60x bus frequency. See Section 6.3.3, “DRAM Interface Timing,” for more information. 0001 1 clock 0010 2 clocks 0011 3 clocks ... ... 1111 15 clocks 0000 16 clocks
14–12	CAS_5	000	$\overline{CAS}$ assertion interval for page mode access. For DRAM only. These bits control the number of clock cycles $\overline{CAS}$ is held asserted during page mode accesses. The value for CAS_5 depends on the specific DRAMs used and the 60x bus frequency. See Section 6.3.3, “DRAM Interface Timing,” for more information. 001 1 clock 010 2 clocks 011 3 clocks ... ... 111 7 clocks 000 8 clocks
11–9	CP_4	000	$\overline{CAS}$ precharge interval. For DRAM only. These bits control the number of clock cycles that $\overline{CAS}$ must be held negated in page mode (to allow for column precharge) before the next assertion of $\overline{CAS}$. See Section 6.3.3, “DRAM Interface Timing,” for more information. 001 1 clock 010 2 clocks 011 3 clocks ... ... 111 7 clocks 000 8 clocks
8–6	CAS_3	000	$\overline{CAS}$ assertion interval for the first access. For DRAM only. These bits control the number of clock cycles $\overline{CAS}$ is held asserted during a single beat or during the first access in a burst. The value for CAS_3 depends on the specific DRAMs used and the 60x bus frequency. See Section 6.3.3, “DRAM Interface Timing,” for more information. 001 1 clock 010 2 clocks 011 3 clocks ... ... 111 7 clocks 000 8 clocks

Table 3-26. Bit Settings for Memory Control Configuration Register 3—0xF8 (Continued)

Bit	Name	Reset Value	Description
5–3	RCD_2	000	$\overline{RAS}$ to $\overline{CAS}$ delay interval. For DRAM only. These bits control the number of clock cycles between the assertion of $\overline{RAS}$ and the first assertion of $\overline{CAS}$. The value for RCD_2 depends on the specific DRAMs used and the 60x bus frequency. However, RCD_2 must be at least two clock cycles. See Section 6.3.3, “DRAM Interface Timing,” for more information. 001 Reserved 010 2 clocks 011 3 clocks ... ... 111 7 clocks 000 8 clocks
2–0	RP_1	000	$\overline{RAS}$ precharge interval. For DRAM only. These bits control the number of clock cycles that $\overline{RAS}$ must be held negated (to allow for row precharge) before the next assertion of $\overline{RAS}$. See Section 6.3.3, “DRAM Interface Timing,” for more information. 001 1 clock 010 2 clocks 011 3 clocks ... ... 111 7 clocks 000 8 clocks

See Figure 3-29 and Table 3-27 for memory control configuration register 4 (MCCR4)bit settings.

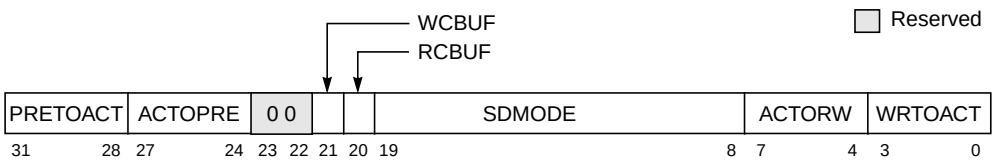


Figure 3-29. Memory Control Configuration Register 4 (MCCR4)

Table 3-27. Bit Settings for Memory Control Configuration Register 4—0xFC

Bit	Name	Reset Value	Description
31–28	PRETOACT	0000	Precharge to activate interval. For SDRAM only. These bits control the number of clock cycles from an SDRAM-precharge command until an SDRAM-activate command is allowed. See Section 6.4.4, “SDRAM Interface Timing,” for more information. 0001 1 clock 0010 2 clocks 0011 3 clocks ... ... 1111 15 clocks 0000 16 clocks

Table 3-27. Bit Settings for Memory Control Configuration Register 4—0xFC (Continued)

Bit	Name	Reset Value	Description
27–24	ACTOPRE	0000	Activate to precharge interval. For SDRAM only. These bits control the number of clock cycles from an SDRAM-activate command until an SDRAM-precharge command is allowed. See Section 6.4.4, “SDRAM Interface Timing,” for more information. 0001 1 clock 0010 2 clocks 0011 3 clocks ... ... 1111 15 clocks 0000 16 clocks
23–22	—	00	Reserved
21	WCBUF	0	Memory write buffer type. This bit controls how the buffer control signals operate. See Section 6.2, “Memory Interface Signal Buffering,” for more information. 0 Flow through or transparent latch type buffer 1 Registered type buffer
20	RCBUF	1	Memory read buffer type. This bit controls how the buffer control signals operate. See Section 6.2, “Memory Interface Signal Buffering,” for more information. 0 Flow through type buffer 1 Transparent latch or registered type buffer
19–8	SDMODE	All 0s	SDRAM mode register. For SDRAM only. These bits specify the SDRAM mode register data to be written to the SDRAM array during power-up configuration. **Bit Description** 19–15 Opcode. For compliance with the JEDEC standard, these bits are set to 0b00000 for normal mode of operation and to 0b00001 for the JEDEC reserved test mode. All other modes of operation are vendor-specific. 14–12 CAS latency 000 Reserved 001 1 010 2 011 3 100 4 101 Reserved 110 Reserved 111 Reserved 11 Wrap type 0 Sequential (Note that the sequential wrap type is required for 60x processor-based systems) 1 Interleaved 10–8 Wrap length 000 Reserved 001 Reserved 010 4 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 3-27. Bit Settings for Memory Control Configuration Register 4—0xFC (Continued)

Bit	Name	Reset Value	Description
7–4	ACTORW	0000	Activate to read/write interval. For SDRAM only. These bits control the number of clock cycles from an SDRAM-activate command until an SDRAM-read (with autoprecharge) or SDRAM-write (with autoprecharge) command is allowed. ACTORW must be at least two clock cycles. See Section 6.4.4, “SDRAM Interface Timing,” for more information. 0001 Reserved 0010 2 clocks 0011 3 clocks ... ... 1111 15 clocks 0000 16 clocks
3–0	WRTOACT	0000	Write to activate interval. For SDRAM only. These bits control the number of clock cycles from an SDRAM-write (with autoprecharge) command until an SDRAM-activate command is allowed. See Section 6.4.4, “SDRAM Interface Timing,” for more information. 0001 1 clock 0010 2 clocks 0011 3 clocks ... ... 1111 15 clocks 0000 16 clocks

3.2.7 Processor Interface Configuration Registers

The processor interface configuration registers (PICRs) control the programmable parameters of the 60x bus interface and the L2 cache interface. There are two 32-bit PICRs—PICR1 and PICR2. See Figure 3-30 and Table 3-28 for PICR1 bit settings.

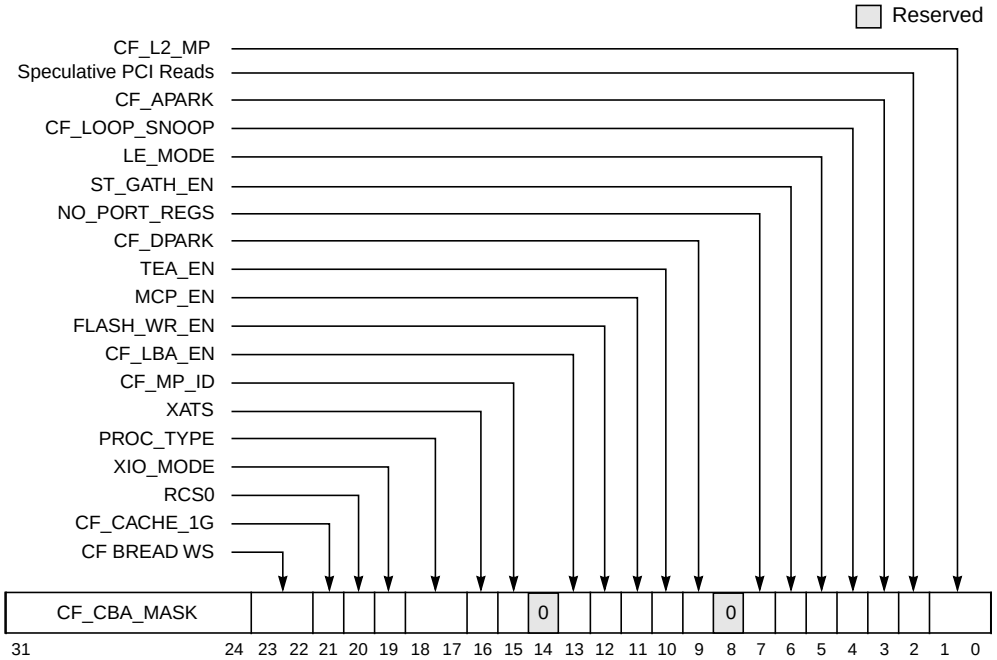


Figure 3-30. Processor Interface Configuration Register 1

Table 3-28. Bit Settings for Processor Interface Configuration Register 1—0xA8

Bit	Name	Reset Value	Description
31–24	CF_CBA_MASK	All 1s	L2 copy-back address mask. The MPC105 uses CF_CBA_MASK to mask off address bits that are not driven by the tag RAM during tag RAM read cycles. If a bit in CF_CBA_MASK is cleared, the corresponding address bit read from the tag RAM will be treated as 0b0 by the MPC105 (it is masked internally), regardless of its actual state. If a bit in CF_CBA_MASK is set, the corresponding address bit read from the tag RAM will be the actual state of the address bit.
23–22	CF_BREAD_WS	00	Burst read wait states. These bits control the number of wait states from $\overline{TS}$ to the first $\overline{TA}$ for burst reads. 00 0 wait states (601, 604) 01 1 wait state (PowerPC 603™ processor in $\overline{DRTRY}$ mode) 10 2 wait states (603 in no-$\overline{DRTRY}$ mode) 11 3 wait states (not recommended)

Table 3-28. Bit Settings for Processor Interface Configuration Register 1—0xA8 (Continued)

Bit	Name	Reset Value	Description
21	CF_CACHE_1G	0	L2 cache 0–1Gbyte only. This bit controls whether the L2 cache caches addresses from 0 to 1 Gbyte or from 0 to 2 Gbyte and the ROM address space. 0 The L2 may cache addresses from 0 to 2 Gbyte and ROM addresses. 1 The L2 may only cache addresses from 0 to 1 Gbyte. No check for hit or miss is performed for addresses from 1 to 2 Gbyte or for ROM addresses.
20	RCS0	x	ROM Location. Read only. This bit indicates the state of the ROM location ($\overline{\text{RCS0}}$) configuration signal at power-on reset. 0 ROM is located on PCI bus. 1 ROM is located on 60x processor/memory data bus.
19	XIO_MODE	0	Address map A contiguous/discontiguous mode. This bit controls whether address map A uses the contiguous or discontiguous I/0 mode. Note that this bit is also accessible from the external configuration register at 0x850. See Section 3.1.1, “Address Map A,” for more information. 0 Contiguous mode 1 Discontiguous mode
18–17	PROC_TYPE	00	Processor type. These bits identify the type of processor used in the system. The MPC105 uses PROC_TYPE to control $\overline{\text{ARTRY}}$ timing (due to differences between the 601 and the 603/604), and the power saving modes (for the 603 or 604). 00 601 01 Reserved 10 603 11 604
16	XATS	x	Address map. Read only. This bit indicates the state of the address map ($\overline{\text{XATS}}$) configuration signal at power-on reset. See Section 3.1, “Address Maps,” for more information. 0 The MPC105 is configured for address map B. 1 The MPC105 is configured for address map A.
15	CF_MP_ID	0	Multiprocessor identifier. Read only. This bit indicates which processor (in a multiprocessor system) is performing the current transaction. CF_MP_ID provides a means for software to identify the processors. 0 Processor 0 is reading PICR1[CF_MP_ID]. 1 Processor 1 is reading PICR1[CF_MP_ID].
14	—	0	Reserved
13	CF_LBA_EN	0	Local bus slave access enable. This bit controls whether the MPC105 allows a local bus slave in the 60x bus address range from 1 Gbyte to 2 Gbyte. See Section 4.4.5, “60x Bus Slave Support,” for more information. 0 Local bus slave access disabled. 1 Local bus slave access enabled. When the local bus slave is accessed, it is responsible for generating $\overline{\text{AACK}}$ and $\overline{\text{TA}}$ to terminate the address and data tenure.

Table 3-28. Bit Settings for Processor Interface Configuration Register 1—0xA8 (Continued)

Bit	Name	Reset Value	Description
12	FLASH_WR_EN	0	Flash write enable. This bit controls whether the MPC105 allows write operations to Flash ROM. 0 Flash writes disabled. 1 Flash writes enabled.
11	MCP_EN	0	Machine check enable. This bit controls whether the MPC105 will assert $\overline{\text{MCP}}$ upon detecting an error. See Chapter 9, “Error Handling,” for more information. 0 Machine check disabled 1 Machine check enabled
10	TEA_EN	0	Transfer error enable. This bit controls whether the MPC105 will assert $\overline{\text{TEA}}$ upon detecting an error. See Chapter 9, “Error Handling,” for more information. 0 Transfer error disabled 1 Transfer error enabled
9	CF_DPARK	0	Data bus park. This bit indicates whether the 60x processor is parked on the data bus. 0 60x processor is not parked on the data bus. 1 60x processor is parked on the data bus.
8	—	0	Reserved
7	NO_PORT_REGS	0	This bit indicates the presence or absence of the external configuration registers. See Section 3.2.9, “External Configuration Registers,” for more information. 0 The system implements the external configuration registers. The MPC105 treats accesses to the external registers as PCI I/O cycles. 1 There are no physical registers for the external configuration registers. The MPC105 services read accesses to the external registers. Note that writes to these registers are always shadowed regardless of the state of this bit.
6	ST_GATH_EN	0	This bit enables/disables store gathering of writes from the processor to PCI memory space. See Chapter 8, “Internal Control,” for more information. 0 Store gathering disabled 1 Store gathering enabled
5	LE_MODE	0	This bit controls the endian mode of the MPC105. Note that this bit is also accessible from the external configuration register at 0x092. See Appendix B, “Bit and Byte Ordering,” for more information. 0 Big-endian mode 1 Little-endian mode

Table 3-28. Bit Settings for Processor Interface Configuration Register 1—0xA8 (Continued)

Bit	Name	Reset Value	Description
4	CF_LOOP_SNOOP	1	This bit causes the MPC105 to repeat a snoop operation (due to a PCI-to-memory transaction) until it is not retried ($\overline{\text{ARTRY}}$ input asserted) by the processor(s) or the L2 cache. Generally, this bit indicates whether the system implements snoop looping using the high-priority snoop request ($\overline{\text{HP\_SNP\_REQ}}$) signal on the 601. See *PowerPC 601 RISC Microprocessor User's Manual* for more information. 0 Snoop looping disabled 1 Snoop looping enabled
3	CF_APARK	0	This bit indicates whether the 60x address bus is parked. See Section 4.3.1, "Address Arbitration," for more information. 0 Indicates that no processor is parked on the 60x address bus 1 Indicates that a processor is parked on the 60x address bus
2	Speculative PCI Reads	0	This bit controls speculative PCI reads from memory. See Chapter 8, "Internal Control," for more information. 0 Indicates that speculative reads are disabled. 1 Indicates that speculative reads are enabled
1–0	CF_L2_MP	00	L2/multiprocessor configuration. These bits indicate the uni/multiprocessor and L2 configuration. 00 Uniprocessor (no L2 cache) configuration 01 Write-through L2 cache configuration 10 Write-back L2 cache configuration 11 Multiprocessor (two 60x processors on the 60x bus) configuration

See Figure 3-31 and Table 3-29 for processor interface configuration register 2 bit settings.

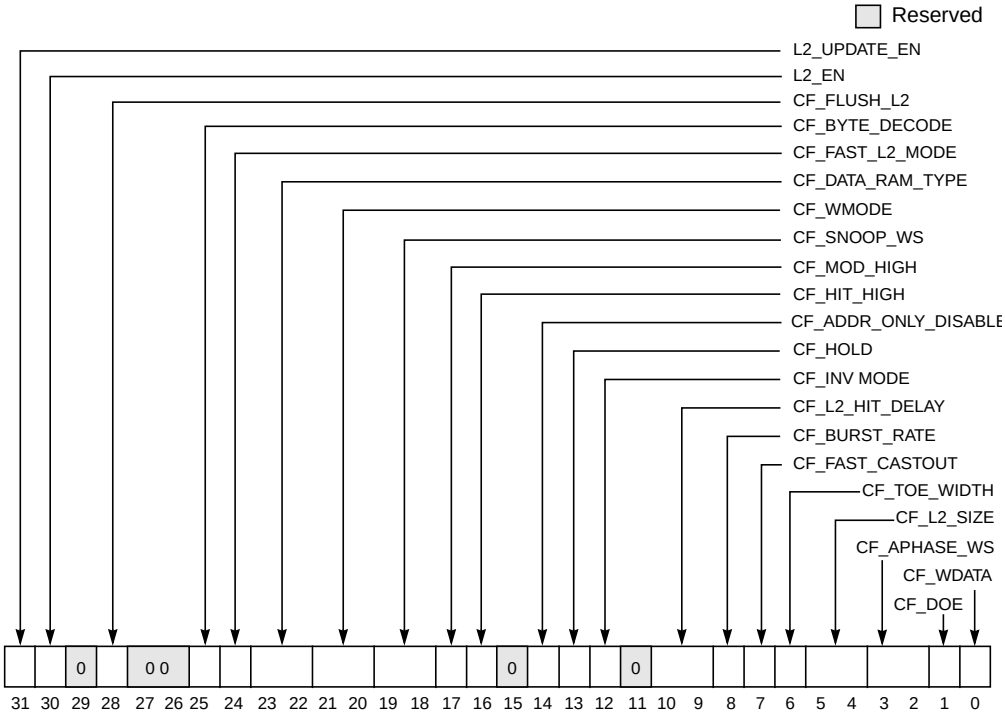


Figure 3-31. Processor Interface Configuration Register 2

Table 3-29. Bit Settings for Processor Interface Configuration Register 2—0xAC

Bit	Name	Reset Value	Description
31	L2_UPDATE_EN	0	This bit controls how the L2 cache handles cache misses. Note that this bit is also accessible from the external configuration register at 0x81C. 0 L2 cache misses bypass the L2 cache. L2 cache contents are not updated. 1 L2 cache misses are serviced by the L2 cache.
30	L2_EN	0	This bit enables/disables the L2 cache. The L2 cache is only enabled if both this bit and PICR1[CF_L2_MP] signify that there is an L2 cache in the system. Note that this bit is also accessible from the external configuration register at 0x81C. 0 The L2 cache is disabled. However, the tags are not invalidated. No L2 snoop operations or data updates are performed while this bit is cleared. 1 The L2 cache is enabled. Indicates normal L2 cache operation.
29	—	0	Reserved

Table 3-29. Bit Settings for Processor Interface Configuration Register 2—0xAC (Continued)

Bit	Name	Reset Value	Description
28	CF_FLUSH_L2	0	L2 cache flush. The transition on this bit from 0 to 1 initiates an L2 flush and invalidate operation, provided PICR2[L2_EN] = 0b0. Note that this bit is also accessible from the external configuration register at 0x81C. 0 Normal cache operation. 1 The transition from 0 to 1 indicates that the L2 cache should write all modified lines to memory and mark all lines as invalid.
27–26	—	00	Reserved
25	CF_BYTE_DECODE	0	On-chip byte-write decode enable. This bit controls whether byte-write decoding for the L2 cache is performed by the MPC105 or by external logic. Note that if external decoding is used, CF_WMODE (PICR2, bits 21 and 20) must be set accordingly. See Chapter 5, “Secondary Cache Interface,” for more information. 0 L2 byte-write decode is performed external to the MPC105. 1 L2 byte-write decode is performed by the MPC105.
24	CF_FAST_L2_MODE	0	Fast L2 mode enable. This bit enables/disables fast L2 mode timing. Fast L2 mode timing allows for no dead cycles between consecutive burst reads that hit in the L2 cache. Note that the 601 and 603 are not capable of using fast L2 mode timing. 0 Enable fast L2 mode timing 1 Disable fast L2 mode timing
23–22	CF_DATA_RAM_TYPE	00	L2 data RAM type. These bits indicate the type of data RAM used for the L2 cache. 00 Synchronous burst SRAM 01 Reserved 10 Asynchronous SRAM 11 Reserved
21–20	CF_WMODE	00	SRAM write timing. These bits control L2 data RAM write timing. For an asynchronous SRAM cache configuration, only mode 01 is valid. See Chapter 5, “Secondary Cache Interface,” for more information. 00 Reserved 01 Normal $\overline{\text{DWE}}$ timing 10 Delayed write timing. When performing an L2 cache write, the MPC105 issues the L2 cache control signals, but delays the assertion of $\overline{\text{TA}}$ by one cycle to allow for external byte write decoding. Not valid for asynchronous SRAMs. 11 Early write timing. The MPC105 speculatively asserts $\overline{\text{DWE}}$ one cycle earlier than the other L2 data RAM control signals for better write performance when using external byte write decoding logic. Not valid for asynchronous SRAMs.

Table 3-29. Bit Settings for Processor Interface Configuration Register 2—0xAC (Continued)

Bit	Name	Reset Value	Description
19-18	CF_SNOOP_WS	11	Snoop wait states. These bits control the minimum number of wait states for the address phase in a snoop cycle. 00 0 clock cycles 01 1 clock cycle 10 2 clock cycles 11 3 clock cycles
17	CF_MOD_HIGH	0	Cache modified signal polarity. This bit controls the active state of the $\overline{\text{DIRTY\_IN/BR1}}$, $\overline{\text{DIRTY\_OUT/BG1}}$, and TV L2 cache signals. Note that the state of this bit has no effect on the polarity of the $\overline{\text{DIRTY\_IN/BR1}}$ and $\overline{\text{DIRTY\_OUT/BG1}}$ signals when used as secondary processor signals. 0 The input signals $\overline{\text{TV}}$ and $\overline{\text{DIRTY\_IN/BR1}}$ are active low and the output signals $\overline{\text{TV}}$ and $\overline{\text{DIRTY\_OUT/BG1}}$ are active low. 1 The input signals TV and DIRTY_IN/$\overline{\text{BR1}}$ are active high and the output signals TV and DIRTY_OUT/$\overline{\text{BG1}}$ are active high.
16	CF_HIT_HIGH	0	Cache $\overline{\text{HIT}}$ signal polarity.This bit controls the active state of the $\overline{\text{HIT}}$ secondary cache signal. 0 $\overline{\text{HIT}}$ is active low. 1 HIT is active high
15	—	0	Reserved
14	CF_ADDR_ONLY_DISABLE	0	Set when the L2 is enabled for normal L2 operation. 0 The L2 responds to CLEAN, FLUSH, and KILL transactions. 1 The L2 ignores CLEAN, FLUSH, and KILL transactions.
13	CF_HOLD	0	L2 tag address hold. This bit controls the hold time of the address, TV, and $\overline{\text{DIRTY\_OUT/BG1}}$ signals with respect to the rising edge (negation) of $\overline{\text{TWE}}$. 0 Synchronous tag RAM configurations. No hold time (0 clocks). 1 Asynchronous tag RAM configurations. Tag address, TV, and $\overline{\text{DIRTY\_OUT/BG1}}$ are held valid for one clock after $\overline{\text{TWE}}$ is negated.
12	CF_INV_MODE	0	L2 invalidate mode enable. When L2 invalidate mode is enabled, any 60x transaction on the 60x bus causes the L2 to invalidate the tag entry indexed by the 60x address. Invalidate mode is used to initialize the tag contents. See Chapter 5, “Secondary Cache Interface,” for more information. 0 L2 invalidate mode is disabled 1 L2 invalidate mode is enabled
11	—	0	Reserved

Table 3-29. Bit Settings for Processor Interface Configuration Register 2—0xAC (Continued)

Bit	Name	Reset Value	Description
10–9	CF_L2_HIT_DELAY	11	L2 cache hit delay. These bits control the number of clock cycles from the assertion of $\overline{TS}$ until $\overline{HIT}$ is valid. 00 Reserved 01 1 clock cycle 10 2 clock cycles 11 3 clock cycles
8	CF_BURST_RATE	0	L2 cache burst rate. This bit controls the burst rate of the data beats for both L2 burst read and burst write transactions. For asynchronous SRAM configurations, CF_BURST_RATE should be set to two clocks (0b1). 0 1 clock 1 2 clocks
7	CF_FAST_CASTOUT	0	Fast L2 castout timing 0 Normal L2 castout timing 1 Fast L2 castout timing for improved performance when using synchronous write TAG RAMs
6	CF_TOE_WIDTH	0	$\overline{TOE}$ active pulse width. This bit controls the number of clock cycles that $\overline{TOE}$ is held asserted during L2 tag cast-out/copy-back read operations. 0 2 clock cycles 1 3 clock cycles
5–4	CF_L2_SIZE	00	L2 cache size. These bits indicate the size of the L2 cache. 00 256 Kbytes 01 512 Kbytes 10 1 Mbyte 11 Reserved
3–2	CF_APHASE_WS	11	Address phase wait states. These bits control the minimum number of address phase wait states (in clock cycles) for processor-initiated operations. 00 0 clock cycles 01 1 clock cycle 10 2 clock cycles 11 3 clock cycles
1	CF_DOE	0	L2 first data read access timing. For synchronous burst SRAM L2 configurations, this bit controls the number of clock cycles from $\overline{DOE}$ asserted-to-valid data on the first read access. 0 1 clock cycle 1 2 clock cycles For asynchronous SRAM L2 configurations, this bit controls the first data access timing of pipelined read cycles. 0 3-2-2-2/2-2-2-2 timing (2 clocks) 1 3-2-2-2/3-2-2-2 timing (3 clocks)

Table 3-29. Bit Settings for Processor Interface Configuration Register 2—0xAC (Continued)

Bit	Name	Reset Value	Description
0	CF_WDATA	0	L2 first data write setup time. For synchronous burst SRAM configurations, this bit indicates the delay from data bus grant to write data valid: 1 2 clocks 0 1 clock (default) For asynchronous SRAMs, indicates the $\overline{\text{DWE}}$ timing: 1 $\overline{\text{DWE}}$ is negated when $\overline{\text{TA}}$ is negated (simultaneously) 0 $\overline{\text{DWE}}$ is negated at the falling clock edge of the cycle when $\overline{\text{TA}}$ is asserted.

3.2.8 Alternate OS-Visible Parameters Registers

The alternate OS-visible parameters registers 1 and 2 provide operating systems an alternate means to access some of the bits in PICR1. These registers are 1 byte each. See Figure 3-32 and Table 3-30 for alternate OS-visible parameters register 1 bit settings.

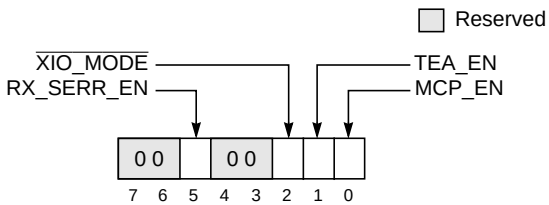


Figure 3-32. Alternate OS-Visible Parameters Register 1

Table 3-30. Bit Settings for Alternate OS-Visible Parameters Register 1—0xBA

Bit	Name	Reset Value	Description
7–6	—	00	Reserved
5	RX_SERR_EN	0	This bit controls whether the MPC105 recognizes the assertion of $\overline{\text{SERR}}$ by another PCI device. 0 The MPC105 ignores the assertion of $\overline{\text{SERR}}$ by another PCI device. 1 The MPC105 recognizes the assertion of $\overline{\text{SERR}}$ by another PCI device.
4–3	—	00	Reserved
2	$\overline{\text{XIO\_MODE}}$	1	Address map A discontiguous/contiguous mode. This bit controls whether address map A uses the discontiguous or contiguous I/0 mode. See Section 3.1.1, “Address Map A,” for more information. Note that this bit is the inverse of bit 19 of PICR1. 0 Discontiguous mode 1 Contiguous mode

Table 3-30. Bit Settings for Alternate OS-Visible Parameters Register 1—0xBA (Continued)

Bit	Name	Reset Value	Description
1	TEA_EN	0	Transfer error enable. This bit controls whether the MPC105 will assert $\overline{\text{TEA}}$ upon detecting an error. Note that this bit is the same as bit 10 of PICR1. 0 Transfer error disabled 1 Transfer error enabled
0	MCP_EN	0	Machine check enable. This bit controls whether the MPC105 will assert $\overline{\text{MCP}}$ upon detecting an error. Note that this bit is the same as bit 11 of PICR1. 0 Machine check disabled 1 Machine check enabled

See Figure 3-33 and Table 3-31 for alternate OS-visible parameter register 2 bit settings.

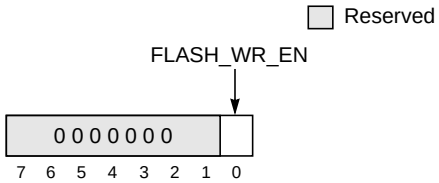


Figure 3-33. Alternate OS-Visible Parameter Register 2

Table 3-31. Bit Settings for Alternate OS-Visible Parameters Register 2—0xBB

Bit	Name	Reset Value	Description
7–1	—	00	Reserved
0	FLASH_WR_EN	0	Flash write enable. This bit controls whether the MPC105 allows write operations to Flash ROM. Note that this bit is the same as bit 12 of PICR1. 0 Flash write disabled 1 Flash write enabled

3.2.9 External Configuration Registers

Certain configuration bits can be accessed by reading or writing addresses 0x8000_0092, 0x8000_081C, or 0x8000_0850 in address map A. These are compatible with the example system described by the *PowerPC Reference Platform Specification*. These external configuration registers should only be accessed as a 1-byte quantity, even though the other bytes in the double word are reserved.

PICR1[NO_PORT_REGS] controls access to these registers. If NO_PORT_REGS is set, then the MPC105 handles all accesses to the external configuration registers. If NO_PORT_REGS is cleared, then the MPC105 treats read accesses to the external configuration registers as PCI I/O read cycles, and write accesses are treated as PCI I/O writes. However, writes to the external configuration registers are always shadowed in PICR1 regardless of the state of NO_PORT_REGS. For example, if bit 1 of the data being written to address 0x8000_0092 is set, then when the configuration write access completes on the PCI bus, the MPC105 enters little-endian mode and PICR1[LE_MODE] is set.

See Figure 3-34 and Table 3-32 for external configuration register 1 bit settings.

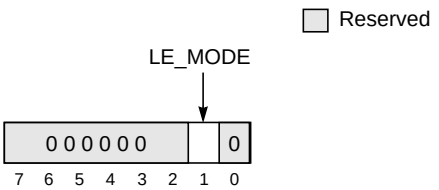


Figure 3-34. External Configuration Register 1—0x8000_0092

Table 3-32. Bit Settings for External Configuration Register 1—0x8000_0092

Bit	Name	Reset Value	Description
7–2	—	All 0s	Reserved
1	LE_MODE	0	This bit controls the endian mode of the MPC105. Note that this bit corresponds to bit 5 of PICR1. See Appendix B, “Bit and Byte Ordering,” for more information. 0 Big-endian mode 1 Little-endian mode
0	—	0	Reserved

See Figure 3-35 and Table 3-32 for external configuration register 2 bit settings.

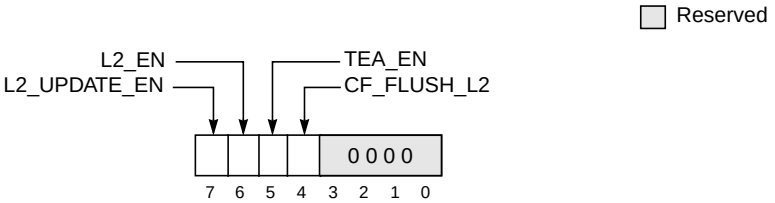


Figure 3-35. External Configuration Register 2—0x8000_081C

Table 3-33. Bit Settings for External Configuration Register 2—0x8000_081C

Bit	Name	Reset Value	Description
7	L2_UPDATE_EN	1	This bit controls how the L2 cache handles cache misses. Note that this bit corresponds to bit 31 of PICR2. 0 L2 cache misses bypass the L2 cache. L2 cache contents are not updated. 1 L2 cache misses are serviced by the L2 cache.
6	L2_EN	1	This bit enables/disables the L2 cache. The L2 cache is only enabled if both this bit and PICR1[CF_L2_MP] signify that there is an L2 cache in the system. Note that this bit corresponds to bit 30 of PICR2. 0 The L2 cache is disabled. However, the tags are not invalidated. No L2 snoop operations or data updates are performed while this bit is negated. 1 The L2 cache is enabled. Indicates normal L2 cache operation.
5	TEA_EN	0	Transfer error enable. This bit controls whether the MPC105 will assert $\overline{\text{TEA}}$ upon detecting an error. Note that this bit corresponds to bit 10 of PICR1. 0 Transfer error disabled 1 Transfer error enabled
4	CF_FLUSH_L2	0	L2 cache flush. The transition on this bit from 0 to 1 initiates an L2 flush and invalidate operation, provided PICR2[L2_EN] = 0b0. Note that this bit corresponds to bit 28 of PICR2. 0 Normal cache operation. 1 The transition from 0 to 1 indicates that the L2 cache should write all modified lines to memory and mark all lines as invalid.
3–0	—	All 0s	Reserved

See Figure 3-36 and Table 3-32 for external configuration register 3 bit settings.

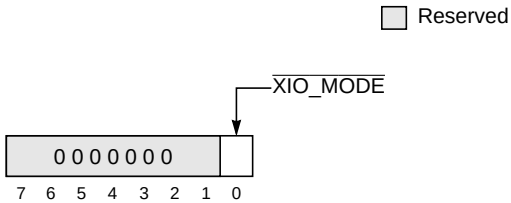


Figure 3-36. External Configuration Register 3—0x8000_0850

Table 3-34. Bit Settings for External Configuration Register 3—0x8000_0850

Bit	Name	Reset Value	Description
7–1	—	All 0s	Reserved
0	$\overline{\text{XIO\_MODE}}$	1	Address map A discontiguous/contiguous mode. This bit controls whether address map A uses the discontiguous or contiguous I/0 mode. See Section 3.1.1, “Address Map A,” for more information. Note that this bit is the inverse of bit 19 of PICR1. 0 Discontiguous mode 1 Contiguous mode

Chapter 4
Processor Bus Interface

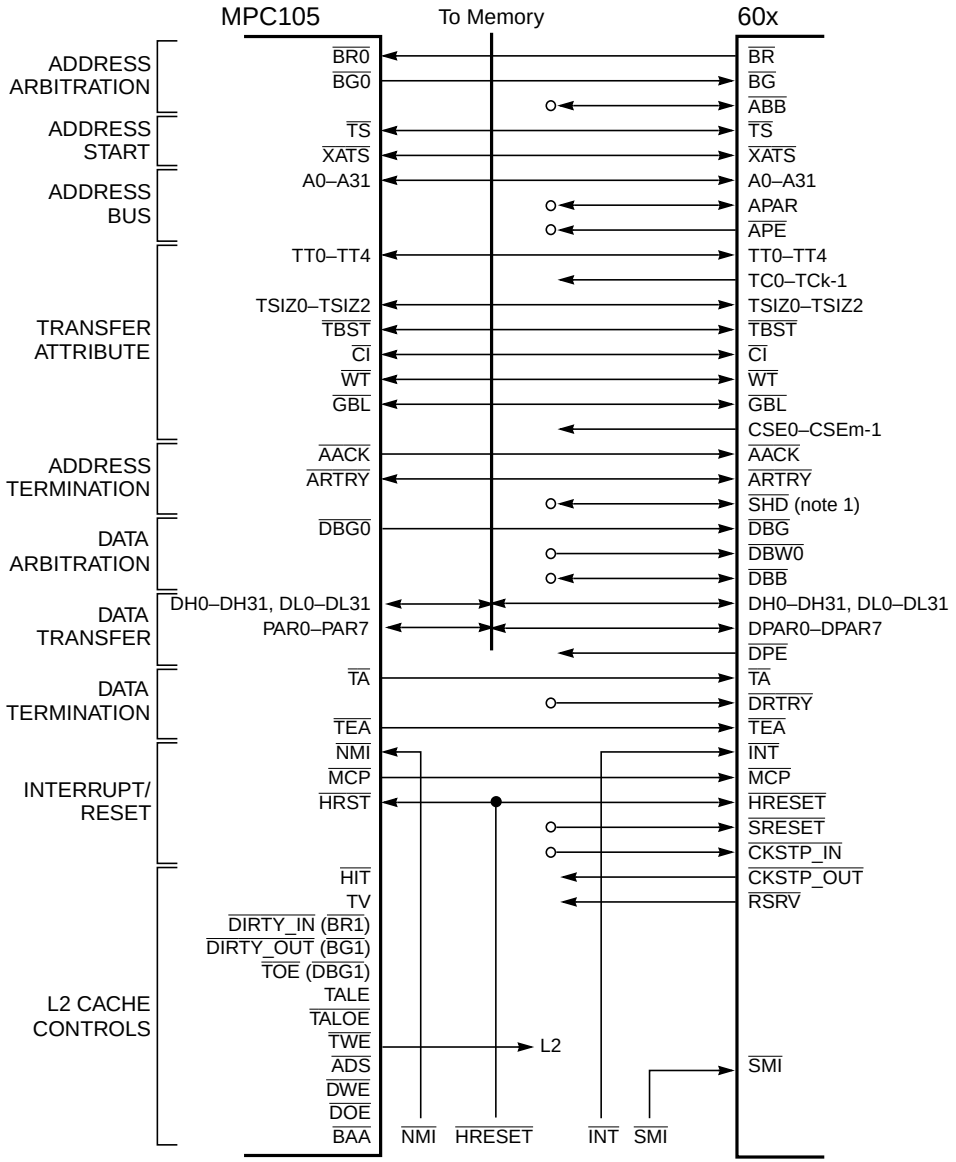
The MPC105 provides flexible support for system designs using the PowerPC 601, PowerPC 603, and PowerPC 604 microprocessors via the processor (60x) bus interface. The MPC105's 60x bus interface provides a 32-bit address bus and a configurable 32- or 64-bit data bus that supports both single-beat and burst data transfers. The address and data buses support synchronous, one-level pipelined transactions. The MPC105's 60x bus interface can be configured to support a single processor, a single processor with a secondary cache, or two processors.

4.1 MPC105 Processor Bus Configuration

The figures in the following sections show how the MPC105 can be connected to support a single processor with an optional L2 cache or a second processor. The term "alternate bus master" is used to refer to the L2 cache or second processor attached to the MPC105 in the following sections.

4.1.1 Single-Processor System Configuration

The MPC105 can be connected to a single 601, 603, or 604 as shown in Figure 4-1. This configuration supports the addition of an L2 cache, and the MPC105 will snoop bus operations to maintain coherency between the primary cache in the processor, the optional L2 cache, and main memory.



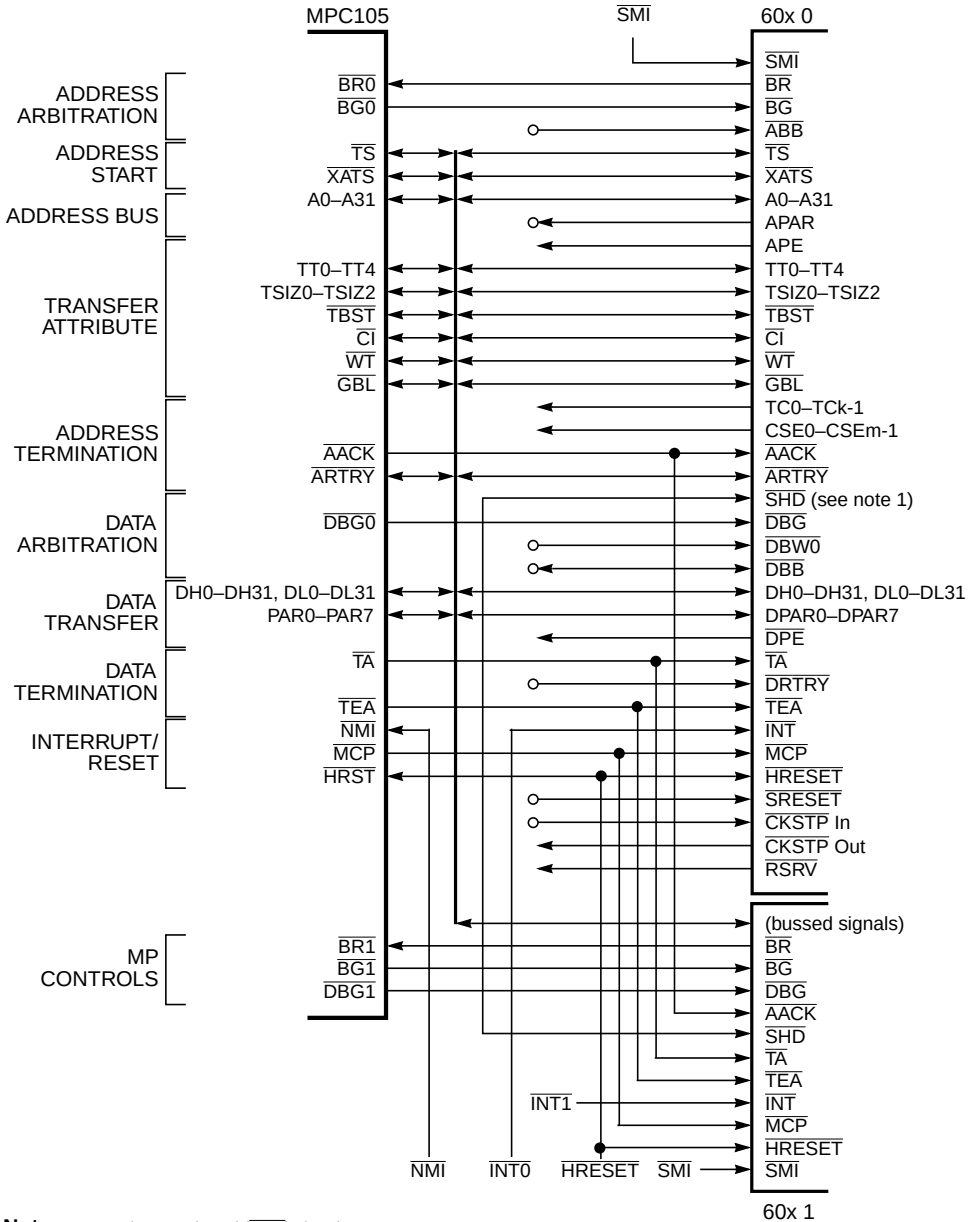
Notes: 1. 603 has no shared (SHD) signal.

○ All bidirectional control signals should have pull-up resistors tied to VDD through a resistor.

Figure 4-1. Single-Processor Configuration with Optional L2 Cache

4.1.2 Multiprocessor System Configuration

Instead of a single processor and an L2 cache, the MPC105 can also be configured to support two 601s, 603s, or 604s. When operating in a multiprocessor configuration, the MPC105 will snoop bus operations and maintain coherency between the two primary caches and main memory. Figure 4-2 shows how two processors are attached to the MPC105.



Notes: 1. 603 has no shared (SHD) signal.

o All bidirectional control signals should have pull-up resistors tied to VDD through a resistor.

Figure 4-2. Multiprocessor Configuration

4.2 Processor Bus Protocol Overview

60x bus accesses are divided into address and data tenures. Each tenure has three phases—bus arbitration, transfer, and termination. Figure 4-3 shows that the address and data tenures are distinct from one another and that both consist of three phases—arbitration, transfer, and termination. Address and data tenures are independent (indicated in Figure 4-3 by the fact that the data tenure begins before the address tenure ends), which allows split-bus transactions to be implemented at the system level in multiprocessor systems. Figure 4-3 shows a data transfer that consists of a single-beat transfer of as many as 64 bits. Four-beat burst transfers of 32-byte cache lines require data transfer termination signals for each beat of data.

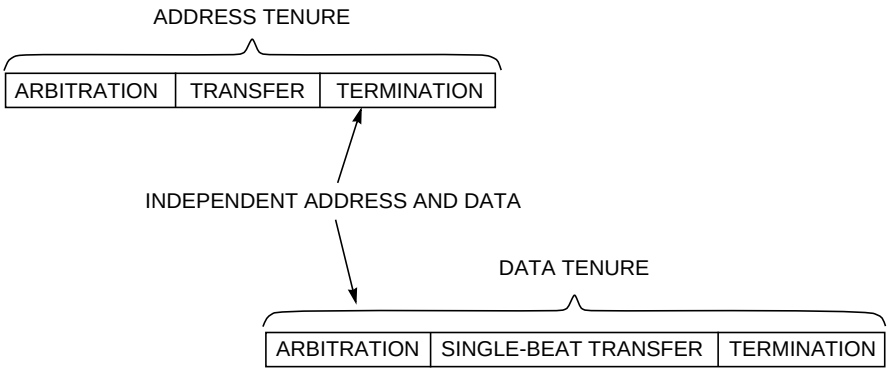


Figure 4-3. Overlapping Tenures on the 60x Bus for a Single-Beat Transfer

The basic functions of the address and data tenures are as follows:

- Address tenure
  - — Arbitration: During arbitration, address bus arbitration signals are used to gain mastership of the address bus.
  - — Transfer: After a bus master is granted the address bus, it transfers the address. The address signals and the transfer attribute signals control the address transfer. The address parity and address parity error signals ensure the integrity of the address transfer.
  - — Termination: After the address transfer, the system signals that the address tenure is complete or that it must be repeated.
- Data tenure
  - — Arbitration: Following the initiation of the address tenure, the bus master arbitrates for mastership of the data bus.
  - — Transfer: After the bus master is granted the data bus, it samples the data bus for read operations or drives the data bus for write operations.

— Termination: Data termination signals are required after each data beat in a data transfer. Note that in a single-beat transaction, the data termination signals also indicate the end of the tenure, while in burst accesses, the data termination signals apply to individual beats and indicate the end of the tenure only after the final data beat.

4.2.1 MPC105 Arbitration

Arbitration for both address and data bus mastership is performed by the MPC105 through the use of the following signals. Note that the MPC105 controls bus access through the use of bus request and bus grant signals, and determines the state of the address and data bus busy signals by monitoring the $\overline{\text{DBG0}}$, $\overline{\text{DBG1}}$, $\overline{\text{TS}}$, $\overline{\text{AACK}}$, and $\overline{\text{TA}}$ signals.

The following signals are used for address bus arbitration:

- $\overline{\text{BR0}}$ and $\overline{\text{BR1}}$(bus request)—Assertion indicates that a bus master is requesting mastership of the address bus.
- $\overline{\text{BG0}}$ and $\overline{\text{BG1}}$(bus grant)—Assertion indicates that a bus master may, with the proper qualification, assume mastership of the address bus. A qualified bus grant occurs when $\overline{\text{BG}}$ is asserted and $\overline{\text{ARTRY}}$ is negated.

The following signals are used for data bus arbitration:

- $\overline{\text{DBG0}}$ and $\overline{\text{DBG1}}$(data bus grant)—Indicates that a bus master may, with the proper qualification, assume mastership of the data bus. A qualified data bus grant occurs when $\overline{\text{DBG}}$ is asserted while $\overline{\text{ARTRY}}$ is negated.

For more detailed information on the arbitration signals, refer to Chapter 2, “Signal Descriptions.”

4.2.2 Address Pipelining and Split-Bus Transactions

The 60x bus protocol provides independent address and data bus capability to support pipelined and split-bus transaction system organizations. Address pipelining allows the address tenure of a new bus transaction to begin before the data tenure of the current transaction has finished.

While this capability does not inherently reduce memory latency, support for address pipelining and split-bus transactions can greatly improve effective bus/memory throughput. For this reason, these techniques are most effective in shared-memory multiprocessor implementations where bus bandwidth is an important measurement of system performance.

External arbitration (as provided by the MPC105) is required in systems in which multiple devices must compete for the system bus. The MPC105 affects pipelining by regulating address bus grants ($\overline{BG0}$ and $\overline{BG1}$), data bus grants ($\overline{DBG0}$ and $\overline{DBG1}$), and the address acknowledge ($\overline{AACK}$) signal. One-level pipelining is implemented by the MPC105 by asserting $\overline{AACK}$ to the current address bus master and granting mastership of the address bus to the next requesting master before the current data bus tenure has completed. Two address tenures can occur before the current data bus tenure completes.

4.3 Address Tenure Operations

This section describes the three phases of the address tenure—address bus arbitration, address transfer, and address termination.

4.3.1 Address Arbitration

The MPC105 provides arbitration for the processor address bus. The bus request ($\overline{BR0}$ and $\overline{BR1}$) for the processor and the alternate master (in a multiprocessor configuration) are external inputs to the arbiter. The bus grant signals for the processor and alternate master ($\overline{BG0}$ and $\overline{BG1}$) are outputs. In addition to the external signals, there are internal request and grant signals for snoop broadcast and L2 cast-out operations. If the MPC105 needs to perform a snoop broadcast or L2 cast-out operation, it asserts the internal bus request. The arbiter negates the external bus grants and asserts the internal bus grant for those operations. Bus accesses are prioritized, with processor L1 cache copy-back operations having the highest priority. L2 cast-out operations have the next highest priority, followed by snoop and 60x bus requests. Bus requests signaled by the assertion of $\overline{BR0}$ and $\overline{BR1}$ have rotating priority, unless an L1 cache copy-back operation is required. In these cases, the MPC105 grants higher priority to the processor requesting the cache copy-back.

Address bus parking is supported by the MPC105 through the use of the PICR2[CF_APARK] bit. When this bit is set, the MPC105 parks the address bus (asserts the address bus grant signal in anticipation of an address bus request) to the 60x processor that most recently had mastership of the bus.

The processor and the alternate bus master qualify $\overline{BG}$ by sampling $\overline{ARTRY}$ in the negated state prior to taking address bus mastership. The negation of $\overline{ARTRY}$ during the address retry window (one cycle after the assertion of $\overline{AACK}$) indicates that no address retry is requested. The processor and the alternate bus master will not accept the address bus grant during the $\overline{ARTRY}$ cycle or the cycle following if an asserted $\overline{ARTRY}$ is detected. The 60x bus master that asserts $\overline{ARTRY}$ due to a modified cache block hit asserts its bus request during the cycle following the assertion of $\overline{ARTRY}$, and assumes the mastership of the bus for the cache block push when it is given a bus grant.

Figure 4-4 shows a series of address transfers to illustrate the transfer protocol when the MPC105 is configured with two processors. Initially processor 0 is parked on the bus with address bus grant asserted, which allows it to initiate an address bus tenure (by asserting $\overline{TS}$) without first having asserted address bus request. During the same clock cycle, the MPC105's internal bus request is asserted to request access to the 60x bus, thereby causing the negation of $\overline{BG0}$. Following processor 0's address tenure, the MPC105 takes the bus and initiates its address transaction. At the completion of the MPC105's address transaction, both $\overline{BR0}$ and $\overline{BR1}$ are asserted. Because processor 0 was the last processor to have mastership of the bus, the MPC105's arbiter grants mastership to processor 1.

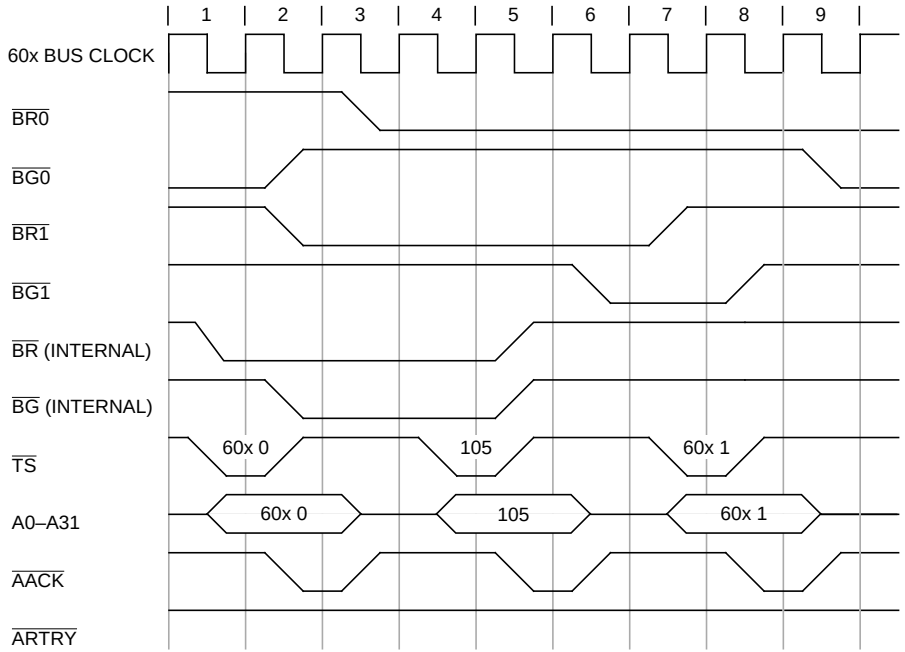


Figure 4-4. Address Bus Arbitration with Dual Processors

The MPC105 supports one level of address pipelining by asserting the $\overline{AACK}$ signal to the current bus master when its data tenure starts and by granting the address bus to the next requesting master before the current data bus tenure has completed. Address pipelining allows a new set of address and control signals to be decoded by the memory control hardware while the current data transaction finishes, thus improving data throughput. The MPC105 performs pipelined data bus operations strictly in order with the associated address operations. Figure 4-5 shows how address pipelining allows address tenures to overlap the associated data tenures.

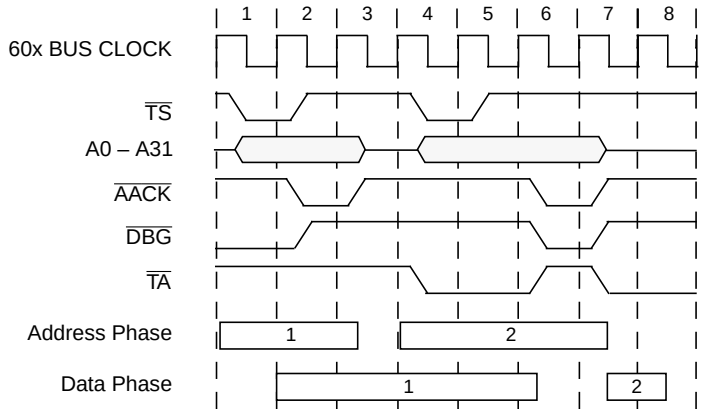


Figure 4-5. Address Pipelining

4.3.2 Address Transfer Attribute Signals

During the address transfer phase of an address tenure, the address of the bus operation to be performed is placed on address signals (A0–A31), along with the appropriate parity bits. In addition to the address signals, the bus master provides three other types of signals during the address transfer to indicate the type and size of the transfer; these are the transfer type (TT0–TT4), transfer size (TSIZ0–TSIZ2), and transfer burst ($\overline{\text{TBST}}$) signals. These signals are discussed in the following sections.

4.3.2.1 Transfer Type Signal Encodings

The transfer type signals define the nature of the transfer that is being requested. The transfer type encoding indicates whether the transfer is to be an address-only transaction or both address and data. These signals can be originated by both the address bus master or the MPC105, depending on the nature of the bus transaction. Transfer type signals originating from the MPC105 occur due to snoop operations caused by PCI bus accesses to memory. Table 4-1 describes the MPC105's response to transfer type signals driven by an address bus master on the 60x bus.

Table 4-1. MPC105 Responses to 60x Transfer Types

TT0–TT4	Bus Operation	Class of Operation	$\overline{\text{TEA}}$	MPC105 Response
01010	Read	Normal	—	Read, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.
01110	Read-with-intent-to-modify	Normal	—	Read, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.
11010	Read atomic	Normal	—	Read, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.

Table 4-1. MPC105 Responses to 60x Transfer Types (Continued)

TT0–TT4	Bus Operation	Class of Operation	$\overline{\text{TEA}}$	MPC105 Response
11110	Read-with-intent-to-modify-atomic	Normal	—	Read, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.
00010	Write w/Flush	Normal	—	Write, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.
00110	Write w/Kill	Normal	—	Write, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.
10010	Write w/Flush atomic	Normal	—	Write, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.
01000	**sync**	Normal	—	Address only, assert $\overline{\text{AACK}}$. Bus grant is negated until MPC105 buffers flushed.
10000	**eieio**	Normal	—	Address only, assert $\overline{\text{AACK}}$. Bus grant is negated until MPC105 buffers flushed.
01100	Kill	Normal	—	Address only operation, $\overline{\text{AACK}}$ is asserted.
01101	**icbi**	Normal	—	Address only operation, $\overline{\text{AACK}}$ is asserted.
01011	Read-with-no-intent-to-cache	Normal	—	Read, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.
00000	Clean	Normal	—	Address only operation. $\overline{\text{AACK}}$ is asserted, and MPC105 takes no further action.
00100	Flush	Normal	—	Address only operation. $\overline{\text{AACK}}$ is asserted, and MPC105 takes no further action.
11000	**tlbi**	Normal	—	Address only operation. $\overline{\text{AACK}}$ is asserted, and MPC105 takes no further action.
00001	**lwarx**, reservation set	Normal	—	Address only operation. $\overline{\text{AACK}}$ is asserted, and MPC105 takes no further action.
00101	**stwcx.**, reservation set	Normal	—	Address only operation. $\overline{\text{AACK}}$ is asserted, and MPC105 takes no further action.
01001	**tlbsync**	Normal	—	Address only operation. $\overline{\text{AACK}}$ is asserted, and MPC105 takes no further action.
10110	<reserved>	Error	—	Illegal operation; signals error. Address only operation. $\overline{\text{AACK}}$ is asserted.
00011	<reserved>	Error	—	Illegal operation; signals error. Address only operation. $\overline{\text{AACK}}$ is asserted.
00111	<reserved>	Error	—	Illegal operation, signals error. Address only operation. $\overline{\text{AACK}}$ is asserted.
01111	<reserved>	Error	—	Illegal operation; signals error. Address only operation. $\overline{\text{AACK}}$ is asserted.
1XXX1	<reserved for customer>	Error	—	Illegal operation; signals error. Address only operation. $\overline{\text{AACK}}$ is asserted.
0100-	Direct-store load request	Error	—	Illegal operation; signals error. Address only operation. $\overline{\text{AACK}}$ is asserted.

Table 4-1. MPC105 Responses to 60x Transfer Types (Continued)

TT0–TT4	Bus Operation	Class of Operation	$\overline{\text{TEA}}$	MPC105 Response
10100	Graphic write (**ecowx**)	Error	$\overline{\text{TEA}}$ asserted	Illegal operation; signals error. $\overline{\text{AACK}}$ is asserted; $\overline{\text{TEA}}$ will be asserted if enabled. If $\overline{\text{TEA}}$ is not enabled, data tenure terminated by $\overline{\text{TA}}$.
11100	Graphic read (**eciwx**)	Error	$\overline{\text{TEA}}$ asserted	Illegal operation; signals error. $\overline{\text{AACK}}$ is asserted; $\overline{\text{TEA}}$ will be asserted if enabled. If $\overline{\text{TEA}}$ is not enabled, data tenure terminated by $\overline{\text{TA}}$.
0101-	Direct-store load immediate	Error	$\overline{\text{TEA}}$ asserted	Illegal operation; signals error. $\overline{\text{AACK}}$ is asserted; $\overline{\text{TEA}}$ will be asserted if enabled. If $\overline{\text{TEA}}$ is not enabled, data tenure terminated by $\overline{\text{TA}}$.
0111-	Direct-store load last	Error	$\overline{\text{TEA}}$ asserted	Illegal operation; signals error. $\overline{\text{AACK}}$ is asserted; $\overline{\text{TEA}}$ will be asserted if enabled. If $\overline{\text{TEA}}$ is not enabled, data tenure terminated by $\overline{\text{TA}}$.
0001-	Direct-store store immediate	Error	$\overline{\text{TEA}}$ asserted	Illegal operation; signals error. $\overline{\text{AACK}}$ is asserted; $\overline{\text{TEA}}$ will be asserted if enabled. If $\overline{\text{TEA}}$ is not enabled, data tenure terminated by $\overline{\text{TA}}$.
0011-	Direct-store store last	Error	$\overline{\text{TEA}}$ asserted	Illegal operation; signals error. $\overline{\text{AACK}}$ is asserted; $\overline{\text{TEA}}$ will be asserted if enabled. If $\overline{\text{TEA}}$ is not enabled, data tenure terminated by $\overline{\text{TA}}$.

The MPC105 propagates snoop broadcast operations to the 60x bus in response to PCI bus memory accesses. The snoop broadcasts generated by the MPC105 are caused by PCI bus operations and are identified as burst, cacheable, write-back, and global accesses. Table 4-2 describes the transfer type encodings generated by the MPC105.

Table 4-2. Transfer Types Generated by the MPC105

TT0–TT4 (Driven by MPC105)	60x Bus Operation	MPC105 Response
00010	Burst-write-with-flush	Generated in response to PCI writes to memory
10010	Burst-write-with-flush-atomic	Generated in response to locked PCI writes to memory
00110	Burst-write-with-kill	Generated in response to nonlocked/locked PCI writes with invalidate to memory
11110	Burst-RWITM-atomic	Read-with-intent-to-modify—generated for locked PCI reads
01010	Burst-read	Generated in response to nonlocked PCI reads to memory

4.3.2.2 $\overline{\text{TBST}}$ and TSIZ0–TSIZ2 Signals and Size of Transfer

The transfer size (TSIZ0–TSIZ2) signals, in conjunction with the transfer burst ($\overline{\text{TBST}}$) signal, indicate the size of the requested data transfer, as shown in Table 4-3. These signals may be used along with address bits A29–A31 to determine which portion of the data bus contains valid data for a write transaction or which portion of the bus should contain valid data for a read transaction. The 60x processors use the eight-word burst transactions for transfer of cache blocks. For these transactions, the TSIZ0–TSIZ2 signals are encoded as 0b010, and address bits A27–A28 determine which double-word transfer should occur first.

The MPC105 supports critical-word-first burst transactions (double-word-aligned) from the 60x processor. The MPC105 transfers this double word of data first, followed by double words from increasing addresses, wrapping back to the beginning of the eight-word block as required. L2 cast-out operations always start at eight-word block boundaries.

Table 4-3. MPC105 Transfer Size Encodings

$\overline{\text{TBST}}$	TSIZ0	TSIZ1	TSIZ2	Transfer Size
Asserted	0	1	0	Eight-word burst
Negated	0	0	0	Eight bytes
Negated	0	0	1	One byte
Negated	0	1	0	Two bytes
Negated	0	1	1	Three bytes
Negated	1	0	0	Four bytes
Negated	1	0	1	Five bytes
Negated	1	1	0	Six bytes
Negated	1	1	1	Seven bytes

4.3.2.3 Burst Ordering During Data Transfers

During burst data transfer operations, 32 bytes of data (one cache line) are transferred to or from the cache in order. Burst write transfers are always performed zero-double-word-first. However, since burst reads are performed critical-double-word-first, a burst-read transfer may not start with the first double word of the cache line, and the cache line fill operation may wrap around the end of the cache line.

Table 4-4 describes the burst ordering when the MPC105 is configured with a 64-bit data bus.

Table 4-4. Burst Ordering—64-Bit Data Bus

Data Transfer	For Starting Address:			
	A27–A28 = 00	A27–A28 = 01	A27–A28 = 10	A27–A28 = 11
First data beat	DW0	DW1	DW2	DW3
Second data beat	DW1	DW2	DW3	DW0
Third data beat	DW2	DW3	DW0	DW1
Fourth data beat	DW3	DW0	DW1	DW2

Note: The A29–A31 signals are always 0b000 for burst transfers by the MPC105.
"U" and "L" represent the upper and lower word of the double word respectively.

Table 4-5 describes the burst ordering when the MPC105 is configured with a 32-bit data bus.

Table 4-5. Burst Ordering—32-Bit Data Bus

Data Transfer	For Starting Address:			
	A27–A28 = 00	A27–A28 = 01	A27–A28 = 10	A27–A28 = 11
First data beat	DW0-U	DW1-U	DW2-U	DW3-U
Second data beat	DW0-L	DW1-L	DW2-L	DW3-L
Third data beat	DW1-U	DW2-U	DW3-U	DW0-U
Fourth data beat	DW1-L	DW2-L	DW3-L	DW0-L
Fifth data beat	DW2-U	DW3-U	DW0-U	DW1-U
Sixth data beat	DW2-L	DW3-L	DW0-L	DW1-L
Seventh data beat	DW3-U	DW0-U	DW1-U	DW2-U
Eighth data beat	DW3-L	DW0-L	DW1-L	DW2-L

Notes: The A29–A31 signals are always 0b000 for burst transfers by the MPC105.
"U" and "L" represent the upper and lower word of the double word respectively.

4.3.2.4 Effect of Alignment on Data Transfers (64-Bit Data Bus)

Table 4-6 lists the aligned transfers that can occur to and from the MPC105 when configured with a 64-bit bus. These are transfers in which the data is aligned to an address that is an integer multiple of the size of the data. For example, Table 4-6 shows that 1-byte data is always aligned; however, for a 4-byte word to be aligned, it must be oriented on an address that is a multiple of 4.

Table 4-6. Aligned Data Transfers (64-Bit Data Bus)

Transfer Size	TSIZ0	TSIZ1	TSIZ2	A29–A31	Data Bus Byte Lane(s)							
					0	1	2	3	4	5	6	7
Byte	0	0	1	000	√	—	—	—	—	—	—	—
	0	0	1	001	—	√	—	—	—	—	—	—
	0	0	1	010	—	—	√	—	—	—	—	—
	0	0	1	011	—	—	—	√	—	—	—	—
	0	0	1	100	—	—	—	—	√	—	—	—
	0	0	1	101	—	—	—	—	—	√	—	—
	0	0	1	110	—	—	—	—	—	—	√	—
	0	0	1	111	—	—	—	—	—	—	—	√
Half word	0	1	0	000	√	√	—	—	—	—	—	—
	0	1	0	010	—	—	√	√	—	—	—	—
	0	1	0	100	—	—	—	—	√	√	—	—
	0	1	0	110	—	—	—	—	—	—	√	√
Word	1	0	0	000	√	√	√	√	—	—	—	—
	1	0	0	100	—	—	—	—	√	√	√	√
Double word	0	0	0	000	√	√	√	√	√	√	√	√

Notes: √ These entries indicate the byte portions of the requested operand that are read or written during that bus transaction.

— These entries are not required and are ignored during read transactions, they are driven with undefined data during all write transactions.

Data bus byte lane 0 corresponds to DH0–DH7, byte lane 7 corresponds to DL24–DL31.

The MPC105 supports misaligned memory operations, although their use may substantially degrade performance. Misaligned memory transfers address memory that is not aligned to the size of the data being transferred (such as, a word read of an odd byte address). The MPC105's processor bus interface supports misaligned transfers within a word (32-bit aligned) boundary, as shown in Table 4-7. Note that the 4-byte transfer in Table 4-7 is only one example of misalignment. As long as the attempted transfer does not cross a word boundary, the MPC105 can transfer the data to the misaligned address (for example, a half-word read from an odd byte-aligned address). An attempt to address data that crosses a word boundary requires two bus transfers to access the data.

Due to the performance degradations associated with misaligned memory operations, they are best avoided. In addition to the double-word straddle boundary condition, the processor's address translation logic can generate substantial exception overhead when the load/store multiple and load/store string instructions access misaligned data. It is strongly recommended that software attempt to align code and data where possible.

Table 4-7. Misaligned Data Transfers (4-Byte Examples)

Transfer Size (Four Bytes)	TSIZ0–TSIZ2	A29–A31	Data Bus Byte Lanes							
			0	1	2	3	4	5	6	7
Aligned	1 0 0	0 0 0	A	A	A	A	—	—	—	—
Misaligned—first access	0 1 1	0 0 1	—	A	A	A	—	—	—	—
second access	0 0 1	1 0 0	—	—	—	—	A	—	—	—
Misaligned—first access	0 1 0	0 1 0	—	—	A	A	—	—	—	—
second access	0 1 1	1 0 0	—	—	—	—	A	A	—	—
Misaligned—first access	0 0 1	0 1 1	—	—	—	A	—	—	—	—
second access	0 1 1	1 0 0	—	—	—	—	A	A	A	—
Aligned	1 0 0	1 0 0	—	—	—	—	A	A	A	A
Misaligned—first access	0 1 1	1 0 1	—	—	—	—	—	A	A	A
second access	0 0 1	0 0 0	A	—	—	—	—	—	—	—
Misaligned—first access	0 1 0	1 1 0	—	—	—	—	—	—	A	A
second access	0 1 0	0 0 0	A	A	—	—	—	—	—	—
Misaligned—first access	0 0 1	1 1 1	—	—	—	—	—	—	—	A
second access	0 1 1	0 0 0	A	A	A	—	—	—	—	—

Notes: A:Byte lane used
—:Byte lane not used

4.3.2.5 Effect of Alignment in Data Transfers (32-Bit Data Bus)

The aligned data transfer cases for 32-bit data bus mode are shown in Table 4-8. All of the transfers require a single data beat except for double-word cases which require two data beats.

Table 4-8. Aligned Data Transfers (32-Bit Data Bus)

Transfer Size	TSIZ0	TSIZ1	TSIZ2	A29–A31	Data Bus Byte Lane(s)							
					0	1	2	3	4	5	6	7
Byte	0	0	1	000	A	—	—	—	x	x	x	x
	0	0	1	001	—	A	—	—	x	x	x	x
	0	0	1	010	—	—	A	—	x	x	x	x
	0	0	1	011	—	—	—	A	x	x	x	x
	0	0	1	100	A	—	—	—	x	x	x	x
	0	0	1	101	—	A	—	—	x	x	x	x
	0	0	1	110	—	—	A	—	x	x	x	x
	0	0	1	111	—	—	—	A	x	x	x	x
Half word	0	1	0	000	A	A	—	—	x	x	x	x
	0	1	0	010	—	—	A	A	x	x	x	x
	0	1	0	100	A	A	—	—	x	x	x	x
	0	1	0	110	—	—	A	A	x	x	x	x
Word	1	0	0	000	A	A	A	A	x	x	x	x
	1	0	0	100	A	A	A	A	x	x	x	x
Double word	0	0	0	000	A	A	A	A	x	x	x	x
Second beat	0	0	0	000	A	A	A	A	x	x	x	x

Notes: A:Byte lane used

—:Byte lane not used

x: Byte lane not used in 32-bit bus mode

Misaligned data transfers when the MPC105 is configured with a 32-bit data bus operate in the same way as when configured with a 64-bit data bus, with the exception that only the DH0–DH31 data bus is used. See Table 4-9 for an example of a 4-byte misaligned transfer starting at each possible byte address within a double word.

Table 4-9. Misaligned 32-Bit Data Bus Transfer (4-Byte Examples)

Transfer Size (Four Bytes)	TSIZ0–TSIZ2	A29–A31	Data Bus Byte Lanes							
			0	1	2	3	4	5	6	7
Aligned	1 0 0	0 0 0	A	A	A	A	x	x	x	x
Misaligned—first access	0 1 1	0 0 1		A	A	A	x	x	x	x
second access	0 0 1	1 0 0	A	—	—	—	x	x	x	x
Misaligned—first access	0 1 0	0 1 0	—	—	A	A	x	x	x	x
second access	0 1 0	1 0 0	A	A	—	x	x	x	x	x
Misaligned—first access	0 0 1	0 1 1	—	—	—	A	x	x	x	x
second access	0 1 1	1 0 0	A	A	A	—	x	x	x	x
Aligned	1 0 0	1 0 0	A	A	A	A	x	x	x	x
Misaligned—first access	0 1 1	1 0 1	—	A	A	A	x	x	x	x
second access	0 0 1	0 0 0	A	—	—	—	x	x	x	x
Misaligned—first access	0 1 0	1 1 0	—	—	A	A	x	x	x	x
second access	0 1 0	0 0 0	A	A	—	—	x	x	x	x
Misaligned—first access	0 0 1	1 1 1	—	—	—	A	x	x	x	x
second access	0 1 1	0 0 0	A	A	A	—	x	x	x	x

A: Byte lane used
—: Byte lane not used
x: Byte lane not used in 32-bit bus mode

4.3.3 Address Transfer Termination

Address transfers are terminated with the assertion of the address acknowledge ($\overline{\text{AACK}}$) signal. A snoop response is indicated by the assertion of the $\overline{\text{ARTRY}}$ signal unti1 one clock after $\overline{\text{AACK}}$; the bus clock cycle after $\overline{\text{AACK}}$ is referred to as the ARTRY window. For address bus transactions initiated by a processor, the snoop response originates from either the other processor or the secondary cache. For transactions initiated by the MPC105, the snoop response can originate from either the processor or the alternate master. The following sections describe how the MPC105 can be configured through its register settings to accommodate a variety of snoop responses and snoop timing requirements.

4.3.3.1 MPC105 Snoop Response

Processors may assert $\overline{\text{ARTRY}}$ because of pipeline collisions or because an address snoop hits a modified line in the processor's L1 cache. When a processor detects a snoop hit due to a modified line in the cache, it will assert its bus request in the window of opportunity (the clock after the ARTRY window) to obtain mastership of the bus for its L1 copy-back cycle.

The MPC105 can be configured to repeat a PCI-to-memory transaction that has been terminated by the assertion of $\overline{\text{ARTRY}}$ by the processor or by the L2 cache through the use of the PICR1[CF_LOOP_SNOOP] bit. If the PICR1[CF_LOOP_SNOOP] is set, the MPC105 repeats snooping until $\overline{\text{ARTRY}}$ is not asserted.

The MPC105 may assert $\overline{\text{ARTRY}}$ because of an L2 cast-out operation, or because the PCI bus is occupied by another PCI bus master in a transaction that requires snooping before the 60x-to-PCI address bus transaction is completed. This can occur, for example, when a 60x processor performs a read operation to a PCI target, or when a 60x processor performs a write operation to the PCI bus, but the MPC105's 60x-to-PCI write buffer is full and another PCI bus master accesses the system RAM requiring a snoop while the 60x processor is waiting.

Figure 4-6 illustrates the sequence of bus actions in the case of a snoop. When a 60x processor detects a snoop hit with the L1 cache in write-back mode, the 60x asserts $\overline{\text{ARTRY}}$ and $\overline{\text{BR}}$. This causes the MPC105 to grant the bus to the 60x processor which then proceeds with a copy-back to main memory of the modified cache line.

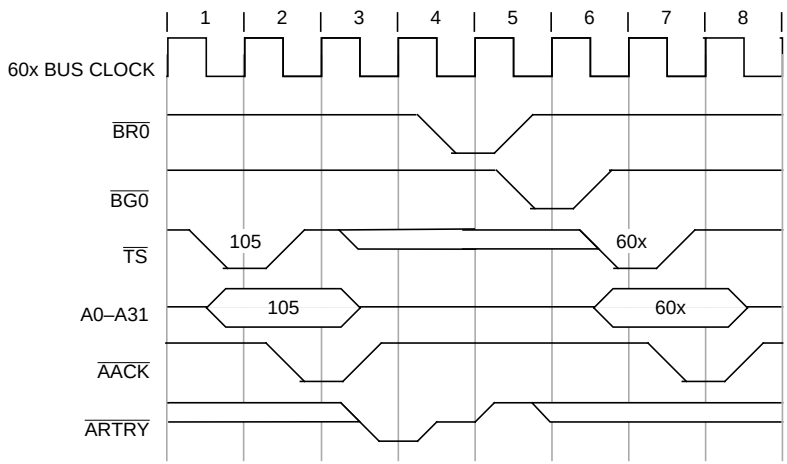


Figure 4-6. Snooped Address Transaction with $\overline{\text{ARTRY}}$ and L1 Cache Copy-Back

4.3.3.2 Address Tenure Timing Configuration

During 60x processor-initiated address tenures, the timing of the assertion of $\overline{\text{AACK}}$ by the MPC105 is determined by the PICR2[CF_APHASE_WS] bits, and the pipeline status of the 60x bus. Since the MPC105 can support one level of pipelining, it uses $\overline{\text{AACK}}$ to control the 60x pipeline condition. To maintain the one-level pipeline, $\overline{\text{AACK}}$ is not asserted for a pipelined address tenure until the current data tenure ends. The MPC105 also withholds the assertion of $\overline{\text{AACK}}$ until no more $\overline{\text{ARTRY}}$ conditions could occur.

The PICR2[CF_APHASE_WS] bits specify the minimum number of address tenure wait states for 60x processor-initiated address operations. Extra wait states may occur because of other MPC105 configuration parameters. Note that in a system implementing an L2 cache, the number of wait states configured by the CF_APHASE_WS bits should be equal to or greater than the value configured in PICR2[CF_L2_HIT_DELAY]. In systems with multiple processors, the number of wait states configured by the CF_APHASE_WS bits should be equal to or greater than the number of wait states selected by the PICR2[CF_SNOOP_WS] bits, since the other processor needs to snoop the 60x access. The CF_SNOOP_WS bits specify the minimum number of address phase wait states required for the snoop response to be valid. For example, additional wait states are required when a 603 is running in 1:1 mode; this case requires at least one wait state to generate the $\overline{\text{ARTRY}}$ response.

For MPC105-initiated transactions, address phase wait states are determined by the PICR2[CF_SNOOP_WS] bits and the 60x bus pipeline status.

4.4 Data Tenure Operations

This section describes the operation of the MPC105 during the data bus arbitration, transfer, and termination phases of the data tenure.

4.4.1 Data Bus Arbitration

The beginning of an address transfer, marked by the assertion of the transfer start ($\overline{\text{TS}}$) signal, is also an implicit data bus request provided that the transfer type (determined by the encoding of the TT0–TT4 signals) indicates the transaction is not address-only.

The MPC105 implements two data bus grant signals ($\overline{\text{DBG0}}$ and $\overline{\text{DBG1}}$), one for each potential master on the 60x interface. These signals are not asserted if the data bus, which is shared with the memory, is busy with a transaction. The internal buffer control circuitry arbitrates the data bus between the 60x processors and the memory controller depending on internal buffer conditions and PCI bus requests.

The PICR1[CF_DPARK] bit specifies whether the MPC105 should park the 60x data bus. If the CF_DPARK bit is asserted, the data bus is parked to the processor which had most recently taken mastership of the 60x address bus.

4.4.2 Data Bus Transfers and Normal Termination

The MPC105 handles data transfers in either single-beat or burst operations. Single-beat operations can transfer from 1 to 8 bytes of data at a time. Burst operations always transfer eight words in four double-word beats (64-bit mode) or eight word beats (32-bit mode). A burst transaction is indicated by the assertion of the $\overline{\text{TBST}}$ signal by the bus master. A transaction is terminated normally by asserting the $\overline{\text{TA}}$ signal.

Figure 4-7 illustrates a sample of both a single-beat and burst data transfer. The $\overline{TA}$ signal is asserted by the MPC105 to mark the cycle in which data is accepted. In a normal burst transfer, the assertion of the fourth $\overline{TA}$ signals the end of a transfer.

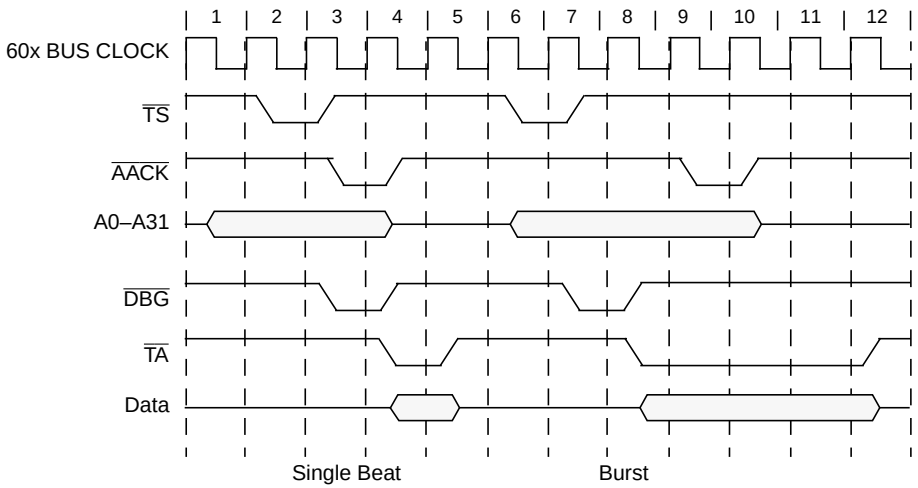


Figure 4-7. Single-Beat and Burst Data Transfers

4.4.3 Data Tenure Timing Configurations

The MPC105 provides PICR1[CF_BREAD_WS] bits to allow the system designer to configure the minimum number of wait states for 60x burst read cycles. The CF_BREAD_WS bits determine the minimum number of wait states from the assertion of the $\overline{TS}$ signal to the assertion of $\overline{TA}$ during a burst read data tenure as required by the various 60x processors. For example, a 603 running in 1:1 mode requires a minimum of one wait state, and at least two wait states if NO_DRTRY mode is also selected.

4.4.4 Data Bus Termination by $\overline{TEA}$

If a processor initiates a transaction which is not supported by the MPC105, the MPC105 signals an error by asserting either the $\overline{TEA}$ or $\overline{MCP}$ signal inputs of the 60x processors. If the transaction is not address only, the MPC105 asserts $\overline{TEA}$ to terminate the transaction provided $\overline{TEA}$ is enabled by setting the PICR1[TEA_EN] bit. The MPC105 can also signal bus transaction errors through the assertion of the $\overline{MCP}$ signal by setting the PICR1[MCP_EN] bit. If the TEA_EN bit is not enabled, the data tenure will be terminated by the appropriate number of $\overline{TA}$ assertions, but the data transferred will be corrupted.

The assertion of the $\overline{TEA}$ signal is only sampled by the processor during the data tenure of the bus transaction, so the MPC105 ensures that the 60x processor receives a qualified data bus grant by asserting the $\overline{DBG}$ signal before asserting $\overline{TEA}$. This sequence is shown in Figure 4-8. In Figure 4-8 the data bus is busy at the beginning of the transaction, thus

delaying the assertion of $\overline{\text{DBG}}$. Note that although $\overline{\text{DBB}}$ is not an input to the MPC105, the state of the bus is always known because the MPC105 controls the data bus grants and either drives or monitors the termination signals.

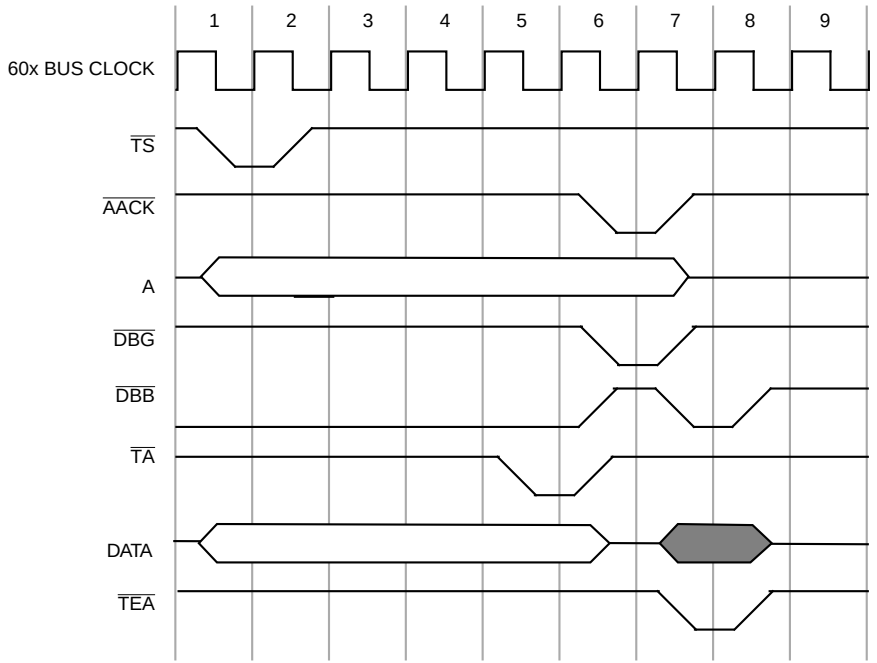


Figure 4-8. Data Tenure Terminated by Assertion of $\overline{\text{TEA}}$

The bus transactions interpreted by the MPC105 as bus errors are as follows:

- Direct-store transactions, as indicated by the assertion of $\overline{\text{XATS}}$ and TT0–TT4
- Any graphics read/write transactions (using the **eciwx** or **ecowx** instructions)
- Write to INTA space (0xBFFF_FFF0) in address map A
- Write to ROM or write to Flash ROM if Flash ROM is not enabled, or if the transfer size is not 1 byte, or transaction is not cache-inhibited or write-through. Only single-byte, cache-inhibited or write-through writes to Flash ROM are allowed.
- Processor read operation from PCI transaction which is target-aborted by the PCI target

4.4.5 60x Bus Slave Support

The MPC105 provides support for a local bus slave that can handle 60x transactions by generating its own $\overline{\text{AACK}}$ and $\overline{\text{TA}}$ responses. The MPC105 is configured for 60x bus slave support through setting the PICR1[CF_LBA_EN] bit. If the CF_LBA_EN bit is cleared, the MPC105 will always drive the $\overline{\text{AACK}}$ and $\overline{\text{TA}}$ signals. If CF_LBA_EN is set to 1, the

MPC105 will only drive $\overline{\text{AACK}}$ and $\overline{\text{TA}}$ when necessary and will remain three-stated otherwise. If the CF_LBA_EN bit is set, and the 60x accesses an address between 1G and 2G (0x40000000–0x7FFFFFFF), and the $\overline{\text{HIT}}$ input signal is asserted one clock after $\overline{\text{TS}}$, the MPC105 will let the 60x bus slave handle the address and data bus tenures, and does not assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$. If $\overline{\text{HIT}}$ is not asserted, the MPC105 will handle the address and data tenures normally. Note that the $\overline{\text{HIT}}$ signal from the 60x bus slave is always active low regardless of the configuration of the PICR2[CF_HIT_HIGH] bit. Note that 60x bus slave access is only allowed in the 1G to 2G address range. In systems implementing an L2 cache, accesses to the 60x bus slave should be cache-inhibited. Figure 4-9 shows an example of a bus transaction performed by a 60x bus slave connected to both an MPC105 and a 60x processor.

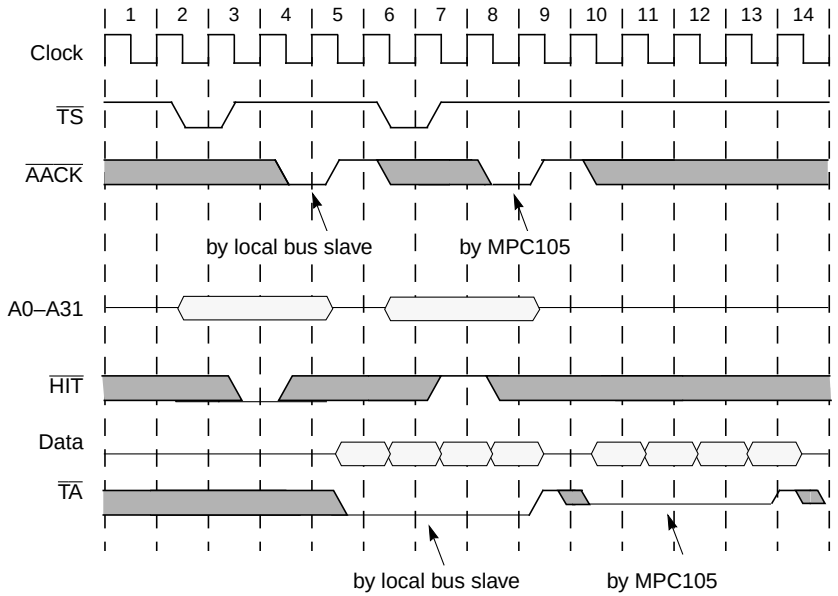
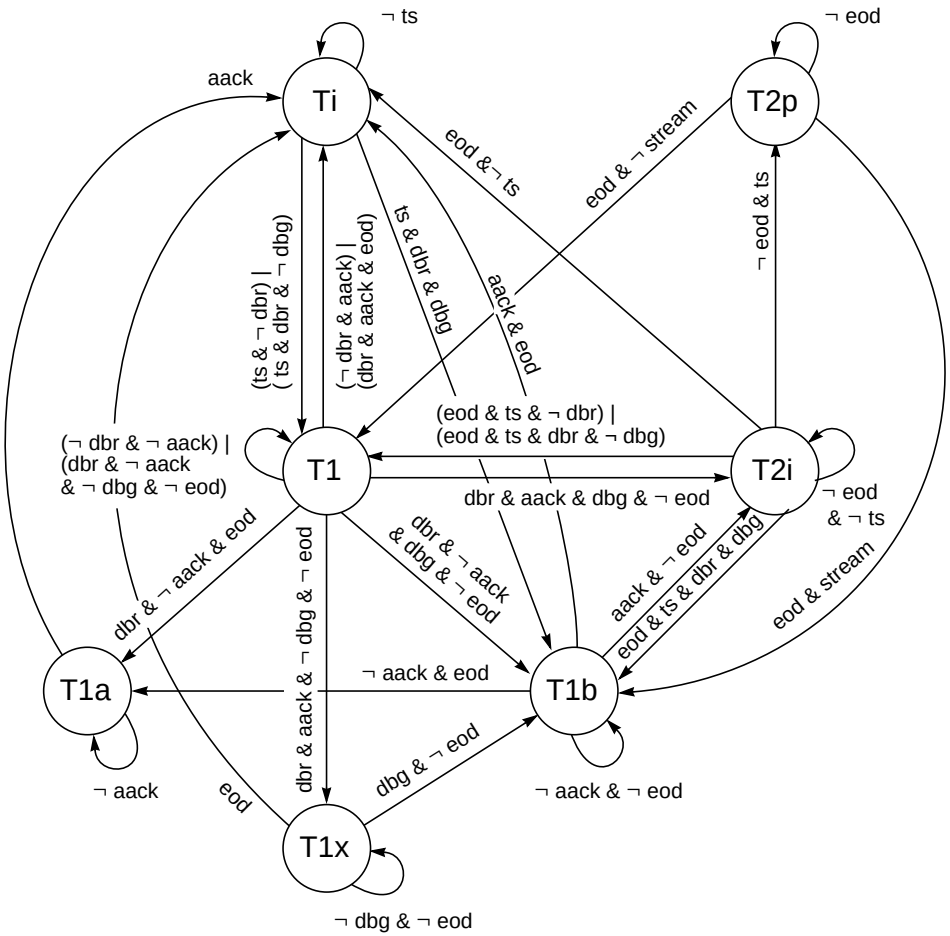


Figure 4-9. 60x Bus Slave Transaction

The MPC105 supports a subset of the 60x bus protocol, and imposes conditions on the assertion of the $\overline{\text{AACK}}$ signal in order to maintain a one-level pipeline, and simplify the bus state model. Designers implementing local bus slave devices should adhere to the MPC105's bus protocol through the same treatment of the $\overline{\text{AACK}}$ and related signals. The $\overline{\text{AACK}}$ signal should only be asserted when the data tenure of a bus transaction has started (that is, $\overline{\text{DBG}}$ is asserted), and $\overline{\text{AACK}}$ for a pipelined bus operation should only be asserted one bus clock cycle after the current data tenure has completed. The $\overline{\text{AACK}}$ signal may be asserted without the assertion of $\overline{\text{DBG}}$ only if a bus operation is terminated by the assertion of the $\overline{\text{ARTRY}}$ signal during the address tenure. Local bus slave devices should implement a state machine equivalent to that shown in Figure 4-10 to track the 60x bus state, and respond accordingly. The signal logic states (true or false) shown in Figure 4-10 reflect the assertion or negation of the signals shown.



Legend:

- ¬ —Condition shown is false
- aack—Address acknowledge
- dbr—Data bus request as indicated by TT0–TT4 that the cycle is not address only.
- dbg—Data bus grant
- eod—End of data phase
- stream—604 fast-L2 mode burst read streaming (fast-L2 mode is enabled and back-to-back burst read is detected)
- ts—Transfer start
- Ti—Idle
- T1—Address phase in progress, data phase not started yet
- T1a—Address phase in progress, data phase completed
- T1b—Address phase and data phase in progress
- T1x—Address phase completed, data phase not started yet.
- T2i—Address phase completed, data phase in progress.
- T2p—Address pipelined, data phase in progress.

Figure 4-10. 60x Bus State Diagram

Chapter 5
Secondary Cache Interface

When the MPC105 is operated with a single 60x processor, the MPC105 provides the system designer the option of implementing a 64-bit, lookaside secondary, or level 2 (L2) cache, which provides 60x processors faster access to instructions and data. The MPC105's L2 cache interface supports L2 cache operation in write-through or write-back mode, cache sizes of 256 Kbytes, 512 Kbytes, and 1 Mbyte, and a cacheable direct-mapped address space of 4 Gbytes. The MPC105 supports an L2 cache line size and coherence granularity of 32 bytes. The MPC105 can perform fast nonpipelined bursts of 3-1-1-1 bus cycles and pipelined bursts of 2-1-1-1 bus cycles. When used with the PowerPC 604 microprocessor in fast L2 mode, the MPC105 can perform pipelined bursts of 1-1-1-1 bus cycles. Use of either burst or asynchronous static RAM is supported by the MPC105.

This chapter describes the operation of the secondary cache interface.

5.1 L2 Cache Interface Operation

The following sections describe L2 cache operation in write-back and write-through mode.

5.1.1 Write-Back Cache Operation

The use of a write-back L2 cache offers several advantages over direct access to the memory system. Since every L1 write operation does not go to main memory but to the L2 cache which can be accessed more quickly, write operation latency is reduced along with contention for the memory system. Subsequent read accesses from the processor that hit in the L2 cache are also expedited in comparison to the memory system. Write-back L2 cache blocks implement a dirty bit in their tag RAM, which indicates whether the contents of the L2 cache block have been modified from that in the memory system. L2 cache blocks that have been modified (dirty bit set) will be written back to memory on L2 cache line replacement, while unmodified L2 cache blocks will be invalidated and overwritten without being cast out to memory.

Figure 5-1 shows the MPC105 configured with a typical write-back L2 cache.

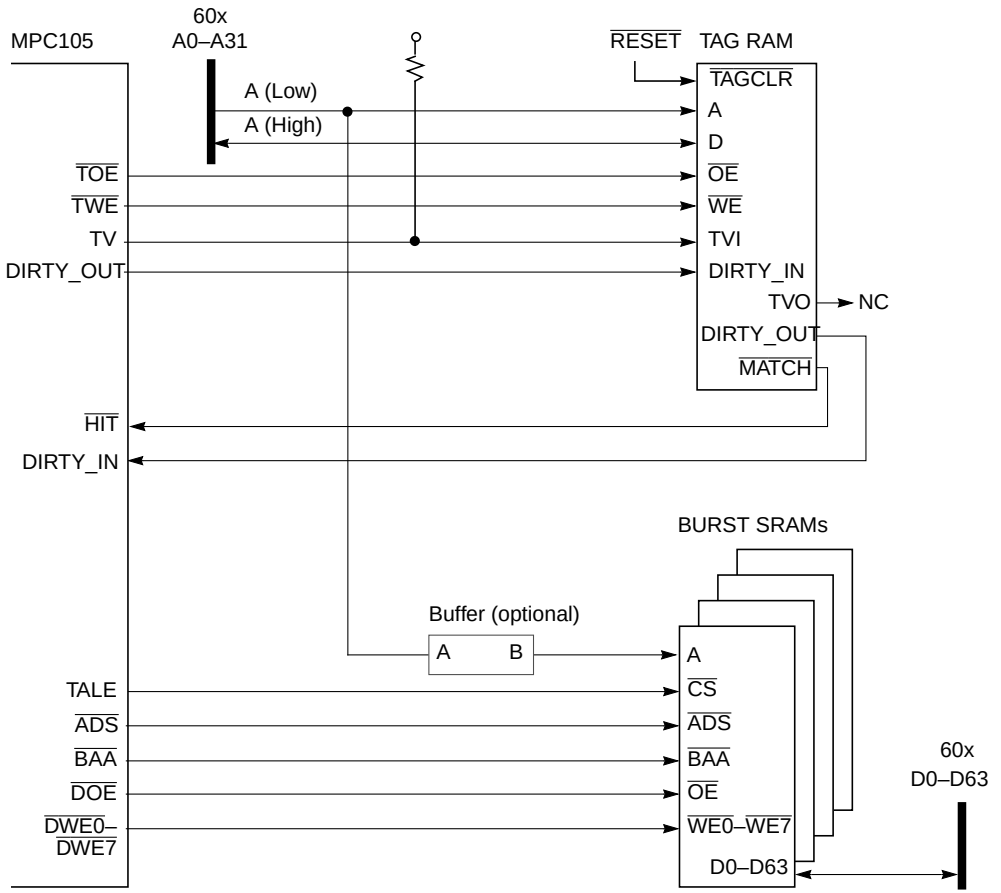


Figure 5-1. MPC105 with Write-Back L2 Cache

5.1.2 Write-Through Cache Operation

Write-through cache operation is also supported by the MPC105. Write-through L2 caches reduce read latency in the same way write-back L2 caches do, but write operations from L1 cache is written to both the L2 cache and the memory system, thereby exhibiting the same latency as an ordinary memory write. The use of write-through L2 cache keeps memory coherent with the contents of the L2 cache, and removes the need for maintenance of a dirty bit in the tag RAM.

Figure 5-2 shows the MPC105 configured with a typical write-through L2 cache.

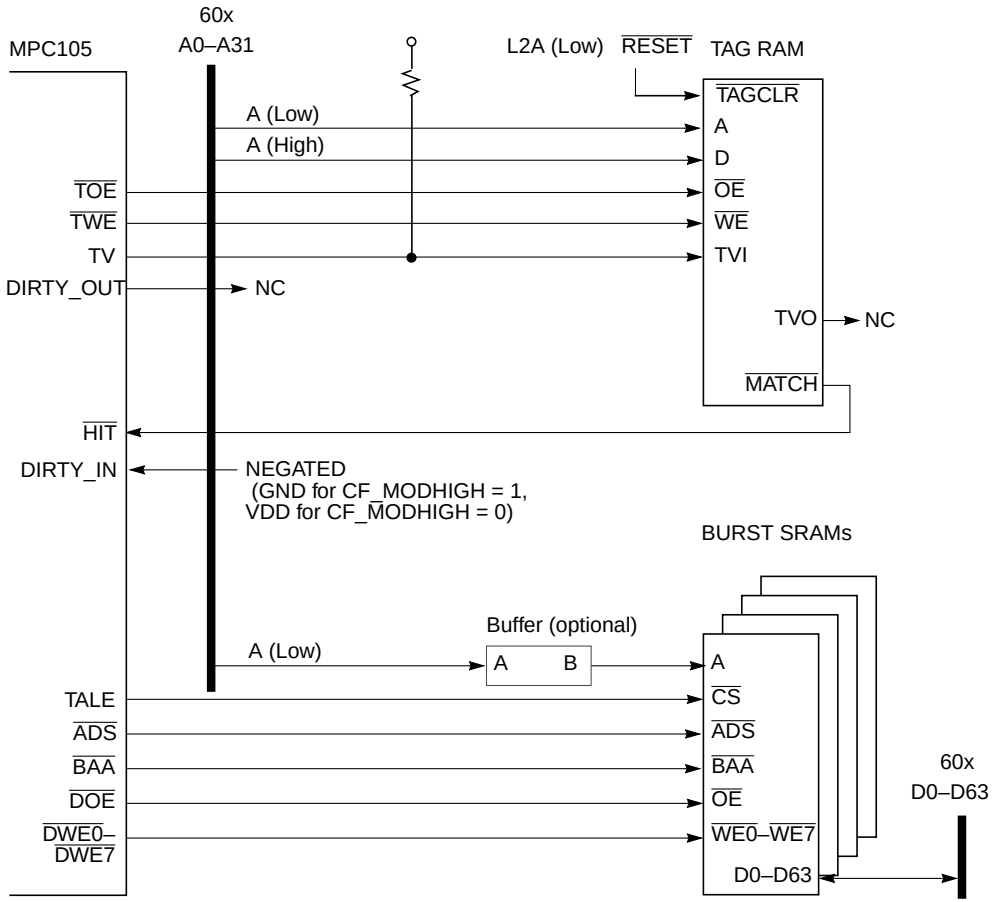


Figure 5-2. MPC105 with Write-Through Cache

5.1.3 L2 Cache Initialization

The MPC105's default power-up configuration holds the L2 cache disabled so that the system software can configure the L2 cache interface parameters prior to enabling the L2 interface. The RESET signal to the tag RAM should initialize the L2 line status to the invalid and unmodified state. If hardware initialization is not available, software can perform tag initialization using the invalidate mode function prior to enabling the L2 interface.

5.1.4 L2 Cache Address Operations

As shown in Figure 5-1 and Figure 5-2, the low-order address bits of the 60x address bus are connected to the address signals of the tag RAM as the tag entry index. The high-order address bits are connected to the tag RAM data signals. The $\overline{\text{TALOE}}$ signal is normally active to drive the high-order bit address for tag lookup and tag write. Depending on the cache size and size of the cacheable address space, different bits from the 60x address bus should be connected to the tag RAM and data RAM. Table 5-1 shows the address signals used by three L2 cache sizes in a 4-Gbyte cacheable space. For smaller cacheable space, the tag RAM data width can be reduced by setting the proper cast-out address mask (CF_CBA_MASK) bits. For example, with a cache size of 512K, an 8-bit tag RAM (plus the TV bit) can be used by masking off the 5 high-order address bits to provide a cacheable space of 128 Mbytes. See Section 3.2.7, “Processor Interface Configuration Registers,” for additional information about tag configuration.

Table 5-1. 60x to Tag and Data RAM Addressing for 4-Gbyte Cacheable Address Space

Cache Size	Tag Address	Tag Data	Data RAM Address
256 Kbyte	A14–A26 (13 bits)	A0–A13 (14 bits)	A14–A28 (15 bits)
512 Kbyte	A13–A26 (14 bits)	A0–A12 (13 bits)	A13–A28 (16 bits)
1 Mbyte	A12–A26 (15 bits)	A0–A11 (12 bits)	A12–A28 (17 bits)

The tag RAM and dirty RAM are updated at the same time when $\overline{\text{TWE}}$ is asserted. The high-order address bits, the TV signal, and DIRTY_OUT signal are used to update the tag RAM and dirty RAM with new line status.

During L2 cast-out cycles, the MPC105 three-states the high-order address bits and the TV signal, deasserts the $\overline{\text{TALOE}}$ signal, and asserts the $\overline{\text{TOE}}$ signal to read the dirty address from tag RAM. The MPC105 latches only the address bits from the tag data signals during tag read cycles. The L2 cache line status is not used.

When both $\overline{\text{TWE}}$ and $\overline{\text{TOE}}$ are deasserted, the tag RAM is in tag lookup mode. During 60x bus operations and MPC105-initiated snoop cycles, the MPC105 uses $\overline{\text{HIT}}$ and DIRTY_IN signal status to determine the current L2 line status and responds accordingly. Polarity of the $\overline{\text{HIT}}$ and DIRTY_IN signal inputs is programmable. The TALE signal is used as the chip select for the data RAM when using the early write timing mode. When other write timing modes are selected, the TALE signal may be used, or the data RAM $\overline{\text{CS}}$ signal may be tied asserted. The TV signal can be used with tag RAMs with separate I/O valid bits or one bidirectional valid bit. For tag RAMs with separate I/O valid bits, the TV signal from the MPC105 is connected to the valid input of the tag RAM. The MPC105 does not sample the TV signal as an input, so the valid output of the tag RAM can be left unconnected. The TV signal is always asserted during the tag lookup, allowing the MPC105 to work with tag RAMs that use the TV signal input for the lookup comparison. The TV signal is three-stated when the MPC105 is in single processor mode without an L2 cache, or in two processor mode.

The MPC105 provides five signals for interfacing to synchronous burst data RAMs. Burst cycles consist of four beats, with 64 bits of data per beat. The data RAM latches the address and $\overline{CS}$ inputs at the beginning of the access when $\overline{ADS}$ is asserted. The $\overline{DOE}$ signal is an asynchronous signal that enables the driving of data onto the 60x data bus during read accesses. The $\overline{BAA}$ signal, when asserted, advances the internal beat counter of the burst RAM. The $\overline{DWE0}$–$\overline{DWE7}$ signals, when asserted, indicate that a write operation to the burst RAM is required. For brevity, when the MPC105 is in the on-chip byte decode mode (PICR2[CF_BYTE_DECODE] set to 1), this manual refers to the data RAM write enable signals as $\overline{DWE0}$–$\overline{DWE7}$, or simply $\overline{DWE}n$. Note that three of the $\overline{DWE}n$ signals have multiple functions—FNR/$\overline{DWE0}$ also functions as the flash/nonvolatile ROM configuration input signal, CK0/$\overline{DWE3}$ also functions as the test clock output, and CKE/$\overline{DWE7}$ also functions as the SDRAM clock enable output.

Parity generation and checking in the L2 cache is controlled through L2_PARITY_ERROR_ENABLE bit in error enabling register 2. The parity signals from the L2 data RAM are connected to the 60x bus parity signals.

Note that L2 cache line fills from ROM accesses do not reflect correct parity, and the MPC105 does not perform parity checking during L2 cache read operations within the ROM address space. Processor parity checking should be disabled while accessing the ROM address space to avoid machine check exceptions, or a checkstop state. Accesses to ROM in the PCI memory space are not cached by the MPC105. When the L2 cache is configured with burst data RAM, 8- and 64-bit local ROM accesses can be cached as write-through. When the L2 cache is configured with asynchronous SRAM, 64-bit local ROM accesses can be cached as write-through, but 8-bit local ROM accesses cannot be cached, and must be treated as cache-inhibited.

5.1.5 Asynchronous SRAM Interface

When the MPC105 is configured for asynchronous SRAM, several signals are redefined. The DALE signal is used as a latch enable for the external address latch. The TALE and $\overline{BAA}$ signals are used as burst address outputs, with TALE providing BA0 signal, and $\overline{BAA}$ providing the BA1 signal. The MPC105's on-chip byte decode logic provides the individual byte write enables to the asynchronous SRAM. Note that PICR2[CF_BYTE_DECODE] bit must be set to 1, and PICR2[CF_WMODE] bits must be set to 01 when using asynchronous SRAM. Figure 5-3 shows the connection of asynchronous SRAM to an MPC105.

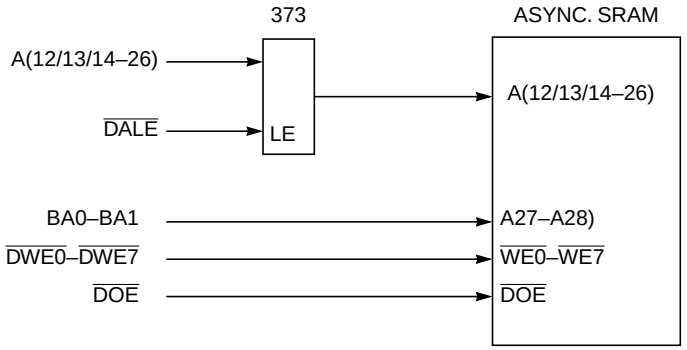


Figure 5-3. Asynchronous SRAM Interface

5.2 L2 Cache Response to Bus Operations

The MPC105's L2 cache interface samples the $\overline{\text{WT}}$, $\overline{\text{CI}}$, $\overline{\text{GBL}}$, $\overline{\text{ARTRY}}$, $\overline{\text{HIT}}$, and DIRTY_IN signals and responds with the activity required by the type of bus operation. The MPC105's L2 cache only supports operations mapped in the 60x processor's ROM or RAM address space, and ignores memory operations mapped in the PCI space. The MPC105's L2 cache supports the following four types of bus operations:

- Normal 60x bus operations (any 60x-initiated operations, with exception of L1 copy-back operations)
- 60x L1 copy-back operations
- L2 cast-out operations (when configured as a write-back cache)
- PCI bus snoop operations

The following sections describe the operation of the L2 cache interface responses in both write-back and write-through configurations.

5.2.1 Write-Back L2 Cache Response

When the MPC105 is configured to support a write-back L2 cache, the L2 cache supplies data on 60x single-beat or burst read hits, read snoop hits, and write snoop hits to modified lines. L2 cache lines are updated on burst read misses, single-beat or burst write hits, and burst write misses. Table 5-2 describes the L2 cache response to normal 60x bus operations, L1 copy-back operations, L2 copy-back operations, and PCI bus snoop operations.

Table 5-2. Write-Back L2 Cache Response

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	$\overline{\text{ARTRY}}$	L2 Hit	L2 Line Status	New L2 Line Status	L2 Controller Response	MPC105 Operation
Normal 60x Bus Operations							
—,R/RWITM	x0x	—	Hit	—	—	L2 —> CPU	Stop MEM access.
B,R/RWITM L2_Update_En = 0[1]	x0x	—	Miss	inv/um	um	MEM —> L2. —> um.	MEM —> CPU
B,R/RWITM with L2_Update_En = 0[1]	x0x	—	Miss	inv/um	—	—	MEM —> CPU
B,R/RWITM L2_Update_En = 0[1]	x0x	A (105)	Miss	mod	—	Set next state as invalid. Wait for L2 copy-back mode.	Stop MEM access. Save low-order bit address. ARTRY the 60x READ cycle. Goes to L2 copy-back mode.
B,R/RWITM with L2_Update_En = 0[1]	x0x	—	Miss	mod	—	—	MEM —> CPU
SB,R/RWITM	x0x	—	Miss	—	—	—	MEM —> CPU
—,R/RWITM	x1x	—	Hit	um	inv	—> invalid.	MEM/PCI —> CPU
—,R/RWITM	x1x	A (105)	Hit	m	—	Set next state as invalid. Wait for L2 copy-back mode.	Stop MEM access. Save low-order bit address. ARTRY the 60x READ cycle. Goes to L2 copy-back mode.
—,R/RWITM	x1x	—	Miss	—	—	—	MEM/PCI —> CPU
B,W	00x	—	Hit	—	mod	CPU —> L2, —> mod.	Stop MEM access.
B,W with L2_Update_En = 0[1]	00X	—	Hit	—	inv	—> invalid.	CPU —> MEM
B,W	10x	—	Hit	—	um	CPU —> L2, —> um.	CPU —> MEM
B,W with L2_Update_En = 0[1]	10X	—	Hit	—	inv	—> invalid.	CPU —> MEM
B,W	00x	—	Miss	inv/um	mod	CPU —> L2, —> mod.	Stop MEM access.
B,W with L2_Update_En = 0[1]	00X	—	Miss	inv/um	—	—	CPU —> MEM

Table 5-2. Write-Back L2 Cache Response (Continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	$\overline{\text{ARTRY}}$	L2 Hit	L2 Line Status	New L2 Line Status	L2 Controller Response	MPC105 Operation
B,W	10x	—	Miss	inv/um	um	CPU —> L2, —> um.	CPU —> MEM
B,W with L2_Update_En = 0[1]	10X	—	Miss	inv/um	—	—	CPU —> MEM
B,W	x0x	A (105)	Miss	mod	—	Set next state as invalid. Wait for L2 copy-back mode.	Stop MEM access. Save low-order bit address. ARTRY the 60x WRITE cycle. Goes to L2 copy-back mode.
B,W with L2_Update_En = 0[1]	x0X	—	Miss	mod	—	—	CPU —> MEM
SB, W	00x	—	Hit	—	mod	CPU —> L2, —> mod.	Stop MEM access.
SB, W with L2_Update_En = 0[1]	00X	—	Hit	um	inv	—> invalid.	CPU —> MEM
SB, W with L2_Update_En = 0[1]	00X	A (105)	Hit	mod	—	Set next state as invalid. Wait for L2 copy-back mode.	Stop MEM access. Save low-order bit address. ARTRY the 60x READ cycle. Goes to L2 copy-back mode
SB, W	10x	—	Hit	um	—	CPU —> L2.	CPU —> MEM
SB, W with L2_Update_En = 0[1]	10X	—	Hit	um	inv	—> invalid.	CPU —> MEM
SB, W	10x	A (105)	Hit	mod	—	Set next state as um. Wait for L2 copy-back mode.	Stop MEM access. Save low-order bit address. ARTRY the 60x WRITE cycle. Goes to L2 copy-back mode.
SB, W	x0x	—	Miss	—	—	—	CPU —> MEM
—,W	x1x	—	Hit	um	inv	—> invalid.	CPU —> MEM/ PCI
B,W	x1x	—	Hit	m	inv	—> invalid.	CPU —> MEM/ PCI

Table 5-2. Write-Back L2 Cache Response (Continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	$\overline{\text{ARTRY}}$	L2 Hit	L2 Line Status	New L2 Line Status	L2 Controller Response	MPC105 Operation
SB,W	x1x	A (105)	Hit	m	—	Set next state as invalid. Wait for L2 copy-back mode	Stop MEM access. Save low-order bit address. ARTRY the 60x WRITE cycle. Goes to L2 copy-back mode.
—,W	x1x	—	Miss	—	—	—	CPU —> MEM/ PCI
—,CLEAN	x0x	—	Hit	um	—	—	—
—,CLEAN	x0x	A (105)	Hit	mod	—	Set next state as um. Wait for L2 copy-back mode.	Save low-order bit address. End the CLEAN address phase. ARTRY any pending 60x address phase. Goes to L2 copy-back mode.
—,CLEAN	x0x	—	Miss	—	—	—	—
—,CLEAN	x1x	—	Hit	um	inv	—	—
—,CLEAN	x1x	A (105)	Hit	m	—	Set next state as invalid. Wait for L2 copy-back mode	Save low-order bit address. ARTRY the 60x cycle. Goes to L2 copy-back mode
—,CLEAN	x1x	—	Miss	—	—	—	—
—,FLUSH	x0x	—	Hit	um	inv	—> invalid.	—
—,FLUSH	x0x	A (105)	Hit	mod	—	Set next state as invalid. Wait for L2 copy-back mode.	Save low-order bit address. ARTRY the 60x cycle. Goes to L2 copy-back mode.
—,FLUSH	x0x	—	Miss	—	—	—	—
—,FLUSH	x1x	—	Hit	um	inv	—> invalid	—
—,FLUSH	x1x	A (105)	Hit	m	—	Set next state as invalid. Wait for L2 copy-back mode	Save low-order bit address. ARTRY the 60x cycle. Goes to L2 copy-back mode

Table 5-2. Write-Back L2 Cache Response (Continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	$\overline{\text{ARTRY}}$	L2 Hit	L2 Line Status	New L2 Line Status	L2 Controller Response	MPC105 Operation
—,FLUSH	x1x	—	Miss	—	—	—	—
—,KILL/ ICBI	x0x	—	Hit	—	inv	—> invalid.	Invalidate internal buffers if Hit.
—,KILL/ ICBI	x0x	—	Miss	—	—	—	Invalidate internal buffers if Hit.
—,KILL/ ICBI	x1x	—	Hit	—	inv	—> invalid.	Invalidate internal buffers if Hit.
—,KILL/ ICBI	x1x	—	Miss	—	—	—	Invalidate internal buffers if Hit.
L1 Copy-Back Operations							
—	xxx	—	Hit	—	PCI read snoop: um PCI read w/ lock or PCI write snoop: inv	If PCI read snoop, replace line with CPU copy-back data. —> um. If PCI read w/ lock or write snoop: L2 —> invalid.	MPC105 detects 60x snoop—push to snooping address and does not assert $\overline{\text{ARTRY}}$. If PCI read or PCI read w/ lock, run 60x cycle and send data to memory and PCI. If PCI write snoop, run 60x cycle and merge 60x data with PCI write data and write to memory. Exit L1 copy-back mode. If in loop-snoop mode, repeat snoop.

Table 5-2. Write-Back L2 Cache Response (Continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	$\overline{\text{ARTRY}}$	L2 Hit	L2 Line Status	New L2 Line Status	L2 Controller Response	MPC105 Operation
L2_Update_En = 0[1]	xxx	—	Hit	—	PCI read, PCI read w/ lock, PCI write: inv	—> invalid.	MPC105 detects 60x snoop—push to snooping address and does not assert $\overline{\text{ARTRY}}$. If PCI read or PCI read w/ lock, run 60x cycle and send data to memory and PCI. If PCI write snoop, run 60x cycle and merge 60x data with PCI write data and write to memory. Exit L1 copy-back mode. If in loop-snoop mode, repeat snoop.
—	xxx	—	Miss	—	—	—	MPC105 detects 60x snoop-push to snooping address and does not assert $\overline{\text{ARTRY}}$. Run 60x cycle. If PCI read or PCI read w/ lock snoop, send data to memory and PCI. If PCI write snoop, send PCI data to memory. Exit L1 copy-back mode. If in loop-snoop mode, repeat snoop.
—	xxx	A (105)	—	—	—	—	60x cycle is not performing snoop-push to snooped address. MPC105 asserts $\overline{\text{ARTRY}}$ to retry 60x cycle. Exit L1 copy-back mode. If in loop-snoop mode, repeat snoop.

Table 5-2. Write-Back L2 Cache Response (Continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	$\overline{\text{ARTRY}}$	L2 Hit	L2 Line Status	New L2 Line Status	L2 Controller Response	MPC105 Operation
L2 Copy-Back Operations							
—	—	—	—	—	um / inv	Send dirty address onto address bus. Burst data onto data bus. —> next state.	Grant address bus to L2. Capture copy-back address. Grant data bus to L2. Send L2 data to memory. Goes to normal mode.
PCI Bus Snoop Operations							
—,—	xx1	A (60x)	—	—	—	—	If 60x asserts $\overline{\text{BR}}$ in window of opportunity, then save snoop type (PCI read, read w/ lock, or write). Goes to L1 copy-back mode. Grant bus to 60x. Else if no $\overline{\text{BR}}$ assertion, repeat snoop.
—,R	xx1	N (60x)	Hit	—	—	L2 —> MPC105	L2 —> PCI.
—,R	xx1	N (60x)	Miss	—	—	—	MEM —> PCI.
—,RWITM	xx1	N (60x)	Hit	—	—	L2 —> MPC105	L2 —> PCI.
—,RWITM	xx1	N (60x)	Miss	—	—	—	MEM —> PCI.
—,W	xx1	N (60x)	Hit	um	inv	—> invalid.	PCI —> MEM.
—,WNK	xx1	N (60x)	Hit	mod	inv	L2 —> MPC105, —> invalid.	L2 —> MEM/PCI.
—,WK	xx1	N (60x)	Hit	mod	inv	—> invalid.	PCI —> MEM.

Table 5-2. Write-Back L2 Cache Response (Continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	$\overline{\text{ARTRY}}$	L2 Hit	L2 Line Status	New L2 Line Status	L2 Controller Response	MPC105 Operation
—,W	xx1	N (60x)	Miss	—	—	—	PCI —> MEM.

Note: [1]The L2_Update_Enable, PICR2[31], when set to 1 allows the L2's cache lines to be updated, and when cleared to 0 allows the cache lines to be read or invalidated only.

A	Asserted	B	Burst
N	Negated	R	Read, Read-atomic, RWNITC
RWNITC	Read-with-no-intent-to-cache	RWITM	Read-with-intent-to-modify, RWITM atomic
W	Write-with-flush, write-with-flush-atomic, write-with-kill	WNK	Write-with-flush, write-with-flush-atomic
WK	Write-with-kill	—,x	Input don't care
inv	Invalid	mod	Modified
um	Unmodified	SB	Single-beat

5.2.2 Write-Through L2 Cache Response

When the MPC105 is configured to support a write-through L2 cache, the L2 cache supplies data on 60x single-beat or burst read hits, and read snoop hits. L2 cache lines are updated on burst read misses, single-beat or burst write hits, and burst write misses. Table 5-3 describes the L2 cache response to normal 60x bus operations, L1 copy-back operations, and PCI bus snoop operations.

Table 5-3. Write-Through L2 Cache Response

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	$\overline{\text{ARTRY}}$	L2 Hit	New L2 Status	L2 Controller Response	MPC105 Operation
Normal 60x Bus Operations						
—, R/RWITM	x0x	—	Hit	—	L2 —> CPU.	Stop MEM access.
B, R/RWITM	x0x	—	Miss	um	MEM —> L2, —> um	MEM —> CPU
B, R/RWITM with L2_Update_En = 0[1]	x0x	—	Miss	—	—	MEM —> CPU
SB, R/RWITM	x0x	—	Miss	—	—	MEM —> CPU
—, R/RWITM	01x	—	Hit	inv	—	MEM/PCI —> CPU
—, R/RWITM	01x	—	Miss	—	—	MEM/PCI —> CPU
B, W	x0x	—	Hit	—	CPU —> L2.	CPU —> MEM
B, W with L2_Update_En = 0[1]	x0x	—	Hit	inv	—> invalid.	CPU —> MEM
B, W	x0x	—	Miss	um	CPU —> L2.	CPU —> MEM

Table 5-3. Write-Through L2 Cache Response (Continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	$\overline{\text{ARTRY}}$	L2 Hit	New L2 Status	L2 Controller Response	MPC105 Operation
B, W with L2_Update_En = 0[1]	x0x	—	Miss	—	—	CPU —> MEM
SB, W	x0x	—	Hit	—	CPU —> L2.	CPU —> MEM
SB, W with L2_Update_En = 0[1]	x0x	—	Hit	inv	—> invalid.	CPU —> MEM
SB, W	x0x	—	Miss	—	—	CPU —> MEM
—, W	x1x	—	Hit	inv	—	CPU —> MEM/PCI
—, W	x1x	—	Miss	—	—	CPU —> MEM/PCI
—, Clean	x0x	—	—	—	—	—
—, Clean	x1x	—	Hit	inv	—> invalid.	—
—, Clean	x1x	—	Miss	—	—	—
—, Flush	x0x	—	Hit	inv	—> invalid.	—
—, Flush	x0x	—	Miss	—	—	—
—, Flush	x1x	—	Hit	inv	—> invalid.	—
—, Flush	x1x	—	Miss	—	—	—
—, Kill/ICBI	x0x	—	Hit	inv	—> invalid.	(Invalidate buffer.)
—, Kill/ICBI	x0x	—	Miss	—	—	(Invalidate buffer.)
—, Kill/ICBI	x1x	—	Hit	inv	—> invalid.	(Invalidate buffer.)
L1 Copy-Back Bus Operations						
—, —	xxx	—	Hit	PCI read snoop: um PCI read w/ lock, or PCI write snoop: inv	If PCI read snoop, CPU —> L2. If PCI read w/ lock, or PCI write snoop: L2 —> invalid.	MPC105 detects 60x snoop-push to snooping address. If PCI read w/ lock, or PCI read, send CPU data to memory and PCI. I f PCI write merge CPU data and PCI data and send merged data to memory. Exit L1 copy-back mode. If loop-snoop mode, repeat snoop.

Table 5-3. Write-Through L2 Cache Response (Continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	ARTRY	L2 Hit	New L2 Status	L2 Controller Response	MPC105 Operation
L2_Update_En = 0[1]	xxx	—	Hit	PCI read or PCI read w/ lock snoop: inv.	—> invalid.	MPC105 detects 60x snoop-push to snooping address. If PCI read w/ lock, or PCI read, send CPU data to memory and PCI. I f PCI write merge CPU data and PCI data and send merged data to memory. Exit L1 copy-back mode. If loop-snoop mode, repeat snoop.
—, —	xxx	—	Miss	—	—	MPC105 detects 60x snoop-push to snooping address. If PCI read, or PCI read w/ lock, send CPU data to memory and PCI. I f PCI write, merge CPU data with PCI data and send merged data to memory. Exit L1 copy-back mode. If loop-snoop mode, repeat snoop.
—, —	xxx	A (105)	—	—	—	MPC105 detects 60x snoop-push that does not match snoop address. MPC105 asserts ARTRY to retry the 60x cycle. Exit L1 copy-back mode. If loop-snoop mode, repeat snoop.
PCI Bus Snoop Operations						
—, —	xx1	A (60x)	—	—	—	If 60x assert BR, then save snoop-type (PCI read, PCI read w/ lock, or PCI write); Goes to L1 copy-back mode; grant bus to 60x. Else repeat snoop.
—, R	xx1	N (60x)	Hit	—	L2 —> MPC105	L2 —> PCI
—, R	xx1	N (60x)	Miss	—	—	MEM —> PCI
—, RWITM	xx1	N (60x)	Hit	—	L2 —> MPC105	L2 —> PCI
—, RWITM	xx1	N (60x)	Miss	—	—	MEM —> PCI

Table 5-3. Write-Through L2 Cache Response (Continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	$\overline{\text{ARTRY}}$	L2 Hit	New L2 Status	L2 Controller Response	MPC105 Operation
—, W	xx1	N (60x)	Hit	inv	—> invalid.	PCI —> MEM
—, W	xx1	N (60x)	Miss	—	—	PCI —> MEM

Note: [1]The L2_Update_Enable, PICR2[31], when set to 1 allows the L2's cache lines to be updated, and when cleared to 0 allows the cache lines to be read or invalidated only.

A	Asserted	B	Burst
N	Negated	R	Read, read-atomic, RWNITC
RWNITC	Read-with-no-intent-to-cache	RWITM	Read-with-intent-to-modify, RWITM atomic
W	Write-with-flush, write-with-flush-atomic, write-with-kill	WNK	Write-with-flush, write-with-flush-atomic
WK	Write-with-kill	—,x	Input don't care
inv	Invalid	mod	Modified
um	Unmodified	SB	Single-beat

5.3 L2 Cache Configuration Registers

The MPC105 is configured at power-up with the L2 cache interface disabled, thereby allowing software to configure the L2 cache parameters prior to enabling the L2 interface. The L2 cache configuration register bits (located in PICR1 and PICR2) shown in the following sections control the behavior of the L2 cache interface, and must be configured in keeping with the system design prior to enabling the L2 cache interface. Where a short description of the configuration register bits is shown, refer to Chapter 3, "Device Programming," for additional information about the specific programming of the configuration bits described.

5.3.1 L2 Cache Interface Mode Configuration

The following configuration register bits control the various operational parameters of the L2 cache interface. These parameters must be set properly before the L2 cache is enabled through PICR2[CF_L2_EN]. The L2 cache interface should be disabled if these parameters are modified.

- CF_L2_MP(1–0). Specifies single or multiprocessor configuration, and write-through or write-back L2 cache interface configuration.
- CF_L2_SIZE(1–0). Specifies L2 cache size.
- CF_HIT_HIGH. Specifies the polarity of the $\overline{\text{HIT}}$ signal.
- CF_MOD_HIGH. Specifies the polarity of the DIRTY_IN, DIRTY_OUT, and TV signals.
- CF_BURST_RATE. Specifies the burst rate of the of the data beats for burst read and burst write operations.

- CF_TOE_WIDTH. Specifies the width of the active $\overline{\text{TOE}}$ pulse during L2 cast-out tag read operations.
- CF_CBA_MASK(7–0). Specifies which bits of the dirty address read from the tag RAM are valid.
- CF_INV_MODE. The L2 cache invalidate enable mode is used to initialize the tag contents before enabling the L2 cache in cases where the hardware initialization of the tag and dirty RAM is not available. To flush the L2 cache, the CF_FLUSHL2 (or port 0x81C[CF_FLUSHL2]) configuration bit can be set.
- CF_CACHE1G. Specifies size of memory space caches by L2 cache.
- CF_DATARAMTYPE(0–1). Specifies the type of SRAM used by the L2 cache.
- CF_FASTL2MODE. Specifies if fast L2 mode is enabled. The use of fast L2 mode is supported only by the 604.
- CF_BYTEDECODE. Specifies whether the byte write decode is on-chip or off-chip.
- CF_FAST_CASTOUT. Specifies timing of L2 cast-out operation.
- CF_HOLD. Specifies the hold time of the address, TV, and DIRTY_OUT signals with respect to the rising edge of the $\overline{\text{TWE}}$ signal.

5.3.2 L2 Cache Interface Control Configuration

The following configuration register bits perform L2 cache control functions, and can be modified after the L2 cache interface has been enabled through PICR2[CF_L2_EN].

- CF_L2_UPDATE_EN. Specifies if L2 cache can be updated. This L2 parameter can also be set through port 0x81C[CF_L2_UPDATE_EN].
- CF_L2EN. Specifies if the L2 cache is enabled. Can also be configured through port 0x81C[CF_L2EN].
- CF_FLUSHL2. Setting this configuration bit causes the L2 cache controller to flush all modified lines to memory, and to invalidate all L2 cache lines. This configuration bit can also be accessed through port 0x81C[CF_FLUSHL2].

5.3.2.1 CF_L2_HIT_DELAY[1–0]

CF_L2_HIT_DELAY[1–0] specify the earliest valid sampling point of the $\overline{\text{HIT}}$ and DIRTY_IN signals from the assertion of $\overline{\text{TS}}$. CF_L2_HIT_DELAY can be configured for a one, two, or three clock delay. For best performance, (3-1-1-1 nonpipelined, and 2-1-1-1 pipelined), CF_L2_HIT_DELAY should be configured for a delay of one clock cycle. Note that the MPC105 may not sample the HIT and DIRTY_IN signals at the earliest sampling point, and should be held valid as long as the address is valid. Figure 5-4 shows the earliest sampling points selected by the configuration of CF_L2_HIT_DELAY.

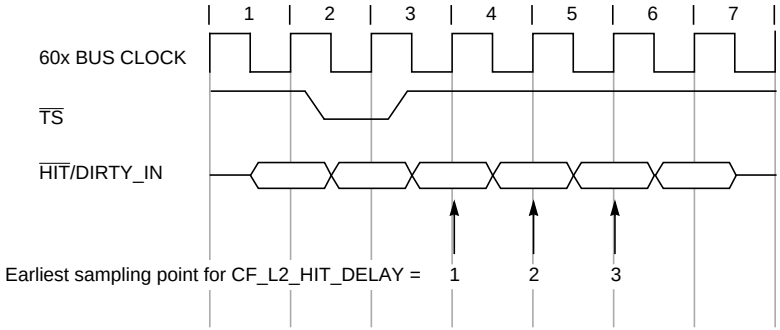


Figure 5-4. HIT and DIRTY_IN Delay Configuration

5.3.2.2 CF_DOE

CF_DOE specifies the time from DOE assertion to the data valid access time of the data RAM for the first data beat. If CF_DOE is cleared to 0, one clock cycle is selected. If CF_DOE is set to 1, two clock cycles are selected, and the MPC105 will assert DOE speculatively at the end of the assertion of TS as long as DBG is asserted in order to minimize the effect of the extra clock delay on read hits. When using asynchronous SRAM, CF_DOE controls the first data beat access time of pipelined read operations. Clearing CF_DOE to 0 provides 3-2-2-2/2-2-2-2 burst read timing, and setting CF_DOE to 1 provides 3-2-2-2/3-2-2-2 burst read timing. Figure 5-5 shows data access timing when CF_DOE is cleared to 0.

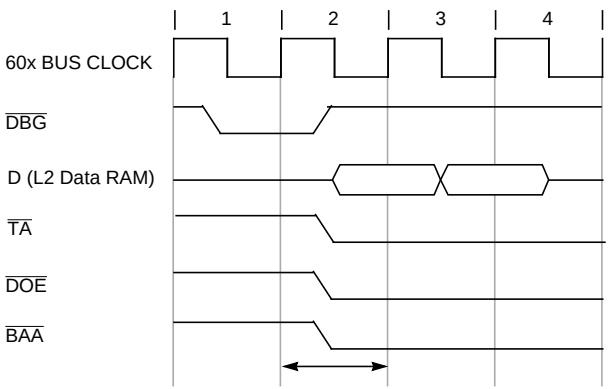


Figure 5-5. Data Access Timing with CF_DOE = 0

Figure 5-6 shows data access timing when CF_DOE is set to 1.

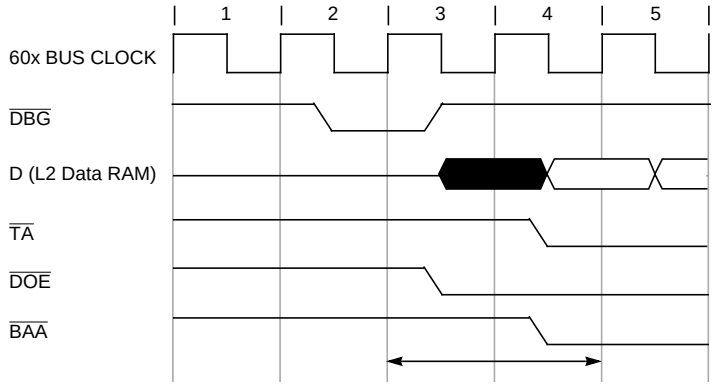


Figure 5-6. Data Access Timing with CF_DOE = 1

5.3.2.3 CF_WDATA

CF_WDATA specifies the write data setup time required for writing the first data beat to data RAM. Clearing CF_WDATA to 0 provides one clock cycle of set-up time, and setting CF_WDATA to 1 provides two clock cycles. When using asynchronous RAM, CF_WDATA controls the write pulse timing. Setting CF_WDATA to 1 causes $\overline{\text{DWE0}}$–$\overline{\text{DWE7}}$ and $\overline{\text{TA}}$ to be deasserted on the same rising edge of the clock during line fills from memory. If CF_WDATA is set to 1, and the transaction is not a cache line fill from memory, The $\overline{\text{DWE0}}$–$\overline{\text{DWE7}}$ signals are deasserted on the falling edge of the clock when $\overline{\text{TA}}$ is asserted, just as when CF_WDATA is cleared to 0. Figure 5-7 shows write data set-up timing with CF_WDATA cleared to 0, and CF_WMODE set to 1.

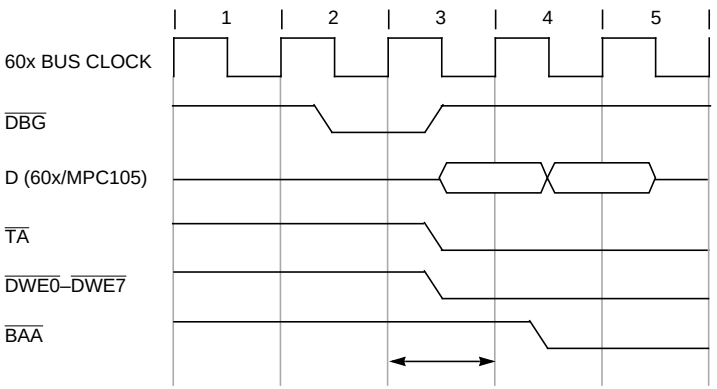


Figure 5-7. Write Data Setup Timing with CF_WDATA = 0

Figure 5-8 shows write data set-up timing with CF_WDATA and CF_WMODE set to 1.

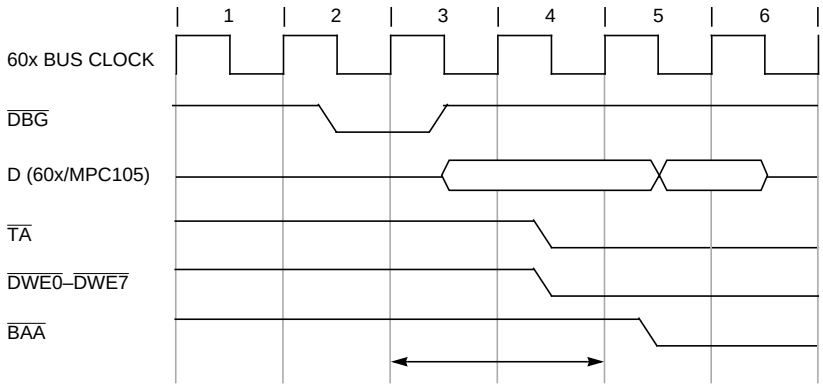


Figure 5-8. Write Data Setup Timing with CF_WDATA = 1

5.3.2.4 CF_WMODE

CF_WMODE selects one of three data RAM write timings. When CF_WMODE is set to 1, normal write timing is selected, and the MPC105 assumes no external delay on the DWE0–DWE7 signals, or on the DWE signal when external byte decode logic is implemented. When using internal byte decoding, or asynchronous RAM, CF_WMODE must be set to 1. Figure 5-9 shows the logic required for external byte decode that requires CF_WMODE to be set to 1, and CF_BYTE_DECODE cleared to 0.

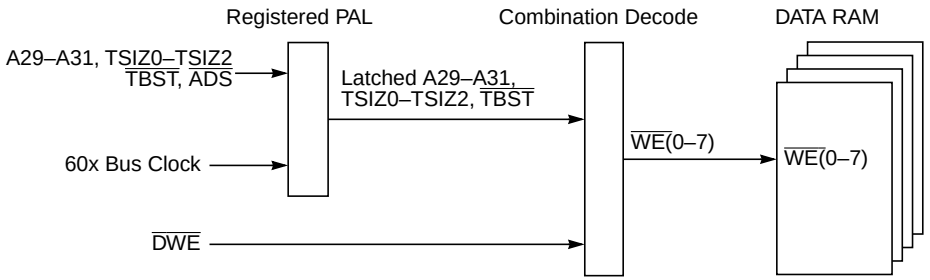


Figure 5-9. External Byte Decode Logic Requiring CF_WMODE =1

Figure 5-10 shows the write mode timing associated with Figure 5-9 for pipelined and nonpipelined bus transactions with CF_WMODE and CF_L2_HIT_DELAY set to 1, and CF_WDATA and CF_BURST_RATE cleared to 0.

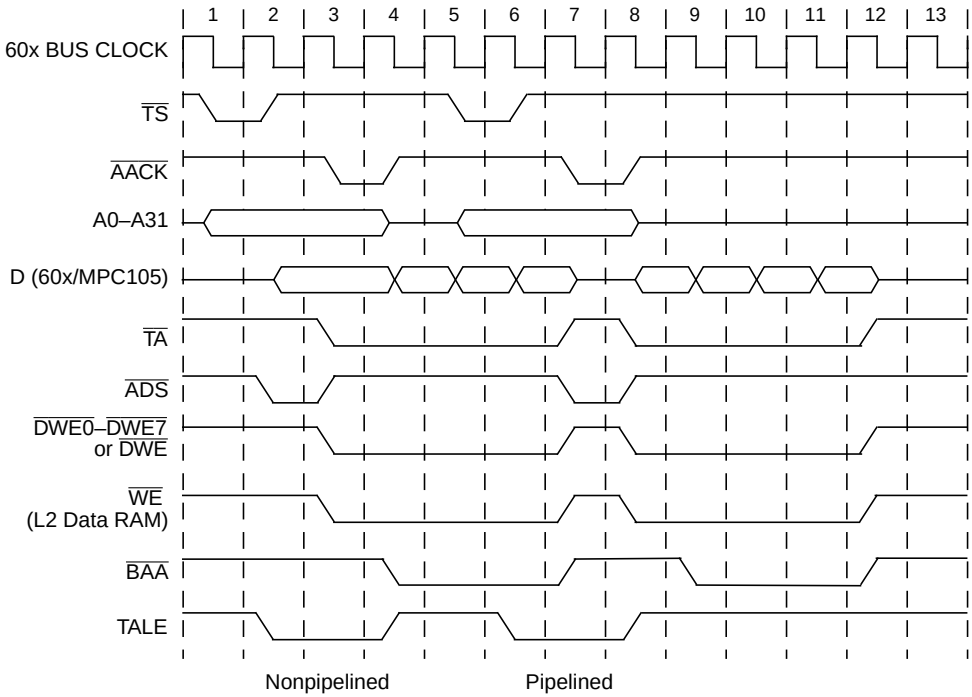


Figure 5-10. Pipelined and Nonpipelined Operations with CF_WMODE = 1

When CF_WMODE is set to 2, one cycle of delayed write timing is provided, allowing for one level of external logic for byte selection. Figure 5-11 shows the logic required for external byte decode that requires CF_WMODE to be set to 2, and CF_BYTE_DECODE cleared to 0.

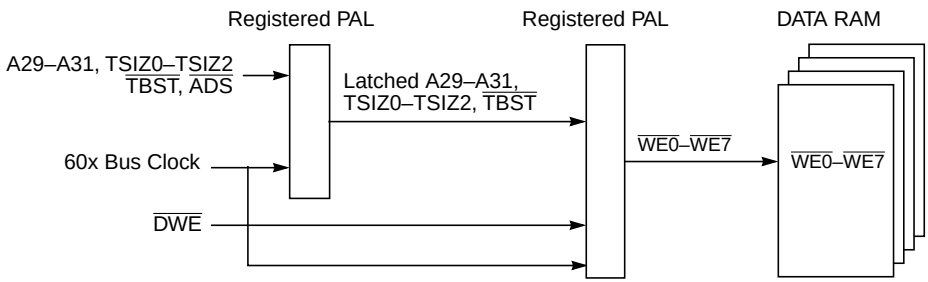


Figure 5-11. External Byte Decode Logic Requiring CF_WMODE = 2

Figure 5-12 shows the write mode timing associated with Figure 5-11 for pipelined and nonpipelined bus transactions with CF_WMODE set to 2, CF_L2_HIT_DELAY set to 1, and CF_WDATA and CF_BURST_RATE cleared to 0.

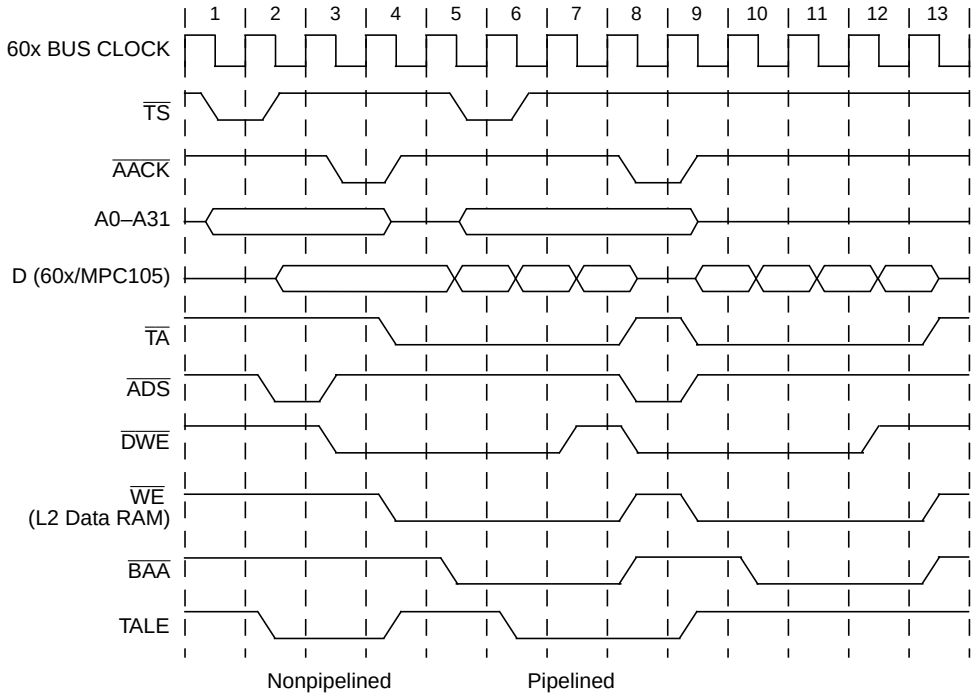


Figure 5-12. Pipelined and Nonpipelined Operations with CF_WMODE = 2

When CF_WMODE is set to 3, early write timing is provided, and $\overline{\text{DWE}}$ is asserted speculatively one clock cycle early to provide better write performance. Figure 5-13 shows the logic required for external byte decode that requires CF_WMODE to be set to 3, and CF_BYTE_DECODE cleared to 0.

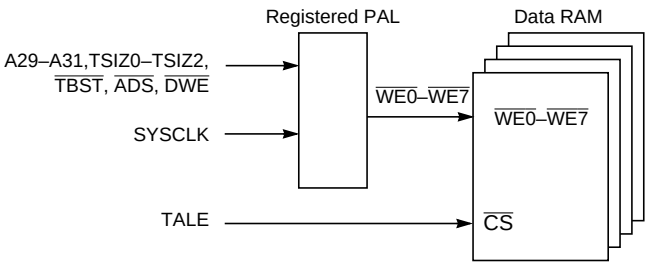


Figure 5-13. External Byte Decode Logic Requiring CF_WMODE = 3

Figure 5-14 shows the write mode timing associated with Figure 5-13 for pipelined and nonpipelined bus transactions with CF_WMODE set to 3, CF_L2_HIT_DELAY set to 1, and CF_WDATA and CF_BURST_RATE cleared to 0.

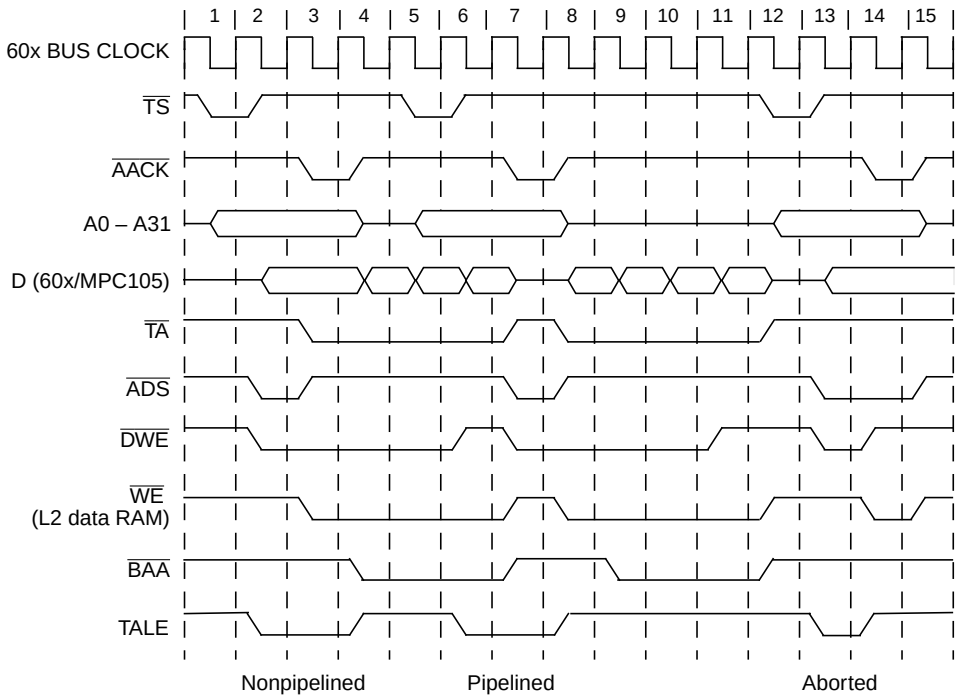


Figure 5-14. Pipelined and Nonpipelined Operations with CF_WMODE = 3

5.4 L2 Cache Interface Timing Examples

The figures in the following sections provide examples of the L2 cache interface signal timing in the course of cache read hits, cache write hits, cache line updates, L2 cache cast-out operations, and snoop operations. L2 cache timing examples are provided for a cache implemented with synchronous burst SRAMs and asynchronous SRAMs.

The symbology shown in Figure 5-15 is applicable to all the figures in the following sections.

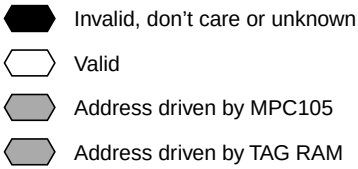


Figure 5-15. Timing Diagram Legend

5.4.1 Synchronous Burst SRAM L2 Cache Timing

The following sections provide timing examples for L2 cache operations in systems implemented using synchronous burst SRAM.

5.4.1.1 L2 Cache Read Hit Timing

Figure 5-16 shows the L2 interface timing for a read hit in the L2 cache. The L2 cache configuration bits CF_APHASE_WS, CF_L2_HIT_DELAY, and CF_DPARK are set to 1, and CF_DOE is cleared to 0.

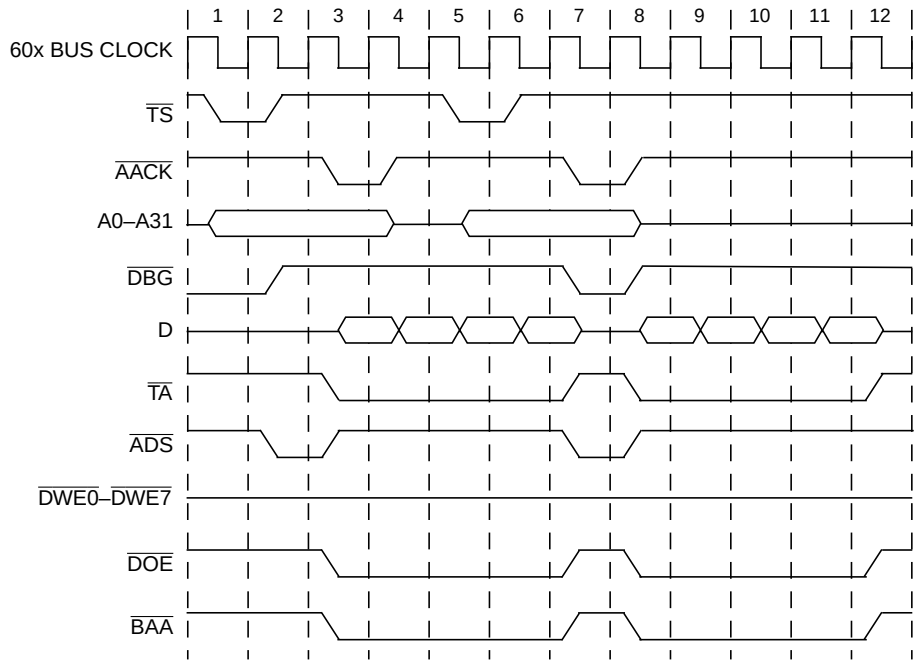


Figure 5-16. L2 Cache Read Hit Timing with CF_DOE = 0

Figure 5-17 shows the L2 interface timing for a read hit with configuration bits CF_APHASE_WS, CF_L2_HIT_DELAY, CF_DPARK, and CF_DOE set to 1.

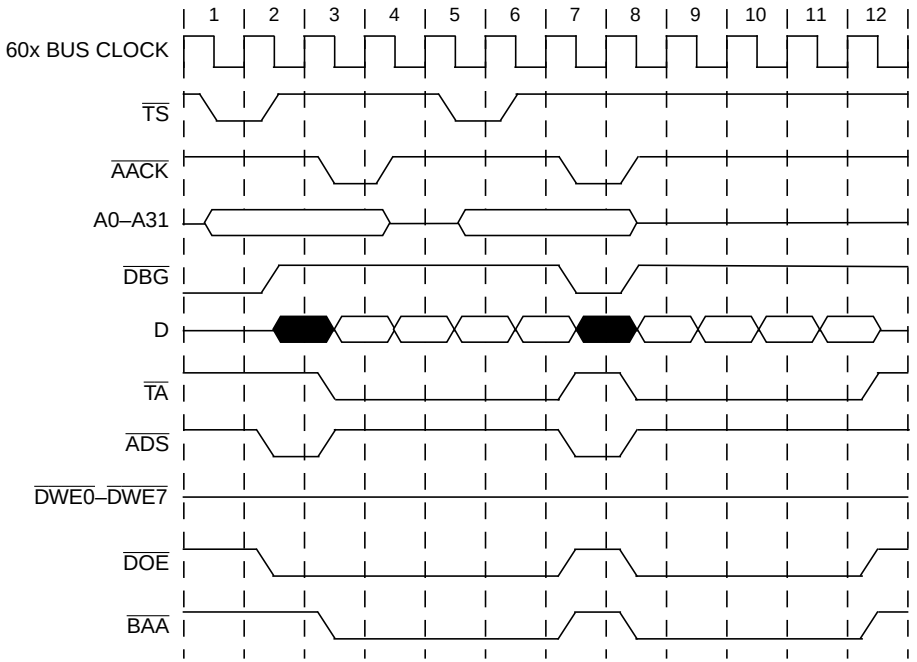


Figure 5-17. L2 Cache Read Hit Timing with CF_DOE = 1

5.4.1.2 L2 Cache Write Hit Timing

Figure 5-18 shows the L2 interface timing for a write hit in the L2 cache. The L2 cache configuration bits CF_APHASE_WS, CF_L2_HIT_DELAY, CF_DPARK, and CF_MOD_HIGH are set to 1, CF_WDATA and CF_HOLD are cleared to 0, and CF_WMODE(0–1) is set to 3.

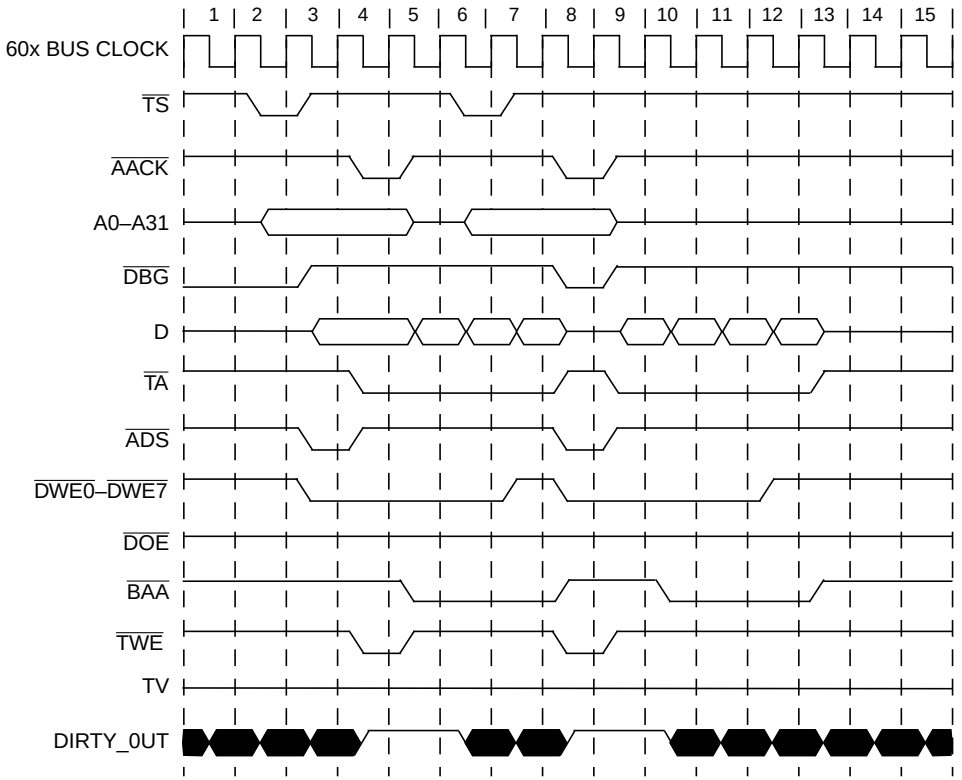


Figure 5-18. L2 Cache Write Hit Timing

5.4.1.3 L2 Cache Line Update Timing

Figure 5-19 shows the L2 interface timing for a cache line update (following a read miss). The L2 cache configuration bits CF_APHASE_WS, CF_L2_HIT_DELAY, CF_DOE, and CF_MOD_HIGH are set to 1, CF_HOLD is cleared to 0, and CF_WMODE(0–1) is set to 3.

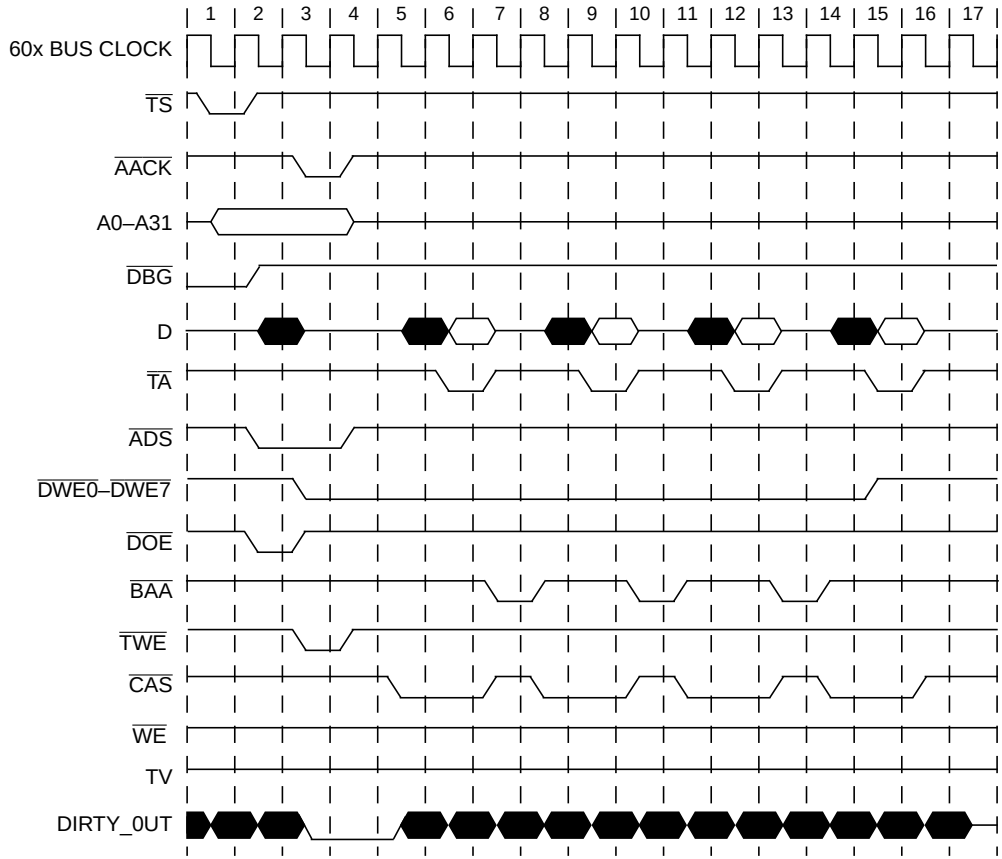


Figure 5-19. L2 Cache Line Update Timing

5.4.1.4 L2 Cache Line Cast-Out Timing

Figure 5-20 shows the L2 interface timing for an L2 cache line cast-out. The L2 cache configuration bits CF_APHASE_WS, CF_L2_HIT_DELAY, and CF_MOD_HIGH are set to 1, CF_DOE, CF_TOE_WIDTH, and CF_HOLD are cleared to 0.

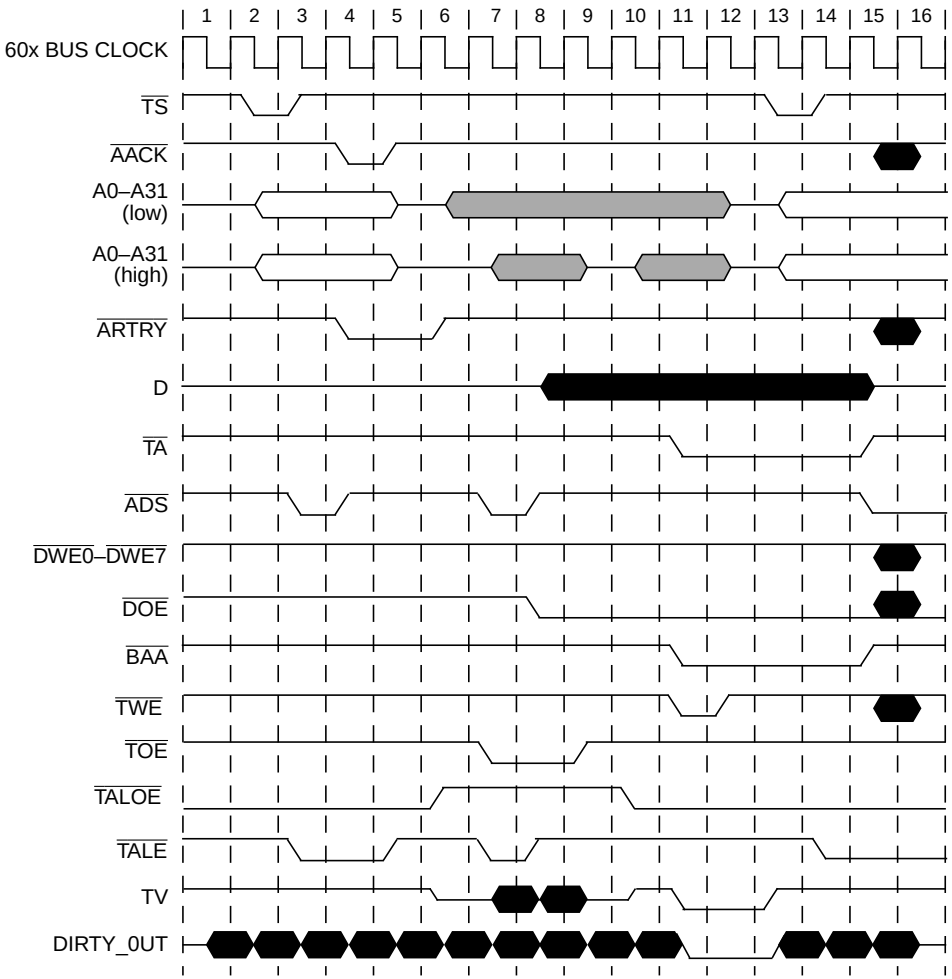


Figure 5-20. L2 Cache Line Cast-Out Timing

5.4.1.5 L2 Cache Hit Timing Following PCI Read Snoop

Figure 5-21 shows the L2 interface timing for an L2 cache hit following a PCI bus read operation that misses in the L1 cache. The L2 cache configuration bits CF_APHASE_WS, CF_L2_HIT_DELAY, and CF_SNOOP_WS are set to 1, and CF_DOE is cleared to 0.

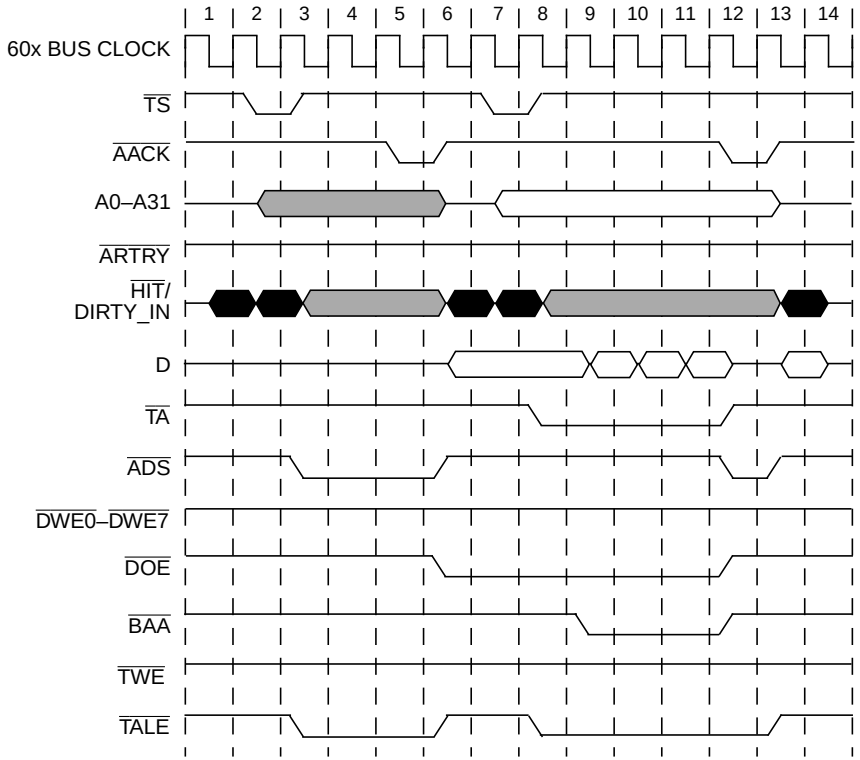


Figure 5-21. L2 Cache Hit Timing Following PCI Read Snoop

5.4.1.6 L2 Cache Line Push Timing Following PCI Write Snoop

Figure 5-22 shows the L2 interface timing for a modified L2 cache line push following a PCI bus write operation that misses in the L1 cache. The L2 cache line is invalidated following the snoop push. The L2 cache configuration bits CF_APHASE_WS, CF_L2_HIT_DELAY, CF_HOLD, CF_MOD_HIGH, and CF_SNOOP_WS are set to 1, and CF_DOE is cleared to 0.

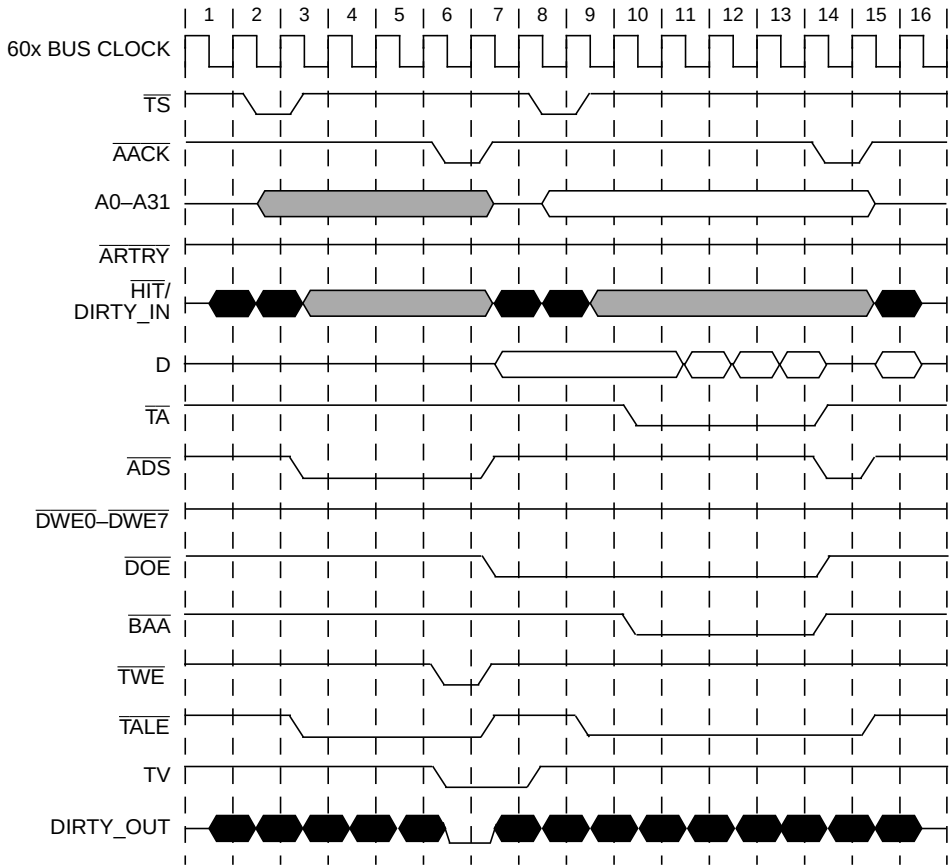


Figure 5-22. Modified L2 Cache Line Push Timing Following PCI Write Snoop

5.4.1.7 L2 Cache Line Invalidate Timing Following PCI Write with Invalidate Snoop

Figure 5-23 shows the L2 interface timing for an L2 cache line invalidate following a PCI bus write with invalidate operation that misses in the L1 cache. The L2 cache configuration bits CF_APHASE_WS, CF_L2_HIT_DELAY, CF_HOLD, CF_MOD_HIGH, and CF_SNOOP_WS are set to 1, and CF_DOE is cleared to 0.

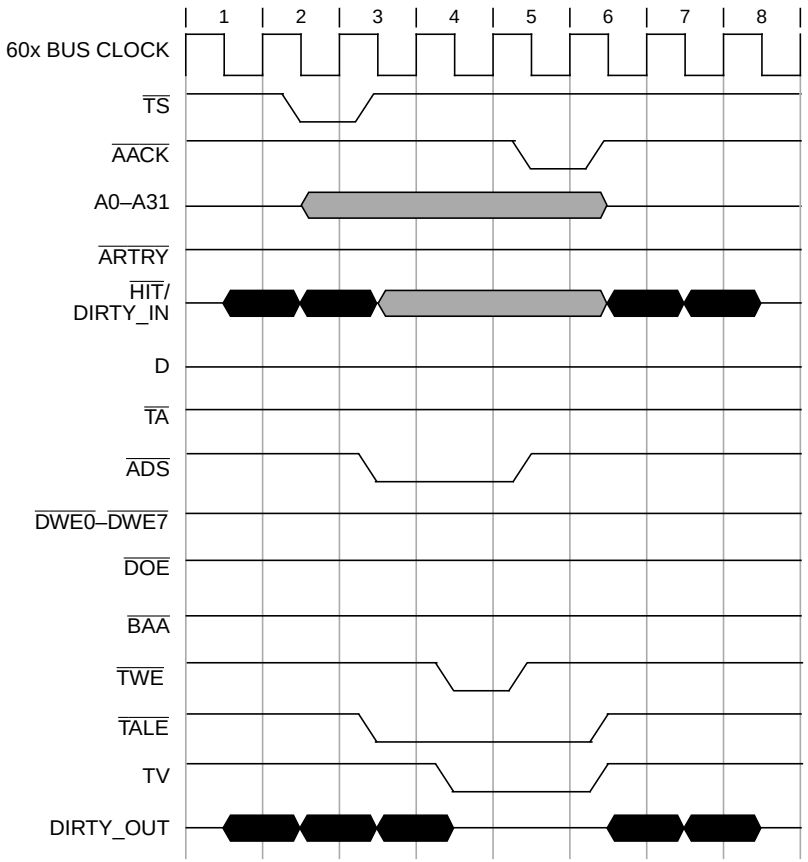


Figure 5-23. L2 Cache Line Invalidate Timing Following PCI Write with Invalidate Snoop

5.4.2 Asynchronous SRAM L2 Cache Timing

The following sections provide timing examples for L2 cache operations in systems implemented using asynchronous SRAM.

5.4.2.1 Burst Read Timing

Figure 5-24 shows the L2 interface timing for an L2 cache burst read operation with 3-2-2-2/2-2-2-2 burst timing. The L2 cache configuration bit CF_DOE is cleared to 0.

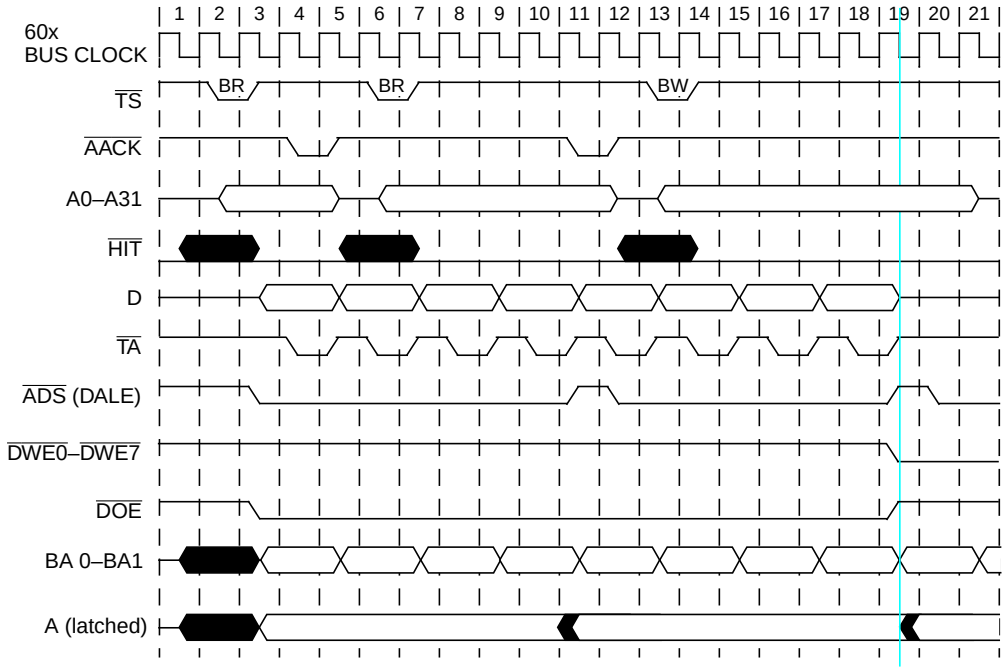


Figure 5-24. L2 Cache Burst Read Timing with CF_DOE = 0

Figure 5-25 shows the L2 interface timing for an L2 cache burst read operation with 3-2-2-2/3-2-2-2 burst timing. The L2 cache configuration bit CF_DOE is set to 1.

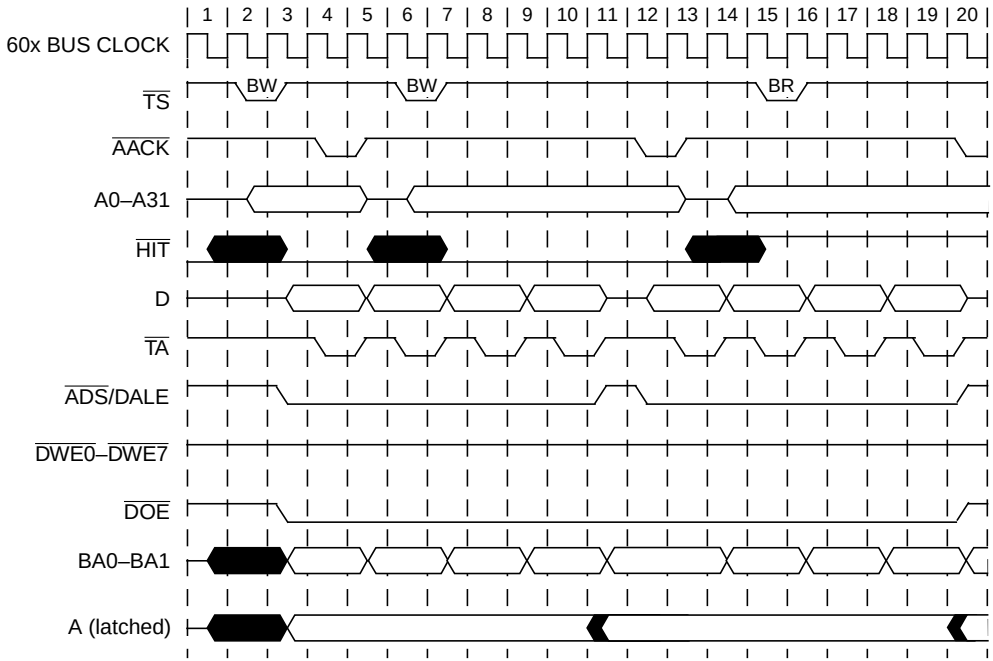


Figure 5-25. L2 Cache Burst Read Timing with CF_DOE = 1

5.4.2.2 L2 Cache Burst Read Line Update Timing

Figure 5-26 shows the L2 interface timing for an L2 cache line update during a burst read operation. The L2 cache configuration bit CF_WDATA is cleared to 0.

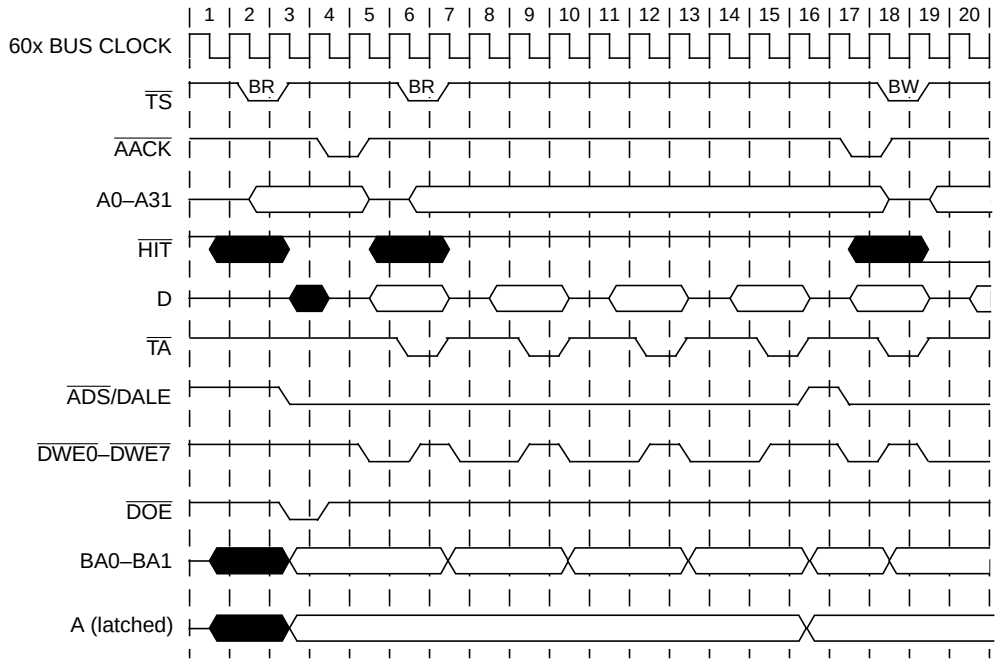


Figure 5-26. L2 Cache Burst Read Line Update Timing with CF_WDATA = 0

Figure 5-27 shows the L2 interface timing for an L2 cache line update during a burst read operation. The L2 cache configuration bit CF_WDATA is set to 1.

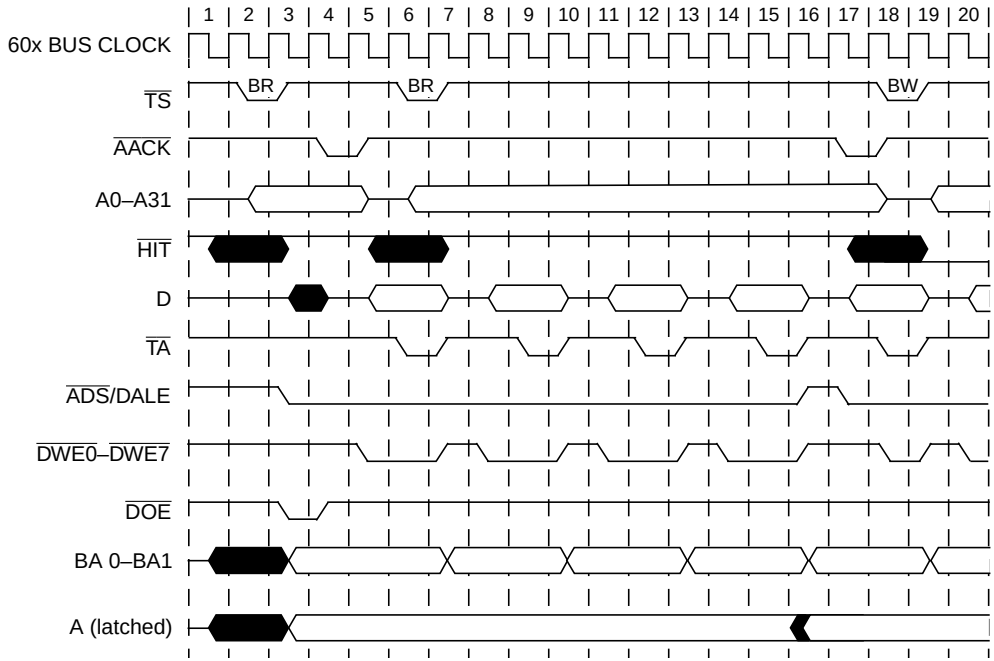


Figure 5-27. L2 Cache Burst Read Line Update Timing with CF_WDATA = 1

5.4.2.3 Burst Write Timing

Figure 5-28 shows the L2 interface timing for an L2 cache line update during a burst read operation. The L2 cache configuration bit CF_WDATA is cleared to 0 for the first bus transaction, and set to 1 for the second transaction.

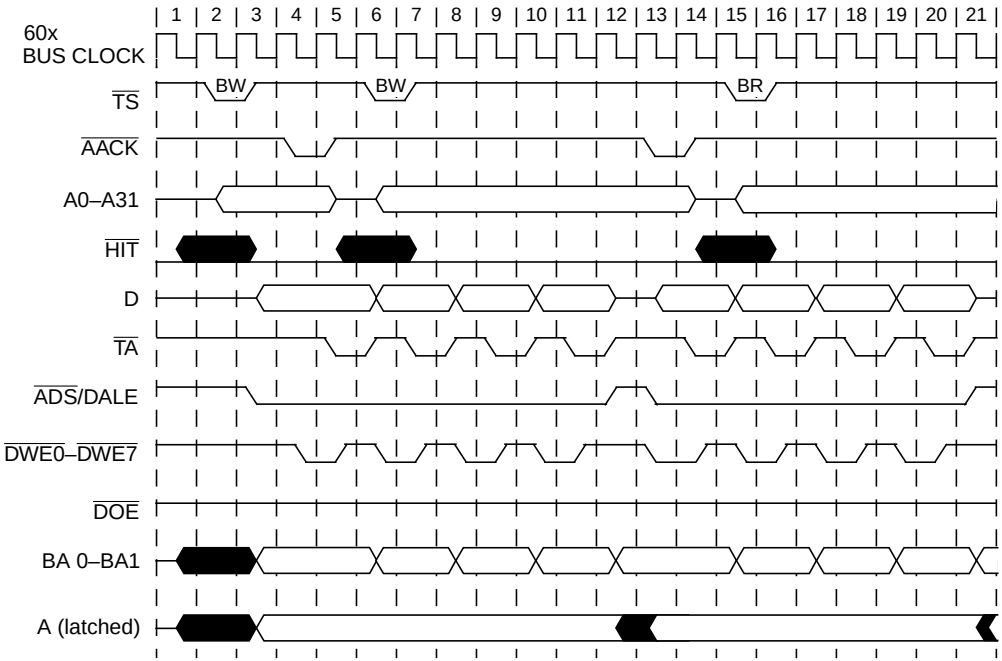


Figure 5-28. L2 Cache Burst Write Timing with CF_WDATA = 0/1

Chapter 6
Memory Interface

The memory interface of the MPC105 controls processor and PCI interactions with main memory. It is capable of supporting a variety of DRAM or SDRAM, and ROM or Flash ROM configurations as main memory. This chapter describes the memory interface on the MPC105—its features and limitations, initialization, buffering requirements, supported device organizations, and timing. Design examples are provided for the DRAM, SDRAM, ROM, and Flash ROM interfaces.

Note that only one of the RAM interfaces (DRAM or SDRAM) and only one of the ROM interfaces (ROM or Flash ROM) can be used in a system. That is, a system cannot mix DRAMs and SDRAMs or ROMs and Flash ROMs.

The maximum supported memory size is 1 Gbyte of DRAM or SDRAM, with 16 Mbytes of ROM or 1 Mbyte of Flash ROM. Software initialization of on-chip configuration registers configures the memory interface to support the various memory sizes.

Chapter 2, “Signal Descriptions,” contains the signal definitions for the memory interface, and Chapter 3, “Device Programming,” details the configurable parameters that are used to initialize the memory interface. In addition, Chapter 8, “Internal Control,” provides information about the internal buffers that permit the MPC105 to coordinate memory accesses between the L2 cache, the 60x processor(s), and devices on the PCI bus.

6.1 Overview

The MPC105 can control either a 64- or 32-bit data path to main memory; SDRAM systems support 64-bit data paths only. Parity protection is provided for the DRAM or SDRAM. The MPC105 handles parity checking and generation, with four parity bits checked or generated for a 32-bit data path, and eight parity bits checked or generated for a 64-bit data path. To reduce loading on the data bus, system designers may choose to implement data buffers between the 60x bus and memory. The MPC105 features configurable data buffer control logic to accommodate several buffer types.

The MPC105 is capable of supporting a variety of DRAM or SDRAM configurations. Note that if SDRAM is used, it must comply with the JEDEC specification for SDRAM. Twelve multiplexed address signals provide for device densities to 16M. Eight row address strobe/command select ($\overline{RAS}/\overline{CS}$) signals support up to eight banks of memory. Each bank can be 8 bytes wide. Eight column address strobe/data qualifier ($\overline{CAS}$/DQM) signals are used to provide byte selection for memory accesses.

The DRAM or SDRAM banks can be built of SIMMs or directly-attached memory chips. The data path to the memory banks must be either 32 or 64 bits wide (36 or 72 with parity). The banks can be constructed using x 1, x 4, x 8, x 9, x 16, or x 18 memory chips. Regardless of whether DRAMs or SDRAMs are used, the memory design must be byte-selectable for writes using the $\overline{CAS}$/DQM signals. The MPC105 provides the capability for initialization software to determine the amount of DRAM or SDRAM in the system, and set the row/column address configuration, at system startup.

The MPC105 memory interface provides for doze, nap, sleep, and suspend power saving modes, defined in Appendix A, “Power Management.” In the sleep and suspend power saving modes, the MPC105 can be configured to put the DRAM array into a self-refresh mode, (if supported by the DRAMs). The MPC105 may be configured to use the RTC input as its refresh time base in suspend mode. If self-refreshing DRAMs are not available or the RTC input is not used (in suspend mode), system software must preserve DRAM data (such as by copying the data to disk) in the sleep or suspend mode. In doze and nap power saving modes and in full-on mode, the MPC105 supplies $\overline{CAS}$ before $\overline{RAS}$ (CBR) refresh to DRAM.

An MPC105 configuration signal (FNR/$\overline{DWE0}$, sampled at reset) determines whether the MPC105 accesses initialization software from ROM or Flash ROM. If the MPC105 is configured to access initialization software from ROM, the corresponding data path must be the same bit width as the DRAM or SDRAM data path (32 or 64 bits). Twenty address bits and two bank selects are provided for ROM systems. If the MPC105 is configured to access initialization software from Flash ROM, the corresponding data path must be 8 bits wide and must be connected to the most significant byte of the data bus. Twenty address bits, one bank select signal, one write enable signal, and one output enable signal are provided for Flash ROM systems.

6.2 Memory Interface Signal Buffering

To reduce loading on the data bus, system designers may choose to implement data buffers between the 60x bus and memory. The signals $\overline{BCTL0}$ and $\overline{BCTL1}$ control the data bus buffers (directional control and high-impedance state). The example design in Figure 6-5 uses bidirectional/tri-state drivers on the data and parity signals.

Note that in addition to the data and parity signals, certain other memory interface signals may also require buffering. Parameters such as the AC characteristics of the MPC105, memory operating frequency, capacitive loading, and board routing loads will dictate which signals require buffering, and which buffer devices are appropriate.

The MPC105 features configurable data buffer control logic to accommodate flow-through, transparent latch, or registered type data buffers. This section describes the different buffer control configurations of the MPC105.

The write buffer type and read buffer type bits in memory control configuration register 4 (MCCR4[WCBUF] and MCCR4[RCBUF]) determine the type of buffer used and the data synchronization for those buffer types. The buffer mode bit in memory control configuration register 2 (MCCR2[BUF]) controls how the buffer control signals, $\overline{\text{BCTL0}}$ and $\overline{\text{BCTL1}}$, operate.

Table 6-1 shows the parameter settings for the different configurations and gives examples of typical buffer devices that might be used in those configurations.

Table 6-1. Buffer Configurations

WCBUF	RCBUF	BUF	Buffer Control Configuration	Typical Buffer Device
0	0	0	Flow-through buffer, alternate protocol	54/7416863
0	0	1	Flow-through buffer, default protocol	54/7416245, 54/74162245, 54/74163245
0	1	0	Transparent latch buffer	54/7416543, 54/74162543
0	1	1	—	—
1	0	0	—	—
1	0	1	—	—
1	1	0	Registered buffer	54/7416952, 54/74162952
1	1	1	—	—

6.2.1 Flow-Through Buffers

The configuration bits RCBUF and WCBUF should both be cleared to indicate a flow-through buffer configuration. The MPC105 supports two protocols for the buffer control signals, $\overline{\text{BCTL0}}$ and $\overline{\text{BCTL1}}$. The buffer mode bit, BUF, determines which protocol the buffer control signals use.

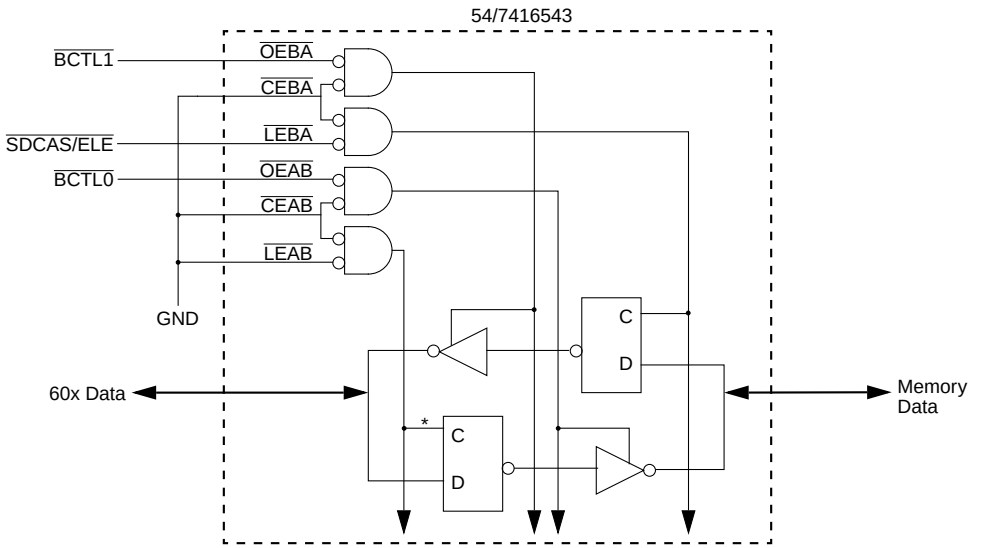
The default protocol (BUF = 1) uses $\overline{\text{BCTL0}}$ as a direction control signal, and $\overline{\text{BCTL1}}$ as a buffer enable signal. Buffers that are compatible with this protocol include the 54/7416245, 54/74162245, and 54/74163245. These buffers are available from several manufacturers.

The alternate protocol (BUF = 0) uses $\overline{\text{BCTL0}}$ as a buffer write enable signal, and $\overline{\text{BCTL1}}$ as a read enable signal. The 54/7416863, available from several manufacturers, is compatible with this protocol.

6.2.2 Transparent Latch-Type Buffers

Configuration bit RCBUF should be set and WCBUF should be cleared to indicate the transparent latch configuration. The $\overline{\text{BCTL0}}$, $\overline{\text{BCTL1}}$ and $\overline{\text{SDCAS/ELE}}$ signals control the buffer. The $\overline{\text{BCTL0}}$ signal acts as a buffer write enable signal, and $\overline{\text{BCTL1}}$ acts as a read enable signal. When inactive, $\overline{\text{SDCAS/ELE}}$ provides additional data hold time, as might be required by an asynchronous L2 cache (refer to the $\overline{\text{SDCAS/ELE}}$ waveform shown in Figure 6-9).

Buffers that are compatible with the transparent latch configuration include the 54/7416543 and 54/74162543. These buffers are available from several manufacturers. Connections to the buffer device are shown in Figure 6-1.



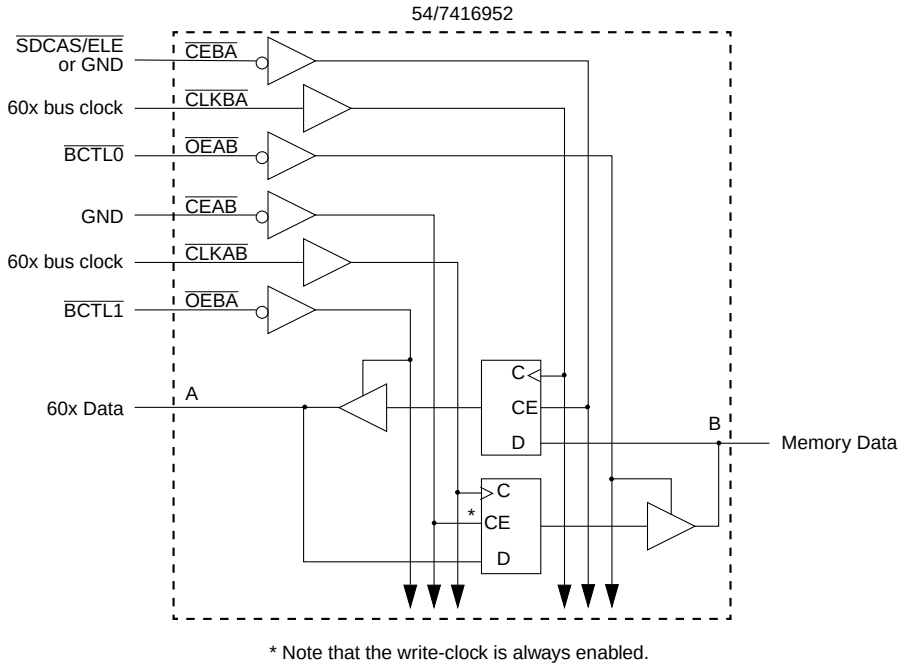
* Note that this configuration intentionally allows the write latch to remain in a flow-through state.

Figure 6-1. Transparent Latch-Type Buffer

6.2.3 Registered Buffers

Configuration bits RCBUF and WCBUF should both be set to indicate the registered configuration. The $\overline{\text{BCTL0}}$ and $\overline{\text{BCTL1}}$ signals control the buffer. For the example in Figure 6-2, $\overline{\text{BCTL0}}$ acts as a buffer write enable signal, and $\overline{\text{BCTL1}}$ acts as a read enable signal. The $\overline{\text{SDCAS/ELE}}$ signal may optionally be used as the register read-clock enable. When negated, $\overline{\text{SDCAS/ELE}}$ provides additional data hold time, as might be required by an asynchronous L2 cache (refer to the $\overline{\text{SDCAS/ELE}}$ waveform shown in Figure 6-9).

Buffers that are compatible with the registered configuration include the 54/7416952 and 54/74162952. These buffers are available from several manufacturers. Connections to this buffer device should be made as shown in Figure 6-2.



* Note that the write-clock is always enabled.

Figure 6-2. Registered Buffer

6.2.4 Parity Path Read Control

A read operation from ROM or Flash ROM located on the 60x/memory data bus can cause a conflict with the memory data parity buffers. To eliminate this conflict, external logic must be employed to decode a separate read enable signal for the parity path for systems with the ROM or Flash ROM on the 60x/memory bus. The parity path read control logic is easily implemented in either programmable or discrete logic. Figure 6-3 shows the parity path read control logic for ROM-based systems.

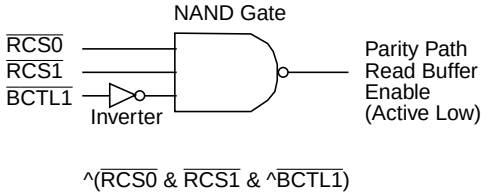


Figure 6-3. Parity Path Read Control Logic for ROM-Based Systems

Figure 6-4 shows the parity path read control logic for Flash ROM-based systems.

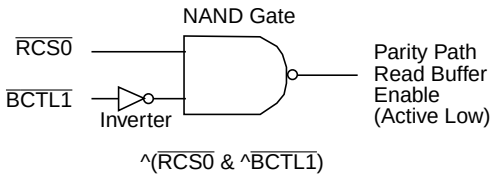


Figure 6-4. Parity Path Read Control Logic for Flash ROM-Based Systems

6.3 DRAM Interface Operation

Up to eight RAM banks can be built of DRAMs or SIMMs that range from 1M to 16M in depth, and from 1 bit to 72 bits in width. The MPC105 provides 1 Gbyte of addressable memory.

In addition to the eight $\overline{\text{CAS}}$/DQM signals, the eight $\overline{\text{RAS/CS}}$ signals, and the 12 row/column multiplexed address signals (MA0–MA11), there are 64 data signals (DH0–DH31 and DL0–DL31), a write enable ($\overline{\text{WE}}$) signal, and one parity bit (PAR*n*) per byte of data.

Figure 6-5 shows an example of a two-bank, 16-Mbyte DRAM system.

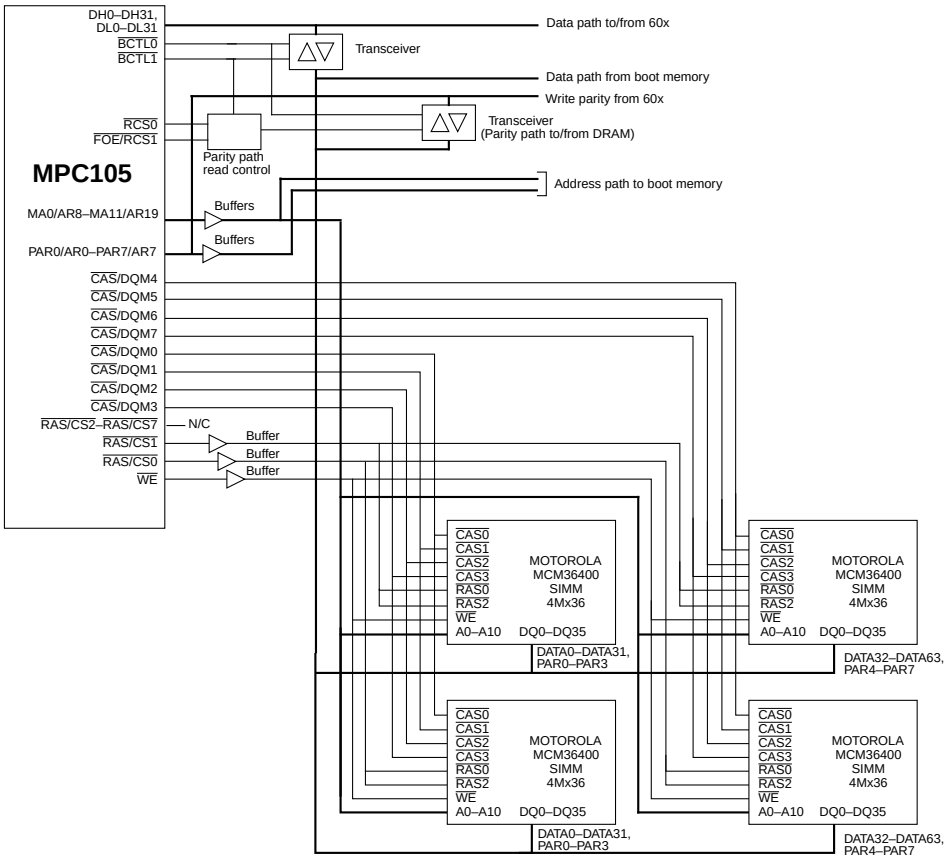


Figure 6-5. 16-Mbyte DRAM System with Parity

6.3.1 Supported DRAM Organizations

It is not necessary to use identical DRAM devices in each memory bank; individual memory banks may be of differing size. Although the MPC105 multiplexes the row and column address bits onto 12 memory address signals, individual DRAM banks may be implemented with memory devices requiring fewer than 24 address bits. The MPC105 can be configured to provide 12, 11, 10, or 9 row address bits to a particular bank, and 12, 11, 10, or 9 column address bits.

The data path to the memory banks must be either 32 or 64 bits wide (36 or 72 with parity). The banks can be constructed using x1, x4, x8, x9, x16, or x18 memory devices. However, the memory system design must use the $\overline{\text{CAS}}$/DQM signals for byte lane selection. Table 6-2 summarizes some of the memory configurations supported by the MPC105.

Table 6-2. Memory Device Configurations Supported with 64-Bit Data Bus

Number of Devices in a Bank	Device Configuration	Row Bits x Column Bits	Bank Size	Maximum Memory (Using All 8 Banks)
64	16M x 1	12 x 12	128 Mbytes	1 Gbyte
16	4M x 4	12 x 10	32 Mbytes	256 Mbytes
64	4M x 1	11 x 11	32 Mbytes	256 Mbytes
16	4M x 4	11 x 11	32 Mbytes	256 Mbytes
8	2M x 8	11 x 10	16 Mbytes	128 Mbytes
16	1M x 4	10 x 10	8 Mbytes	64 Mbytes
64	1M x 1	10 x 10	8 Mbytes	64 Mbytes
4	1M x 16	10 x 10	8 Mbytes	64 Mbytes
8	512K x 8	10 x 9	4 Mbytes	32 Mbytes
4	256K x 16	9 x 9	2 Mbytes	16 Mbytes
16	256K x 4	9 x 9	2 Mbytes	16 Mbytes

By using a memory polling algorithm at power-on reset, system firmware configures the MPC105 to correctly map the size of each bank in memory (the memory boundary registers). The MPC105 uses its bank map to assert the appropriate $\overline{\text{RAS/CS}}$ signal for memory accesses according to the provided bank depths.

System software must also configure the MPC105 at power-on reset to appropriately multiplex the row and column address bits for each bank. This is done by writing the row address configuration into a specific configuration register. Address multiplexing will then occur according to the configuration settings, as shown in Figure 6-6 for 64-bit bus mode, and in Figure 6-7 for 32-bit bus mode.

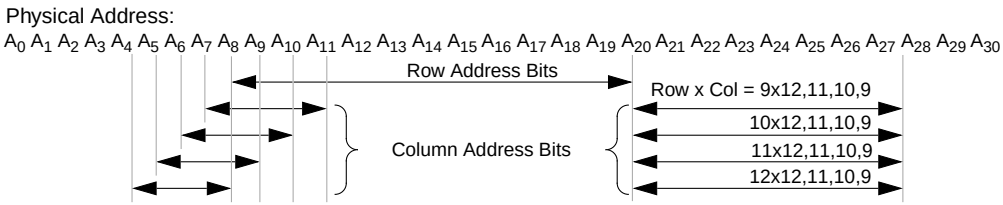


Figure 6-6. DRAM Address Multiplexing—64-Bit Data Bus Mode

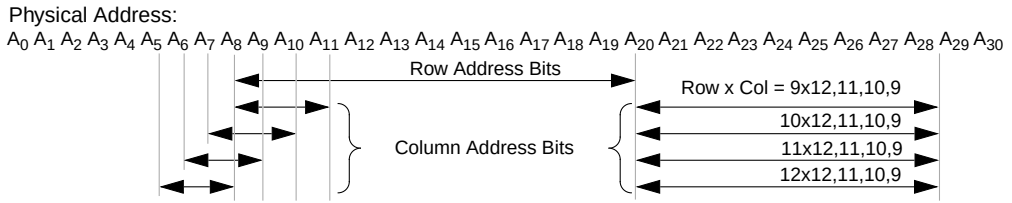


Figure 6-7. DRAM Address Multiplexing—32-Bit Data Bus Mode

6.3.2 DRAM Power-On Initialization

At system reset, initialization software (boot code) must set up the programmable parameters in the memory interface configuration registers (MICRs). These include the memory boundary registers, the memory bank enable register, and the memory control configuration registers (MCCRs). See Section 3.2.6, “Memory Interface Configuration Registers,” for more detailed descriptions of the MICRs.

The programmable parameters relevant to DRAM-based systems are:

- Memory bank starting and ending addresses (memory boundary registers)
- Memory bank enables (memory bank enable register)
- SREN—self-refresh enable (MCCR1[18])
- RAMTYP—RAM type (MCCR1[17])
- PCKEN—memory interface parity checking/generation enable (MCCR1[16])
- Row address bit count for each bank (MCCR1[15–0])
- SRF—self-refresh entry delay (MCCR2[31–18])
- REFINT—refresh interval (MCCR2[15–2])
- BUF—buffer mode (MCCR2[1])
- RAS_{6P}—$\overline{RAS}$ assertion interval for CBR refresh (MCCR3[18–15])
- CAS_5—$\overline{CAS}$ assertion interval for page mode access (MCCR3[14–12])
- CP_4—$\overline{CAS}$ precharge interval (MCCR3[11–9])
- CAS_3—$\overline{CAS}$ assertion interval for a single beat, or for the first access in a burst (MCCR3[8–6])
- RCD_2—$\overline{RAS}$ to $\overline{CAS}$ delay interval (MCCR3[5–3])
- RP_1—$\overline{RAS}$ precharge interval (MCCR3[2–0])
- WCBUF—memory write buffer type (MCCR4[21])
- RCBUF—memory read buffer type (MCCR4[20])

Once all the memory parameters are configured, then system software should set the MEMGO bit (MCCR1, bit 19) to enable the memory interface. The MPC105 then performs eight refreshes on the DRAM array (in accordance with REFINT) to prepare it for system access. When these refreshes are complete, the memory array is ready for access.

Note that 100 µs must elapse after the negation of $\overline{\text{HRST}}$ before the MEMGO bit can be set, so a delay loop in the initialization code may be necessary.

6.3.3 DRAM Interface Timing

System software is responsible for optimal configuration of the DRAM programmable timing parameters (RP_1, RCD_2, CAS_3, CP_4, and CAS_5) at reset. The programmable timing parameters are applicable to both read and write timing configuration. The configuration process must be completed, before any accesses to DRAM are attempted.

It is very important to note that the MPC105 should never be programmed such that successive assertions of $\overline{\text{CAS}}$/DQM within a burst will have more than eight PCI clocks between the start of assertions. This should be easy to achieve with existing DRAM technology.

Table 6-3 represents suggested timing configurations for Motorola DRAM technology. The actual values used by initialization software depend upon the specific memory technology used in the system.

Table 6-3. Suggested DRAM Timing Configurations

Bus Frequency	RP_1		RCD_2	CAS_3		CP_4	CAS_5	
	70 ns	60 ns		70 ns	60 ns		70 ns	60 ns
25 MHz	2	2	2	1	1	1	1	1
33 MHz	2	2	2	1–2	1	1	1–2	1
50 MHz	3	3	2	2–3	2	1	2	2
66 MHz	4	3	2–3	3–4	2–3	1	2–3	2

Figure 6-8 through Figure 6-12 illustrate DRAM timing for various types of accesses; see Figure 6-8 for a single-beat read operation, Figure 6-9 for a burst-of-four read operation, Figure 6-10 for a burst-of-eight read operation, Figure 6-11 for a single-beat write operation, and Figure 6-12 for a burst-of-four write operation. Note that all signal transitions occur on the rising edge of the 60x bus clock.

Table 6-4 describes the acronyms used in the DRAM timing diagrams. The acronyms with subscripted numbers represent the programmable timing parameters of the MPC105.

Table 6-4. DRAM Timing Parameters

Symbol	Timing Parameter
AA	Access time from column address
ASC	Column address setup time
ASR	Row address setup time
CAC	Access time from $\overline{\text{CAS}}$

Table 6-4. DRAM Timing Parameters (Continued)

Symbol	Timing Parameter
CAH	Column address hold time
CAS_3	$\overline{CAS}$ assertion interval for a single beat, or for the first access in a burst
CAS_5	$\overline{CAS}$ assertion interval for page mode access
CHR	$\overline{CAS}$ hold time for CBR refresh
CP	$\overline{CAS}$ precharge time
CP_4	$\overline{CAS}$ precharge interval during page-mode access
CRP	$\overline{CAS}$ to $\overline{RAS}$ precharge time
CSH	$\overline{CAS}$ hold time
CSR	$\overline{CAS}$ setup time for CBR refresh
DH	Data in hold time
DS	Data in setup time
PC	Fast page mode cycle time
RAC	Access time from $\overline{RAS}$
RAD	$\overline{RAS}$ to column address delay time
RAH	Row address hold time
RAL	Column address to $\overline{RAS}$ lead time
RAS_{6P}	$\overline{RAS}$ assertion interval for CBR refresh
RASP	$\overline{RAS}$ pulse width (page-mode)
RASS	Self-refresh interval (power saving modes only)
RC	Random access (read, write, or refresh) cycle time
RCD_2	$\overline{RAS}$ to $\overline{CAS}$ delay interval
RHCP	$\overline{RAS}$ hold time from $\overline{CAS}$ precharge (page mode only)
RP_1	$\overline{RAS}$ precharge interval
RPC	$\overline{RAS}$ precharge to $\overline{CAS}$ active time
RSH	$\overline{RAS}$ hold time
WCH	Write command hold time (referenced to $\overline{CAS}$)
WCS	Write command setup time
WP	Write command pulse width
WRH	Write to $\overline{RAS}$ hold time (CBR refresh)
WRP	Write to $\overline{RAS}$ precharge time (CBR refresh)

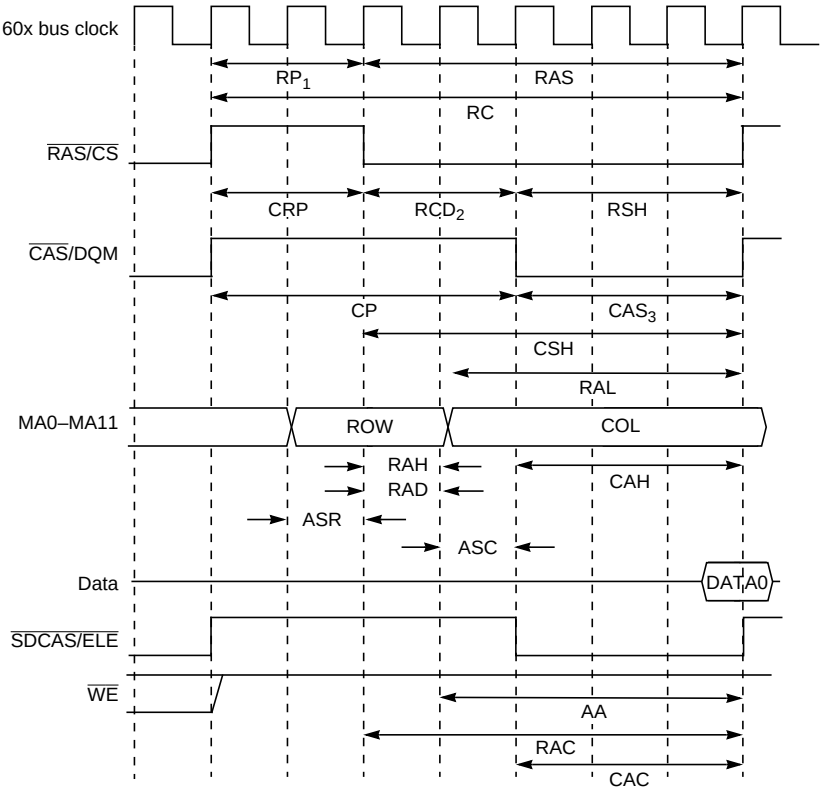


Figure 6-8. DRAM Single-Beat Read Timing

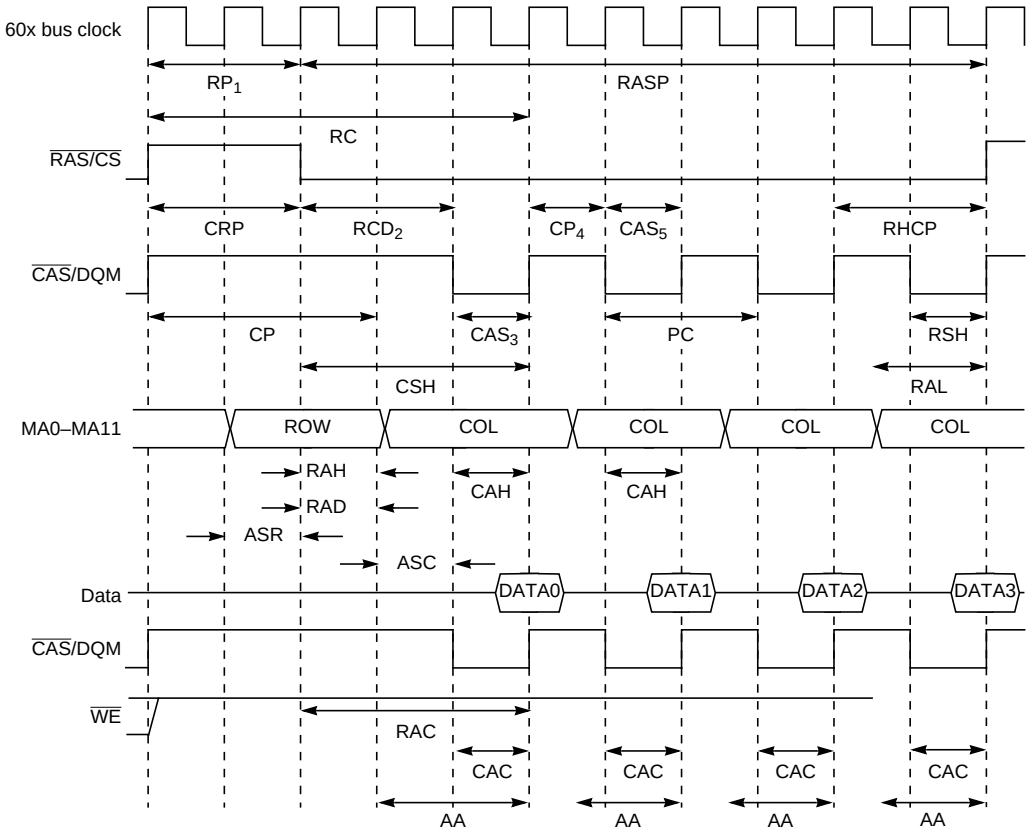


Figure 6-9. DRAM Burst-of-Four Read Timing

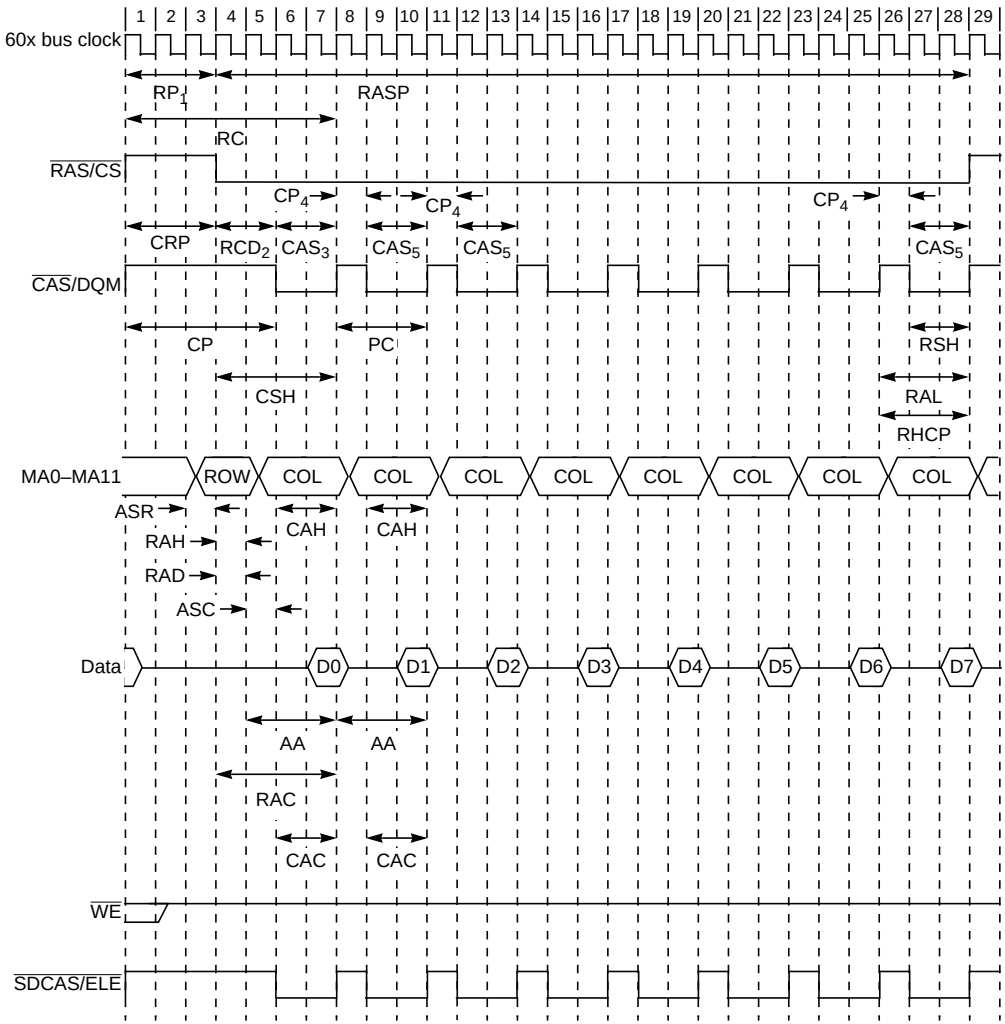


Figure 6-10. DRAM Burst-of-Eight Read Timing

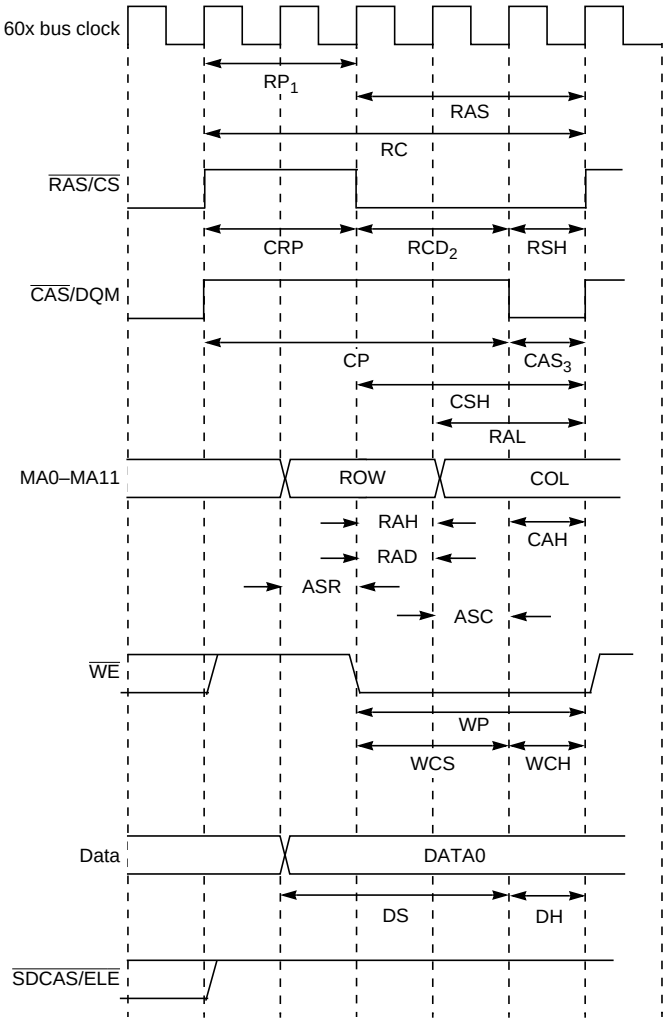


Figure 6-11. DRAM Single-Beat Write Timing

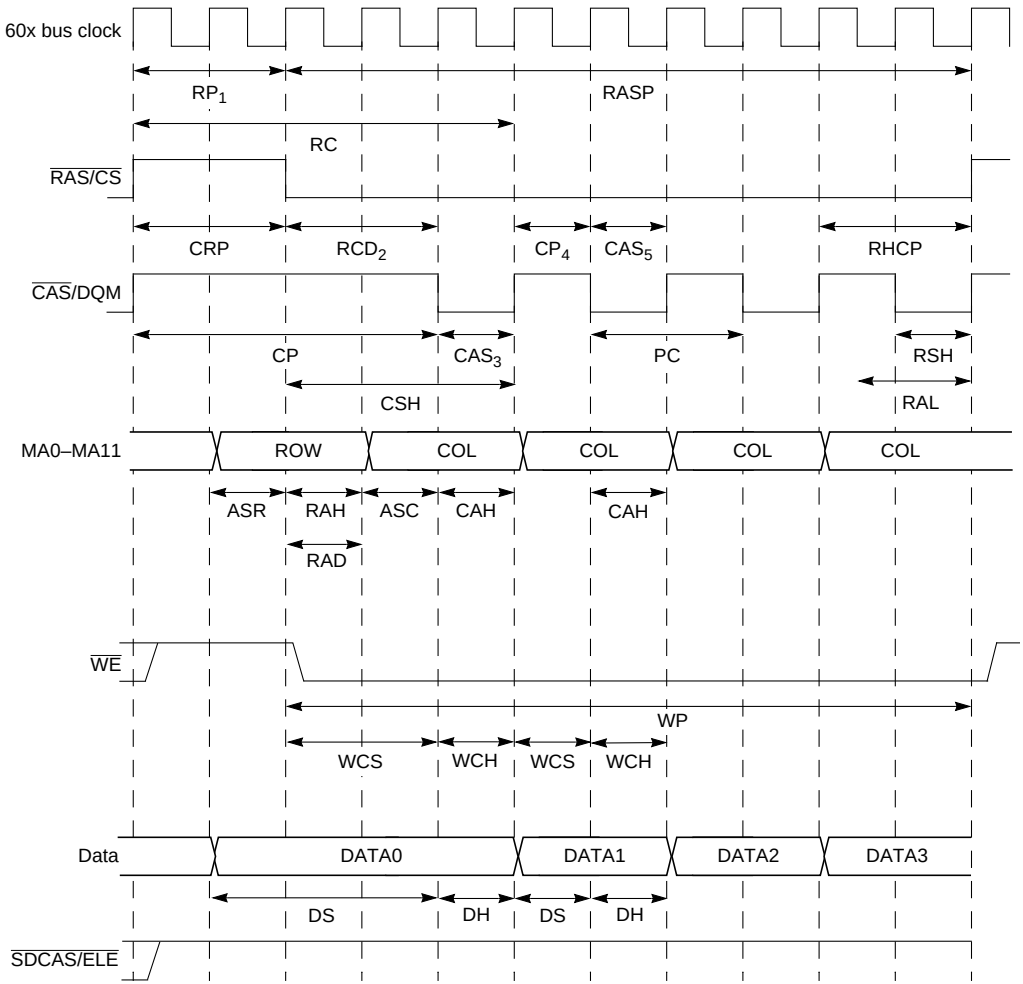


Figure 6-12. DRAM Burst-of-Four Write Timing

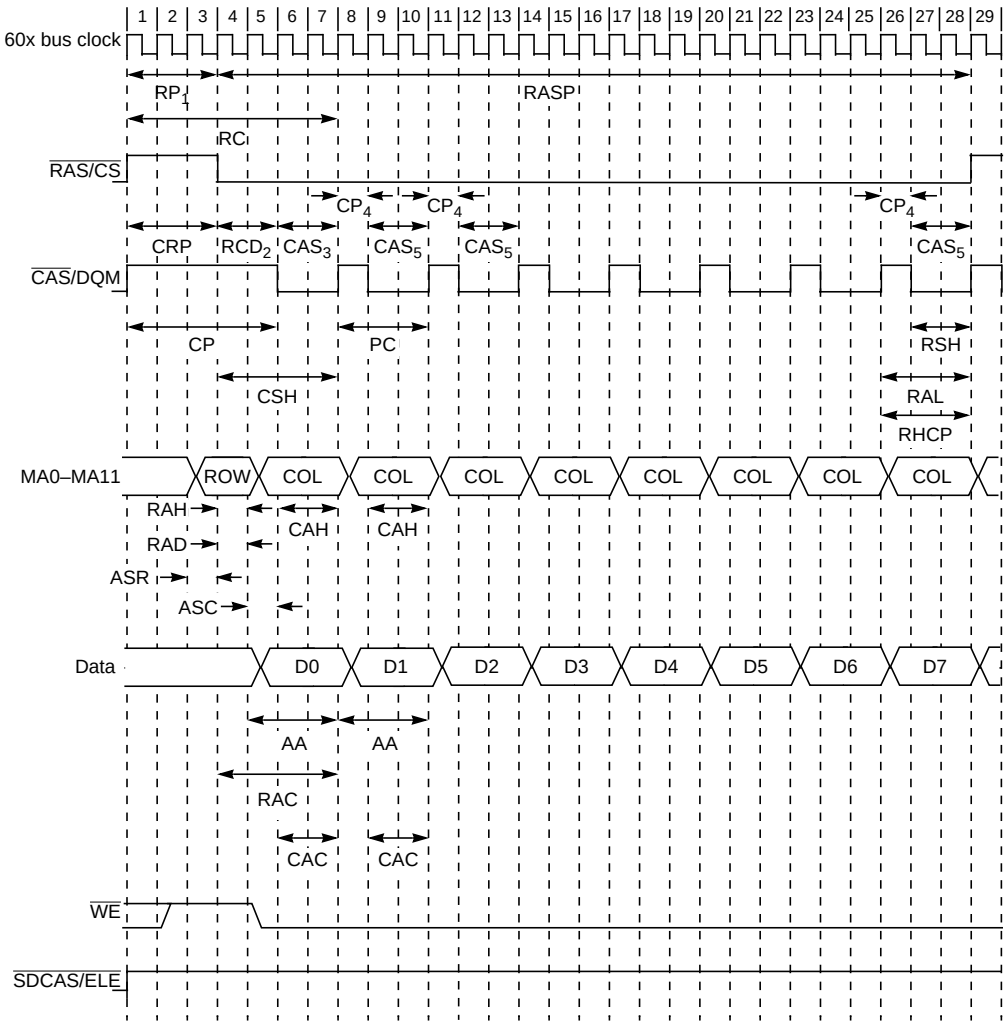


Figure 6-13. DRAM Burst–of–Eight Write Timing

6.3.3.1 DRAM Burst Wrap

The MPC105 supports burst-of-four data beats for accesses with a 64-bit data path and burst-of-eight data beats for accesses with a 32-bit data path. The burst is always sequential, and the critical double word is always supplied first as detailed in the following two examples:

1. Using a 64-bit data path, if the 60x requests the third double word of a cache line, the MPC105 reads double words from memory in the order 2-3-0-1.
2. Using a 32-bit data path, if the 60x requests the third double word of a cache line, the MPC105 reads words from memory in the order 4-5-6-7-0-1-2-3.

6.3.3.2 DRAM Latency

Table 6-5 summarizes the estimated memory latency (in processor cycles) from the assertion of $\overline{TS}$ by the 60x processor to the time the first data is returned. The latency calculations assume $\overline{RAS}$ has been precharged, the MPC105 is idle, and the 60x processor and the MPC105 are operating at the same frequency.

Table 6-5. Estimated Memory Latency

DRAM Access Time	System Timing	Processor Frequency			
		25 MHz	33 MHz	50 MHz	66MHz
60 ns	Aggressive (lightly loaded)	5-2-2-2	5-2-2-2	6-3-3-3	8-3-3-3
	Conservative (heavily loaded)	6-2-2-2	6-2-2-2	7-3-3-3	9-4-4-4
70 ns	Aggressive (lightly loaded)	5-2-2-2	5-3-3-3	7-3-3-3	9-4-4-4
	Conservative (heavily loaded)	6-2-2-2	6-3-3-3	8-3-3-3	10-4-4-4

6.3.4 DRAM Refresh

The memory interface supplies $\overline{CAS}$ before $\overline{RAS}$ (CBR) refreshes to DRAM according to the refresh interval, MCCR2[REFINT]. The value stored in REFINT represents the number of 60x bus clock cycles required between CBR refreshes. The value for REFINT depends on the specific DRAMs used and the operating frequency of the MPC105. The value should allow for any potential collisions between DRAM accesses and refresh cycles. The period of the refresh interval must be greater than the access time to insure that read and write operations complete successfully.

If a burst read is in progress at the time a refresh operation is to be performed, the refresh waits for the read to complete. In the worst case, the refresh must wait the number of clock cycles required by the longest programmed access time. The value stored to REFINT should be the number of clock cycles between row refreshes reduced by the number of clock cycles required by the longest access time (to allow for potential collisions).

For example, given a DRAM with 4096 rows and a cell refresh time of 64 ms, the number of clocks between row refreshes would be 64 ms ÷ 4096 rows = 15.6 µs. If the 60x bus clock is running at 66 MHz, the number of clock cycles per row refresh is 15.6 µs x 66 MHz = 1030 clock cycles. If the number of clock cycles for the longest burst access time is 24 clocks, then the value stored in REFINT would be 0b00_0011_1110_1110 (in decimal, 1030 – 24 = 1006).

6.3.4.1 DRAM Refresh Timing

The refresh timing for DRAM is controlled by the programmable timing parameter MCCR3[RAS_{6P}]. RAS_{6P} determines the number of 60x bus clock cycles that $\overline{RAS/CS}$ is held asserted during a CBR refresh. The MPC105 implements bank staggering for CBR refreshes, as shown in Figure 6-14. Table 6-4 describes the acronyms used in Figure 6-14. The acronyms with subscripted numbers represent the programmable timing parameters of the MPC105.

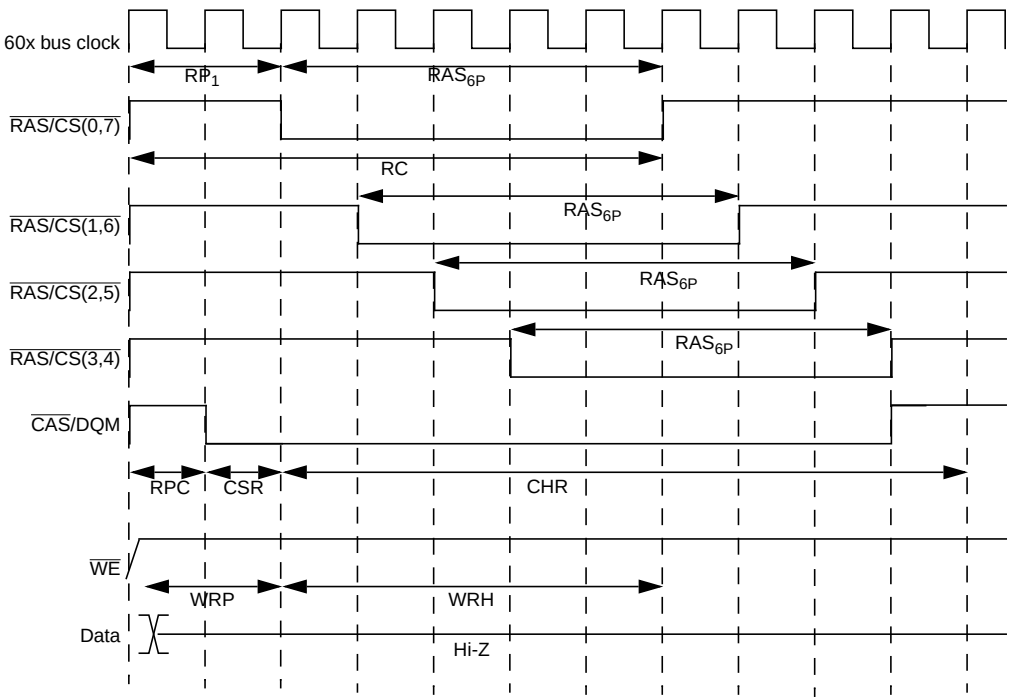


Figure 6-14. DRAM Bank Staggered CBR Refresh Timing

System software is responsible for optimal configuration of RAS_{6P} at reset. The configuration process must be completed before any accesses to DRAM are attempted. Table 6-6 represents suggested refresh timing configurations for Motorola DRAM technology. The actual values used by initialization software depend upon the specific memory technology used in the system.

Table 6-6. Suggested DRAM Refresh Timing Configurations

Processor Frequency	RAS_{6P}	
	70 ns	60 ns
25 MHz	2	2
33 MHz	3	2
50 MHz	4	3
66 MHz	5	4

6.3.4.2 DRAM Refresh and Power Saving Modes

The MPC105 memory interface provides for doze, nap, sleep, and suspend power saving modes defined in Appendix A, “Power Management.”

In doze and nap power saving modes and in full-on mode, the MPC105 supplies the normal CBR refresh to DRAM. In sleep mode, the MPC105 can be configured to take advantage of self-refreshing DRAMs, provide normal refresh to DRAM, or provide no refresh support. In suspend mode, the MPC105 can be configured to take advantage of self-refreshing DRAMs, provide CBR refreshes based on the RTC input signal, or provide no refresh support. Normal refresh is not available for DRAM in suspend mode. Table 6-7 summarizes the refresh types available in each of the power saving modes and the relevant configuration parameters.

Table 6-7. DRAM Power Saving Modes Refresh Configuration

Power Saving Mode	Refresh Type	$\overline{\text{SUSPEND}}$ Signal	Power Management Control Register (PMCR)					MCCR1 [SREN]
			[PM]	[DOZE]	[NAP]	[SLEEP]	[LP_REF_EN]	
Doze	Normal	Negated (High)	1	1	0	0	—	—
Nap	Normal	Negated (High)	1	—	1	0	—	—
Sleep	Self	Negated (High)	1	—	—	1	1	1
	Normal	Negated (High)	1	—	—	1	1	0
	None	Negated (High)	1	—	—	1	0	—

Table 6-7. DRAM Power Saving Modes Refresh Configuration (Continued)

Power Saving Mode	Refresh Type	$\overline{\text{SUSPEND}}$ Signal	Power Management Control Register (PMCR)					MCCR1 [SREN]
			[PM]	[DOZE]	[NAP]	[SLEEP]	[LP_REF_EN]	
Suspend[1]	Self	Asserted (Low)	1	—	—	0	1	1
	RTC	Asserted (Low)	1	—	—	0	1	0
	None	Asserted (Low)	1	—	—	0	0	—
	Note: [1]Normal refresh is not available in suspend mode.							

Note that in the absence of refresh support, system software must preserve DRAM data (such as by copying the data to disk) before entering the power saving state.

6.3.4.2.1 Self-Refresh in Sleep and Suspend Modes

The setup timing for self-refreshing DRAMs is shown in Figure 6-15; see Table 6-4 for acronyms used in Figure 6-15.

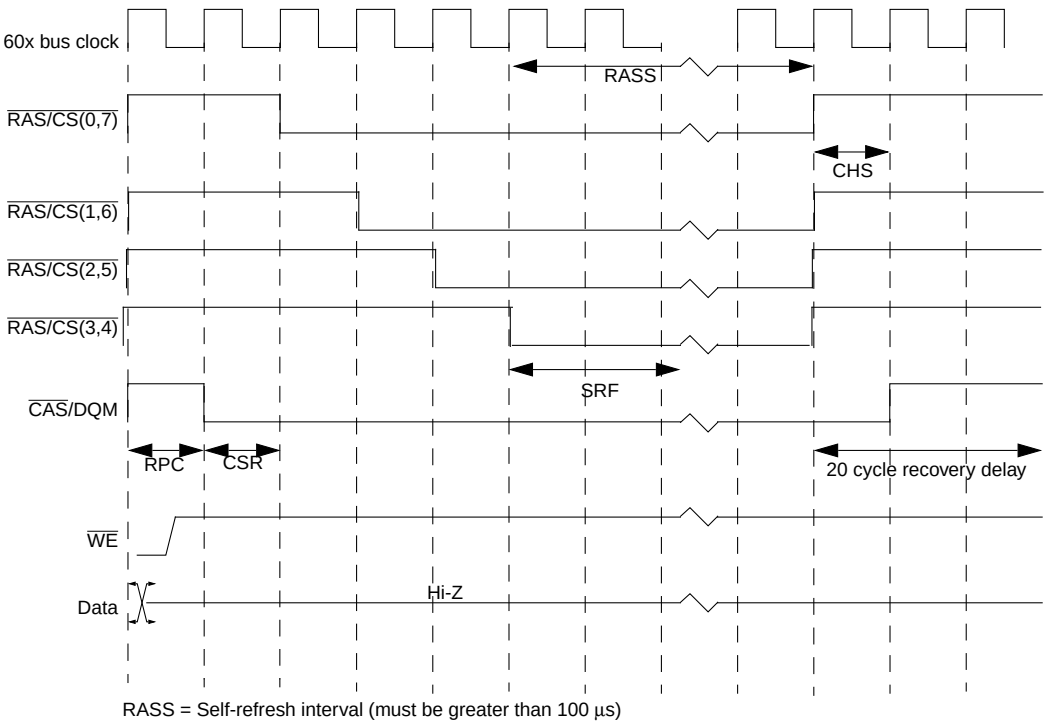


Figure 6-15. DRAM Self-Refresh Timing in Sleep and Suspend Modes

6.3.4.2.2 RTC Refresh in Suspend Mode

If the MPC105 is configured to provide refresh cycles based on the RTC signal in suspend mode, $\overline{\text{RAS}}$/$\overline{\text{CS}}$ and $\overline{\text{CAS}}$/DQM are driven as shown in Figure 6-16. The DRAMs used in this mode of operation must have a distributed refresh interval no less than 4 times the period of the RTC input signal. For example, if the frequency of RTC is 32.768 KHz, the DRAMs must have a low-power refresh interval greater than 122 µs to allow use of the RTC refresh feature.

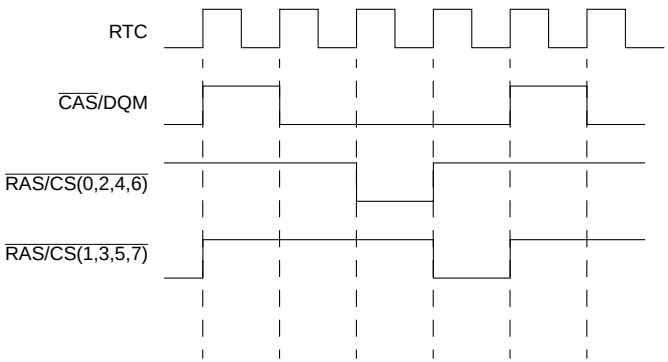


Figure 6-16. Suspend Mode—Real Time Clock Refresh

6.4 SDRAM Interface Operation

The MPC105 provides the necessary control functions and signals for JEDEC-compliant SDRAM devices. The MPC105 provides 1 Gbyte of addressable memory.

Twelve row/column multiplexed address signals (MA0–MA11) provide for SDRAM densities from 1M to 16M in depth. Eight command select ($\overline{\text{RAS}}$/$\overline{\text{CS}}$) signals support up to eight banks of memory. Eight data qualifier ($\overline{\text{CAS}}$/DQM) signals are used to provide byte selection for 64-bit accesses.

In addition to the $\overline{\text{CAS}}$/DQM, $\overline{\text{RAS}}$/$\overline{\text{CS}}$, and MA0–MA11, there are 64 data signals (DH0–DH31 and DL0–DL31), a write enable ($\overline{\text{WE}}$) signal, a column address strobe ($\overline{\text{SDCAS}}$/$\overline{\text{ELE}}$) signal, a row address strobe ($\overline{\text{SDRAS}}$), a memory clock enable (CKE) signal, and one parity bit (PAR*n*) per byte of data.

The MPC105 supports burst-of-four data transfers for SDRAM-based systems. However, the MPC105 does not support burst-of-eight transfers or 32-bit data bus mode for SDRAM-based systems. The SDRAM memory bus must be 64 bits wide (or 72 bits with parity).

Figure 6-17 shows an example of an eight-bank, 128-Mbyte system, comprised of 2M bit x 9 SDRAMs. Note that the SDRAM memory clock must operate at the same frequency as, and phase aligned with, the 60x processor bus clock.

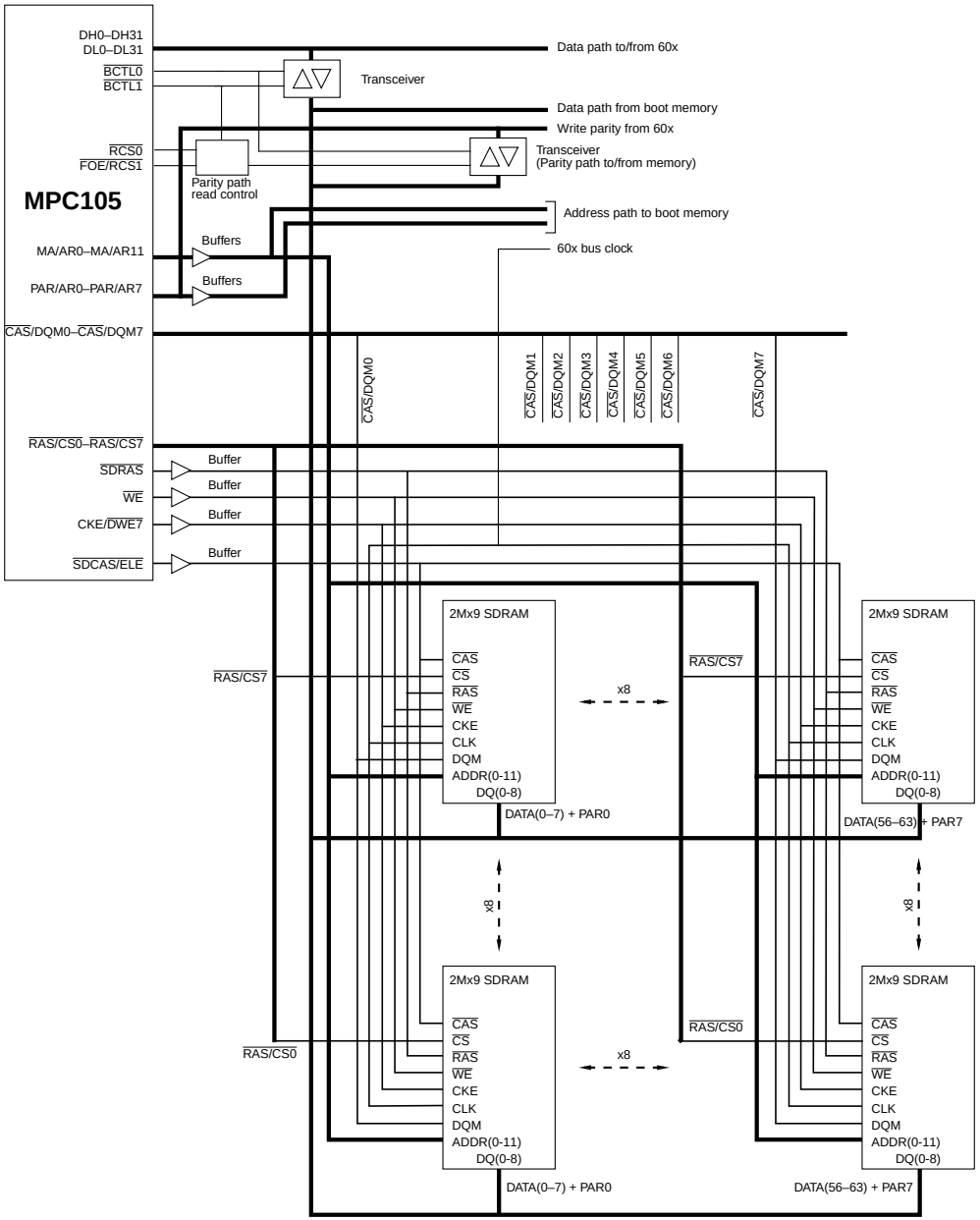


Figure 6-17. 128-Mbyte SDRAM System with Parity

6.4.1 Supported SDRAM Organizations

It is not necessary to use identical memory devices in each memory bank; individual memory banks may be of differing size. Although the MPC105 multiplexes the row and column address bits onto 12 memory address signals, individual SDRAM banks may be implemented with memory devices requiring fewer than 24 address bits. The MPC105 can be configured to provide 12, 11, 10, or 9 row address bits to a particular bank, and 12, 11, 10, or 9 column bits.

The data path to the SDRAM banks must be 64 bits wide (72 with parity). Table 6-8 summarizes some of the memory configurations supported by the MPC105.

Table 6-8. Memory Device Configurations Supported

Number of Devices in a Bank	Device Configuration	Row Bits x Column Bits	Bank Size	Maximum Memory (Using All 8 Banks)
16	4M x 4	12 x 12	32 Mbytes	256 Mbytes
8	2M x 8, 9	12 x 12	16 Mbytes	128 Mbytes
4	1M x 16, 18	12 x 12	8 Mbytes	64 Mbytes

By using a memory polling algorithm at power-on reset, system firmware configures the MPC105 to correctly map the size of each bank in memory (the memory boundary registers). The MPC105 uses its bank map to assert the appropriate $\overline{\text{RAS/CS}}$ signal for memory accesses according to the provided bank depths.

System software must also configure the MPC105 at power-on reset to appropriately multiplex the row and column address bits for each bank. This is done by writing the row address configuration into one of the memory control configuration registers. Address multiplexing will then occur according to the configuration settings.

6.4.2 SDRAM Power-On Initialization

At system reset, initialization software must set up the programmable parameters in the memory interface configuration registers (MICRs). These include the memory boundary registers, the memory banks enable register, and the memory control configuration registers (MCCRs). See Section 3.2.6, “Memory Interface Configuration Registers,” for more detailed descriptions of the MICRs and MCCRs.

The programmable parameters relevant to SDRAM-based systems are:

- Memory bank starting and ending addresses (memory boundary registers)
- Memory bank enables (memory bank enable register)
- SREN—self-refresh enable (MCCR1[18])
- RAMTYP—RAM type (MCCR1[17])

- PCKEN—memory interface parity checking/generation enable (MCCR1[16])
- Row address bit count for each bank (MCCR1[15–0])
- REFINT—refresh interval (MCCR2[15–2])
- BUF—buffer mode (MCCR2[1])
- RDTOACT—read-to-activate interval (MCCR3[31–28])
- REFREC—refresh recovery interval (MCCR3[27–24])
- RDLAT—data latency from read command (MCCR3[23–20])
- PRETOACT—precharge-to-activate interval (MCCR4[31–28])
- ACTOPRE—activate-to-precharge interval (MCCR4[27–24])
- WCBUF—memory write buffer type (MCCR4[21])
- RCBUF—memory read buffer type (MCCR4[20])
- SDMODE—SDRAM mode register (MCCR4[19–8])
- ACTORW—activate-to-read/write interval (MCCR4[7–4])
- WRTOACT—write-to-activate interval (MCCR4[3–0])

Once all the memory parameters are configured, then system software should set the MEMGO bit (MCCR1, bit 19) to enable the memory interface. (Note that 100 µs must elapse after the negation of $\overline{\text{HRST}}$ before the MEMGO bit can be set, so a delay loop in the initialization code may be necessary.)

The MPC105 then proceeds with the following initialization sequence:

1. Issues a precharge-all-banks command
2. Issues eight refresh commands
3. Issues a mode-set command to initialize the mode register

See Section 6.4.3, “JEDEC Standard SDRAM Interface Commands,” for detailed information about the SDRAM commands in the above sequence. When the sequence completes, the SDRAM array is ready for access.

6.4.3 JEDEC Standard SDRAM Interface Commands

The MPC105 performs all accesses to SDRAM by using JEDEC standard SDRAM interface commands. The SDRAM device samples the command and data inputs on the rising edge of the 60x bus clock. Data at the output of the SDRAM device must be sampled on the rising edge of the 60x bus clock.

The MPC105 provides the following SDRAM interface commands:

- Bank-activate—Latches the row address and initiates a memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored by issuing a precharge command before another bank-activate is issued.
- Precharge—Restores data from the sense amplifiers to the appropriate row. Also initializes the sense amplifiers in preparation for reading another row in the SDRAM array. A precharge command must be issued after a read or write, if the row address changes on the next access. Note that the MPC105 uses MA10 to distinguish the precharge-all-banks command. The SDRAMs must be compatible with this format.
- Read-with-autoprecharge—Latches the column address and transfers data from the selected sense amplifier to the output buffer as determined by the column address. During each succeeding clock, additional data is output without additional read commands. The amount of data transferred is determined by the burst size. At the end of the burst, a precharge is performed by the SDRAM without the MPC105 issuing a precharge command. Note that the MPC105 uses MA10 to distinguish the read-with autoprecharge command. The SDRAMs must be compatible with this format.
- Write-with-autoprecharge—Latches the column address and transfers data from the data signals to the selected sense amplifier as determined by the column address. During each succeeding clock, additional data is transferred to the sense amplifiers from the data signals without additional write commands. The amount of data transferred is determined by the burst size. At the end of the burst, a precharge is performed by the SDRAM without the MPC105 issuing a precharge command. Note that the MPC105 uses MA10 to distinguish the write-with-autoprecharge command. The SDRAMs must be compatible with this format.
- Refresh—Causes a row to be read in both memory banks (JEDEC SDRAM) as determined by the refresh row address counter (similar to CBR). The refresh row address counter is internal to the SDRAM device. After being read, a row is automatically rewritten into the memory array. Before execution of refresh, both memory banks must be in a precharged state.
- Mode-set—Allows setting of SDRAM options. The options are $\overline{\text{CAS}}$ latency, burst type, and burst length.

  $\overline{\text{CAS}}$ latency depends upon the SDRAM device used (some SDRAMs provide $\overline{\text{CAS}}$ latency of 1, 2, or 3, some provide $\overline{\text{CAS}}$ latency of 1, 2, 3, or 4, etc.).

  Burst type must be chosen according to the 60x cache wrap (sequential).

  Although some SDRAMs provide burst lengths of 1, 2, 4, 8, or a page, the MPC105 only supports a burst of four. Burst lengths of 1, 2, 8, and a page for SDRAMs are not supported by the MPC105.

  The mode register data ($\overline{\text{CAS}}$ latency, burst length, and burst type) is programmed into MCCR4[SDMODE] by initialization software at reset. After MCCR1[MEMGO] is set, the MPC105 then transfers the information in

MCCR4[SDMODE] to the SDRAM array by issuing the mode-set command. See Section 6.4.4.2, “SDRAM Mode-Set Command Timing,” for timing information.

- Self-refresh—Used when the SDRAM device is in standby for very long periods of time (corresponding with sleep or suspend mode on the MPC105). Internal refresh cycles are automatically generated by the SDRAM to keep the data in both memory banks refreshed. Before execution of this command, both memory banks must be in a precharged state.

Note that a precharge cycle begins immediately after termination of a read or write operation (read-with-autoprecharge, or write-with-autoprecharge), regardless of pending memory transactions from the PCI bus or 60x. The MPC105 can perform precharge cycles (including the precharge that automatically follows a read or write transaction) concurrent with snoop broadcasts for PCI transactions.

The SDRAM interface command encodings are summarized in Table 6-9.

Table 6-9. SDRAM Command Encodings

	Command						
	Bank-Activate	**Precharge-All- Banks**	**Read-with-Autoprecharge**	**Write-with-Autoprecharge**	**Refresh (CBR)**	**Mode-Set**	**Self-Refresh**
Previous CKE	High	High	High	High	High	High	High
Current CKE	x	x	x	x	High	x	Low
$\overline{\text{RAS}}$/$\overline{\text{CS}}n$	Low	Low	Low	Low	Low	Low	Low
$\overline{\text{SDRAS}}$	Low	Low	High	High	Low	Low	Low
$\overline{\text{SDCAS}}$/$\overline{\text{ELE}}$	High	High	Low	Low	Low	Low	Low
$\overline{\text{WE}}$	High	Low	High	Low	High	Low	High
$\overline{\text{CAS}}$/DQMn	x	x	Low	Low	x	x	x
MA11	Bank select	x	Bank select	Bank select	x	Opcode	x
MA10	Row	High	High	High	x	Opcode	x
MA9–MA0	Row	x	Column	Column	x	Opcode and mode	x

6.4.4 SDRAM Interface Timing

System software is responsible for optimal configuration of the SDRAM programmable timing parameters (RDLAT, RDTOACT, WRTOACT, and ACTORW) at reset. The actual values used by initialization software depend upon the specific SDRAM devices used in the system design.

Note that data latency is programmable for read operations (RDLAT). For write operations, the first beat of write data is supplied concurrent with the write command.

The following figures illustrate SDRAM timing for various types of accesses; Figure 6-18 shows a single-beat read operation, Figure 6-19 shows a burst-of-four read operation, Figure 6-20 shows a single-beat write operation, and Figure 6-21 shows a burst-of-four write operation.

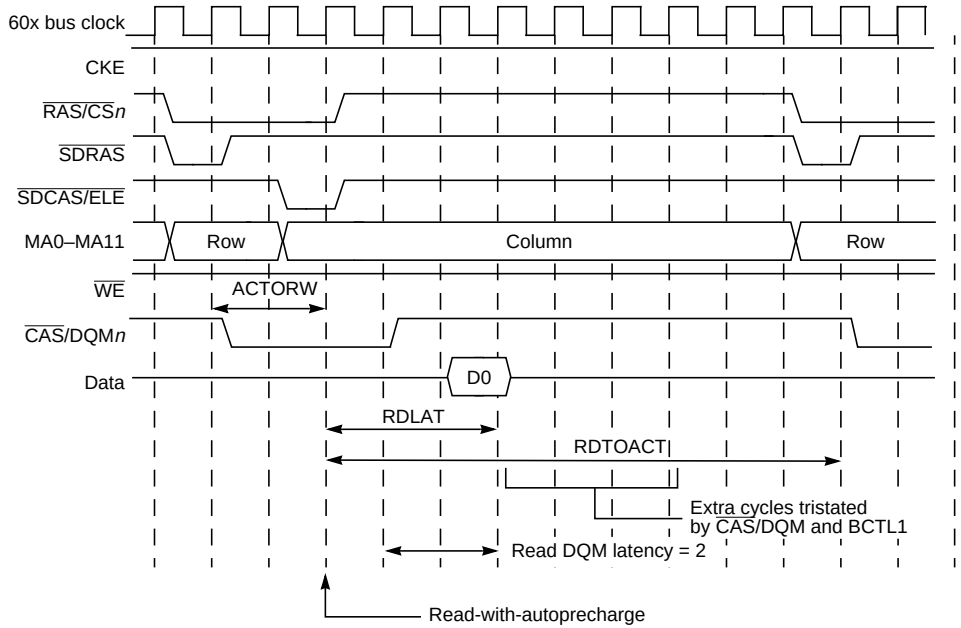


Figure 6-18. SDRAM Single-Beat Read Timing

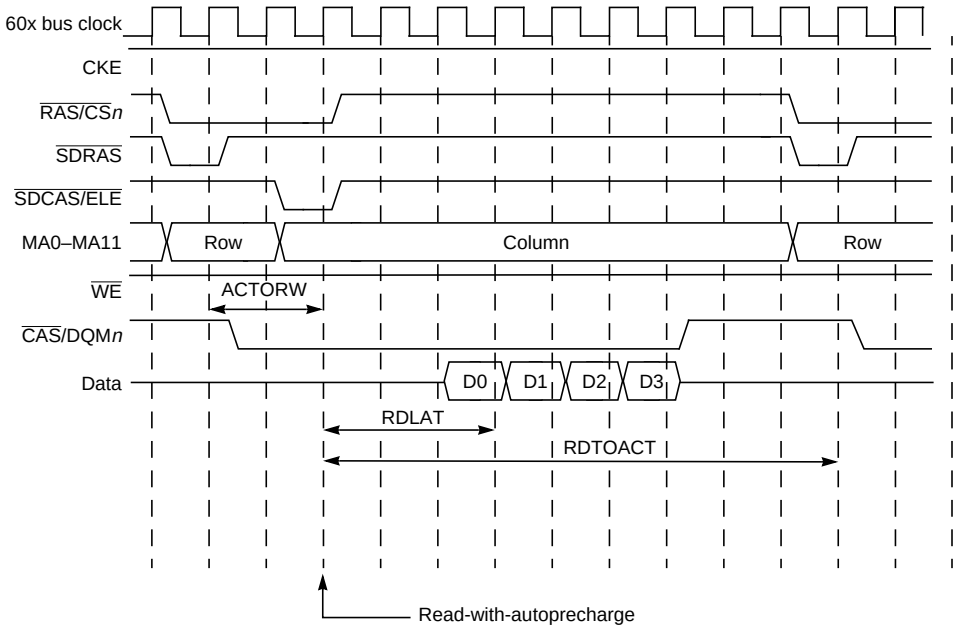


Figure 6-19. SDRAM Burst-of-Four Read Timing

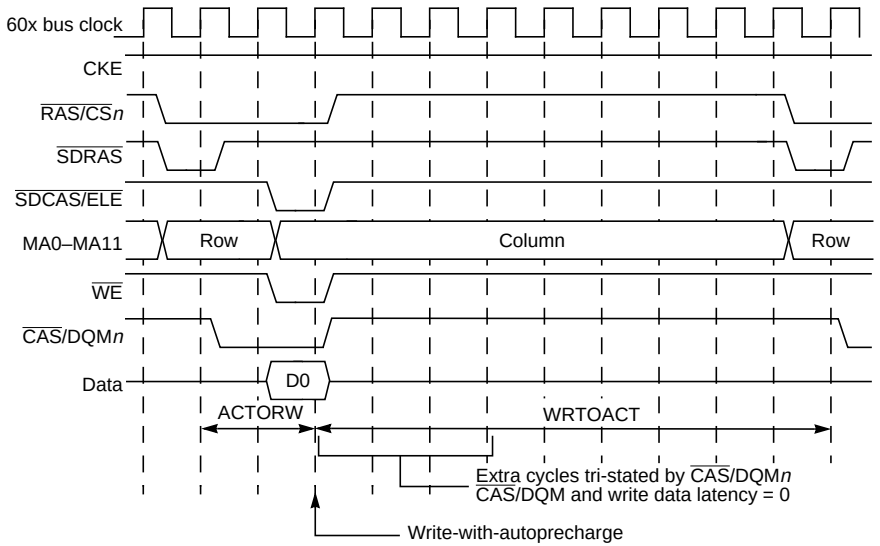


Figure 6-20. SDRAM Single-Beat Write Timing

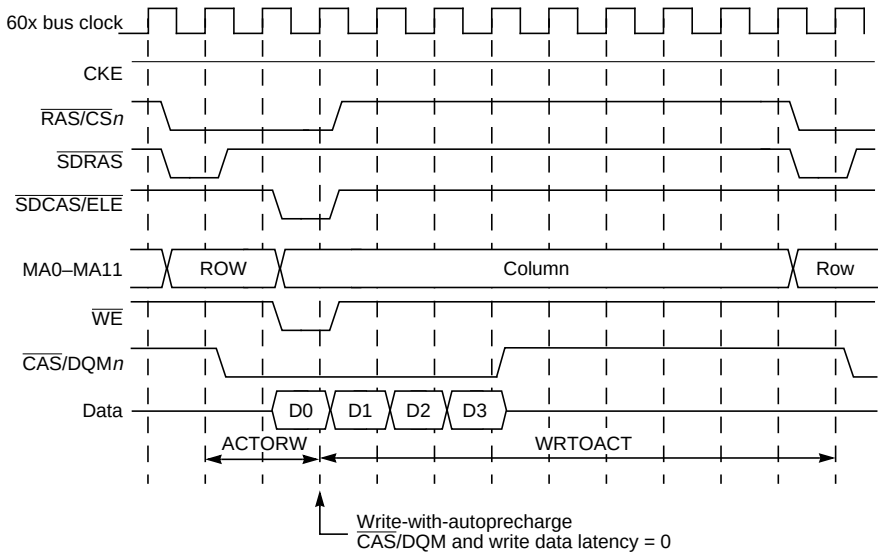


Figure 6-21. SDRAM Burst-of-Four Write Timing

6.4.4.1 SDRAM Burst and Single-Beat Transactions

The MPC105 runs a burst-of-four for every transaction (burst and single-beat). For single-beat read transactions, the extraneous data in the burst is ignored. For single-beat write transactions, the MPC105 protects nontargeted addresses by driving $\overline{\text{CAS}}$/DQM*n* high on the irrelevant cycles of the burst. The MPC105 only supports burst lengths of four for SDRAM-based systems. With an SDRAM-based system, burst lengths of eight are not supported.

For single-beat transactions, the bursts cannot be terminated early. That is, if the relevant data is in the first data phase, the subsequent data phases of the burst must run to completion even though the data is irrelevant.

6.4.4.2 SDRAM Mode-Set Command Timing

The MPC105 transfers the mode register data, ($\overline{\text{CAS}}$ latency, burst length, and burst type) stored in MCCR4[SDMODE] to the SDRAM array by issuing the mode-set command. The timing of the mode-set command is shown in Figure 6-22.

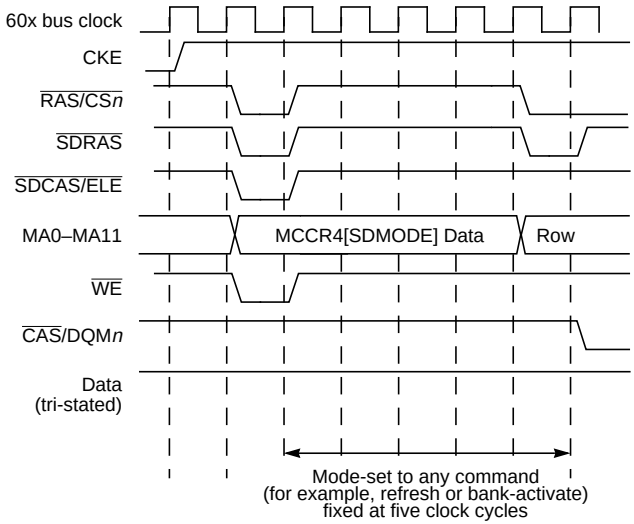


Figure 6-22. SDRAM Mode-Set Command Timing

6.4.5 SDRAM Refresh

The memory interface supplies CBR refreshes to SDRAM according to the interval specified in MCCR2[REFINT]. The value stored in REFINT represents the number of 60x bus clock cycles required between CBR refreshes. The value for REFINT depends on the specific SDRAM devices used and the operating frequency of the MPC105. This value should allow for a potential collision between memory accesses and refresh cycles. The period of the refresh interval must be greater than the access time to insure that read and write operations complete successfully.

If a burst read is in progress at the time a refresh operation is to be performed, the refresh waits for the read to complete. In the worst case, the refresh must wait the number of clock cycles required by the longest access time. The value stored to REFINT should be the number of clock cycles per row refresh reduced by the number of clock cycles required by the longest access time (to allow for potential collisions).

For example, given an SDRAM with 4096 rows and a cell refresh time of 64 ms, the per row refresh interval would be 64 ms ÷ 4096 rows = 15.6 µs. If the 60x bus clock is running at 66 MHz, the number of clock cycles per row refresh is 15.6 µs x 66 MHz = 1030 clock cycles. If the number of clock cycles for the longest access time is 24 clocks, value stored in REFINT would be 0b00_0011_1110_1110 (in decimal, 1030 – 24 = 1006).

6.4.5.1 SDRAM Refresh Timing

The CBR refresh timing for SDRAM is controlled by the programmable timing parameter MCCR3[REFREC]. REFREC represents the number of clock cycles from the refresh command until a bank-activate command is allowed. The AC specifications of the specific SDRAM device will provide a minimum refresh to activate interval.

The MPC105 implements bank staggering for CBR refreshes, as shown in Figure 6-23. This reduces instantaneous current consumption for memory refresh operations.

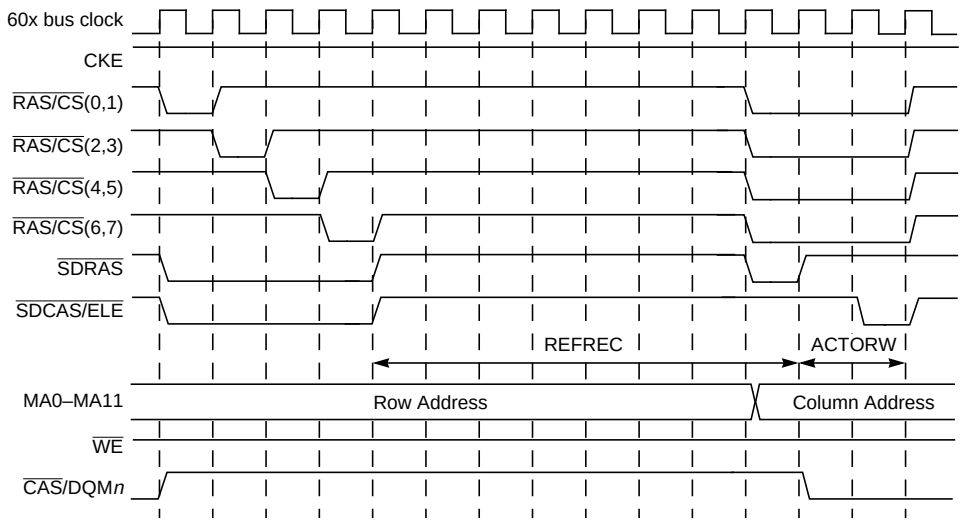


Figure 6-23. SDRAM Bank-Staggered CBR Refresh Timing

6.4.5.2 SDRAM Refresh and Power Saving Modes

The MPC105 memory interface provides for doze, nap, sleep, and suspend power saving modes defined in Appendix A, “Power Management.”

In doze and nap power saving modes and in full-on mode, the MPC105 supplies the normal CBR refresh to SDRAM. In sleep mode, the MPC105 can be configured to take advantage of self-refreshing DRAMs, provide normal refresh to DRAM, or provide no refresh support. In suspend mode, the MPC105 can be configured to take advantage of self-refreshing DRAMs, or provide no refresh support. Normal and real-time clock (RTC) refresh are not available for SDRAM in suspend mode—all JEDEC-compliant SDRAMs allow self-refresh which consumes less power than normal or RTC refresh. Table 6-7 summarizes the refresh types available in each of the power saving modes and the relevant configuration parameters.

Table 6-10. SDRAM Power Saving Modes Refresh Configuration

Power Saving Mode	Refresh Type	$\overline{\text{SUSPEND}}$ Signal	Power Management Control Register (PMCR)					MCCR1 [SREN]
			[PM]	[DOZE]	[NAP]	[SLEEP]	[LP_REF_EN]	
Doze	Normal	Negated (High)	1	1	0	0	—	—
Nap	Normal	Negated (High)	1	—	1	0	—	—
Sleep	Self	Negated (High)	1	—	—	1	1	1
	Normal	Negated (High)	1	—	—	1	1	0
	None	Negated (High)	1	—	—	1	0	—
Suspend[1]	Self	Asserted (Low)	1	—	—	0	1	1
	None	Asserted (Low)	1	—	—	0	0	—
	Note: [1]Normal and RTC refresh are not available in suspend mode.							

Note that in the absence of refresh support, system software must preserve DRAM data (such as by copying the data to disk) before entering the power saving state.

The entry timing for self-refreshing SDRAMs is shown in Figure 6-24. The exit timing for self-refreshing SDRAMs is shown in Figure 6-25.

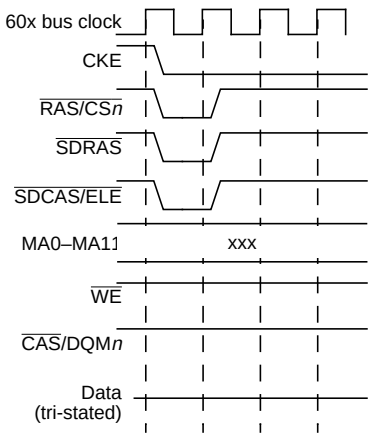


Figure 6-24. SDRAM Self-Refresh Entry Timing

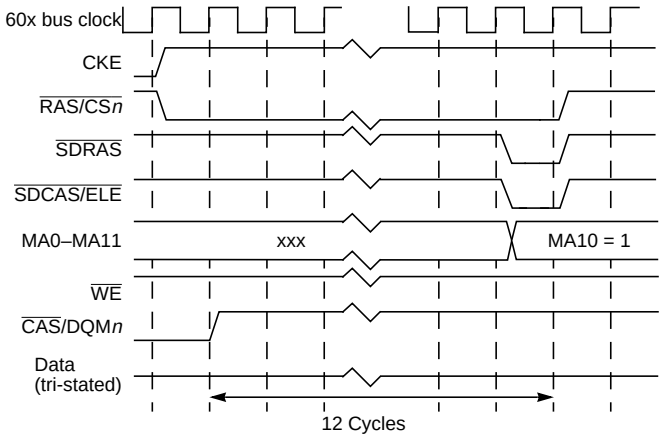


Figure 6-25. SDRAM Self-Refresh Exit Timing

6.5 ROM Interface Operation

The MPC105 can be configured to support ROM devices located on the 60x processor/memory bus or on the PCI bus. The $\overline{\text{RCS0}}$ signal is sampled at reset to determine the location of ROM. See Section 2.2.7, “Configuration Signals,” for more information. The ROM interface of the MPC105 controls ROM located on the 60x processor memory bus.

The MPC105 provides 20 ROM address bits and two ROM chip select outputs for access to ROM. Bits 0–7 of the ROM address (AR0–AR7) are multiplexed with the parity signals (PAR0–PAR7) of the RAM interface. Bits 8–19 of the ROM address (AR8–AR19) are multiplexed with the RAM address (MA0–MA11) signals. AR0 is the msb and AR19 is the lsb of the ROM address. The two ROM chip select outputs ($\overline{\text{RSC0}}$ and $\overline{\text{RCS1}}$) function as bank selects. Figure 6-26 shows an example of a 16-Mbyte ROM system.

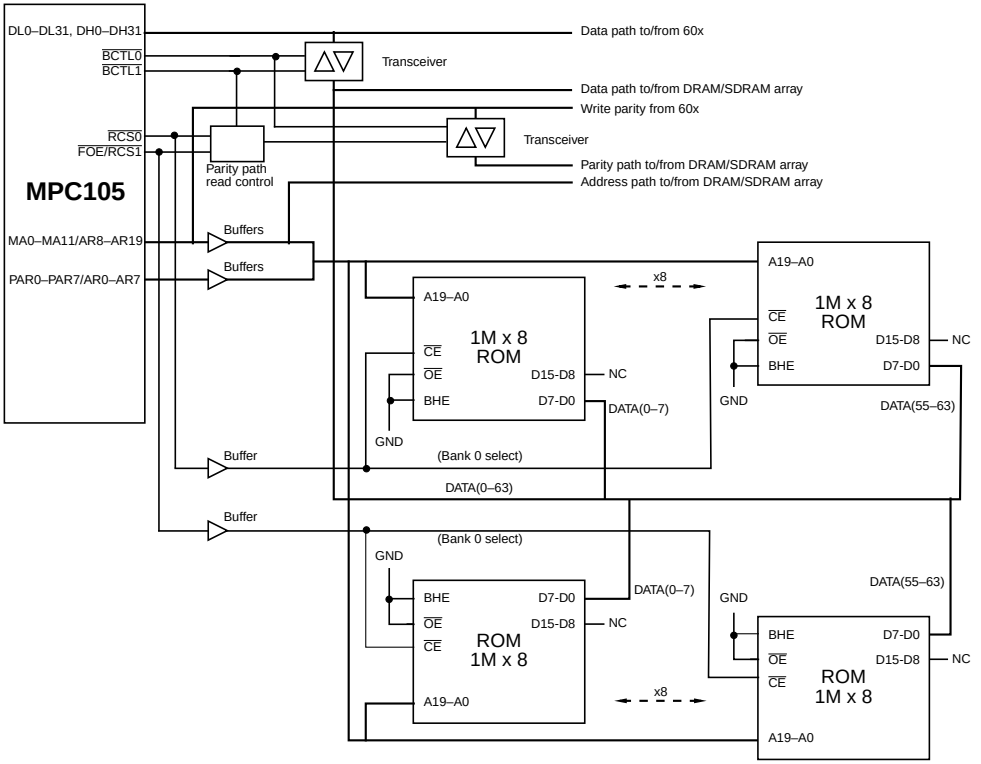


Figure 6-26. 16-Mbyte ROM System

The MPC105 can access up to 16 Mbytes of ROM in 64-bit data bus implementations (8 Mbytes per bank). In this configuration, addresses 0xFF00_0000–0xFF7F_FFFF activate $\overline{\text{RCS0}}$ and addresses 0xFF80_0000–0xFFFF_FFFF activate $\overline{\text{RCS1}}$. Implementations that require less than 16 Mbytes may allocate the required ROM to one or both banks.

For example, an implementation that requires only 4 Mbytes of ROM could locate the ROM entirely within the range for $\overline{\text{RCS1}}$ at addresses 0xFFC0_0000–0xFFFF_FFFF. Alternately, the ROM could be split across both banks with 2 Mbytes in the $\overline{\text{RCS0}}$ range at 0xFF60_0000–0xFF7F_FFFF, and 2 Mbytes in the $\overline{\text{RCS1}}$ range at 0xFFE0_0000–0xFFFF_FFFF.

The MPC105 can access up to 8 Mbytes of ROM in 32-bit data bus implementations (4 Mbytes per bank). In this configuration, addresses 0xFF80_0000–0xFFBF_FFFF activate $\overline{\text{RCS0}}$ and addresses 0xFFC0_0000–0xFFFF_FFFF activate RSC1. Implementations that require less than 8 Mbytes may allocate the required ROM to one or both banks.

6.5.1 ROM Interface Timing

The ROM interface of the MPC105 supports burst and nonburst ROMs. The MPC105 provides programmable access timing for the ROM interface. The programmable timing parameters for the ROM interface are MCCR1[ROMNAL], MCCR1[ROMFAL], MCCR1[BURST], and MCCR2[WMODE]. See Section 3.2.6.3, "Memory Control Configuration Registers," for more information.

MCCR1[ROMFAL] controls the latency for nonburst ROMs. For burst ROMs, ROMFAL controls the latency for the first data beat only. The actual latency cycle count is three cycles more than the value specified in ROMFAL. For example, when ROMFAL = 0b0_0000, the latency is three clock cycles; when ROMFAL = 0b0_0001, the latency is four clock cycles; when ROMFAL = 0b0_0010, the latency is five clock cycles; and so on.

MCCR1[ROMNAL] controls the latency for burst ROMs for the burst data beats following the first data beat. The actual latency cycle count is two cycles more than the value specified in ROMNAL. For example, when ROMNAL = 0b0000, the latency is two clock cycles; when ROMNAL = 0b0001, the latency is three clock cycles; when ROMNAL = 0b0010, the latency is four clock cycles; and so on.

MCCR1[BURST] controls whether the MPC105 uses burst (BURST = 1) or nonburst (BURST = 0) access timing. MCCR2[WMODE] controls the number of data beats in a burst to ROM.

ROMFAL and ROMNAL are set to their maximum value at reset in order to accommodate initial boot code fetches. BURST is cleared at reset to indicate nonburst ROM timing. WMODE is set at reset to indicate burst-of-four burst ROM access.

Figure 6-27 and Figure 6-28 illustrate the ROM interface timing for nonburst and burst ROM configurations.

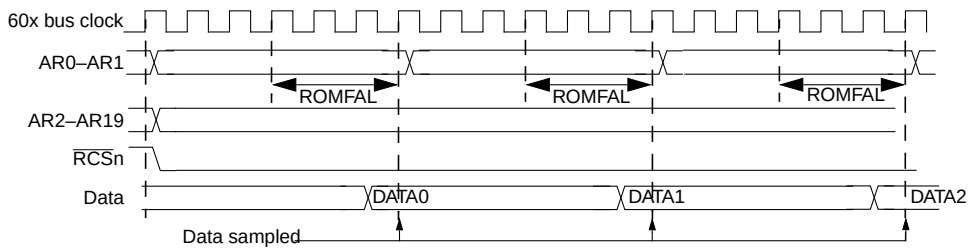


Figure 6-27. ROM Nonburst Read Timing

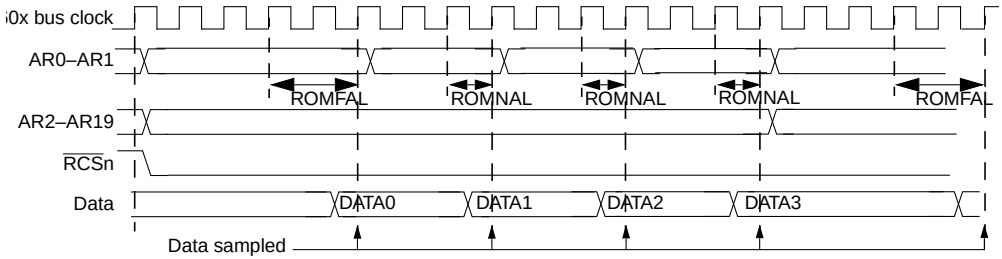


Figure 6-28. ROM Burst Read Timing

6.6 Flash ROM Interface Operation

The MPC105 provides 20 address bits (AR0–AR19), one chip select output ($\overline{\text{RCS0}}$), one output enable output ($\overline{\text{FOE}}$/$\overline{\text{RCS1}}$), and one write enable output ($\overline{\text{WE}}$) for read and write access to Flash memory. Bits 0–7 of the Flash ROM address (AR0–AR7) are multiplexed with the parity signals (PAR0–PAR7) of the RAM interface. Bits 8–19 of the Flash ROM address (AR8–AR19) are multiplexed with the RAM address (MA0–MA11) signals. AR0 is the msb and AR19 is the lsb of the Flash ROM address. Figure 6-29 shows an example of a 1-Mbyte Flash ROM system.

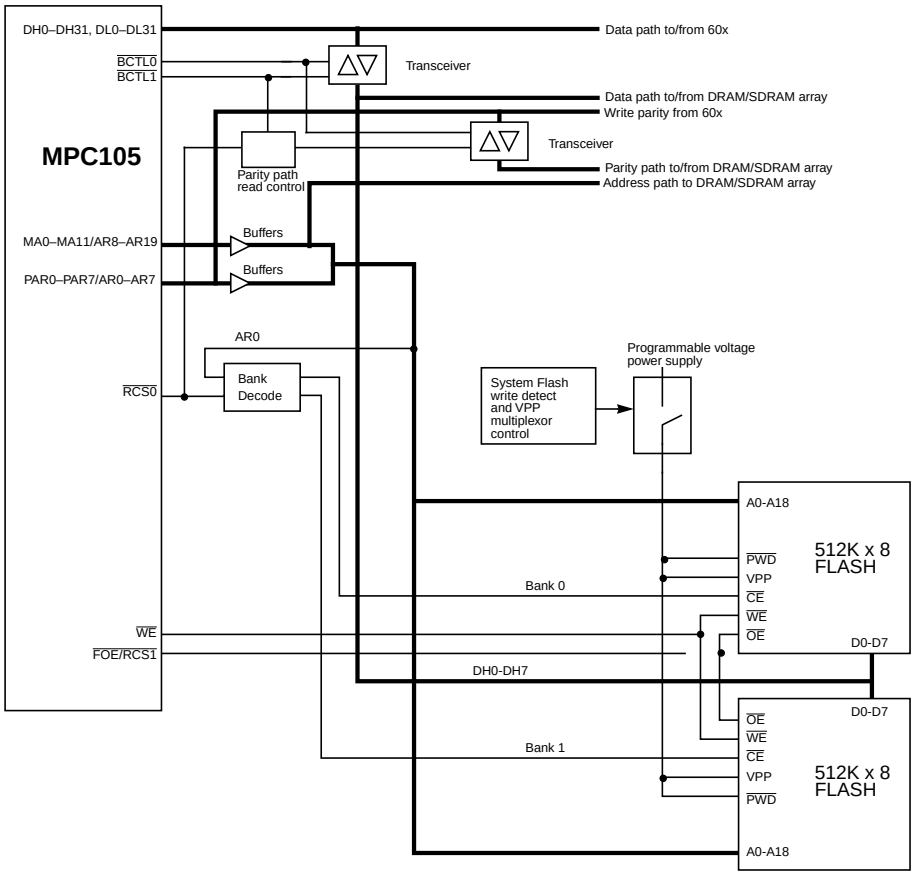


Figure 6-29. One-Mbyte Flash ROM System

The MPC105 supports x8 organizations of Flash ROM only, using the DH0–DH7 signals for data flow into and out of the Flash ROM device(s). The MPC105 performs byte lane alignment for single-byte reads from Flash ROM. This configuration provides the access to 1 Mbyte of Flash ROM space. Accesses in the address range 0xFFF0_0000–0xFFFF_FFFF activate $\overline{\text{RCS0}}$ for Flash ROM device selection.

Although the Flash ROM interface only supports 8-bit devices, the MPC105 gathers up to eight bytes from the Flash ROM for read operations. The data bytes are gathered and aligned within the MPC105, then forwarded to the 60x processor. Note that data in Flash ROM may be cached, but only in write-through mode.

6.6.1 Flash ROM Interface Timing

The MPC105 provides programmable latency for accesses to Flash ROM so that systems of various clock frequencies may properly interface to the Flash ROM devices. The programmable timing parameter MCCR1[ROMFAL] controls the access latency to Flash ROM. The actual latency cycle count is two cycles more than the value specified in ROMFAL. For example, when ROMFAL = 0b0_0000, the latency is two clock cycles; when ROMFAL = 0b0_0001, the latency is three clock cycles; when ROMFAL = 0b0_0010, the latency is four clock cycles; and so on. ROMFAL is set to its maximum value at reset in order to accommodate initial boot code fetches.

The following figures illustrate the Flash ROM interface timing for various read accesses. Figure 6-30 shows a single-byte read access. Figure 6-31 shows a half-word read access. Word and double-word accesses require using the burst access timing shown in Figure 6-32.

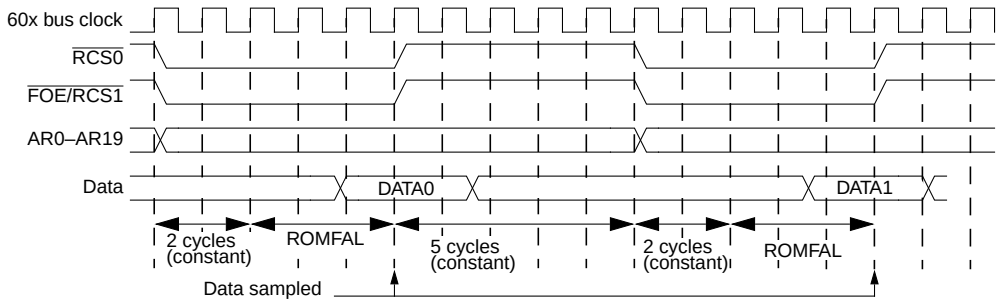


Figure 6-30. Flash ROM Single-Byte Read Timing

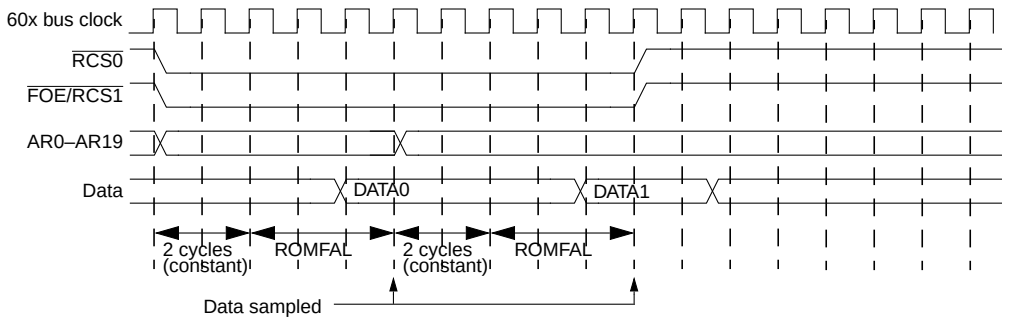


Figure 6-31. Flash ROM Half-Word Read Timing

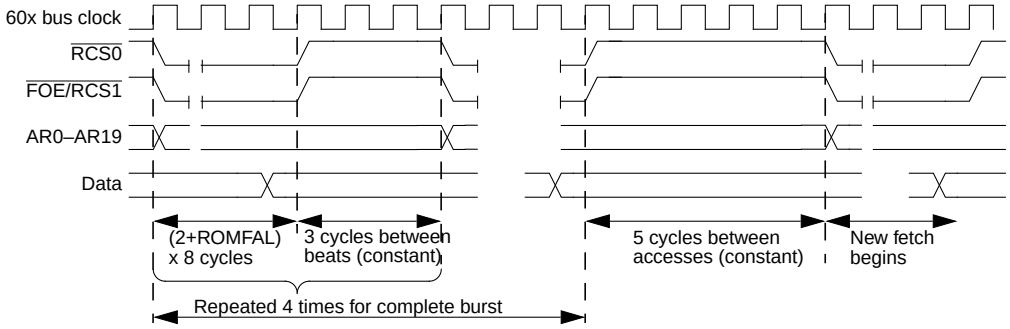


Figure 6-32. Flash ROM Burst Read Timing

6.6.2 Writing to Flash ROM

The MPC105 accommodates only single-byte writes to Flash memory. Software must partition larger data into single-byte write operations. If an attempt is made to write data larger than one byte to Flash memory, the MPC105 will assert $\overline{\text{TEA}}$ (provided $\overline{\text{TEA}}$ is enabled in PICR1).

PICR1[FLASH_WR_EN] must be set when performing write operations to Flash memory. FLASH_WR_EN controls whether write operations to Flash memory are allowed. System logic (external to the MPC105) is responsible for multiplexing high voltage to the Flash memory as required for write operations.

The MPC105 provides programmable latency for write operations to Flash memory so that systems of various clock frequencies may properly interface to the Flash ROM devices. The programmable timing parameter MCCR1[ROMFAL] controls the read and write access latency to Flash memory. The actual latency cycle count is two cycles more than the value specified in ROMFAL. For example, when ROMFAL = 0b0000, the latency is two clock cycles; when ROMFAL = 0b0001, the latency is three clock cycles; when ROMFAL = 0b0010, the latency is four clock cycles; and so on.

MCCR1[ROMNAL] controls the Flash memory write recovery time (that is, the number of cycles between write pulse assertions). The actual recovery cycle count is three cycles more than the value specified in ROMNAL. For example, when ROMNAL = 0b0000, the write recovery time is three clock cycles; when ROMNAL = 0b0001, the write recovery time is four clock cycles; when ROMNAL = 0b0010, the write recovery time is five clock cycles; and so on.

ROMFAL and ROMNAL are set to their maximum value at reset in order to accommodate initial boot code fetches. FLASH_WR_EN is cleared at reset to disable write operations to Flash memory.

Figure 6-33 illustrates the write access timing of the Flash ROM interface.

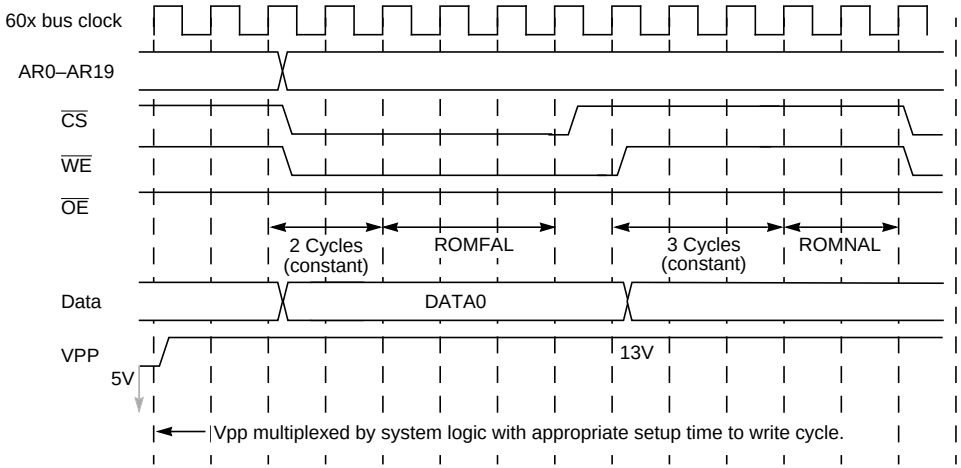


Figure 6-33. Flash Memory Write Timing

Chapter 7
PCI Bus Interface

One of the primary functions of the MPC105 is to serve as a bridge between the 60x processor bus (the host bus) and the PCI bus. The MPC105's PCI interface is compliant with the *PCI Local Bus Specification, Revision 2.0* and follows the guidelines in the *PCI System Design Guide, Revision 1.0*, for host bridge architecture.

It is well beyond the scope of this manual to document the intricacies of the PCI bus. This chapter provides a rudimentary description of PCI bus operations. The specific emphasis is directed at how the MPC105 implements the PCI bus. It is strongly advised that anyone designing a system incorporating PCI devices should refer to the *PCI Local Bus Specification, Revision 2.0* and the *PCI System Design Guide, Revision 1.0,* for a thorough description of the PCI local bus.

NOTE

Much of the available PCI literature refers to a 16-bit quantity as a "word," and a 32-bit quantity as a "double word." As this is inconsistent with the terminology in this manual, the terms word and double word will not be used in this chapter. Instead, the number of bits or bytes will indicate the exact quantity.

7.1 PCI Interface Overview

The PCI interface connects the processor and memory buses to the PCI bus, to which I/O components are connected. The PCI bus uses a 32-bit multiplexed, address/data bus, plus various control and error signals. The MPC105 supports a range of bus speeds in full-on mode. (In the sleep and suspend power saving modes, the PCI bus clock may be turned off.) For complete timing information, see the MPC105 Hardware Specifications. The PCI interface supports address and data parity with error checking and reporting.

The PCI interface of the MPC105 functions as both a master (initiator) and target device. Internally, the PCI interface of the MPC105 is controlled by two state machines (one for master and one for target) running independently of each other. This allows the MPC105 to run two separate transactions simultaneously. For example, if the master is trying to run a burst-write to a PCI device, it may get disconnected before finishing the transaction. If another PCI device is granted the PCI bus and requests a burst-read from system memory,

the MPC105 can accept the burst-read transfer. When the MPC105 is granted the PCI bus, the burst-write transaction continues.

As a master, the MPC105 supports read and write operations to the PCI memory space, the PCI I/O space, and the PCI 256-byte configuration space. The MPC105 also supports PCI special cycles and interrupt acknowledge cycles. As a target, the MPC105 supports read and write operations to system memory.

Buffers are provided for operations between the PCI bus and the 60x processor or system memory. Processor read and write operations each have a 32-byte buffer, and memory operations have one 32-byte read buffer and two 32-byte write buffers. See Section 8.1, “Internal Control,” for more information.

The interface can be programmed for either little-endian or big-endian formatted data, and provides the data swapping, byte enable swapping, and address translation in hardware. See Appendix B, “Bit and Byte Ordering,” for more information on the bi-endian features of the MPC105.

7.1.1 The MPC105 as a PCI Master

Upon detecting a 60x-to-PCI transaction, the MPC105 requests the use of the PCI bus. For write operations to the PCI bus, the MPC105 requests the PCI bus when the 60x completes the write operation on the 60x processor bus. For read operations from the PCI bus, the MPC105 requests the PCI bus when it decodes that the access is for PCI address space.

Once granted, the MPC105 drives the 32-bit PCI address (AD31–AD0) and the 4-bit bus command ($\overline{\text{C/BE3}}$–$\overline{\text{C/BE0}}$) signals. The master interface supports reads and writes of up to 32-bytes without inserting master-initiated wait states.

The master part of the interface can initiate master-abort cycles, recognizes target-abort, target-retry, and target-disconnect cycles, and supports various device selection timings. The master interface does not run fast back-to-back or interlocked accesses.

7.1.2 The MPC105 as a PCI Target

As a target, upon detection of a PCI address phase the MPC105 decodes the address and bus command to determine if the transaction is for system memory. If the transaction is destined for system memory, the target interface latches the address and decode the command and forwards them to an internal control unit. On writes to system memory, data is forwarded along with the byte enables to the internal control unit. On reads, four bytes of data are provided to the PCI bus and the byte enables determine which byte lanes contain valid data.

The target interface of the MPC105 can issue target-abort, target-retry, and target-disconnect cycles. Also, The target interface supports fast back-to-back transactions, and interlocked accesses using the PCI lock protocol.

The target interface uses the fastest device selection timing and can accept burst writes to system memory of up to 32 bytes with no wait states. Burst reads from system memory are also accepted with wait states inserted depending upon the device timing of system memory. The target interface disconnects when a transaction reaches the end of a cache line (32-bytes) so a new address can be provided to snoop the 60x bus.

7.2 PCI Bus Arbitration

The PCI arbitration approach is access-based. Bus masters must arbitrate for each access performed on the bus. PCI uses a central arbitration scheme where each master has its own unique request ($\overline{\text{REQ}}$) and grant ($\overline{\text{GNT}}$) signal. A simple request-grant handshake is used to gain access to the bus. Arbitration for the bus occurs during the previous access so that no PCI bus cycles are consumed due to arbitration (except when the bus is idle).

The MPC105 does not function as the central PCI bus arbiter. It is the responsibility of the system designer to provide for PCI bus arbitration. Regardless of the implementation, the arbitration algorithm must be defined to establish a basis for a worst-case latency guarantee. Latency guidelines are provided in the *PCI Local Bus Specification*. There are devices available that integrate the central arbiter, DMA controller, interrupt controller, and PCI-to-ISA bridge functions into a single device.

7.2.1 Exclusive Access

PCI provides an exclusive access mechanism referred to as a resource lock. The mechanism locks only the selected PCI resource but allows other nonexclusive accesses to proceed. A full description of exclusive access is contained in the *PCI Local Bus Specification*.

The $\overline{\text{LOCK}}$ signal indicates an exclusive access is underway. The assertion of $\overline{\text{GNT}}$ does not guarantee control of the $\overline{\text{LOCK}}$ signal. Control of $\overline{\text{LOCK}}$ is obtained in conjunction with $\overline{\text{GNT}}$. When using resource lock, agents performing nonexclusive accesses are free to proceed even while another master retains ownership of $\overline{\text{LOCK}}$.

The MPC105 implements the $\overline{\text{LOCK}}$ signal to guarantee complete access exclusion in system memory. The 60x processor must honor the resource lock. If a master on the PCI bus asserts $\overline{\text{LOCK}}$ for a transaction to system memory, the MPC105 flushes all internal PCI-to-memory write buffers, performs a snoop transaction on the 60x bus, prevents masters on the 60x bus from starting any new transactions (except for a snoop copy-back), and then completes the locked transaction. The 60x bus remains locked as long as the MPC105 is locked and the $\overline{\text{LOCK}}$ signal is asserted.

7.3 PCI Bus Protocol

This section provides a general description of the PCI bus protocol by presenting the basic transfer control mechanisms. Specific PCI bus transactions are described in Section 7.4, “PCI Bus Transactions.” Refer to Figure 7-1 and Figure 7-2 for examples of the transfer control mechanisms described in this section.

All signals are sampled on the rising edge of the PCI bus clock (SYSCLK). Each signal has a setup and hold aperture with respect to the rising clock edge, in which transitions are not allowed. Outside this aperture, signal values or transitions have no significance.

7.3.1 Basic Transfer Control

The basic PCI bus transfer mechanism is a burst. A burst is composed of an address phase followed by one or more data phases. Fundamentally, all PCI data transfers are controlled by three signals—$\overline{\text{FRAME}}$, $\overline{\text{IRDY}}$, and $\overline{\text{TRDY}}$. A master asserts $\overline{\text{FRAME}}$ to indicate the beginning of a PCI bus transaction and negates $\overline{\text{FRAME}}$ to indicate the end of a PCI bus transaction. A master negates $\overline{\text{IRDY}}$ (initiator ready) to force wait cycles. A target negates $\overline{\text{TRDY}}$ (target ready) to force wait cycles.

The PCI bus is considered idle when both $\overline{\text{FRAME}}$ and $\overline{\text{IRDY}}$ are negated. The first clock cycle in which $\overline{\text{FRAME}}$ is asserted indicates the beginning of the address phase. The address and bus command code are transferred in that first cycle. The next cycle begins the first of one or more data phases. Data is transferred between master and target in each cycle that both $\overline{\text{IRDY}}$ and $\overline{\text{TRDY}}$ are asserted. Wait cycles may be inserted in a data phase by the master (by negating $\overline{\text{IRDY}}$) or by the target (by negating $\overline{\text{TRDY}}$).

Once a master has asserted $\overline{\text{IRDY}}$, it cannot change $\overline{\text{IRDY}}$ or $\overline{\text{FRAME}}$ until the current data phase completes regardless of the state of $\overline{\text{TRDY}}$. Once a target has asserted $\overline{\text{TRDY}}$ or $\overline{\text{STOP}}$, it cannot change $\overline{\text{DEVSEL}}$, $\overline{\text{TRDY}}$ or $\overline{\text{STOP}}$ until the current data phase completes. In simpler terms, once a master or target has committed to the data transfer, it cannot change its mind.

When the master intends to complete only one more data transfer (which could be immediately after the address phase), $\overline{\text{FRAME}}$ is negated and $\overline{\text{IRDY}}$ is asserted (or kept asserted) indicating the master is ready. After the target indicates the final data transfer (by asserting $\overline{\text{TRDY}}$), the PCI bus may return to the idle state (both $\overline{\text{FRAME}}$ and $\overline{\text{IRDY}}$ are negated) unless a fast back-to-back transaction is in progress. In the case of a fast back-to-back transaction, an address phase immediately follows the last data phase.

7.3.2 PCI Bus Commands

A PCI bus command is encoded in the $\overline{\text{C/BE3}}$–$\overline{\text{C/BE0}}$ signals during the address phase of a PCI transaction. The bus command indicates to the target the type of transaction the master is requesting. Table 7-1 describes the PCI bus commands as implemented by the MPC105.

Table 7-1. PCI Bus Commands

$\overline{C/BE3}$–$\overline{C/BE0}$	PCI Bus Command	MPC105 Supports as a Master Device	MPC105 Supports as a Target Device	Definition
0000	Interrupt-acknowledge	Yes	No	The interrupt-acknowledge command is a read (implicitly addressing the system interrupt controller). Only one device on the PCI bus should respond to the interrupt acknowledge command. Other devices ignore the interrupt-acknowledge command. See Section 7.4.6.1, “Interrupt Acknowledge,” for more information.
0001	Special-cycle	Yes	No	The special-cycle command provides a mechanism to broadcast select messages to all devices on the PCI bus. See Section 7.4.6.2, “Special Cycle,” for more information.
0010	I/O-read	Yes	No	The I/O-read command accesses agents mapped into the PCI I/O space.
0011	I/O-write	Yes	No	The I/O-write command accesses agents mapped into the PCI I/O space.
0100	Reserved[1]	No	No	—
0101	Reserved[1]	No	No	—
0110	Memory-read	Yes	Yes	The memory-read command accesses either system memory, or agents mapped into PCI memory space, depending on the address. When a PCI master issues a memory-read command to system memory, the MPC105 (the target) fetches from the requested address to the end of the cache line (32 bytes) from system memory, even though all of the data may not be requested by (or sent to) the master.
0111	Memory-write	Yes	Yes	The memory-write command accesses either system memory, or agents mapped into PCI memory space, depending on the address.
1000	Reserved[1]	No	No	—
1001	Reserved[1]	No	No	—
1010	Configuration-read	Yes	No	The configuration-read command accesses the 256-byte configuration space of a PCI agent. A specific agent is selected when its IDSEL signal is asserted during the address phase. See Section 7.4.5, “Configuration Cycles,” for more detail of PCI configuration cycles.

Table 7-1. PCI Bus Commands (Continued)

$\overline{C/BE3}$–$\overline{C/BE0}$	PCI Bus Command	MPC105 Supports as a Master Device	MPC105 Supports as a Target Device	Definition
1011	Configuration-write	Yes	No	The configuration-write command accesses the 256-byte configuration space of a PCI agent. A specific agent is selected when its IDSEL signal is asserted during the address phase. See Section 7.4.5.2, “Accessing the PCI Configuration Space,” for more detail of configuration accesses.
1100	Memory-read-multiple	No	Yes	The memory-read-multiple command functions the same as the memory-read command on the MPC105. If prefetching is desired, speculative PCI reads should be enabled. See Section 8.1.3.2.1, “Speculative PCI Reads from System Memory,” for more information.
1101	Dual-address-cycle	No	No	The dual-address-cycle command is used to transfer a 64-bit address (in two 32-bit address cycles) to 64-bit addressable devices. The MPC105 does not respond to this command.
1110	Memory-read-line	No	Yes	The memory-read-line command functions the same as the memory-read command on the MPC105.
1111	Memory-write-and-invalidate	No	Yes	The memory-write-and-invalidate command functions the same as the memory-write command, but invalidates any cache line hit that is marked dirty.

Note: [1]Reserved command encodings are reserved for future use. The MPC105 does not respond to these commands.

7.3.3 Addressing

PCI defines three physical address spaces—PCI memory space, PCI I/O space, and PCI configuration space. Access to the PCI memory and I/O space is straightforward, although one must take into account the MPC105 memory map (A or B) being used. The memory maps are described in Section 3.1, “Address Maps.” Access to the PCI configuration space is described in Section 7.4.5, “Configuration Cycles.”

Address decoding on the PCI bus is performed by every device for every PCI transaction. Each agent is responsible for decoding its own address. PCI supports two types of address decoding—positive decoding and subtractive decoding. For positive decoding, each device is looking for accesses in the address range that the device has been assigned. For subtractive decoding, one device on the bus is looking for accesses that no other device has claimed. See Section 7.3.4, “Device Selection,” for information about claiming transactions.

The information contained in the two low-order address bits (AD1–AD0) varies by the address space (memory, I/O, or configuration). Regardless of the encoding scheme, the two low-order address bits are always included in parity calculations.

For memory accesses, PCI defines two types of ordering controlled by the two low-order bits of the address—linear burst ordering (AD1–AD0 = 0b00) and cache line toggle (AD1–AD0 = 0b01). Cache line toggle is an optional feature of PCI and is not implemented on the MPC105. Since the MPC105 implements linear burst ordering only, AD1–AD0 must be 0b00 during the address phase of any memory access. As a target, the MPC105 executes a target disconnect after the first data phase completes if AD1–AD0 ≠ 0b00 during the address phase of a system memory access. As a master, the MPC105 always encodes AD1–AD0 = 0b00 for PCI memory space accesses.

The memory address is encoded/decoded using AD31–AD02. Thereafter, the address is incremented by 4 bytes after each data phase completes until the transaction is terminated or completed (a 4-byte data width per data phase is implied). Note that the two low-order bits of the address bus are still included in all parity calculations.

For PCI I/O accesses, all 32 address/data signals are used to provide an address with granularity of a single byte. The AD1–AD0 signals are used for the generation of $\overline{\text{DEVSEL}}$ and indicate the least significant valid byte involved in the transfer.

Once a target has claimed an I/O access, it must determine if it can complete the entire access as indicated by the byte enable signals. If all the selected bytes are not in the address range of the target, the entire access cannot complete. In this case, the target does not transfer any data, and terminates the transaction with a target-abort.

PCI supports two types of configuration access, which use different formats for the AD31–AD0 signals during the address phase. The two low-order bits of the address indicate the format used for the configuration address phase—Type 0 (AD1–AD0 = 0b00) or Type 1 (AD1–AD0 = 0b01). Both address formats identify a specific device and a specific configuration register for that device. See Section 7.4.5, “Configuration Cycles,” for descriptions of the two formats.

7.3.4 Device Selection

The $\overline{\text{DEVSEL}}$ signal is driven by the target of the current transaction. $\overline{\text{DEVSEL}}$ indicates to the other devices on the PCI bus that the target has decoded the address and claimed the transaction. $\overline{\text{DEVSEL}}$ may be driven one, two, or three clocks (fast, medium, or slow device select timing) following the address phase. Device select timing is encoded into the device’s configuration space status register. If no agent asserts $\overline{\text{DEVSEL}}$ within three clocks of $\overline{\text{FRAME}}$, the agent responsible for subtractive decoding may claim the transaction by asserting $\overline{\text{DEVSEL}}$.

A target must assert DEVSEL (claim the transaction) before or coincident with any other target response (assert its TRDY, STOP, or data signals). In all cases except target-abort, once a target asserts DEVSEL, it must not negate DEVSEL until FRAME is negated (with IRDY asserted) and the last data phase has completed. With normal termination, negation of DEVSEL coincides with the negation of TRDY or STOP.

If the first access maps into a target's address range, that target asserts DEVSEL to claim the access. But if the master attempts to continue the burst access across the resource boundary, then the target must issue a target disconnect.

The MPC105 is hardwired for fast device select timing (PCI status register[10–9] = 0b00). Therefore, when the MPC105 is the target of a transaction (system memory access), it asserts DEVSEL one clock cycle following the address phase.

As a master, if the MPC105 does not see the assertion of DEVSEL within four clocks after the address phase (five clocks after it asserts FRAME), it terminates the transaction with a master-abort.

7.3.5 Byte Alignment

The byte enable (C/BE3–C/BE0, during a data phase) signals are used to determine which byte lanes carry meaningful data. The byte enable signals may enable different bytes for each of the data phases. The byte enables are valid on the edge of the clock that starts each data phase and will stay valid for the entire data phase. Note that parity is calculated on all bytes regardless of the byte enables. See Section 7.5.1, "Parity," for more information.

If the MPC105, as a target, sees no byte enables asserted, it will complete the current data phase with no permanent change. This implies that on a read transaction, the MPC105 expects that the data is not changed, and on a write transaction, the data is not stored.

7.3.6 Bus Driving and Turnaround

A turnaround-cycle is required, to avoid contention, on all signals that may be driven by more than one agent. The turnaround-cycle occurs at different times for different signals. The IRDY, TRDY, DEVSEL, and STOP signals use the address phase as their turnaround cycle. FRAME, C/BE3–C/BE0, and AD31–AD0 signals use the idle cycle between transactions as their turnaround cycle. The PERR signal has a turnaround cycle on the fourth clock after the last data phase. An idle cycle is when both FRAME and IRDY are negated.

The address/data signals, AD31–AD0, are driven to a stable condition during every address/data phase. Even when the byte enables indicate that byte lanes carry meaningless data, the signals carry stable values. Parity is calculated on all bytes regardless of the byte enables. See Section 7.5.1, "Parity," for more information.

7.4 PCI Bus Transactions

This section provides descriptions of the PCI bus transactions. All bus transactions follow the protocol as described in Section 7.3, “PCI Bus Protocol.” Read and write transactions are similar for the memory and I/O spaces, so they are treated as a generic “read transaction” or a generic “write transaction.”

The timing diagrams show the relationship of significant signals involved in bus transactions.When a signal is drawn as a solid line, it is actively being driven by the current master or target. When a signal is drawn as a dashed line, no agent is actively driving it. Tri-stated signals are indicated to have indeterminate values when the dashed line is between the two rails.

The terms “edge” and “clock edge” always refer to the rising edge of the clock. The terms “asserted” and “negated” always refer to the globally visible state of the signal on the clock edge, and not to signal transitions. “  ” represents a turnaround-cycle in the timing diagrams.

7.4.1 Read Transactions

Figure 7-1 illustrates an example read transaction. The transaction starts with the address phase, occurring when a master asserts $\overline{\text{FRAME}}$. During the address phase, AD31–AD0 contain a valid address and $\overline{\text{C/BE3}}$–$\overline{\text{C/BE0}}$ contain a valid bus command.

The first data phase of a read transaction requires a turnaround-cycle. This allows the transition from the master driving AD31–AD0 as address signals to the target driving AD31–AD0 as data signals. The turnaround-cycle is enforced by the target using the $\overline{\text{TRDY}}$ signal. The earliest the target can provide valid data is one cycle after the turnaround-cycle. The target must drive the address/data signals when $\overline{\text{DEVSEL}}$ is asserted.

During the data phase, the command/byte enables indicate which byte lanes are involved in the current data phase. A data phase may consist of a data transfer and wait cycles. The $\overline{\text{C/BE3}}$–$\overline{\text{C/BE0}}$ signals remain actively driven for both reads and writes from the first clock of the data phase through the end of the transaction.

A data phase completes when data is transferred, which occurs when both $\overline{\text{IRDY}}$ and $\overline{\text{TRDY}}$ are asserted on the same clock edge. When either $\overline{\text{IRDY}}$ or $\overline{\text{TRDY}}$ is negated, a wait cycle is inserted and no data is transferred. The master indicates the last data phase by negating $\overline{\text{FRAME}}$ when $\overline{\text{IRDY}}$ is asserted. The transaction is considered complete when data is transferred in the last data phase.

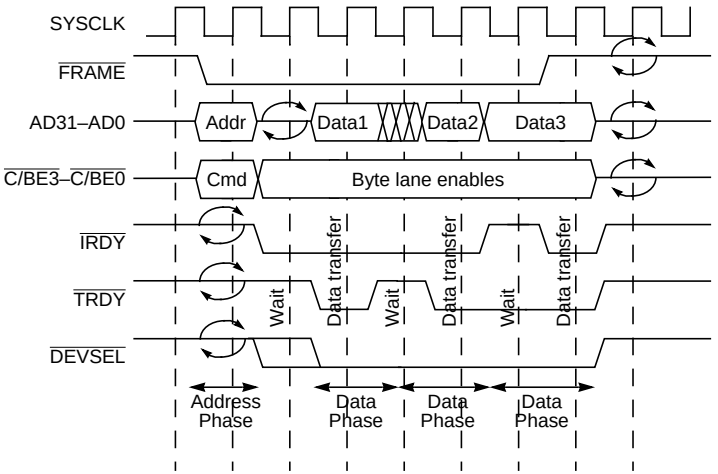


Figure 7-1. Example PCI Read Operation

7.4.2 Write Transactions

Figure 7-2 shows an example write transaction. The transaction starts with the address phase, occurring when a master asserts FRAME. A write transaction is similar to a read transaction except no turnaround-cycle is needed following the address phase because the master provides both address and data. The data phases are the same for both read and write transactions. Note that for the third data phase, even though the master is not ready to provide valid data (IRDY negated), the master must still drive the byte enable signals.

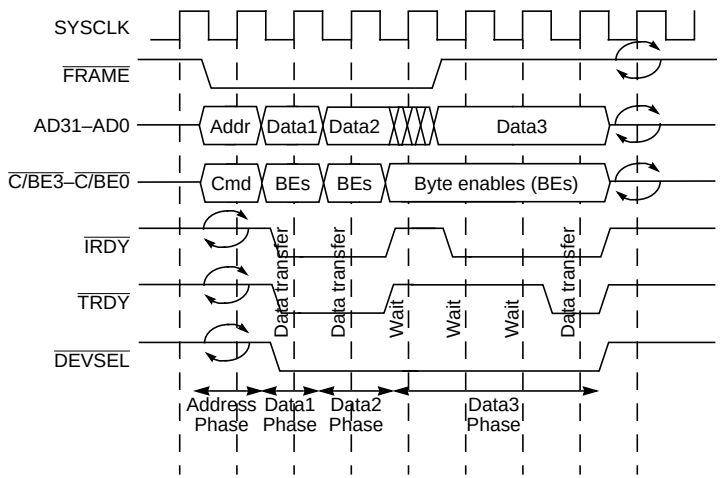


Figure 7-2. Example PCI Write Operation

7.4.3 Transaction Termination

Termination of a PCI transaction may be initiated by either the master or the target. The master is ultimately responsible for bringing all transactions to conclusion, regardless of the cause of the termination. All transactions are concluded when $\overline{\text{FRAME}}$ and $\overline{\text{IRDY}}$ are both negated, indicating the bus is idle.

7.4.3.1 Master-Initiated Termination

Normally, a master initiates termination by negating $\overline{\text{FRAME}}$ and asserting $\overline{\text{IRDY}}$. This indicates to the target that the final data phase is in progress. The final data transfer occurs when both $\overline{\text{TRDY}}$ and $\overline{\text{IRDY}}$ are asserted. The transaction is considered complete when data is transferred in the last data phase. After the final data phase, both $\overline{\text{FRAME}}$ and $\overline{\text{IRDY}}$ are negated (the bus becomes idle).

Normal master-initiated termination occurs for two reasons:

- Completion — Completion refers to termination when the master has concluded its intended transaction. This is the most common reason for termination.
- Timeout — Timeout refers to termination when the master loses its bus grant (GNT is negated) and its internal latency timer has expired. The intended transaction is not necessarily concluded. Note that the MPC105 does not have a latency timer. Latency for the MPC105 acting as a master is determined by the target.

Master-abort is an abnormal case of master-initiated termination. If no device (including the subtractive decoding agent) asserts $\overline{\text{DEVSEL}}$ to claim a transaction, then the master terminates the transaction with a master-abort. For a master-abort termination, the master negates $\overline{\text{FRAME}}$ and then negates $\overline{\text{IRDY}}$ on the next clock. If a transaction is terminated by master-abort (except for a special-cycle), the received master-abort bit (bit 13) of the PCI status register is set.

As a master, if the MPC105 does not detect the assertion of $\overline{\text{DEVSEL}}$ within four clocks following the address phase (five clocks after asserting $\overline{\text{FRAME}}$), it terminates the transaction with a master-abort.

7.4.3.2 Target-Initiated Termination

By asserting the $\overline{\text{STOP}}$ signal, a target may request that the master terminate the current transaction. Once asserted, the target holds $\overline{\text{STOP}}$ asserted until the master negates $\overline{\text{FRAME}}$. Data may or may not be transferred during the request for termination. If $\overline{\text{TRDY}}$ and $\overline{\text{IRDY}}$ are asserted during the assertion of $\overline{\text{STOP}}$, data is transferred. However, if $\overline{\text{TRDY}}$ is negated when $\overline{\text{STOP}}$ is asserted, it indicates that the target will not transfer any more data, and the master therefore does not wait for a final data transfer as it would in a completion termination.

Target-initiated termination occurs for two reasons:

- Disconnect — Disconnect refers to termination requested because the target is unable to respond within eight PCI clocks (not including the first data phase). Disconnect implies that some data was transferred. The master may restart the transaction at a later time starting with the address of the next untransferred data. (That is, data transfer may resume where it left off.)
- Retry — Retry refers to termination requested because the target is currently in a state where it is unable to process the transaction. Retry implies that no data was transferred. The master may start the entire transaction over again at a later time.

Note that as a target, the MPC105 executes a target disconnect after the first data phase completes if AD1–AD0 ≠ 0b00 during the address phase of a system memory access. See Section 7.3.3, “Addressing,” for more information.

When a transaction is terminated by $\overline{\text{STOP}}$, the master must negate its $\overline{\text{REQ}}$ signal for a minimum of two PCI clocks, one being when the bus goes to the idle state ($\overline{\text{FRAME}}$ and $\overline{\text{IRDY}}$ negated). If the master intends to complete the transaction, it must reassert its $\overline{\text{REQ}}$ immediately following the two clocks or potential starvation may occur. If the master does not intend to complete the transaction, then it can assert $\overline{\text{REQ}}$ whenever it needs to use the PCI bus again.

Target-abort is an abnormal case of target-initiated termination. Target-abort is used when a fatal error has occurred, or when a target will never be able to respond. Target-abort is indicated by asserting $\overline{\text{STOP}}$ and negating $\overline{\text{DEVSEL}}$. This indicates that the target requires termination of the transaction and does not want the transaction retried. If a transaction is terminated by target-abort, the received target-abort bit (bit 12) of the master’s status register and the signaled target-abort bit (bit 11) of the target’s status register is set. Note that any data transferred in a target-aborted transaction may be corrupt.

7.4.4 Fast Back-to-Back Transactions

The PCI bus allows fast back-to-back transactions by the same master. During a fast back-to-back transaction, the master starts the next transaction immediately without an idle state. The last data phase completes when $\overline{\text{FRAME}}$ is negated, and $\overline{\text{IRDY}}$ and $\overline{\text{TRDY}}$ are asserted. The current master starts another transaction on the clock immediately following the last data transfer for the previous transaction.

Fast back-to-back transactions must avoid contention on the $\overline{\text{TRDY}}$, $\overline{\text{DEVSEL}}$, $\overline{\text{PERR}}$, and $\overline{\text{STOP}}$ signals. There are two types of fast back-to-back transactions—those that access the same target, and those that access multiple targets (sequentially). The first type places the burden of avoiding contention on the master; the second type places the burden of avoiding contention on all potential targets.

As a master, the MPC105 does not perform any fast back-to-back transactions. As a target, the MPC105 supports both types of fast back-to-back transactions.

During fast back-to-back transactions, the MPC105 monitors the bus states to determine if it is the target of a transaction. If the previous transaction was not directed to the MPC105 and the current transaction is directed at the MPC105, the MPC105 delays the assertion of $\overline{\text{DEVSEL}}$ (as well as $\overline{\text{TRDY}}$, $\overline{\text{STOP}}$, and $\overline{\text{PERR}}$) for one cycle to allow the other target to get off the bus.

7.4.5 Configuration Cycles

This section describes configuring standard PCI devices using PCI configuration cycles. Configuring the internal registers of the MPC105 is described in Chapter 3, “Device Programming.” The PCI configuration space is intended for configuration, initialization, and catastrophic error handling functions only. Access to the configuration space should be limited to initialization and error handling software.

7.4.5.1 The Configuration Space Header

The first 64 bytes of the 256-byte configuration space consists of a predefined header that every PCI device must support. The predefined header is shown in Figure 7-3. The rest of the 256-byte configuration space is device-specific.

The first 16 bytes of the predefined header are defined the same for all PCI devices; the remaining 48 bytes of the header may have differing layouts depending on the function of the device. Most PCI devices will use the configuration header layout shown in Figure 7-3.

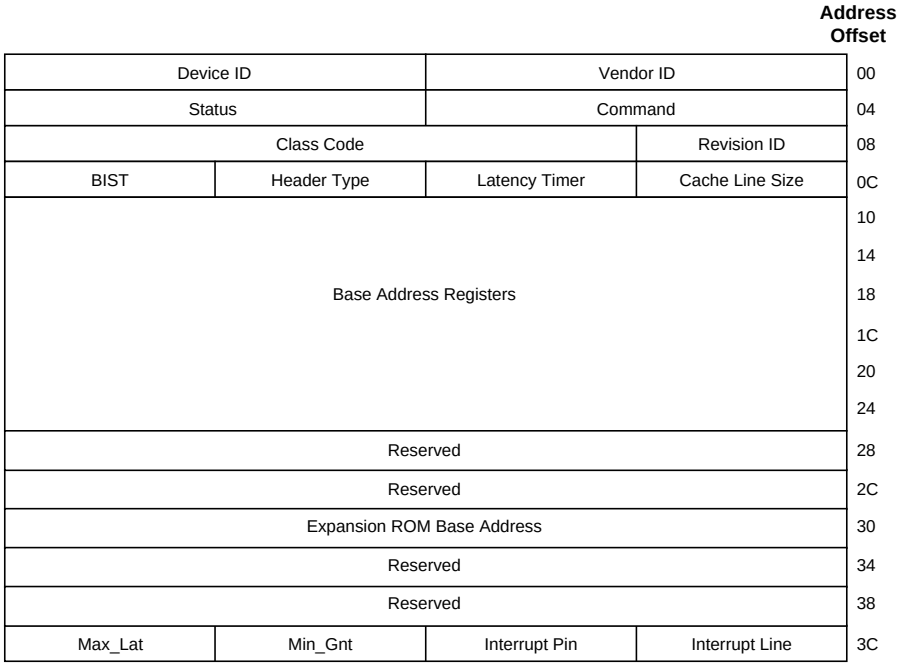


Figure 7-3. Standard PCI Configuration Header

Table 7-2 summarizes the registers of the configuration header. Detailed descriptions of these registers are provided in the *PCI Local Bus Specification*.

Table 7-2. PCI Configuration Space Header Summary

Address Offset	Register Name	Description
00	Vendor ID	Identifies the manufacturer of the device (assigned by the PCI SIG to ensure uniqueness)
02	Device ID	Identifies the particular device (assigned by the vendor)
04	Command	Provides coarse control over a device's ability to generate and respond to PCI bus cycles
06	Status	Records status information for PCI bus-related events
08	Revision ID	Specifies a device-specific revision code (assigned by vendor)
09	Class code	Identifies the generic function of the device and (in some cases) a specific register-level programming interface
0C	Cache line size	Specifies the system cache line size in 32-bit units
0D	Latency timer	Specifies the value of the latency timer for this bus master in PCI bus clock units

Table 7-2. PCI Configuration Space Header Summary (Continued)

Address Offset	Register Name	Description
0E	Header type	Bits 0–6 identify the layout of bytes 10-3F; bit 7 indicates a multifunction device. The most common header type (0x00) is shown in Figure 7-3 and in this table.
0F	BIST	Optional register for control and status of built-in self test (BIST)
10–27	Base address registers	Address mapping information for memory and I/O space
28	—	Reserved for future use
2C	—	Reserved for future use
30	Expansion ROM base address	Base address and size information for expansion ROM contained in an add-on board
34	—	Reserved for future use
38	—	Reserved for future use
3C	Interrupt line	Contains interrupt line routing information
3D	Interrupt pin	Indicates which interrupt pin the device (or function) uses
3E	Min_Gnt	Specifies the length of the device's burst period in 0.25 µs units
3F	Max_Lat	Specifies how often the device needs to gain access to the bus in 0.25 µs units

7.4.5.2 Accessing the PCI Configuration Space

To support hierarchical bridges, two types of configuration accesses are supported. The first type of configuration access, type 0, is used to select a device on the local PCI bus (the PCI bus connected to the MPC105). Type 0 configuration accesses are not propagated beyond the local PCI bus and must be claimed by a local device or terminated with a master-abort. The second type of configuration access, type 1, is used to pass a configuration request on to another PCI bus (through a PCI-to-PCI bridge). Type 1 accesses are ignored by all targets except PCI-to-PCI bridges.

To access the configuration space, a 32-bit value must be written to the CONFIG_ADDR register that specifies the target PCI bus, the target device on that bus, and the configuration register to be accessed within that device. A read or write to the CONFIG_DATA register causes the MPC105 to translate the access into a PCI configuration cycle (provided the enable bit in CONFIG_ADDR is set and the device number is not 0b1_1111).

The CONFIG_ADDR register is located at different addresses depending on the memory address map in use. The address maps are described in Section 3.1, "Address Maps." For address map A in the contiguous mode, the 60x can access the CONFIG_ADDR register through the MPC105 at 0x8000_0CF8. For address map A in the discontiguous mode, the 60x can access the CONFIG_ADDR register through the MPC105 at 0x8006_7018. For address map B, the 60x can access the CONFIG_ADDR register at any location in the address range from 0xF080_0000 to 0xF0BF_FFFF. For simplicity, the address for

CONFIG_ADDR is sometimes referred to as “CF8,” “0x*nnnn_n*CF8,” or (in the PCI literature as) “CF8h.” Although systems implementing address map B can use any address in the range from 0xF080_0000 to 0xF0BF_FFFF for the CONFIG_ADDR register, the address 0xF080_0CF8 may be the most intuitive location.

The format of CONFIG_ADDR is shown in Figure 7-4.

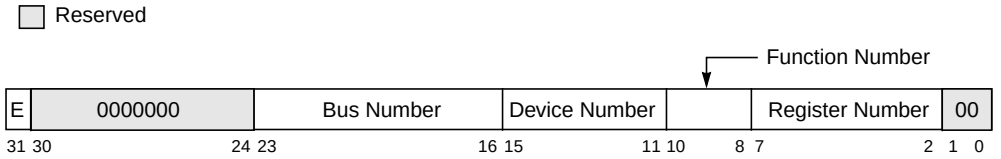


Figure 7-4. Layout of CONFIG_ADDR Register

Table 7-3 describes the fields within CONFIG_ADDR.

Table 7-3. CONFIG_ADDR Register Fields

Bits	Field Name	Description
31	E(nable)	Enable Flag. This bit controls whether accesses to CONFIG_DATA are translated into PCI configuration cycles. 1 Enabled 0 Disabled
30–24	—	Reserved (must be 0b000_0000)
23–16	Bus number	This field is an encoded value used to select the target bus of the configuration access. For target devices on the PCI bus connected to the MPC105, this field should be set to 0x00.
15–11	Device number	This field is used to select a specific device on the target bus.
10–8	Function number	This field is used to select a specific function in the requested device. Single-function devices should respond to function number = 0b000.
7–2	Register number	This field is used to select the address offset in the configuration space of the target device.
1–0	—	Reserved (must be 0b00)

As with the CONFIG_ADDR register, the CONFIG_DATA register is located at different addresses depending on the memory address map in use. For address map A in the contiguous mode, the 60x can access the CONFIG_DATA register through the MPC105 at 0x8000_0CFC–0x8000_0CFF. For address map A in the discontiguous mode, the 60x can access the CONFIG_DATA register through the MPC105 at 0x8006_701C–0x8006_701F. For address map B, the 60x can access the CONFIG_DATA register at any location in the address range from 0xF0C0_0000 to 0xF0DF_FFFF. For simplicity, the address for CONFIG_DATA is sometimes referred to as “CFC,” “0x*nnnn_n*CFC,” or (in the PCI literature as) “CFCh.” Although systems implementing address map B can use any address

in the range from 0xF0C0_0000 to 0xF0DF_FFFF for the CONFIG_DATA register, the address 0xF0C0_0CFC–0xF0C0_0CFF may be the most intuitive location.

Note that the CONFIG_DATA register may contain 1, 2, 3, or 4 bytes depending on the register number.

When the MPC105 detects an access to the CONFIG_DATA register, it checks the enable flag and the device number in the CONFIG_ADDR register. If the enable bit is set, and the device number is not 0b1_1111, the MPC105 performs a configuration cycle translation and runs a configuration-read or configuration-write transaction on the PCI bus. The device number 0b1_1111 is used for performing interrupt-acknowledge and special-cycle transactions. See Section 7.4.6, “Other Bus Transactions,” for more information. If the bus number corresponds to the local PCI bus (bus number = 0x00), the MPC105 performs a type 0 configuration cycle translation. If the bus number indicates a non-local PCI bus, the MPC105 performs a type 1 configuration cycle translation.

For type 0 configuration cycle translations, the MPC105 translates the device number field of the CONFIG_ADDR register into a unique IDSEL signal for up to 21 different devices. Each device connects its IDSEL input to one of the AD31–AD11 signals. If the device number field contains 0b0_1011, the MPC105 drives AD11 high and AD31–AD12 low during the address phase of the configuration cycle; if the device number field contains 0b0_1100, the MPC105 drives AD11 low, AD12 high, and AD31–AD13 low during the address phase of the configuration cycle; continuing on until for a device number of 0b1_1110, the MPC105 drives AD31 low, AD30 high, and AD29–AD11 low during the address phase. The one exception to this translation is for a device number of 0b0_1010; the MPC105 drives AD31 high and AD30–AD11 low during the address phase.

For type 0 translations, the function number and register number fields are copied without modification onto the AD10–AD2 signals during the address phase. The AD1–AD0 signals are driven to 0b00 during the address phase for type 0 configuration cycles. Figure 7-5 shows the type 0 translation from the CONFIG_ADDR register to the AD31–AD0 signals on the PCI bus during the address phase of the configuration cycle.

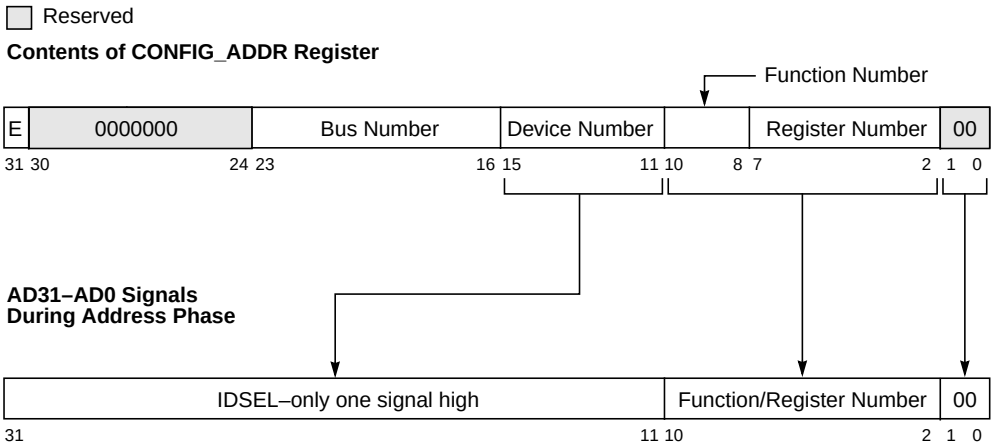


Figure 7-5. Type 0 Configuration Translation

For type 1 translations, the MPC105 copies without modification the 30 high-order bits of the CONFIG_ADDR register onto the AD31–AD2 signals during the address phase. The MPC105 automatically translates AD1–AD0 into 0b01 during the address phase to indicate a type 1 configuration cycle.

7.4.6 Other Bus Transactions

There are two other PCI transactions that the MPC105 supports—interrupt-acknowledge cycles and special cycles. As a master, the MPC105 may initiate both interrupt-acknowledge and special cycles; however, as a target, the MPC105 ignores interrupt-acknowledge and special cycles. Both transactions make use of the CONFIG_ADDR and CONFIG_DATA registers discussed in Section 7.4.5.2, “Accessing the PCI Configuration Space.”

7.4.6.1 Interrupt Acknowledge

The PCI bus supports an interrupt-acknowledge transaction. The interrupt-acknowledge command is a read implicitly addressed to the system interrupt controller.

When the MPC105 detects a read to the CONFIG_DATA register, it checks the enable flag and the device number in the CONFIG_ADDR register. If the enable bit is set, the device number is 0b1_1111, and the bus number corresponds to the local PCI bus (bus number = 0x00), the MPC105 performs an interrupt-acknowledge transaction.

The address phase contains no valid information other than the command signals. There is no explicit address, however AD31–AD0 are driven to a stable state and parity is generated. Only one device (the system interrupt controller) on the PCI bus should respond to the interrupt-acknowledge command by asserting $\overline{\text{DEVSEL}}$. All other devices on the bus should ignore the interrupt-acknowledge command.

During the data phase, the responding device returns the interrupt vector on AD31–AD0 when $\overline{\text{TRDY}}$ is asserted. The size of the interrupt vector returned is indicated by the byte enable signals.

7.4.6.2 Special Cycle

The special-cycle command provides a mechanism to broadcast select messages to all devices on the PCI bus. The special-cycle command contains no explicit destination address, but is broadcast to all PCI agents.

When the MPC105 detects a write to the CONFIG_DATA register, it checks the enable flag and the device number in the CONFIG_ADDR register. If the enable bit is set, the device number is 0b1_1111, and the bus number corresponds to the local PCI bus (bus number = 0x00), then the MPC105 performs a special-cycle transaction on the local PCI bus. If the bus number indicates a nonlocal PCI bus, the MPC105 performs a type 1 configuration cycle translation, similar to any other configuration cycle for which the bus number does not match.

The address phase contains no valid information other than the command signals. There is no explicit address, however AD31–AD0 are driven to a stable state and parity is generated. During the data phase, AD31–AD0 contain the message and an optional data field. The message is encoded on the 16 least significant bits (AD15–AD0); the optional data field is encoded on the most significant 16 lines (AD31–AD16). The special-cycle message encodings are assigned by the PCI SIG Steering Committee. The current list of defined encodings and how the MPC105 implements them are provided in Table 7-4.

Table 7-4. Special-Cycle Message Encodings

AD15–AD0	Message	Description
0x0000	SHUTDOWN	Indicates the MPC105 is entering the sleep power saving mode
0x0001	HALT	Indicates the MPC105 is entering either the nap or sleep power saving mode
0x0002	x86 architecture-specific	This message type is not used by the MPC105.
0x0003–0xFFFF	—	Reserved

Note that the power management configuration register (PMCR) controls which special-cycle messages (if any) the MPC105 broadcasts to the PCI bus. See Section 3.2.4, “Power Management Configuration Register (PMCR),” for a description of the PMCR.

Each receiving agent must determine whether the message is applicable to itself. Assertion of $\overline{\text{DEVSEL}}$ in response to a special-cycle command is not necessary. The master of the special-cycle can insert wait states but since there is no specific target, the message and data are valid on the first clock $\overline{\text{IRDY}}$ is asserted. All special-cycles are terminated by master-abort; however, the received master-abort bit in the master's status register is not set for special-cycle terminations.

7.5 PCI Error Functions

PCI provides for parity and other system errors to be detected and reported. This section describes generation and detection of parity and error reporting for the PCI bus.

The PCI command register and error enabling register 1 provide for selective enabling of specific PCI error detection. The PCI status register, error detection register 1, the PCI bus error status register, and the CPU/PCI error address register provide PCI error reporting. These registers are described in Section 3.2.3, "PCI Registers," and Section 3.2.5, "Error Handling Registers."

7.5.1 Parity

Generating parity is not optional; it must be performed by all PCI-compliant devices. All PCI transactions, regardless of type, calculate parity the same way—the number of "1s" on AD31–AD0, $\overline{\text{C/BE3}}$–$\overline{\text{C/BE0}}$, and PAR all sum to an even number.

Parity provides a way to determine, on each transaction, if the master successfully addressed the target and transferred valid data. The $\overline{\text{C/BE3}}$–$\overline{\text{C/BE0}}$ signals are included in the parity calculation to insure that the correct bus command is performed (during the address phase) and that correct data is transferred (during the data phase). The agent that is responsible for driving the bus is also responsible for driving even parity on PAR one clock after a valid address phase or valid data transfer.

During the address and data phases, parity covers all 32 address/data signals and the four command/byte enable signals regardless of whether all lines carry meaningful information. Byte lanes not actually transferring data must contain stable (albeit meaningless) data and are included in parity calculation. During configuration, special-cycle or interrupt-acknowledge commands, some address lines are not defined but are driven to stable values and are included in parity calculation.

Agents that support parity checking must set the detected parity error bit in the status register when a parity error is detected. Any additional response to a parity error is controlled by the parity error response bit in the command register. If the parity error response bit is cleared, the agent ignores all parity errors.

7.5.2 Error Reporting

PCI provides for the detection and signaling of both parity and other system errors. Two signals are used to report these errors—$\overline{\text{PERR}}$ and $\overline{\text{SERR}}$. The $\overline{\text{PERR}}$ signal is used exclusively to report data parity errors on all transactions except special-cycles. $\overline{\text{SERR}}$ is used for other error signaling including data parity errors on special-cycles, address parity errors, and may be used to signal other system errors. Refer to Section 9.3.3, “PCI Interface,” for a complete description of MPC105 actions on due to parity and other errors.

7.6 MPC105-Implemented PCI Sideband Signals

The PCI specification loosely defines a sideband signal as any signal not part of the PCI specification that connects two or more PCI-compliant agents, and has meaning only to those agents. The MPC105 implements three PCI sideband signals—$\overline{\text{ISA\_MASTER}}$, $\overline{\text{FLSHREQ}}$, and $\overline{\text{MEMACK}}$. This section describes the use of these signals in a PCI bus design.

7.6.1 $\overline{\text{ISA\_MASTER}}$

The $\overline{\text{ISA\_MASTER}}$ signal provides a mechanism to access system memory for ISA devices (or a PCI-to-ISA bridge) that cannot generate a full 32-bit address.

Normally, when using address map A, a PCI memory read or write command to addresses in the range 0x8000_0000– 0xFFFF_FFFF generates an access to system memory. However, if the PCI-to-ISA bridge runs a memory transaction that does not use a full 32-bit address, access to system memory is impossible. Assertion of $\overline{\text{ISA\_MASTER}}$ indicates that an ISA master is requesting access to system memory.

The $\overline{\text{ISA\_MASTER}}$ signal should be asserted coincident with the PCI-to-ISA bridge receiving a PCI bus grant. When the MPC105 detects $\overline{\text{ISA\_MASTER}}$ asserted (during the address phase), the MPC105 automatically asserts $\overline{\text{DEVSEL}}$ to claim the transaction regardless of the address in AD31–AD0. Due to the automatic assertion of $\overline{\text{DEVSEL}}$ when $\overline{\text{ISA\_MASTER}}$ is detected, possible bus contention can occur if the current transaction is not truly intended for the MPC105 (system memory access).

If the PCI-to-ISA bridge can generate a full 32-bit address, the $\overline{\text{ISA\_MASTER}}$ signal is unnecessary and may be tied to V_{DD} (high).

7.6.2 $\overline{\text{FLSHREQ}}$ and $\overline{\text{MEMACK}}$

The $\overline{\text{FLSHREQ}}$ signal allows a PCI agent to request that the MPC105 flush its internal buffers. The $\overline{\text{MEMACK}}$ allows the MPC105 to acknowledge that it has flushed its internal buffers.

If a master on the PCI bus asserts $\overline{\text{FLSHREQ}}$, the MPC105 stops accepting new transactions from the 60x bus (except snoop copy-back operations), completes all outstanding transactions, and then asserts $\overline{\text{MEMACK}}$. The MPC105 holds $\overline{\text{MEMACK}}$ asserted until two cycles after the master negates $\overline{\text{FLSHREQ}}$. When $\overline{\text{FLSHREQ}}$ is negated, the master must wait until after $\overline{\text{MEMACK}}$ is negated before it can reassert $\overline{\text{FLSHREQ}}$.

Chapter 8
Internal Control

The MPC105 uses internal buffering to store addresses and data moving through it, and to maximize opportunities for concurrent operations. A central control unit directs the flow of transactions through the MPC105, performing internal arbitration and coordinating the internal and external snooping. This chapter describes the internal buffering and arbitration logic of the MPC105.

8.1 Internal Buffers

For most operations, the data is latched internally in one of seven data buffers. The exception is processor accesses to system memory. Data transfers between the 60x processor and system memory, with the exception of snoop copy-backs, occur directly on the shared data bus, so no data buffering is required for those transactions.

There are eight address buffers which correspond to the seven data buffers plus an eighth buffer for the address of the current 60x processor/system memory access. All transactions entering the MPC105 have their addresses stored in the internal address buffers. The address buffers allow the addresses to be snooped as other transactions attempt to go through the MPC105. This is especially important for write transactions that enter the MPC105 because the update of memory may be performed out-of-order with respect to other transactions.

Figure 8-1 depicts the organization of the internal buffers.

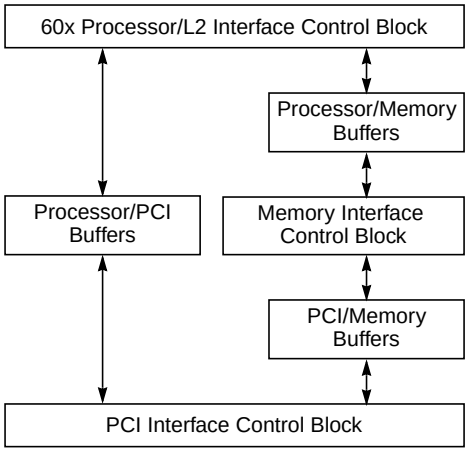


Figure 8-1. MPC105 Internal Buffer Organization

8.1.1 60x Processor/System Memory Buffers

Because systems using the MPC105 have a shared data bus between the processor, L2 cache, and system memory, for most cases it is unnecessary to buffer data transfers between these devices. However, there is a 32-byte copy-back buffer which is used for temporary storage of L1 copy-backs due to snooping PCI-initiated reads from memory, and L2 castouts.

Figure 8-2 shows the address and data buffers between the 60x processor bus and the system memory bus.

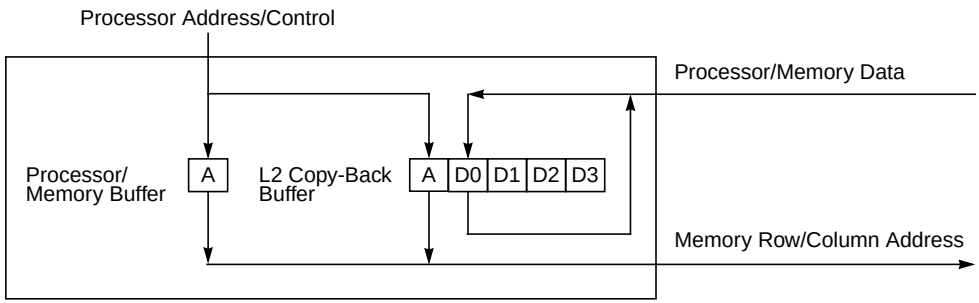


Figure 8-2. Buffers between the 60x Processor Bus and the System Memory Bus

The copy-back buffer can only be in one of two states—invalid, or modified with respect to system memory. Since the buffer is only used for burst write data, the entire cache line in the buffer is always valid if any part of the cache line is valid.

L2 cast-outs are caused by a 60x processor transaction that misses in the L2 cache, and the cache line in the L2 that will be replaced currently holds modified data. The MPC105 latches the modified data from the L2 to minimize the latency of the original 60x processor/system memory transaction. The MPC105 completes the flush of the data in the buffer to system memory at the earliest available opportunity.

In the case of a snoop for a PCI read from system memory that causes an L1 copy-back, the copy-back data is latched in the copy-back buffer and simultaneously forwarded to the PCI bus. Once the L1 copy-back is complete and the PCI agent has finished reading from the copy-back buffer, the MPC105 flushes the data to system memory at the earliest available opportunity.

Once the copy-back buffer has been filled, the data remains in the buffer until the system memory bus is available to complete the copy-back to system memory. During the time that modified data waits in the copy-back buffer, all transactions to system memory space are snooped against the copy-back buffer, and in some cases the transaction can read from or write to the copy-back buffer instead of system memory. Since it is possible for the L1 cache in the 60x processor to contain a more recently modified version of a cache line than that in the copy-back buffer, all PCI-initiated transactions that hit in the copy-back buffer cause a snoop broadcast on the 60x processor bus.

8.1.2 60x Processor/PCI Buffers

There are three data buffers for processor accesses to PCI—one 32-byte buffer for processor reads from PCI (PRPRB), and two 16-byte buffers for processor writes to PCI (PRPWBs) each with an associated address buffer. Figure 8-3 shows the address and data buffers between the 60x processor bus and the PCI bus.

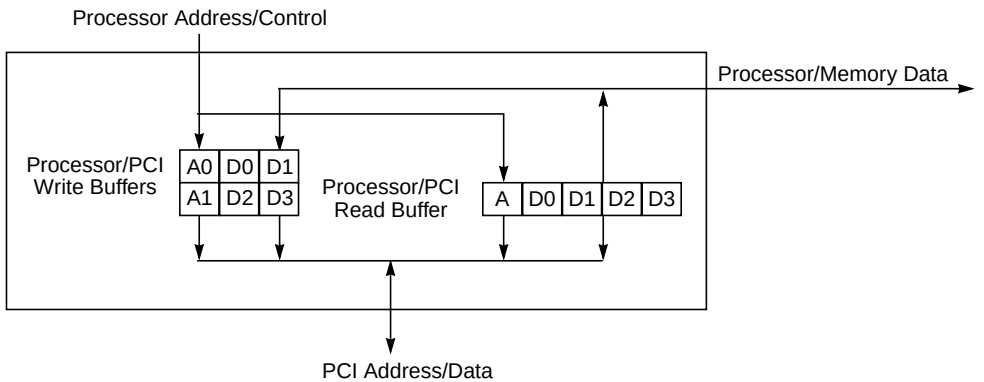


Figure 8-3. Buffers between the 60x Processor Bus and the PCI Bus

8.1.2.1 Processor-Read-from-PCI Buffer (PRPRB)

60x processor reads from PCI require buffering for two primary reasons. First, the processor bus uses a critical-word-first protocol, while the PCI bus uses a zero-word-first protocol. The MPC105 requests the data zero-word-first, latches the requested data, and then delivers the data to the 60x processor critical-word first.

The second reason is that if the target for a processor read from PCI disconnects part way through the data transfer, the MPC105 may have to handle a system memory access from an alternate PCI master before the disconnected transfer can continue.

When the processor requests data from the PCI space, the data received from PCI is stored in the PRPRB until all requested data has been latched. The MPC105 does not terminate the address tenure of the 60x transaction until all requested data is latched in the PRPRB. If the PCI target disconnects in the middle of the data transfer and an alternate PCI master acquires the bus and initiates a read from system memory, the MPC105 retries the 60x processor so that the incoming PCI transaction can be snooped. A PCI-initiated read from system memory may require a snoop transaction on the 60x processor bus, and a copy-back may be necessary.

The PCI interface of the MPC105 continues to request the PCI bus until the processor's original request is completed. When the next processor transaction starts, the address is snooped against the address of the previous transaction (in the internal address buffer) to verify that the same data is being requested. Once all the requested data is latched, the MPC105 asserts $\overline{\text{AACK}}$ and $\overline{\text{DBG}}n$ (as soon as the 60x data bus is available) and completes the data transfer to the processor. If a second processor starts a new transaction, the address cannot match the disconnected transaction address. If the new transaction is not a read from PCI, it proceeds normally; if the transaction is a read from PCI, it must wait until the disconnected transaction completes before proceeding.

For example, if the processor initiates a critical-word-first burst read, starting with the second double word of the cache line, the read on the PCI bus begins with the cache-line-aligned address. If the PCI target disconnects after transferring the first half of the cache line, the MPC105 re-arbitrates for the PCI bus, and when granted, initiates a new transaction with the address of the third double word of the line. If an alternate PCI master requests data from system memory while the MPC105 is waiting for the PCI bus grant, the MPC105 retries the processor transaction to allow the PCI-initiated transaction to snoop the processor bus. When the processor snoop is complete, the subsequent processor transaction is compared to the latched address and attributes of the PCI read buffer to ensure that the processor is requesting the same data. Once all data requested by the processor is latched in the PCI read buffer, the data is transferred to the processor, completing the transaction.

8.1.2.2 Processor-to-PCI-Write Buffers (PRPWBs)

There are two 16-byte buffers for processor writes to PCI. These buffers can be used together as one 32-byte buffer for processor burst writes to PCI, or separately for single-beat writes to PCI. This allows the MPC105 to support both burst transactions and streams of single-beat transactions. The MPC105 performs store gathering (if enabled) within the 16-byte range that makes up either the first or second half of the cache line. All transfer sizes are gathered if enabled (PICR1[ST_GATH_EN] = 1).

The internal buffering minimizes the effect of the slower PCI bus on the higher speed 60x processor bus. Once the processor write data is latched internally, the 60x processor bus is available for subsequent transactions without having to wait for the write to the PCI target to complete. Note that both PCI memory and I/O accesses are buffered. Device drivers must take into account that writes to I/O devices on the PCI bus are posted. The processor may believe that the write has completed while the MPC105 is still trying to acquire mastership of the PCI bus.

If the processor initiates a burst write to PCI, the 60x data transfer is delayed until all previous writes to PCI are completed, and then the burst data from the 60x processor fills the two PRPWBs. The address and transfer attributes are stored in the first address buffer.

For a stream of single-beat writes, the data for the first transaction is latched in the first buffer and the MPC105 initiates the transaction on the PCI bus. The second single-beat write is then stored in the second buffer. For subsequent single-beat writes, store gathering is possible if the incoming write is to the same half cache line as the previously latched data. Store gathering is only used for writes to PCI memory space, not for writes to PCI I/0 space. The store gathering continues until the buffer is scheduled to be flushed or until the processor issues a synchronizing transaction.

For example, if both PRPWBs are empty and the 60x processor issues a single-beat write to PCI, the data is latched in the first buffer and the PCI interface of the MPC105 attempts to acquire the PCI bus for the transfer. The data for the next 60x-to-PCI write transaction is latched in the second buffer, even if the second transaction's address falls within the same half cache line as the first transaction. As long as the PCI interface is busy with the first transfer, any processor single-beat writes to the same half cache line as the second transfer are gathered in the second buffer until the PCI bus becomes available.

8.1.3 PCI/System Memory Buffers

There are three data buffers for PCI accesses to system memory—one 32-byte buffer for PCI reads from system memory (PCMRB) and two 32-byte buffers for PCI writes to system memory (PCMWBs) each with an associated address buffer. Figure 8-4 shows the address and data buffers between the PCI bus and the system memory.

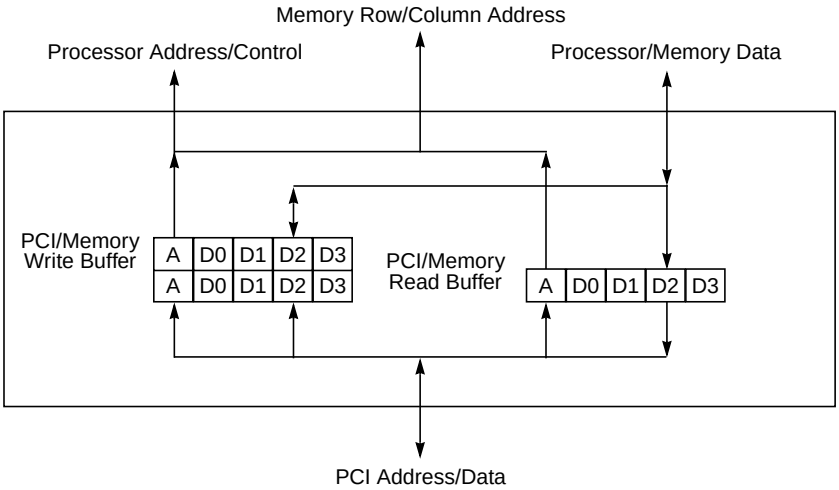


Figure 8-4. Buffers between the PCI bus and System Memory

Note that many PCI accesses to system memory are snooped on the 60x processor bus to ensure coherency between the PCI bus, system memory, the L1 cache of the 60x processor, and the L2 cache (if present). Table 8-1 summarizes the snooping behavior of PCI read and write transactions that hit in one of the internal buffers.

Table 8-1. Snooping Behavior Caused by a Hit in an Internal Buffer

PCI Transaction	Hit in Internal Buffer	Snoop Required
Read	Copy-back	Yes
Read (not locked)	PCMRB	No
Read (first access of a locked transfer)[1]	PCMRB	Yes
Read (not locked)	PCMWB	No
Read (first access of a locked transfer)[1]	PCMWB	Yes
Write	Copy-back	Yes
Write	PCMRB	Yes
Write	PCMWB	No

[1] Only reads can start an exclusive access (locked transfer). The first locked transfer must be snooped so that the cache line in the L1 is invalidated.

8.1.3.1 PCI-Read-from-System-Memory Buffer (PCMRB)

When a PCI device initiates a read from system memory, the address is snooped on the 60x processor bus. The memory access is started simultaneous with the snoop. If the snoop results in a hit in either the L1 or L2 cache, the MPC105 cancels the system memory access.

Depending on the outcome of the snoop, the requested data is latched into either the 32-byte PCI-read-from-system-memory buffer (PCMRB), or into the copy-back buffer (as described in Section 8.1.1, “60x Processor/System Memory Buffers”).

- If the snoop hits in the L1, the copy-back data is written to the copy-back buffer, forwarded to PCI, and then written to memory when the PCI transfer is complete.
- If the snoop hits in the L2, the data is written to the PCMRB and sent to PCI without changing the internal state of the data in the L2. Note that a copy-back to system memory is unnecessary because the state of the data in the L2 remains unchanged.
- If the snoop does not hit in either the L1 or L2, the PCMRB is filled from system memory starting at the requested address to the end of the cache line.

The data is forwarded to PCI as soon as it is received, not when the complete cache line has been written into the PCMRB. The addresses for subsequent PCI reads are compared to the existing address, so if the new access falls within the same cache line and the requested data is already latched in the buffer, the data can be forwarded to PCI without requiring a snoop or another memory transaction.

If a PCI write address hits in the PCMRB, the buffer is invalidated and the address is snooped on the processor bus. If the 60x processor accesses the address in the PCMRB, the PCMRB is invalidated.

8.1.3.2 PCI-to-System-Memory-Write Buffers (PCMWBs)

For PCI write transactions to system memory, the MPC105 employs two PCMWBs. The PCMWBs hold up to one cache line (32-bytes) each. Before PCI data is transferred to system memory, the address must be snooped on the 60x processor bus. The buffers allow for the data to be latched while waiting for a snoop response. The write data can be accepted without inserting wait states on the PCI bus. Also, two buffers allow a PCI master to write to one buffer, while the other buffer is flushing its contents to system memory. Both PCMWBs are capable of gathering for writes to the same cache line.

If the snoop on the 60x processor bus hits modified data in either the L1 or L2 cache, the snoop copy-back data is merged with the data in the PCMWB, and the full cache line is sent to memory. For the PCI memory-write-and-invalidate command, a snoop hit in either the L1 or L2 cache invalidates any modified cache line without requiring a copy-back.

Note that a PCI transaction that hits in either of the PCMWBs does not require a snoop on the 60x processor bus. However, if a PCI write address hits in the PCI-read-from-system-memory buffer (PCMRB), the MPC105 invalidates the PCMRB and snoops the address on the 60x processor bus.

When the PCI write is complete and the snooping is resolved, the data is flushed to memory at the first available opportunity.

For a stream of single-beat writes, the data for the first transaction is latched in the first buffer and the MPC105 initiates the snoop transaction on the 60x processor bus. For subsequent single-beat writes, gathering is possible if the incoming write is to the same cache line as the previously latched data. Gathering to the first buffer can continue until the buffer is scheduled to be flushed, or until a write occurs to a different address. If there is valid data in both buffers, further gathering is not supported until one of the buffers has been flushed.

8.1.3.2.1 Speculative PCI Reads from System Memory

To minimize the latency for large block transfers, the MPC105 includes a selectable speculative read feature. When this feature is enabled (PICR1[2] = 1), the MPC105 starts the snoop of the next sequential cache line address when the current PCI read is accessing the third double word (the second half) of the cache line in the PCMRB.

Once the speculative snoop response is known and PCI has completed the read, the data at the speculative address is fetched from system memory and loaded into the buffer in anticipation of the next PCI request. Note that the assertion of CAS is delayed until PCI is finished reading the data currently latched in the buffer. If a different address is requested, the speculative operation is halted and any data latched in the PCI read buffer is invalidated.

8.2 Internal Arbitration

The arbitration for the PCI bus is performed externally. All processor-PCI transactions are performed strictly in-order. However, the MPC105 performs arbitration internally for the shared processor/memory data bus. The arbitration for the processor/memory data bus employs the following priority scheme:

1. 60x processor read from system memory
2. 60x processor-to-L2 cache transfer
3. L2 copy-back (or PCMRB data transfer) due to a read snoop hit
4. Priority PCMWB flush
5. Priority copy-back buffer flush
6. PCI read from system memory (with snoop complete)
7. 60x processor write to system memory
8. Snoop copy-back due to PCI write snoop
9. 60x processor read from or write to PCI access
10. Load copy-back buffer
11. PCI read from system memory (snoop not complete)
12. Normal copy-back buffer flush
13. Normal PCMWB flush
14. Speculative PCMRB read

Note that the PCMWB and copy-back buffer flushes can be assigned a higher priority under certain conditions. The normal copy-back buffer flush becomes a priority copy-back buffer flush in the following conditions—a 60x processor read hit, a 60x processor single-beat write hit, a PCI read hit, or the buffer is full and new data needs to be written to it. A normal PCMWB flush becomes a priority PCMWB flush in the following conditions—a PCI read hit, the buffer is full and another PCI write starts, or a 60x processor to system memory read or write hit.

Chapter 9
Error Handling

The MPC105 provides error detection and reporting on the three primary interfaces (60x processor interface, memory interface, and PCI interface). This chapter describes how the MPC105 handles different error (or interrupt) conditions.

Errors detected by the MPC105 are reported to the 60x processor by asserting the machine check ($\overline{\text{MCP}}$) or transfer error acknowledge ($\overline{\text{TEA}}$) signal. The system error ($\overline{\text{SERR}}$) and parity error ($\overline{\text{PERR}}$) signals are used to report errors on and to the PCI bus. The MPC105 provides the NMI signal for ISA bridges to report errors on the ISA bus. The MPC105 internally synchronizes any asynchronous error signals.

The PCI command and status registers, and the error handling registers enable or disable the reporting and detection of specific errors. These registers are described in Chapter 3, “Device Programming.”

The MPC105 detects illegal transfer types from the 60x processor, illegal Flash ROM write transactions, L2 cache parity errors, memory parity errors, accesses to memory addresses out of the range of physical memory, PCI address and data parity errors, PCI master-abort cycles, and PCI received target-abort errors.

The MPC105 latches the address and type of transaction that caused the error in the error status registers to assist diagnostic and error handling software. See Section 3.2.5.3, “Error Status Registers,” for more information. Chapter 2, “Signal Descriptions,” contains the signal definitions for the interrupt signals.

9.1 Priority of Externally-Generated Interrupts

Table 9-1 describes the relative priorities and recoverability of externally-generated interrupts.

Table 9-1. Externally-Generated Interrupt Priorities

Priority	Exception	Cause	Processor Recoverability
0	System reset	$\overline{\text{HRST}}$ or power-on reset (POR)	Nonrecoverable in all cases
1	Machine check ($\overline{\text{MCP}}$ or $\overline{\text{TEA}}$)	Memory select error or memory data read parity error	Nonrecoverable in most cases
2	Machine check ($\overline{\text{MCP}}$)	Illegal transaction type or Flash ROM write error	Nonrecoverable in most cases
3	Machine check ($\overline{\text{MCP}}$)	PCI address parity error ($\overline{\text{SERR}}$), PCI data parity error ($\overline{\text{PERR}}$), PCI master-abort, received PCI target-abort	Nonrecoverable in most cases
4	Machine check ($\overline{\text{MCP}}$)	NMI	Nonrecoverable in most cases

Note that for priority 1 through 4, the interrupt is the same. The machine check exception and the priority are related to additional error information provided by the MPC105 (for example, the address provided in the 60x/PCI error address register).

9.2 Interrupt And Error Signals

Although Chapter 2, “Signal Descriptions,” contains the signal definitions for the interrupt and error signals, this section describes the interactions between system components when an interrupt or error signal is asserted.

9.2.1 System Reset

The system reset interrupt is an asynchronous, nonmaskable interrupt that occurs at power-on reset (POR) or when the hard reset ($\overline{\text{HRST}}$) input signal is asserted.

When a system reset request is recognized ($\overline{\text{HRST}}$ or POR), the MPC105 aborts all current internal and external transactions, tri-states all bidirectional I/O signals, ignores the input signals (except for SYSCLK, and the configuration signals FNR/$\overline{\text{DWE0}}$, $\overline{\text{RCS0}}$, DL0, $\overline{\text{XATS}}$, and PLL0–PLL3), and drives most of the output signals to an inactive state. The MPC105 then initializes its internal logic. For proper initialization, the assertion of $\overline{\text{HRST}}$ must satisfy the minimum active pulse width. The minimum active pulse width and other timing requirements for the MPC105 are given in the MPC105 Hardware Specifications.

During system reset, the latches dedicated to JTAG functions are not initialized. The IEEE 1149.1 standard prohibits the device reset from resetting the JTAG logic. The JTAG reset ($\overline{\text{TRST}}$) signal is used to reset the dedicated JTAG logic during POR.

9.2.2 60x Processor Bus Error Signals

The MPC105 provides two signals to the 60x processor bus for error reporting—$\overline{\text{MCP}}$ and $\overline{\text{TEA}}$.

9.2.2.1 Machine Check ($\overline{\text{MCP}}$)

The MPC105 asserts $\overline{\text{MCP}}$ to signal to the 60x processor that a nonrecoverable error has occurred during system operation. The assertion of $\overline{\text{MCP}}$ depends upon whether the error handling registers of the MPC105 are set to report the specific error.

Assertion of $\overline{\text{MCP}}$ causes the 60x processor to conditionally take a machine check exception or enter the checkstop state based on the setting of the MSR[ME] in the 60x processor. The programmable parameter PICR1[MCP_EN] is used to enable or disable the assertion of $\overline{\text{MCP}}$ by the MPC105.

The $\overline{\text{MCP}}$ signal may be asserted on any cycle. The current transaction may or may not be aborted depending upon the software configuration.

The MPC105 holds $\overline{\text{MCP}}$ asserted until the 60x processor has taken the exception. The MPC105 decodes an interrupt acknowledge cycle by detecting 60x processor reads from the two possible machine check exception addresses at 0x0000_0200 and 0xFFF0_0200.

9.2.2.2 Transfer Error Acknowledge ($\overline{\text{TEA}}$)

The MPC105 asserts $\overline{\text{TEA}}$ to signal to the 60x processor that a nonrecoverable error has occurred during data transfer on the 60x processor data bus. The assertion of $\overline{\text{TEA}}$ depends upon whether the error handling registers of the MPC105 are set to report the specific error.

The assertion of $\overline{\text{TEA}}$ causes the 60x processor to conditionally take a machine check exception or enter the checkstop state based on the setting of MSR[ME] in the 60x processor. Note that the assertion of $\overline{\text{TEA}}$ does not prevent corrupt data from being written into the cache or GPRs of the 60x processor.

The $\overline{\text{TEA}}$ signal may be asserted on any cycle that $\overline{\text{DBB}}$ is asserted. The assertion of $\overline{\text{TEA}}$ terminates the data tenure immediately, even if in the middle of a burst. The MPC105 asserts $\overline{\text{TEA}}$ for only one clock.

The programmable parameter PICR1[TEA_EN] is used to enable or disable the assertion of $\overline{\text{TEA}}$ by the MPC105. If PICR1[TEA_EN] is programmed to disable the assertion of $\overline{\text{TEA}}$, and a 60x processor data transfer error occurs, then the MPC105 asserts $\overline{\text{TA}}$ the appropriate number of times to complete the transaction, but the data is unpredictable.

9.2.3 PCI Bus Error Signals

The MPC105 uses three error signals to interact with the PCI bus—$\overline{\text{SERR}}$, $\overline{\text{PERR}}$, and NMI.

9.2.3.1 System Error ($\overline{\text{SERR}}$)

The $\overline{\text{SERR}}$ signal is used to report PCI address parity errors, PCI data parity errors on a special-cycle command, target-abort, or any other errors where the result is potentially catastrophic. The $\overline{\text{SERR}}$ signal is also asserted for master-abort, except if it happens for a PCI configuration access or special-cycle transaction.

The agent responsible for driving AD31–AD0 on a given PCI bus phase is responsible for driving even parity one PCI clock later on the PAR signal. That is, the number of 1's on AD31–AD0, $\overline{C/BE3}$–$\overline{C/BE0}$, and PAR equals an even number.

The $\overline{SERR}$ signal is driven for a single PCI clock cycle by the agent that is reporting the error. The target agent is not allowed to terminate with retry or disconnect if $\overline{SERR}$ is activated due to an address parity error.

Bit 8 of the PCI command register controls whether the MPC105 asserts $\overline{SERR}$ upon detecting one of the error conditions. Bit 14 of the PCI status register reports when the MPC105 asserts the $\overline{SERR}$ signal.

9.2.3.2 Parity Error ($\overline{PERR}$)

The $\overline{PERR}$ signal is used to report PCI data parity errors during all PCI transactions, except for a PCI special-cycle. The agent responsible for driving AD31–AD0 on a given PCI bus phase is responsible for driving even parity one PCI clock later on the PAR signal. That is, the number of 1's on AD31–AD0), $\overline{C/BE3}$–$\overline{C/BE0}$ and PAR equals an even number.

The $\overline{PERR}$ signal must be asserted by the agent receiving data two PCI clocks following the data phase for which a data parity error was detected. Only the master may report a read data parity error and only the selected target may report a write data parity error.

Bit 6 of the PCI command register controls whether the MPC105 ignores $\overline{PERR}$. Bit 15 and bit 8 of the PCI status register are used to report when the MPC105 has detected or reported a data parity error.

9.2.3.3 Nonmaskable Interrupt (NMI)

The NMI signal is, effectively, a PCI sideband signal between the PCI-to-ISA bridge and the MPC105. The NMI signal is driven by the PCI-to-ISA bridge to report any nonrecoverable error detected on the ISA bus (normally, through the $\overline{IOCHCK}$ signal on the ISA bus). The name nonmaskable interrupt is misleading due to its history in ISA bus designs. The NMI signal should be connected to GND if it is not used. If PICR1[MCP_EN] is set, the MPC105 reports the NMI error to the 60x processor by asserting $\overline{MCP}$.

9.3 Error Reporting

Error detection registers 1 and 2 (ErrDR1 and ErrDR2) indicate which specific error has been detected. Associated with these two registers, error enabling registers 1 and 2 (ErrEnR1 and ErrEnR2) are used to enable the latching of the error flags and the corresponding error information which results in the assertion of $\overline{MCP}$, provided PICR1[MCP_EN] is set.

ErrDR1[3] (60x/PCI Cycle) and ErrDR2[7] (invalid error address) together with the 60x/PCI error address register, the 60x bus error status register, and the PCI bus error status register are used to provide additional information about the detected error. Once an error is detected, the associated information is latched inside these registers until all the error flags are cleared. Subsequent errors cannot set the status flags until the previous error's flags are cleared.

9.3.1 60x Processor Interface

The 60x processor interface of the MPC105 detects Flash ROM write errors and unsupported 60x bus cycle errors. In either case, both ErrDR1[3] and ErrDR2[7] are cleared, indicating that the error is due to a 60x bus transaction and the address in the 60x/PCI error address register is valid. The MPC105 asserts either $\overline{\text{TEA}}$ or $\overline{\text{TA}}$ (depending on the value of PICR1[TEA_EN]) to terminate the data tenure.

9.3.1.1 Flash ROM Write Error

The MPC105 allows single-byte writes to the ROM space when it is configured for Flash ROM and PICR1[FLASH_WR_EN] is set. Otherwise, any 60x processor write transaction to the ROM space results in a Flash ROM write error. Write transactions of more than one data byte must be broken into a series of single-byte writes by software. When a Flash ROM write error occurs, ErrDR2[0] is set.

9.3.1.2 Unsupported 60x Bus Error

When an unsupported 60x bus cycle error occurs, ErrDR1[1–0] is set to reflect the error type. Unsupported 60x bus transactions include $\overline{\text{XATS}}$-initiated transactions and transactions with unsupported transfer attributes. Unsupported transfer attributes include writes to the PCI/ISA INTACK address (0xBFFF_FFF*n*) using address map A, and illegal and reserved transfer types defined in Table 4-1.

9.3.2 Memory Interface

The memory interface of the MPC105 detects parity errors on the data bus during memory (DRAM or SDRAM transaction) read cycles or during L2 cache (SRAM) read cycles. The MPC105 also detects errors with system memory transaction addresses that fall outside of the physical memory boundaries.

If the memory read transaction was initiated by a PCI master, ErrDR1[3] is set; if the memory read transaction was initiated by the 60x processor, ErrDR1[3] is cleared.

ErrDR2[7] is cleared to indicate that the error address in the 60x/PCI error address register is valid. However, for L2 data parity errors, the MPC105 cannot provide the error address and the corresponding bus status. Thus, ErrDR2[7] is set to indicate that the error address in the 60x/PCI error address register is not valid.

If the read transaction is initiated by the 60x processor, or by a PCI master with bit 6 of the PCI command register cleared, then the error status information is latched, but the transaction continues and terminates normally. If the transaction is initiated by a PCI master and bit 6 of the PCI command register is set, the PCI interface of the MPC105 signals a target-abort for the current transaction (if it has not completed).

9.3.2.1 System Memory Read Data Parity Error

When MCCR1[PCKEN] is set, the MPC105 checks memory parity on every memory data read cycle and generates the parity data on every memory data write cycle that emanates from the MPC105 but does not check the parity data. The 60x processor generates parity on 60x writes to system memory. When a read parity error occurs, ErrDR1[2] is set.

9.3.2.2 System Memory Select Error

A memory select error occurs when a system memory transaction address falls outside of the physical memory boundaries. When a memory select error occurs, ErrDR1[5] is set.

If a write transaction causes the memory select error, the write data is simply ignored. If a read transaction causes the memory select error, meaningless data is returned. No $\overline{\text{RAS}}$ signals are asserted in either case.

9.3.2.3 L2 Cache Read Data Parity Error

When ErrEnR2[4] is set, the MPC105 checks L2 cache parity on every L2 cache data read cycle and generates the parity data on every L2 cache data write cycle that emanates from the MPC105 but does not check the parity data. The 60x processor generates parity on 60x writes to the L2 cache. When an L2 cache read parity error occurs, ErrDR2[4] is set.

9.3.3 PCI Interface

The MPC105 supports the error detection and reporting mechanism specified in the *PCI Local Bus Specification, Revision 2.0*. The MPC105 keeps error information and sets the appropriate error flags when a PCI error occurs (provided the corresponding enable bit is set), independent of whether the PCI command register is programmed to respond to or detect the specific error.

In cases of PCI errors, ErrDR1[3] is set to indicate that the error is due to a PCI transaction. In most cases, ErrDR2[7] is cleared to indicate that the error address in the 60x/PCI error address register is valid. In these cases, the error address is the address as seen by the PCI bus, not the 60x bus address.

If NMI is asserted, the MPC105 cannot provide the error address and the corresponding bus error status. In such cases, ErrDR2[7] is set to indicate that the error address in the 60x/PCI error address register is not valid.

9.3.3.1 Address Parity Error

If the MPC105 is acting as a PCI master, and the target detects and reports (by asserting $\overline{\text{SERR}}$) a PCI address parity error, then the MPC105 sets ErrDR1[7] and sets the detected parity error bit (bit 15) in the PCI status register. This is independent of the settings in the PCI command register.

If the MPC105 is acting as a PCI target and detects a PCI address parity error, the PCI interface of the MPC105 sets the status bit in the PCI status register (bit 15). If bits 8 and 6 of the PCI command register are set, the MPC105 reports the address parity error by asserting $\overline{\text{SERR}}$ to the master (two clocks after the address phase) and sets bit 14 of the PCI status register. Also the MPC105 will target-abort and set bit 11 of the PCI status register. Also, if PICR1[MCP_EN] is set, the MPC105 reports the error to the 60x processor by asserting $\overline{\text{MCP}}$.

Note that for the MPC105 to recognize the assertion of $\overline{\text{SERR}}$ by another PCI agent, bit 5 (RX_SERR_EN) of the alternate OS-visible parameter register 1 must be set.

9.3.3.2 Data Parity Error

If the MPC105 is acting as a PCI master and a data parity error occurs, the MPC105 sets bit 15 of the PCI status register. This is independent of the settings in the PCI command register.

If the PCI command register of the MPC105 is programmed to respond to parity errors (bit 6 of the PCI command register is set) and a data parity error is detected or signaled during a PCI bus transaction, the MPC105 sets the appropriate bits in the PCI status register (bit 15 is set, and possibly bit 8 is set, as described in the following paragraphs).

If a data parity error is detected by the MPC105 acting as the master (for example, during a 60x processor-read-from-PCI transaction), and if bit 6 of the PCI command register is set, the MPC105 reports the error to the PCI target by asserting $\overline{\text{PERR}}$ and by setting bit 8 of the status register and tries to complete the transaction, if possible. Also, if PICR1[MCP_EN] is set, the MPC105 asserts $\overline{\text{MCP}}$ to report the error to the 60x processor. These actions also occur if the MPC105 is the master and detects the assertion of $\overline{\text{PERR}}$ by the target (for a write).

If the MPC105 is acting as a PCI target when the data parity error occurs (on a write), the MPC105 asserts $\overline{\text{PERR}}$, sets ErrDR1[6] (PCI target $\overline{\text{PERR}}$), and signals a target-abort (if it hasn't already completed the transfer). If the data had been transferred, the MPC105 completes the operation but discards the data. Also, if PICR1[MCP_EN] is set, the MPC105 asserts $\overline{\text{MCP}}$ to report the error to the 60x processor.

In the case that $\overline{\text{PERR}}$ is asserted by the master during a memory read, the MPC105 target-aborts causing the master to discontinue. In this case, the address of the transfer will be logged in the error address register and $\overline{\text{MCP}}$ is optionally asserted. If an address is out of range, the MPC105 sends a target-abort command.

9.3.3.3 Master-Abort Transaction Termination

If the MPC105, acting as a master, initiates a PCI bus transaction (excluding special-cycle and configuration transactions), but there is no response from any PCI agent ($\overline{\text{DEVSEL}}$ has not been asserted within five PCI bus clocks from the start of the address phase), the MPC105 terminates the transaction with a master-abort and sets the master-abort flag (bit 13) in the PCI status register.

If ErrEnR1[1] and PICR1[MCP_EN] are both set and the MPC105 terminates a transaction with a master-abort, the MPC105 reports the error to the 60x processor by asserting $\overline{\text{MCP}}$.

9.3.3.4 Cases of Target-Abort Signaled by MPC105

In summary, the MPC105 signals a target-abort command for the following cases if the parity error response bit (but 6) of the PCI command register is set:

- Memory out of range on a read transaction for MPC105 acting as a target
- Memory out of range on a posted write when the MPC105 is currently acting as a target
- Memory parity error on a read transaction for MPC105 acting as a target
- The $\overline{\text{SERR}}$ signal asserted while MPC105 acting as a target
- Data or address parity error detected while MPC105 acting as a target
- The $\overline{\text{PERR}}$ signal asserted on a read transaction for MPC105 acting as a target

9.3.3.5 Received Target-Abort Error

If a PCI transaction initiated by the MPC105 is terminated by target-abort, the received target-abort flag (bit 12) of the PCI status register is set. If ErrEnR1[7] and PICR1[MCP_EN] are both set and the MPC105 receives a target-abort, the MPC105 reports the error to the 60x processor by asserting $\overline{\text{MCP}}$.

Note that any data transferred in a target-aborted transaction may be corrupt.

9.3.3.6 NMI (Nonmaskable Interrupt)

If PICR1[MCP_EN] is set and a PCI agent (typically the system interrupt controller) asserts NMI to the MPC105, the MPC105 reports the error to the 60x processor by asserting $\overline{\text{MCP}}$.

When the NMI signal is asserted, no error flags are set in the status registers of the MPC105. The agent that drives NMI should provide the error flag for the system and the mechanism to reset that error flag. The NMI signal should then remain asserted until the error flag is cleared.

9.4 Interrupt Latencies

Latencies for taking various interrupts are variable based on the state of the MPC105 when the conditions to produce an interrupt occur. The minimum latency is one cycle. In this case, the interrupt is signaled in the cycle following the appearance of the interrupt-producing conditions.

9.5 Example Signal Connections

This section provides two examples of connecting the interrupt signals between the 60x processor, the MPC105, and an interrupt controller on the PCI bus. Typically the interrupt controller is integrated into the PCI-to-ISA bridge. Figure 9-1 shows a 603- or 604-based system design. Figure 9-2 shows a 601-based system design.

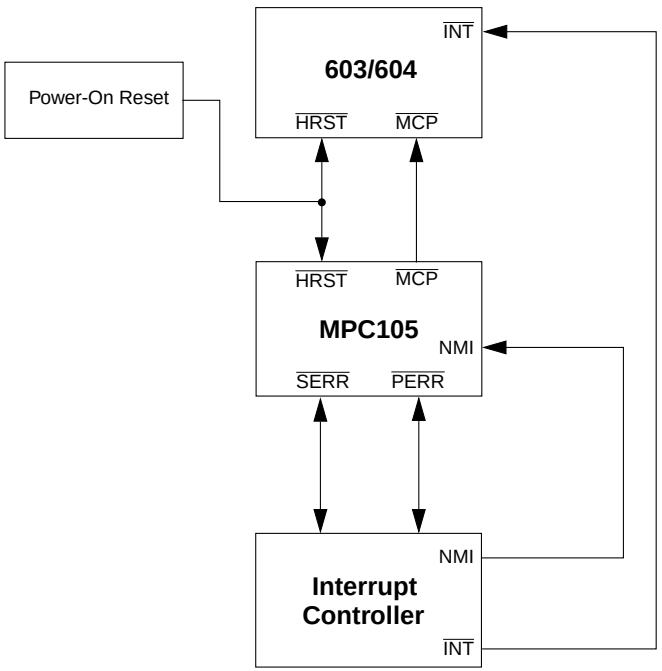


Figure 9-1. Example Interrupt Signal Configuration—603-/604-Based System

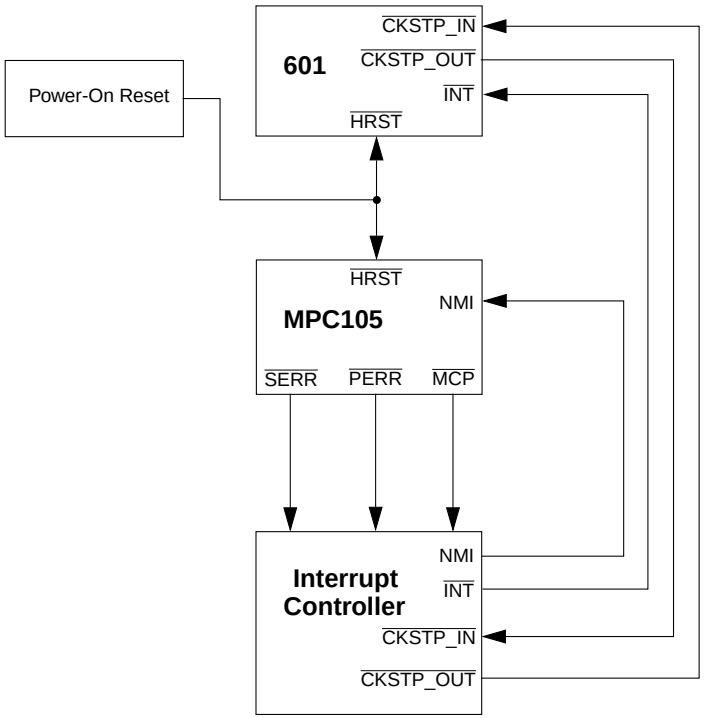


Figure 9-2. Example Interrupt Signal Configuration—601-Based System

Appendix A
Power Management

The MPC105 provides the system designer hardware resources to flexibly reduce system power consumption through the use of software and system hardware power control mechanisms. This appendix describes the hardware support provided by the MPC105 for power management.

A.1 MPC105 Power Modes

The MPC105 implements four levels of power reduction—doze, nap, sleep, and suspend, with power consumption reduced with each step from doze to suspend. The doze, nap, and sleep modes are entered through software setting the required configuration register bit in the power management configuration register (PMCR). For more information about this register, see section 3.2.4, "Power Management Configuration Register (PMCR)." The suspend mode is entered by the assertion of the $\overline{\text{SUSPEND}}$ signal, as described in section 2.2.4, "PCI Interface Signals." All of the power management modes are enabled by the configuration of the global power management bit, PMCR[PM].

A.1.1 MPC105 Power Mode Transition

While the doze, nap, and sleep modes are enabled by setting the corresponding bits in the PMCR, in the case of the nap and sleep modes the power management mode is entered upon the assertion of the $\overline{\text{QREQ}}$ signal. The MPC105 responds by entering the power management mode selected, and asserts $\overline{\text{QACK}}$ to signal to the processor that the power management mode has been entered. The doze mode is entered directly by configuring the doze bit in the PMCR, and does not require the assertion of $\overline{\text{QREQ}}$.

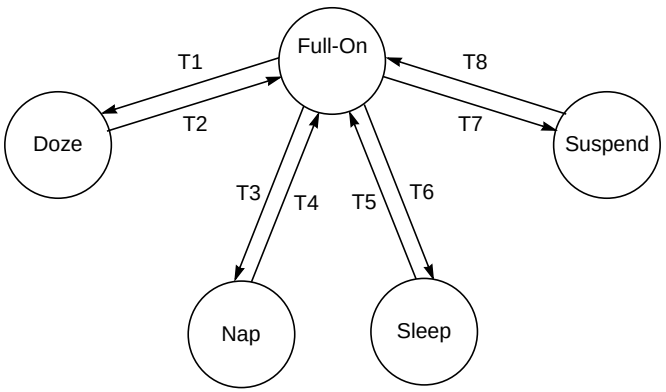
The configuration of the MPC105 and the processor signals asserted differ depending on which processor (601, 603, or 604) the MPC105 is connected to. The response of the MPC105 is configured through the setting of the processor type bits in PICR1[PROC_TYPE].

In a system designed using the 601, the MPC105 can be configured to ignore the state of the $\overline{\text{QREQ}}$ signal, and enters the nap or sleep mode directly upon the setting of the required PMCR bits. This is controlled through PMCR[601_NEED_QREQ], which when cleared to 0 allows the immediate invocation of the nap or sleep mode without assertion of $\overline{\text{QREQ}}$, and when set to 1 requires the assertion of $\overline{\text{QREQ}}$ by system power control logic to enter the desired power management mode.

In systems designed using the 603, the power control signals $\overline{\text{QREQ}}$ and $\overline{\text{QACK}}$ are connected to the corresponding signals on the MPC105, and the doze, nap, or sleep mode is entered following the configuration of the required bit in the PMCR, and the assertion of $\overline{\text{QREQ}}$ (in nap and sleep modes) to the MPC105 by the 603.

In systems designed using the 604, the MPC105's $\overline{\text{QREQ}}$ signal is connected to the 604's HALT signal, and the MPC105's $\overline{\text{QACK}}$ signal is connected to the 604's RUN signal, and the doze, nap, or sleep mode is entered following the configuration of the required bit in the PMCR, and the assertion of $\overline{\text{QREQ}}$ (in nap and sleep modes) to the MPC105 by the 604. Configuring the processor type bits in PICR1[PROC_TYPE] for the 604 causes the signal levels sampled and driven by the MPC105's $\overline{\text{QREQ}}$ and $\overline{\text{QACK}}$ signals to correspond to the levels required by the 604's RUN and HALT signals.

Figure A-1 shows the five power modes of the MPC105, and the conditions required for entering and exiting those modes.



T1: PMCR[DOZE]) =1 & PMCR[PM] =1
T2: hard reset, BRx = 0, PCI address hit, NMI
T3: PMCR[NAP]=1 & PMCR[PM] =1 & $\overline{\text{QREQ}}$ = 0 (or HALT = 1 in 604 system)
T4: hard reset, BRx = 0, PCI address hit, NMI
T5: PMCR[SLEEP]=1 & PMCR[PM] = 1 & $\overline{\text{QREQ}}$ = 0 (or HALT = 1 in 604 system)
T6: hard reset, BRx = 0, NMI
T7: $\overline{\text{suspend}}$ = 0 & PMCR[PM] =1
T8: $\overline{\text{suspend}}$ = 1

Figure A-1. MPC105 Power Modes

The following sections provide a detailed description of the power modes of the MPC105.

A.1.2 Full-On Mode

This is the default power mode of the MPC105. In this mode, the MPC105 is fully powered and the internal functional units are operating at full clock speed.

A.1.3 Doze Mode

In this power management mode, all of the MPC105's functional units are disabled except for PCI address decoding, system RAM refresh logic, processor bus request monitoring (through $\overline{BR0}$ and $\overline{BR1}$), and NMI signal monitoring. Once the doze power management mode is entered, a hard reset, a PCI transaction referenced to the system memory, a bus request from $\overline{BR0}$ or $\overline{BR1}$, or assertion of NMI (with PICR1[MCP_EN] set to 1), brings the MPC105 out of the doze mode and into the full-on mode.

After the system request has been serviced, the system returns to the doze mode if neither PMCR[DOZE] nor the PMCR(PM) has been cleared and there are no further pending service requests.

In doze mode, the PLL is required to be running and locked to SYSCLK. The transition to the full-on mode will take no more than a few processor cycles. The MPC105's doze mode is totally independent of the power saving mode of the CPU.

A.1.4 Nap Mode

Additional power savings can be achieved through the nap mode. When invoking the MPC105's nap mode, both the MPC105 and the processor should be programmed to enable the nap mode. The processor may also be programmed to enter sleep mode while the MPC105 enters nap mode.

As in doze mode, all the MPC105's functional units are disabled except for the PCI address decoding, system RAM refresh logic, processor bus request monitoring (through $\overline{BR0}$ and $\overline{BR1}$), and NMI signal monitoring. Once the nap mode is entered, a hard reset, a PCI transaction referenced to the system memory, a bus request from $\overline{BR0}$, a bus request from $\overline{BR1}$ (in a multiprocessor system with PMCR[BR1_WAKE] set to 1), or an asserted NMI (PICR1[MCP_EN] set to 1) will bring the MPC105 out of the nap mode.

In nap mode, the PLL is required to be running and locked to SYSCLK. The transition to the full-on mode will take no more than a few processor cycles.

When the MPC105 is awakened by an access other than a PCI bus initiated transaction, the transaction will be serviced and PMCR[PM] will be cleared. This means that the MPC105 will not automatically re-enter the nap mode. For PCI bus initiated transactions, PMCR[PM] is not be cleared, and the MPC105 will return to nap mode after the transaction has been serviced.

While the MPC105 is servicing a PCI bus transaction in systems using a 603, if the 603 is still in a power management mode the 603 will not respond to any snoop cycles. Software should therefore flush the 603's L1 cache before allowing the system to enter the nap mode if the system allows a PCI bus access to wake up the MPC105. However, in systems using the 604, the 604 can be forced to respond to a snoop cycle if the RUN signal (connected to the $\overline{\text{QACK}}$ signal from the MPC105) is asserted. This response by the 604 is enabled by clearing PMCR[NO_604_RUN] to 0. If the MPC105 is configured to allow snoop responses by the 604, there is no need to flush the L1 cache before the 604 enters the nap mode.

Before entering the nap mode, $\overline{\text{QREQ}}$ from 603 or HALT from 604 will be sampled active by the MPC105, which will then respond with a $\overline{\text{QACK}}$ signal when it is ready to nap, thereby allowing the processor to enter either the nap or sleep mode.

A.1.5 Sleep Mode

Sleep mode provides additional power savings when compared to nap mode. As in nap mode, both MPC105 and the processor should be configured to enable the sleep mode (although the processor may optionally be configured for nap mode while the MPC105 is in sleep mode). While the MPC105 is in sleep mode, no functional units are operating except the system RAM refresh logic (optional), processor bus request monitoring (through $\overline{\text{BR0}}$ or $\overline{\text{BR1}}$), and NMI signal monitoring. A hard reset, a bus request from $\overline{\text{BR0}}$, a bus request from $\overline{\text{BR1}}$ (in a multiprocessor system with PMCR[BR1_WAKE] set to 1), or assertion of NMI (with PICR1[MCP_EN] set to 1) will wake the MPC105 from the sleep mode. The PMCR[PM] bit will always be cleared after the MPC105 is awakened from the sleep mode.

The PLL and SYSCLK input may be disabled by an external power management controller (PMC) for additional power savings. The PLL can be disabled by setting the PLL_CFG[0–3] signals in the PLL bypass mode. When recovering from sleep mode, the external PMC has to re-enable the PLL and SYSCLK first, and then wake up the system after 100 microseconds of PLL re-lock time.

In sleep mode, the system can retain the system memory content through the use of three different methods. The first method is the normal CBR refresh which is supported by every system. The second method is to enable the self refresh mode of the system memory. This can be supported only if the system memory is capable of supporting the self refresh mode. The final method is supported by the operating system by copying all the system memory data to a hard disk. In this case, there is no need to continue the memory refresh operation.

The programming options for the three memory retention methods are defined by the configuration of PMCR[LP_REF_EN] and MCCR8[SREN]. If the LP_REF_EN bit is cleared to 0, there will be no memory refresh operation when the MPC105 is in the sleep or suspend mode. If PMCR[LP_REF_EN] is set to 1, memory refresh will be carried out even when the MPC105 is in a low-power mode. In this case, MCCR8[SREN] is used to

determine whether the refresh is a self refresh (MCCR8[SREN] set to 1) or a CBR refresh (MCCR8[SREN] cleared to 0).

When the MPC105 is in the sleep mode using CBR refresh and keeping the PLL in locked operation, the wake up latency should be comparable to nap mode. However, additional wake up latency will be needed if the system uses the self refresh mode and/or turns off the PLL during sleep mode operation.

Before entering the sleep mode, $\overline{\text{QREQ}}$ from 603 or HALT from 604 should be sampled active. The MPC105 will then respond with a $\overline{\text{QACK}}$ signal when it is ready to enter the sleep mode, thereby allowing the processor to enter into either the nap or sleep mode.

Turning off the PLL and/or external clock during sleep mode requires waiting until the assertion of the $\overline{\text{QACK}}$ signal. The external PMC chip should trap all the wake up events so that it can turn on the PLL (observing the recommended PLL relock time) and/or the external clock source before forwarding the wake up event to the MPC105.

A.1.6 Suspend Mode

Suspend mode provides the greatest reduction of power consumption. It is activated through the assertion of the $\overline{\text{SUSPEND}}$ signal, which is driven by an external I/O device (in most cases an external (system level) power management controller). In suspend mode, no functional units are operating except the system RAM refresh logic (optional) and the internal logic monitoring the $\overline{\text{SUSPEND}}$ signal. The MPC105 will remain in the suspend mode until the $\overline{\text{SUSPEND}}$ signal is negated.

The PLL and SYSCLK input may be disabled by an external power management controller (PMC) for additional power savings. The PLL can be disabled by setting the PLL_CFG[0–3] pins into the PLL bypass mode. When recovering from suspend mode, the external PMC has to re-enable the PLL and SYSCLK first, and then wake up the system after the PLL has had time to relock (100 microseconds).

In suspend mode, the system can retain the contents of system memory through the use of three different methods. The first method is the low-frequency refresh (RTC refresh) which can be supplied very easily by most systems. A low frequency clock signal is supplied by the system to the real time clock (RTC) input of the MPC105. The second method is to enable the self refresh mode of the system memory. This can be supported only if the system memory is capable of supporting the self refresh mode. The third method is supported by the operating system by copying all the system memory data to the hard disk. In this case, there is no need to continue the memory refresh operation.

The programming options for the three memory retention methods is defined by the configuration of PMCR[LP_REF_EN] and MCCR8[SREN]. If PMCR[LP_REF_EN] is cleared to 0, there will be no memory refresh operation when the MPC105 is in suspend mode. If PMCR[LP_REF_EN] is set to 1, memory refresh will be carried out even when the MPC105 is in suspend mode.

In this case, MCCR8[SREN] will be used to determine whether the refresh is a self refresh (MCCR8[SREN] set to 1) or a low-frequency refresh (MCCR8[SREN] cleared to 0). Note that if the memory system is configured for SDRAM, it will be treated as no refresh required, and no low-frequency refresh is supported.

In suspend mode, all bidirectional and output signals (except the memory refresh-related signals if RTC refresh is being used) will be at high impedance and all input signals (including $\overline{\text{HRST}}$), with the exception of the PLL configuration signals, will be ignored.

After the assertion of the $\overline{\text{SUSPEND}}$ signal, the system should not turn off the PLL and/or the external clock source for at least 60 microseconds (two RTC clock periods). Before the de-assertion of the $\overline{\text{SUSPEND}}$ signal, the system should allow sufficient time for the PLL to stabilize.

A.2 MPC105 Power Management Support

The MPC105 provides hardware for the support of power management activities that is accessible to software and external system-level power management controllers. The fully static design allows internal logic states to be preserved during all power saving operations. System software is expected to handle the majority of power management tasks through access to the PMCR. The following sections provide a description of the power management features and capabilities provided by the MPC105.

A.2.1 Power Management Configuration Register

The PMCR provides software access to the power management modes, enables, and configurations for different processors. Refer to section 3.2.4, “Power Management Configuration Register (PMCR),” for a detailed description of the PMCR.

A.2.2 Clock Configuration

In doze and nap modes, the PLL must be running and locked to SYSCLK in order to provide clocks to the internal logic units that need to be awake, and to minimize the transition time required in coming out of a power saving mode to the full-on mode. The power mode transition occurs with the assumption that the PLL is locked with SYSCLK. The electrical characteristics of the SYSCLK signal and the PLL configuration should remain the same whether the MPC105 is in the full-on mode, or in doze or nap mode. In sleep or suspend mode, the external PMC (if it exists) may disable the PLL and the SYSCLK input for further power savings. However, it is the external PMC’s responsibility to guarantee that there is sufficient relock time for the PLL of the MPC105 before MPC105 is awakened by any event. Doze and nap modes are intended to be used dynamically due to their fast recovery time; sleep and suspend modes are intended for longer periods of power savings with the PLL and SYSCLK off.

A.2.3 PCI Address Bus Decoding

PCI address bus decoding is enabled while the MPC105 is in the doze or nap mode. A PCI transaction to system memory awakens the MPC105 from the doze or nap power saving mode.

After servicing the PCI transaction, the MPC105 returns to the previous power saving mode (doze or nap) if there are no additional PCI bus service requests. In systems using a 603, the L1 cache should be flushed prior to entering the nap mode. Systems designed using the 601 or 604 are not required to flush their L1 caches prior to entering the nap mode.

A.2.4 PCI Bus Special-Cycle Operations

Before the MPC105 enters the nap or sleep mode, it will broadcast the “halt” or “shutdown” message over the PCI bus by means of special bus cycle. See Section 7.4.6.2, “Special Cycle,” for a description of PCI special-cycle operations.

In nap mode, if PMCR[NO_NAP_MSG] is cleared to 0, the MPC105 broadcasts the “halt” message over the PCI bus. If PMCR[NO_NAP_MSG] is set to 1, the MPC105 does not broadcast any message to the PCI bus.

In sleep mode, if PMCR[NO_SLEEP_MSG] is cleared to 0, the MPC105 broadcasts either the “halt” or “shutdown” message over the PCI bus depending on whether PMCR[SLEEP_MSG_TYPE] is cleared or set. If PMCR[NO_SLEEP_MSG] is set to 1, the MPC105 does not broadcast any message to the PCI bus and the configuration of PMCR[SLEEP_MSG_TYPE] is ignored.

A.2.5 Processor Bus Request Monitoring

In doze, nap, and sleep modes, the MPC105 monitors the $\overline{\text{BR0}}$ signal. When $\overline{\text{BR0}}$ is asserted, (for example, due to the processor’s time base interrupt service routine), the MPC105 exits its power saving mode and returns to the full-on mode to service the request.

Additionally, in a multiprocessor system, $\overline{\text{BR1}}$ can be used to awaken the MPC105. In nap or sleep mode, $\overline{\text{BR1}}$ is treated as a wake up event if PMCR[BR1_WAKE] is set to 1. In doze mode, it is unconditional, and does not depend upon the condition of the bit in PMCR[BR1_WAKE].

A.2.6 Memory Refresh Operations in Sleep/Suspend Mode

In sleep or suspend mode, all functional units, including the system memory refresh logic, will not be operating. The system memory contents can be maintained either by enabling the memory’s self-refresh mode or by having the system software copy all the memory contents to a hard disk before the MPC105 enters the sleep or suspend mode. However, if the memory does not have self-refresh capability or the system software has not copied the memory contents to the hard disk, the refresh logic of the MPC105 can continue to operate even if in sleep or suspend mode. This is configured through PMCR[LP_REF_EN], which when set to 1 allows the refresh logic to continue to perform refresh cycles for system

memory when the MPC105 is in the sleep or suspend mode. If PMCR[LP_REF_EN] is cleared to 0, memory refresh operations will cease when the MPC105 enters the sleep or suspend mode. For additional detail on memory refresh operations, refer to Section 6.3.4.2, “DRAM Refresh and Power Saving Modes,” and Section 6.4.5.2, “SDRAM Refresh and Power Saving Modes.”

A.2.7 Device Drivers

Since operating systems service I/O requests by system calls to the device drivers, the device drivers must be modified for power management. When a device driver is called to reduce the power of a device, it needs to be able to check the power mode of the device, save the device configuration parameters, and put the device into a power saving mode. Furthermore, every time the device driver is called it needs to check the power status of the device, and if the device is in a power saving mode, restore the device to the full-on mode.

Appendix B
Bit and Byte Ordering

The MPC105 supports both little-endian and big-endian formatted data on the PCI bus. This appendix provides examples of the little- and big-endian modes of operation. PICR1[LE_MODE] controls the endian mode of the MPC105. LE_MODE is also accessible from the external configuration register at 0x092.

When designing little-endian or bi-endian systems using the MPC105, system designers and programmers must consider the following:

- The PCI bus uses a little-endian bit format (the most significant bit is 31), while the 60x bus uses a big-endian bit format (the most significant bit is 0). Thus, PCI address bit AD31 equates to the 60x address bit A0, while PCI address bit AD0 equates to the 60x address bit A31.
- For data comprised of more than 1 byte, the endian mode affects the byte ordering. For little-endian data, the least-significant byte is stored at the lowest (or starting) address while the most-significant byte is stored at the highest (or ending) address. For big-endian data, the most-significant byte is stored at the lowest (or starting) address while the least-significant byte is stored at the highest (or ending) address.
- For 60x processors, the conversion to little-endian mode does not occur on the data bus. The bus interface unit (BIU) of the 60x processor uses a technique called *munging* to reverse the address order of every 8 bytes stored to memory. See Section 3.2, “Data Organization in Memory and Data Transfers” in *PowerPC Microprocessor Family: The Programming Environments* (Motorola Order Number MPCFPE/AD), for more information. External to the processor all the byte lanes must be reversed (MSB to LSB, etc.) and the addresses must be unmunged. The unmunging/byte lane reversing mechanism can either be between the processor and system memory or between the PCI bus and system memory. The MPC105 unmunges the address and reverses the byte lanes between the PCI bus and system memory.

B.1 Big-Endian Mode

When the 60x processor is running in big-endian mode, no address modification is performed. The MPC105 reverses the byte lanes to PCI to make the PCI memory space and PCI I/O space appear big-endian.

Figure B-1 shows a 4-byte write to PCI memory space in big-endian mode.

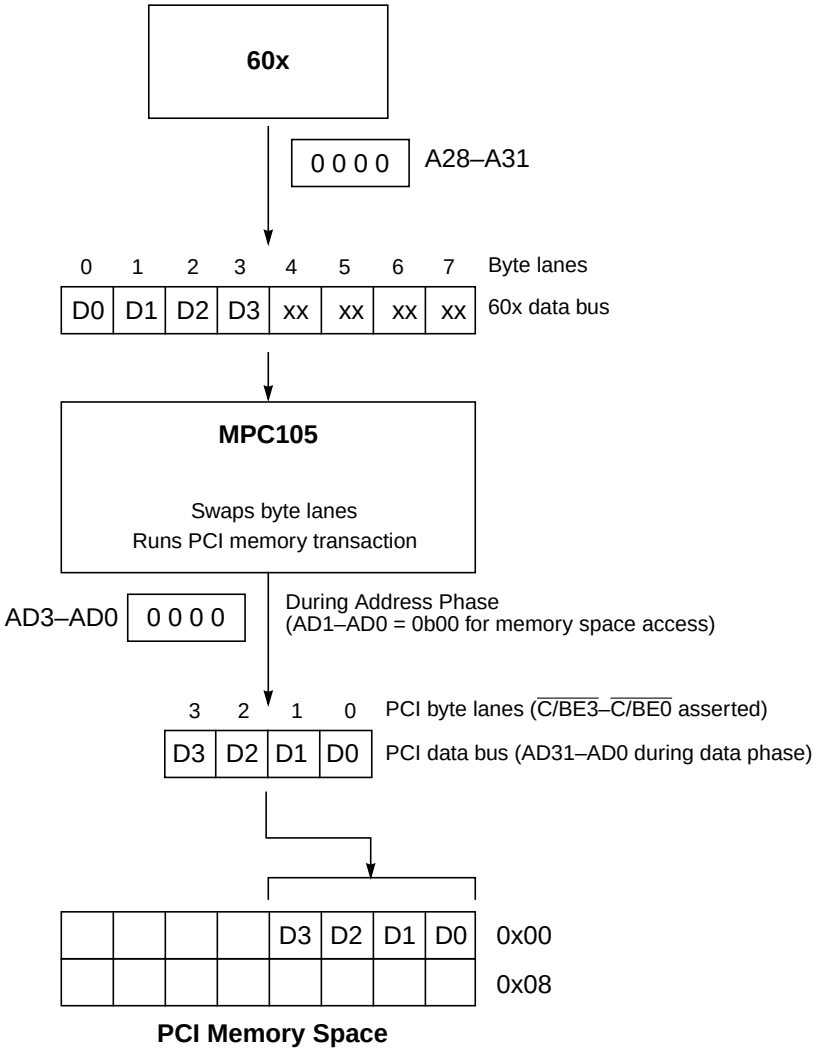


Figure B-1. Four-Byte Transfer to PCI Memory Space—Big-Endian Mode

Note that the most significant byte, D0, is placed on the least significant byte lane on the PCI bus. This occurs so that D0 appears at address 0x*nnnn_nn*00 and not at address 0x*nnnn_nn*03 in the PCI space.

For example, starting with a program that wishes to do the following:

```
store string (“hello, world”) at 0x000
store pointer (0xFEDCBA98) at 0x010
store halfword (0d1234) at 0x00E
store byte (0x55) at 0x00D
```

If the data is stored into system memory, it appears as shown in Figure B-2.

	MSB A0-7	A8-15	A16-23	A24-31	A32-39	A40-47	A48-55	LSB A56-63
0x000	‘h’	‘e’	‘l’	‘l’	‘o’	‘,’	‘ ’	‘w’
0x008	‘o’	‘r’	‘l’	‘d’		0x55	12	34
0x010	0xFE	0xDC	0xBA	0x98				

Figure B-2. Big-Endian Memory Image in System Memory

Note that the stored data has big-endian ordering. The “h” is at address 0x000.

But, if the data is stored to the PCI memory space, the MPC105 sends the addresses out as is, but the data is put on different byte lanes, but at the same byte address, as it goes out to the PCI bus. The data appears in PCI memory space as shown in Figure B-3.

MSB AD(31-24)	AD(23-16)	AD(15-8)	LSB AD(7-0)	
‘l’	‘l’	‘e’	‘h’	0x000
‘w’	‘ ’	‘,’	‘o’	0x004
‘d’	‘l’	‘r’	‘o’	0x008
34	12	0x55	0x00	0x00C
0x98	0xBA	0xDC	0xFE	0x010

Figure B-3. Big-Endian Memory Image in Big-Endian PCI Memory Space

Note that the string “hello, world” starts at address 0x000. The other data are stored to the desired locations with big-endian byte ordering.

B.2 Little-Endian Mode

When the 60x processor is running in little-endian mode, its internal BIU performs a modification on each address. This address modification is called munging. The 60x munges the address by exclusive-ORing the three low-order address bits with a three-bit value that depends on the length of the operand (1, 2, 4 or 8 bytes), as shown in Table B-1.

Table B-1. Address Modification for Individual Aligned Scalars

Data Length (in Bytes)	Address Modification
8	No change
4	XOR with 0b100
2	XOR with 0b110
1	XOR with 0b111

Note that these are the only legal data lengths supported by the 60x processor in little-endian mode.

The munged address is used by the memory interface of the MPC105 to access system memory. For PCI accesses, the MPC105 unmunges the address to its original value and the byte lanes are reversed.

Starting with the same program as before:

```
store string (“hello, world”) at 0x000
store pointer (0xFEDCBA98) at 0x010
store halfword (0d1234) at 0x00E
store byte (0x55) at 0x00D
```

If the data is stored to system memory, the MPC105 stores the data to the 60x-munged addresses as shown in Table B-4.

	MSB A0-7	A8-15	A16-23	A24-31	A32-39	A40-47	A48-55	LSB A56-63
0x000	‘w’	‘ ’	‘,’	‘o’	‘l’	‘l’	‘e’	‘h’
0x008	12	34	0x55		‘d’	‘l’	‘r’	‘o’
0x010					0xFE	0xDC	0xBA	0x98

Figure B-4. Munged Memory Image in System Memory

Note how munging has changed the addresses of the data. The ‘h’ is now at address 0x007.

If the data is stored to the PCI memory space, the MPC105 unmunges the addresses before sending them out to the PCI bus. The data is stored to little-endian PCI memory space as shown in Figure B-5.

MSB AD(31-24)	AD(23-16)	AD(15-8)	LSB AD(7-0)	
‘l’	‘l’	‘e’	‘h’	0x000
‘w’	‘ ’	‘,’	‘o’	0x004
‘d’	‘l’	‘r’	‘o’	0x008
12	34	0x55	0x00	0x00C
0xFE	0xDC	0xBA	0x98	0x010

Figure B-5. Little-Endian Memory Image in Little-Endian PCI Memory Space

Note that the string “hello, world” starts at address 0x000. The other data are stored to the desired locations with little-endian byte ordering.

Figure B-6 through Figure B-11 show the munging/unmunging process for transfers to the PCI memory space and to the PCI I/O space.

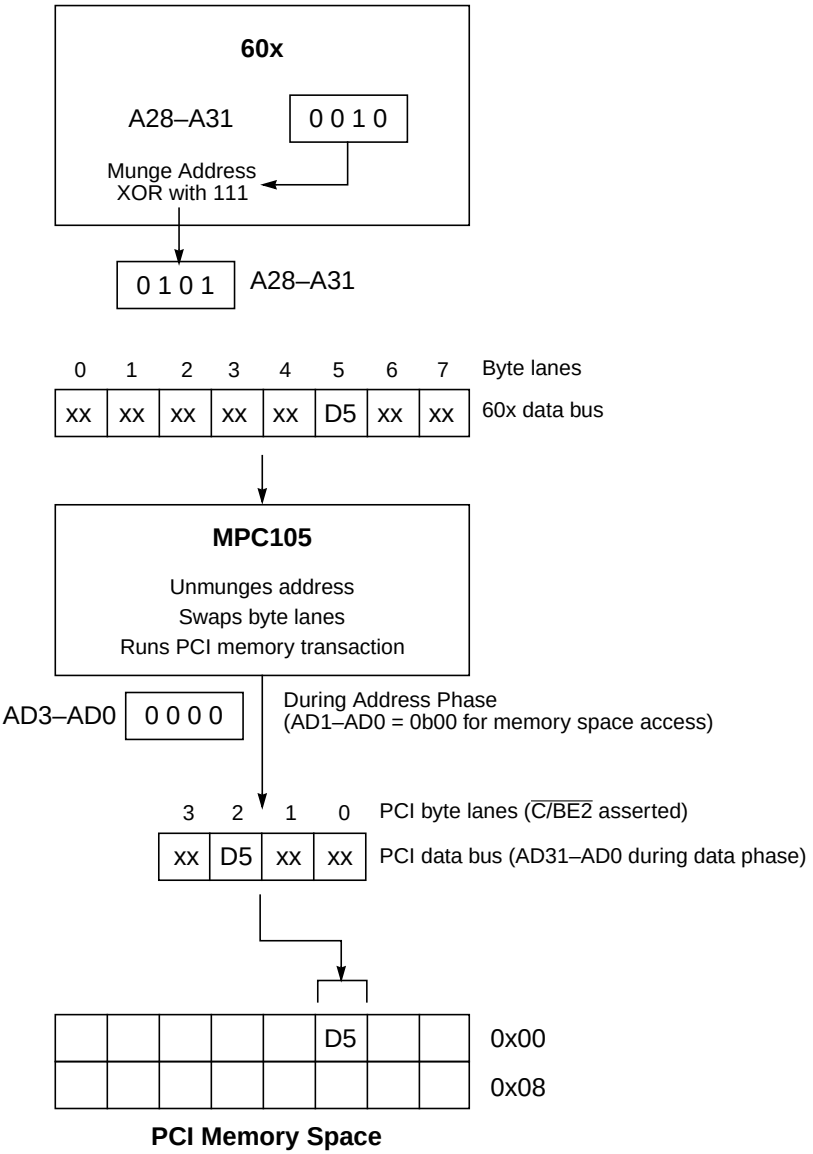


Figure B-6. One-Byte Transfer to PCI Memory Space—Little Endian Mode

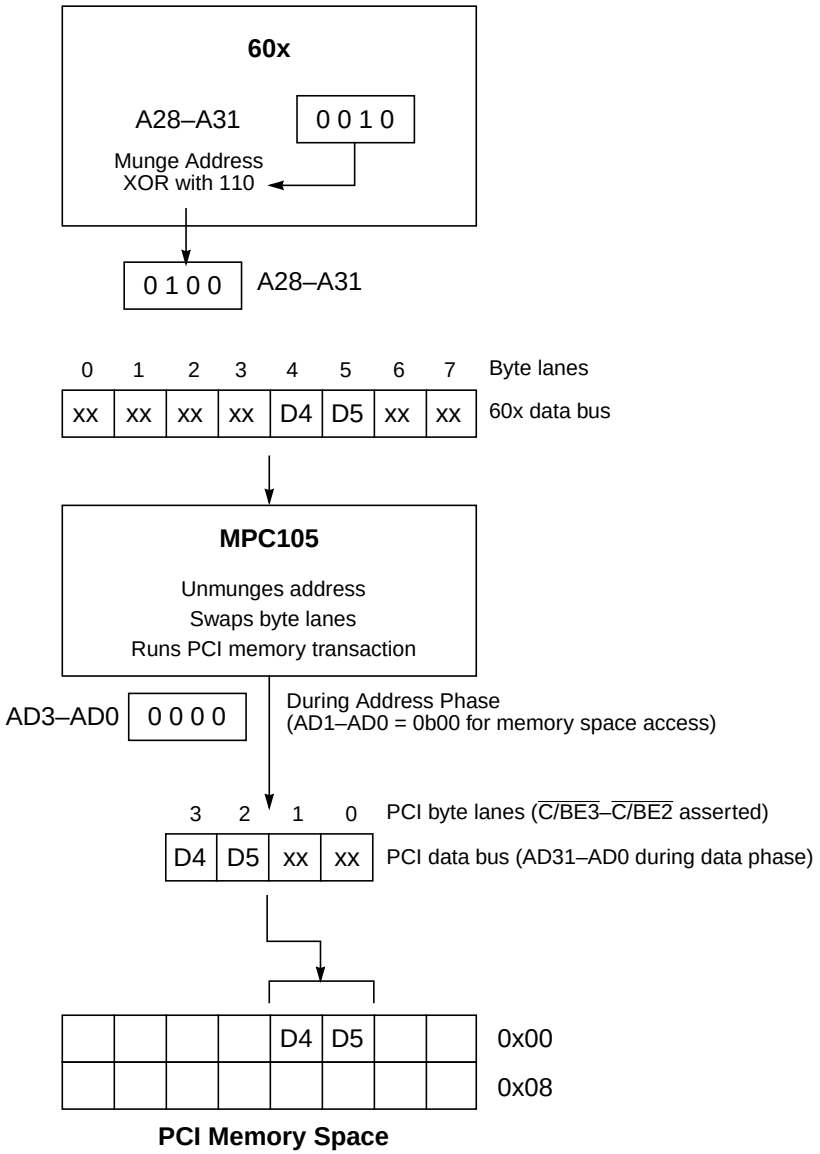


Figure B-7. Two-Byte Transfer to PCI Memory Space—Little Endian Mode

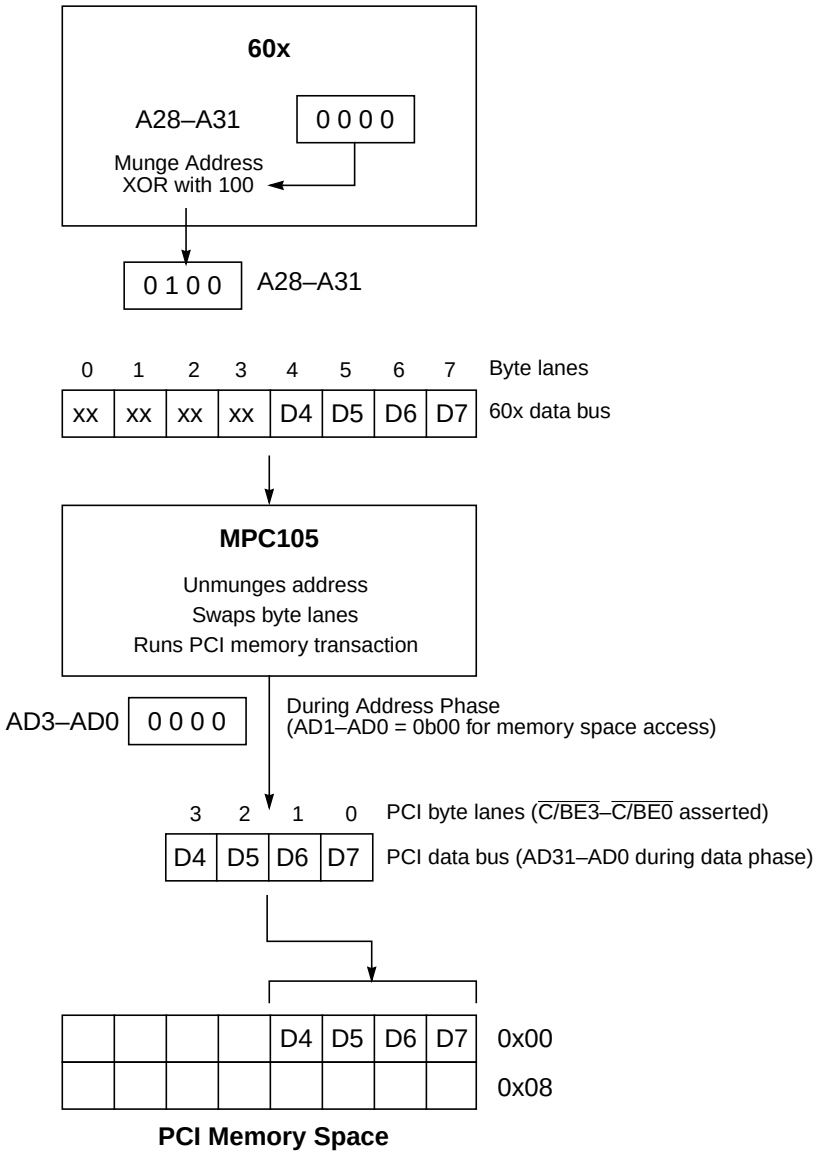


Figure B-8. Four-Byte Transfer to PCI Memory Space—Little Endian Mode

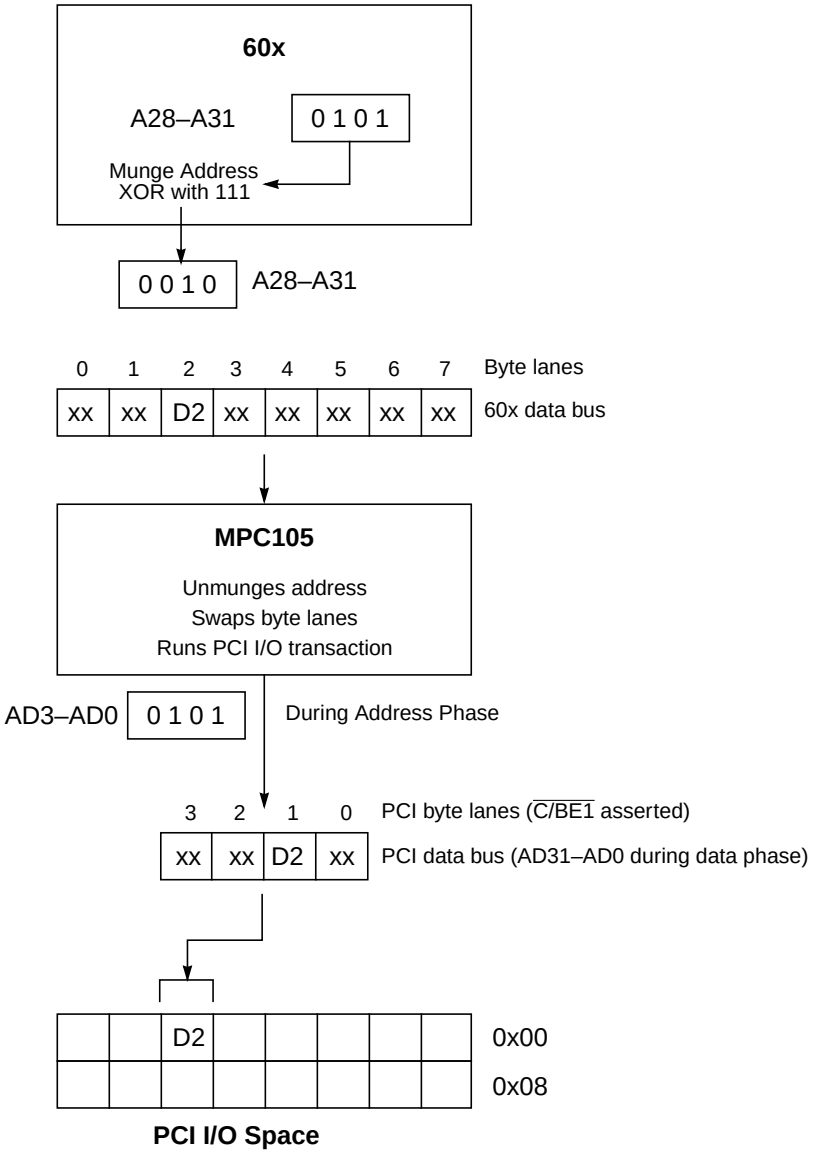


Figure B-9. One-Byte Transfer to PCI I/O Space—Little Endian Mode

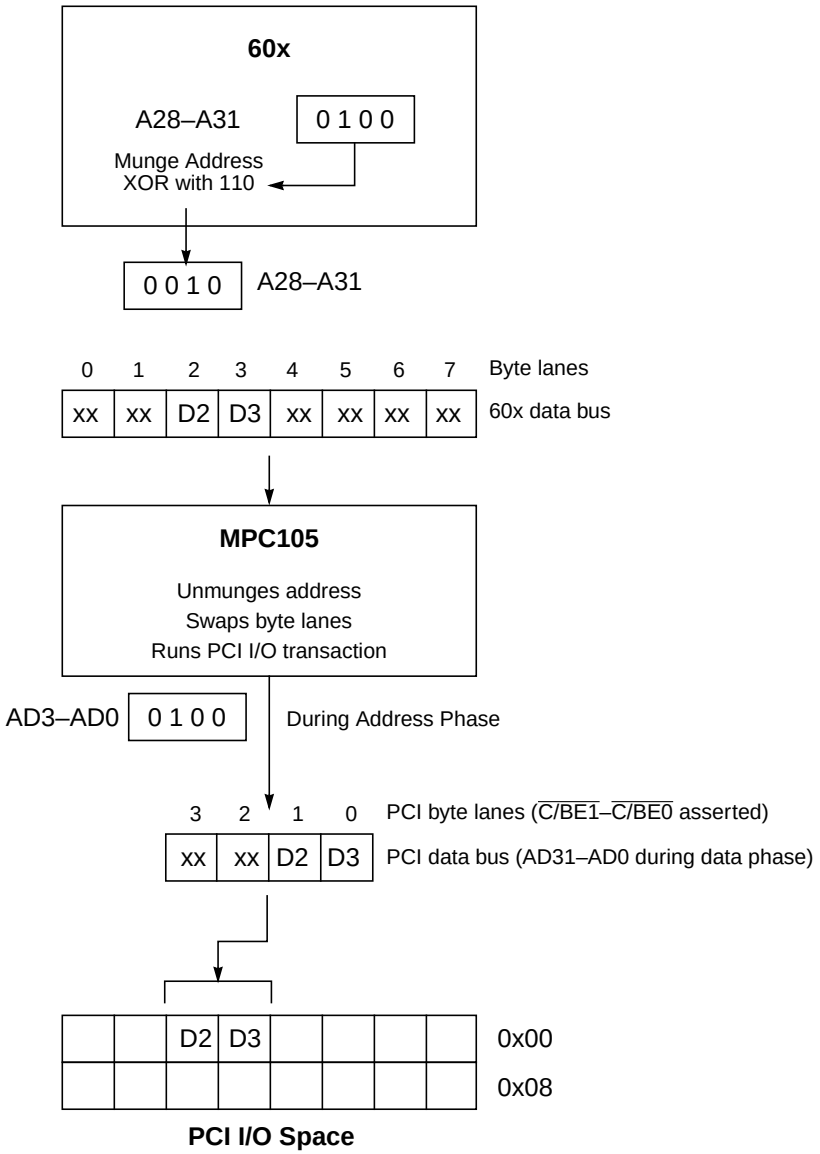


Figure B-10. Two-Byte Transfer to PCI I/O Space—Little Endian Mode

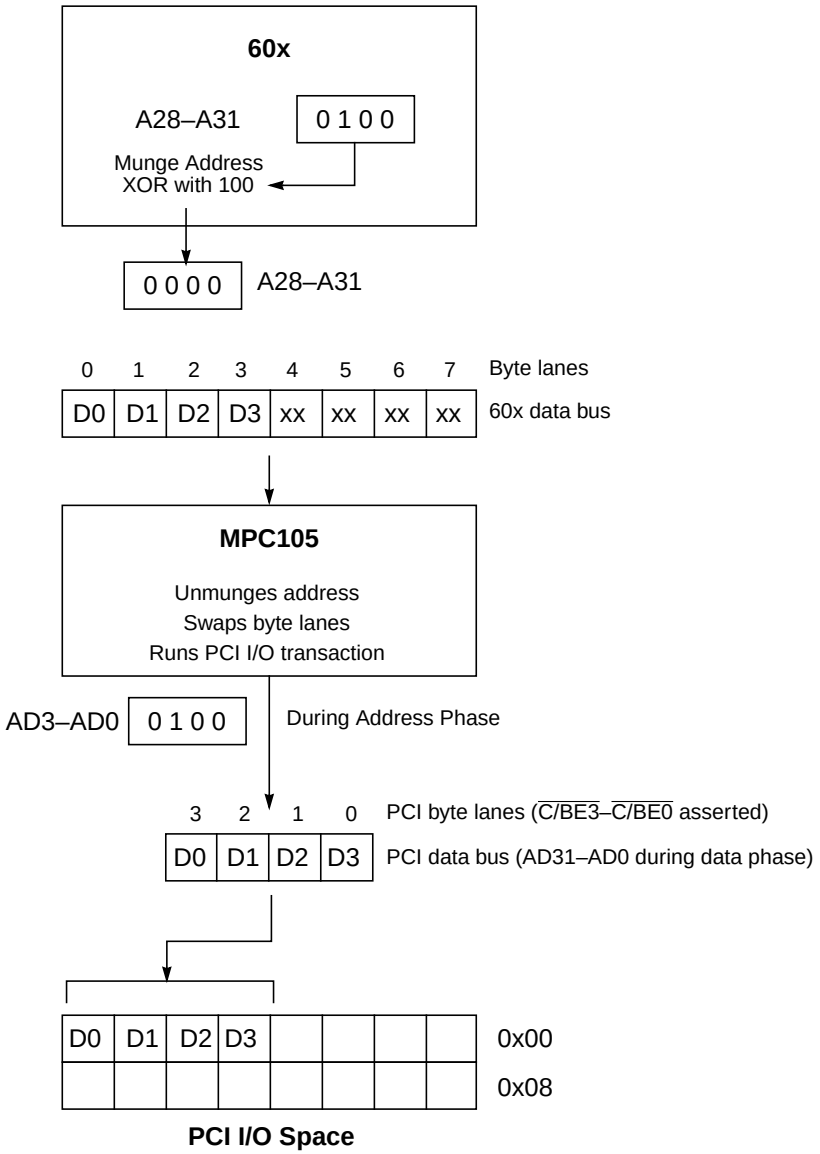


Figure B-11. Four-Byte Transfer to PCI I/O Space—Little Endian Mode

Appendix C
JTAG/Testing Support

The MPC105 provides a joint test action group (JTAG) interface to facilitate boundary-scan testing. The JTAG interface implements the five test port signals required to be fully compliant with the IEEE 1149.1 specification. For additional information about JTAG operations, refer to the IEEE 1149.1 boundary-scan specification.

C.1 JTAG Interface Description

The JTAG interface consists of a set of five signals, three JTAG registers, and a test access port (TAP) controller, described in the following sections. A block diagram of the JTAG interface is shown in Figure C-1.

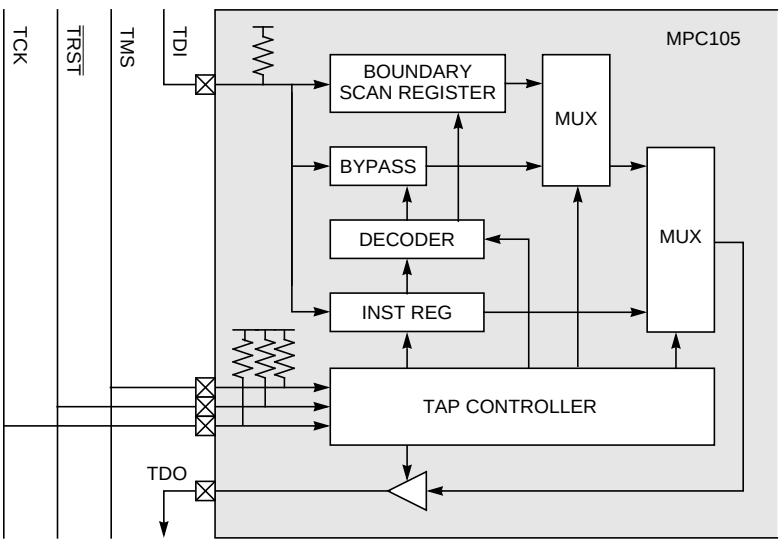


Figure C-1. JTAG Interface Block Diagram

C.1.1 JTAG Signals

The MPC105 provides five dedicated JTAG signals; test data input (TDI), test mode select (TMS), test reset ($\overline{\text{TRST}}$), test clock (TCK), and test data output (TDO). The TDI and TDO signals are used to input and output instructions and data to the JTAG scan registers. The boundary-scan operations are controlled by the TAP controller through commands received by means of the TMS signal. Boundary-scan data is latched by the TAP controller on the rising edge of the TCK signal. The $\overline{\text{TRST}}$ signal is specified as optional by the IEEE 1149.1 specification, and is used to reset the TAP controller asynchronously. The assertion of the $\overline{\text{TRST}}$ signal at power-on reset assures that the JTAG logic does not interfere with the normal operation of the MPC105.

Section 2.2.6, “IEEE 1149.1 Interface Signals” provides additional detail about the operation of these signals.

C.1.2 JTAG Registers and Scan Chains

The bypass, boundary-scan, and instruction JTAG registers and their associated scan chains are implemented by the MPC105. These registers are mandatory for compliance with the IEEE 1149.1 specification.

C.1.2.1 Bypass Register

The bypass register is a single-stage register used to bypass the boundary-scan latches of the MPC105 during board-level boundary-scan operations involving components other than the MPC105. The use of the bypass register reduces the total scan string size of the boundary-scan test.

C.1.2.2 Boundary-Scan Registers

The JTAG interface provides a chain of registers dedicated to boundary-scan operations. To be JTAG-compliant, these registers cannot be shared with any functional registers of the MPC105. The boundary-scan register chain includes registers controlling the direction of the input/output drivers, in addition to the registers reflecting the signal value received or driven.

The boundary-scan registers capture the input or output state of the MPC105’s signals during a Capture_DR TAP controller state. When a data scan is initiated following the Capture_DR state, the sampled values are shifted out through the TDO output while new boundary-scan register values are shifted in through the TDI input. At the end of the data scan operation, the boundary-scan registers are updated with the new values during an Update_DR TAP controller state.

Note that the $\overline{\text{LSSD\_MODE}}$ signal (used for factory testing) is not included in the boundary-scan register chain.

C.1.2.3 Instruction Register

The 8-bit JTAG instruction register serves as an instruction and status register. As TAP controller instructions are scanned in through the TDI input, the TAP controller status bits are scanned out through the TDO output.

C.1.3 TAP Controller

The MPC105 provides a standard JTAG TAP controller that controls instruction and data scan operations. The TMS signal controls the state transitions of the TAP controller.

Appendix D
Initialization Example

This appendix contains PowerPC assembly language initialization code for the MPC105-based Big Bend system. Big Bend is an example PowerPC reference platform-compliant system design that can be used in designing systems using PowerPC microprocessors. For information on the Big Bend evaluation system, contact your Motorola sales office.

```
;# standard definitions

;# MPC105 Registers
#define EAGLE_REG        0x8000
#define VEND_ID_REG      0x0000
#define DEV_ID_REG       0x0002
#define PCI_CMD          0x0004
#define PCI_STAT         0x0006
#define MEM_STA_03       0x0080
#define MEM_STA_47       0x0084
#define EXT_MEM_STA_03   0x0088
#define EXT_MEM_STA_47   0x008c
#define MEM_END_03       0x0090
#define MEM_END_47       0x0094
#define EXT_MEM_END_03   0x0098
#define EXT_MEM_END_47   0x009c
#define MEM_BANK_EN      0x00a0
#define PROC_CFG_A8      0x00a8
#define PROC_CFG_AC      0x00aC
#define ALT_OSV_1        0x00ba
#define ALT_OSV_2        0x00bb
#define ERR_EN_REG1      0x00c0
#define MEM_ERRD_REG     0x00c1
#define CPU_BES_REG      0x00c3
#define ERR_EN_REG2      0x00c4
#define ERR_DET_REG2     0x00c5
#define PCI_BES_REG      0x00c7
#define MEM_CFG_1       0x00f0
#define MEM_CFG_2       0x00f4
#define MEM_CFG_3       0x00f8
#define MEM_CFG_4       0x00fc
```

```
;# TCPCI Memory
#define BANK0_START_ADDR  0x00000000
#define BANK1_START_ADDR  0x00800000
#define BANK2_START_ADDR  0x01000000
#define BANK3_START_ADDR  0x01800000
#define BANK4_START_ADDR  0x02000000
#define BANK5_START_ADDR  0x02800000
#define BANK6_START_ADDR  0x03000000
#define BANK7_START_ADDR  0x03800000
#define BOOTROM_START_ADDR  0xffe00000
#define PCIMEM_START_ADDR  0xc0000000

        .toc
T..main:
        .tc     ..main[tc], main[ds]

        .globl  main[ds]
        .csect  main[ds]
        .long   .main[pr], TOC[tc0], 0
        .globl  .main[pr]
        .csect  .main[pr]
        .align  4

;#============================================================================================
;#
;# Register usage:
;#
;# r0 is continually loaded with masking patterns
;# r1 = 0x 8000 0cf8  CONFIG_ADDRESS
;# r2 = 0x 8000 0cfc  CONFIG_DATA
;# r3 is used to define which register number is to be stored at CONFIG_ADDRESS
;# r4 is used for loading and storing data to/from CONFIG_DATA
;#
;#============================================================================================

.main:
    lis         r0, EAGLE_REG       # r0 = 0x 8000 0000  BASE_ADDRESS
    ori         r1, r0, 0x0cf8      # r1 = 0x 8000 0cf8  CONFIG_ADDRESS
    ori         r2, r0, 0x0cfc      # r2 = 0x 8000 0cfc  CONFIG_DATA
;#
;#  -+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
;#  This small section of code speeds up accesses to the Boot ROM, but
;#  is written SPECIFICALLY FOR THE MDC2.  This section of code is
;#  duplicated at the beginning of the initmdc2 routine found below.
;#  When the MDC Boot ROM isn't really Fast SRAM, this section must
;#  be removed.
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, MEM_CFG_1  # register number 0xf0
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR
;#
```

```
    lwbrx       r4, 0, r2           # load r4 from CONFIG_DATA
    lis         r0, 0x0016          # REDUCE WAIT STATES FOR ROM ACCESSES
    ori         r0, r0, 0x5555      # (contains no reserved bits)
    and         r4, r4, r0          # clears the desired bits
    or          r4, r4, r0          # sets the desired bits
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
;#  End of Boot ROM speed-up specific to the MDC2
;#  -+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
;#
;# This section of code initializes the MPC105's PCI Interface Registers
;#

initpci:

    isync

;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, PCI_CMD     # register number 0x0004
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lhbrx       r4, 0, r2           # load r4 from CONFIG_DATA
    lis         r0, 0x0000          #
    ori         r0, r0, 0x0106      #
    or          r4, r4, r0          # sets the desired bits
    sthbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, PCI_STAT    # register number 0x0006
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    li          r3, 0x0002
    lhbrx       r4, r3, r2          # load r4 from CONFIG_DATA
    ori         r4, r4, 0xffff      # Writing all ones will clear all bits in PCI_STAT
    sthbrx      r4, r3, r2          # write the modified data to CONFIG_DATA
;#
;#=============================================================================
;#
;# This section of code initializes the MPC105's Processor Interface Registers
;# for use with the PDC1 (single 603 and 256 kB of Async L2) at 60 - 66 MHz.
;#
;# Note that the PROC_CFG_AC register is initialized BEFORE the PROC_CFG_A8 register.

initproc:

    isync

;#
```

```
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, PROC_CFG_AC # register number 0xac
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lwbrx       r4, 0, r2           # load r4 from CONFIG_DATA
    lis         r0, 0xae9e          # Reserved bits are 29, 27, 26, 15, and 11
    ori         r0, r0, 0xed0e      # bit 31 is MSb, bit 0 is LSb (see page 230 and 231)
    and         r4, r4, r0          # clears the desired bits
    lis         r0, 0x829e          #
    ori         r0, r0, 0x650e      #
    or          r4, r4, r0          # sets the desired bits
    stwbrx      r4, 0, r2           # write 0x a29e 650e to CONFIG_DATA

;# These next five lines are necessary if you later want to turn on the L2 cache

    lwbrx       r4, 0, r2
    lis         r0, 0x4000          # Now that we've written to the 0x ac register,
   ori        r0, r0, 0x0000     # keep pattern the same, but set the L2_EN bit (see next reg)
    or          r4, r4, r0          # by setting bit 30 to a 1
    stwbrx      r4, 0, r2           # write 0x c29e 650e to CONFIG_DATA

;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, PROC_CFG_A8 # register number 0xa8
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lwbrx       r4, 0, r2           # load r4 from CONFIG_DATA
    lis         r0, 0xff75          # bits 14 and 8 are the only two reserved bits
    ori         r0, r0, 0x479a      # bit 31 is MSb, bit 0 is LSb
    and         r4, r4, r0          # clears the desired bits
    lis         r0, 0xff75          #
    ori         r0, r0, 0x0698      # LEAVE THE L2 CACHE OFF (CF_L2_MP = 00 for Uniprocessor)
;#  ori         r0, r0, 0x069a      # TURN ON THE L2 CACHE (CF_L2_MP = 10 for Write-Back)
    or          r4, r4, r0          # sets the desired bits
    stwbrx      r4, 0, r2           # write 0x ff75 069a to CONFIG_DATA

;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, ALT_OSV_1   # register number 0xba
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lbz         r4, 2(r2)           # load r4 from CONFIG_DATA
    lis         r0, 0x0000          #
    ori         r0, r0, 0x0026      # see description of Alt OS Visible Param Reg 1
    or          r4, r4, r0          # sets the desired bits
    stb         r4, 2(r2)           # write the modified data to CONFIG_DATA

;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, ALT_OSV_2   # register number 0xbb
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR
```

```
    lbz         r4, 3(r2)           # load r4 from CONFIG_DATA
    lis         r0, 0x0000          #
    ori         r0, r0, 0x0000      # see description of Alt OS Visible Param Reg 2
    or          r4, r4, r0          # sets the desired bits
    stb         r4, 3(r2)           # write the modified data to CONFIG_DATA

;#
;#  reading error handling registers
;#    lis         r3, EAGLE_REG       # start building new register number
;#    ori         r3, r3, ERR_EN_REG1 # register number 0xc0
;#    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

;#    lbz         r4, 0(r2)           # load r4 from CONFIG_DATA
;#  lis         r0, 0x0000          #
;#  ori         r0, r0, 0x0040      # see description of ErrEnR1
;#  or          r4, r4, r0          # sets the desired bits
;#  stb         r4, 0(r2)           # write the modified data to CONFIG_DATA

;#
;#    lis         r3, EAGLE_REG       # start building new register number
;#    ori         r3, r3, MEM_ERRD_REG # register number 0xc1
;#    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

;#    lbz         r4, 1(r2)           # load r4 from CONFIG_DATA
;#  lis         r0, 0x0000          #
;#  ori         r0, r0, 0x0000      # see description of ErrDR1
;#  or          r4, r4, r0          # sets the desired bits
;#  stb         r4, 1(r2)           # write the modified data to CONFIG_DATA

;#
;#    lis         r3, EAGLE_REG       # start building new register number
;#    ori         r3, r3, CPU_BES_REG # register number 0xc3
;#    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

;#    lbz         r4, 3(r2)           # load r4 from CONFIG_DATA
;#  lis         r0, 0x0000          #
;#  ori         r0, r0, 0x0000      # see description of 60x Bus Error Status Reg
;#  or          r4, r4, r0          # sets the desired bits
;#  stb         r4, 3(r2)           # write the modified data to CONFIG_DATA

;#
;#    lis         r3, EAGLE_REG       # start building new register number
;#    ori         r3, r3, ERR_EN_REG2 # register number 0xc4
;#    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

;#    lbz         r4, 0(r2)           # load r4 from CONFIG_DATA
;#   lis         r0, 0x0000          #
;#   ori         r0, r0, 0x0000      # see description of ErrEnR2
;#   or          r4, r4, r0          # sets the desired bits
;#   stb         r4, 0(r2)           # write the modified data to CONFIG_DATA
```

```
;#
;#    lis         r3, EAGLE_REG       # start building new register number
;#    ori         r3, r3, ERR_DET_REG2 # register number 0xc5
;#    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

;#    lbz         r4, 1(r2)           # load r4 from CONFIG_DATA
;#   lis         r0, 0x0000          #
;#   ori         r0, r0, 0x0000      # see description of ErrDR2
;#   or          r4, r4, r0          # sets the desired bits
;#   stb         r4, 1(r2)           # write the modified data to CONFIG_DATA

;#
;#    lis         r3, EAGLE_REG       # start building new register number
;#    ori         r3, r3, PCI_BES_REG # register number 0xc7
;#    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

;#    lbz         r4, 3(r2)           # load r4 from CONFIG_DATA
;#   lis         r0, 0x0000          #
;#   ori         r0, r0, 0x0000      # see description of PCI Bus Error Status Reg
;#   or          r4, r4, r0          # sets the desired bits
;#   stb         r4, 3(r2)           # write the modified data to CONFIG_DATA

;#
;#=============================================================================
;#
;# This section of code initializes the MPC105's memory configuration registers
;# for use with the MDC2 (64 MB DRAM and 2 MB SRAM BootROM) at 60 - 66 MHz.
;#

initmdc2:

    isync
;#

    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, MEM_CFG_1  # register number 0xf0
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR
;#
    lwbrx       r4, 0, r2           # load r4 from CONFIG_DATA
    lis         r0, 0x0016          # REDUCE WAIT STATES FOR ROM ACCESSES
    ori         r0, r0, 0x5555      # (contains no reserved bits)
    and         r4, r4, r0          # clears the desired bits
    or          r4, r4, r0          # sets the desired bits
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, MEM_CFG_2  # register number 0xf4
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR
```

```
    lwbrx       r4, 0, r2           # load r4 from CONFIG_DATA
    lis         r0, 0x0000          # Self-Refresh value (not used for MDC2 or MDC3)
    ori         r0, r0, 0x0c35      # 0x30d (decimal 781) clocks between refresh,
 BUF=E/E
                                   ;# 781 clocks for 50 MHz, 1041 for 66.6 MHz
    and         r4, r4, r0          # clears the desired bits
    or          r4, r4, r0          # sets the desired bits
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, MEM_CFG_3  # register number 0xf8
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lwbrx       r4, 0, r2           # load r4 from CONFIG_DATA
    lis         r0, 0x0002          # RAS6P=0101, CAS5=010, CP4=001,
    ori         r0, r0, 0xa294      # CAS3= 010, RCD2=010, RP1=100
    and         r4, r4, r0          # clears the desired bits
    or          r4, r4, r0          # sets the desired bits
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
;# For MDC2, 64MB total of DRAM
;#
;# Bank0 = 0x 0000 0000 - 0x 007f ffff
;# Bank1 = 0x 0080 0000 - 0x 00ff ffff
;# Bank2 = 0x 0100 0000 - 0x 017f ffff
;# Bank3 = 0x 0180 0000 - 0x 01ff ffff
;# Bank4 = 0x 0200 0000 - 0x 027f ffff
;# Bank5 = 0x 0280 0000 - 0x 02ff ffff
;# Bank6 = 0x 0300 0000 - 0x 037f ffff
;# Bank7 = 0x 0380 0000 - 0x 03ff ffff
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, MEM_STA_03  # register number 0x80
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lis         r4, 0x1810          # Each bank on MDC2 is 8MB
    ori         r4, r4, 0x0800      #  (no reserved bits)
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, MEM_STA_47  # register number 0x84
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lis         r4, 0x3830          # Each bank on MDC2 is 8MB
    ori         r4, r4, 0x2820      #  (no reserved bits)
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, EXT_MEM_STA_03  # register number 0x88
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR
```

```
    lwbrx       r4, 0, r2           # load r4 from CONFIG_DATA
    lis         r0, 0xfcfc          # Each bank on MDC2 is 8MB
    ori         r0, r0, 0xfcfc      #
    and         r4, r4, r0          # clears all non-reserved bits
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, EXT_MEM_STA_47  # register number 0x8c
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lwbrx       r4, 0, r2           # load r4 from CONFIG_DATA
    lis         r0, 0xfcfc          # Each bank on MDC2 is 8MB
    ori         r0, r0, 0xfcfc      #
    and         r4, r4, r0          # clears all non-reserved bits
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, MEM_END_03  # register number 0x90
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lis         r4, 0x1f17          # Each bank on MDC2 is 8MB
    ori         r4, r4, 0x0f07      # (no reserved bits)
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, MEM_END_47  # register number 0x94
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lis         r4, 0x3f37          # Each bank on MDC2 is 8MB
    ori         r4, r4, 0x2f27      # (no reserved bits)
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, EXT_MEM_END_03  # register number 0x98
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lwbrx       r4, 0, r2           # load r4 from CONFIG_DATA
    lis         r0, 0xfcfc          # Each bank on MDC2 is 8MB
    ori         r0, r0, 0xfcfc      #
    and         r4, r4, r0          # clears all non-reserved bits
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, EXT_MEM_END_47  # register number 0x9c
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lwbrx       r4, 0, r2           # load r4 from CONFIG_DATA
    lis         r0, 0xfcfc          # Each bank on MDC2 is 8MB
    ori         r0, r0, 0xfcfc      #
    and         r4, r4, r0          # clears all non-reserved bits
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
```

```
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, MEM_BANK_EN  # register number 0xa0
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lis         r4, 0x0000          # ENABLE ALL 8 BANKS OF DRAM
    ori         r4, r4, 0x00ff      # (no reserved bits)
    stb         r4, 0(r2)           # write the modified data to CONFIG_DATA
;#
;# DRAM SHOULD NOW BE CONFIGURED AND ENABLED - MUST WAIT 100 us BEFORE ACCESSING
;#
    li          r0, 0x1800          # decimal 6144
    mtctr       r0
wait100us:
    bdnz        wait100us
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, MEM_CFG_1  # register number 0xf0
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lwbrx       r4, 0, r2           # load r4 from CONFIG_DATA
    lis         r0, 0x0008          # MEMGO=1
    ori         r0, r0, 0x0000
    or          r4, r4, r0          # set the MEMGO bit
    stwbrx      r4, 0, r2           # write the modified data to CONFIG_DATA
;#
    li          r0, 0x2000          # approx decimal 8000
    mtctr       r0
wait8ref:
    bdnz        wait8ref
;#
;# DRAM ON MDC2 IS NOW AVAILABLE FOR USE - ASSERT DRAMINIT FLAG IN TCSRO REGISTER
;#
    lis         r3, TCPCI_REGS      #
    ori         r3, r3, PH_REG      # r3 = 0x 8000 505c for PortHole
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    li          r3, 0x0002          # 2-byte write to PH_REG+2 to assert DRAMINIT
    sthbrx      r4, r3, r2          # write the CONFIG_DATA
;#
;#=============================================================================
;#
;# To turn on the L2 cache ; uncomment the desired mode below
;#
    lis         r3, EAGLE_REG       # start building new register number
    ori         r3, r3, PROC_CFG_A8 # register number 0xa8
    stwbrx      r3, 0, r1           # write this value to CONFIG_ADDR

    lwbrx       r4, 0, r2           # load r4 from CONFIG_DATA
;#    ori         r4, r4, 0x0001      # set bit 0 for write-through L2
;#    stwbrx      r4, 0, r2           # write 0x ff75 0699 to CONFIG_DATA
```

```
      ori         r4, r4, 0x0002      # set bit 1 for write-back L2
      stwbrx      r4, 0, r2           # write 0x ff75 069a to CONFIG_DATA
;#
;#=============================================================================
;#
;#  remove this infinite branch after pasting into your code !!
EndLoop:
         b        EndLoop
```

Glossary of Terms and Abbreviations

The glossary contains an alphabetical list of terms, phrases, and abbreviations used in this book.

A

Atomic. A bus access that attempts to be part of a read-write operation to the same address uninterrupted by any other access to that address (the term refers to the fact that the transactions are indivisible). The 60x processor initiates the read and write separately, but signals the memory system that it is attempting an atomic operation. If the operation fails, status is kept so that the 60x can try again. The 60x implements atomic accesses through the **lwarx**/**stwcx.** instruction pair, which asserts the TT0 signal.

B

Beat. A single state on the 60x interface that may extend across multiple bus cycles. A 60x transaction can be composed of multiple address or data beats.

Big-endian. A byte-ordering method in memory where the address n of a word corresponds to the most significant byte. In an addressed memory word, the bytes are ordered (left to right) 0, 1, 2, 3, with 0 being the most significant byte.

Buffer. A temporary storage mechanism for queuing data.

Burst. A multiple beat data transfer whose total size is typically equal to a cache line (32-bytes).

Bus clock. Clock that synchronizes the bus state transitions.

Bus master. The owner of the address or data bus; the device that initiates or requests the transaction.

C

Cache. High-speed memory containing recently accessed data and/or instructions (subset of main memory).

Cache coherency. Caches are coherent if a processor performing a read from its cache is supplied with data corresponding to the most recent value written to memory or to another processor's cache.

Cast-outs. Cache line that must be written to memory when a snoop miss causes the least recently used cache line with modified data to be replaced.

Copy-back operation. A cache operation in which a cache line is copied back to memory to enforce cache coherency. Copy-back operations consist of snoop push-out operations and cache cast-out operations.

D

Disconnect.The termination of a PCI cycle initiated by the target because it is unable to respond within eight PCI clock cycles.

E

E**xception**. An unusual or error condition encountered by the processor that results in special processing.

Exception handler. A software routine that executes when an exception occurs. Normally, the exception handler corrects the condition that caused the exception, or performs some other meaningful task (such as aborting the program that caused the exception). The addresses of the exception handlers are defined by a two-word exception vector that is branched to automatically when an exception occurs.

Exclusive state. A cache state in which only one caching device contains data that is also in system memory.

F

Flush. An operation that causes a modified cache line to be invalidated and the data to be written to memory.

G

General-purpose register. Any of the 32 registers in the 60x register file. These registers provide the source operands and destination results for all 60x data manipulation instructions. Load instructions move data from memory to registers, and store instructions move data from registers to memory.

I

Interrupt. An external signal that causes the 60x to suspend current execution and take a predefined exception.

Invalid state. A cache state that indicates that the cache line does not contain valid data.

K

Kill. An operation that causes a cache line to be invalidated.

L

Latency. The number of clock cycles necessary to execute an instruction and make ready the results of that instruction.

Little-endian. A byte-ordering method in memory where the address *n* of a word corresponds to the least significant byte. In an addressed memory word, the bytes are ordered (left to right) 3, 2, 1, 0, with 3 being the most significant byte.

Livelock. A state in which processors interact in a way such that no processor makes progress.

M

Memory-mapped accesses. Accesses whose addresses use the segmented or block address translation mechanisms provided by the 60x memory management unit (MMU) and that occur externally with the bus protocol defined for memory.

Memory coherency. Refers to memory agreement between caches and system memory (for example, MESI cache coherency, in which there are four cache states– modified, exclusive, shared, and invalid).

Memory consistency. Refers to levels of memory with respect to a single processor and system memory (for example, primary cache, secondary cache, and system memory).

Memory management unit. The functional unit in the 60x that translates the logical address bits to physical address bits.

Modified state. A cache state in which one, and only one, caching device has the valid data for that address. The data at this address in external memory is not valid.

Munging. A technique used to alter the byte-ordering of data by modifying its address; commonly used when translating between big-endian and little-endian data formats.

N

No-op. No-operation. A single-cycle operation that does not affect registers or generate bus activity.

P

Page. A 4-Kbyte area of memory, aligned on a 4-Kbyte boundary.

Park. The act of allowing a bus master to maintain mastership of the bus without having to arbitrate.

PCI (peripheral component interconnect).A computer bus standard, managed by an industry consortium called the PCI SIG (Special Interest Group), that uses a 32- or 64-bit multiplexed address/data bus and provides the interconnect mechanism between peripheral components.

Pipelining. A technique that breaks instruction execution into distinct steps so that multiple steps can be performed at the same time.

Primary (L1) cache.The cache resource that is most readily available to a processor (for example, the internal cache of a 60x processor). See also secondary (L2) cache.

R

Retry. Resending the current address or data beat until it can be accepted.

Refresh.Periodic charging a device that cannot hold its content. Dynamic RAM (DRAM) devices require refresh cycles every few milliseconds to preserve their charged bit patterns.

S

Scan interface. The 60x's test interface.

Secondary (L2) cache.The cache resource that is next-to-the-most readily available to a processor, the primary (or L1) cache being the most readily available. This cache is typically larger, offers slower access time than a primary cache, and may be accessed by multiple devices. The use of a secondary cache improves performance by reducing the number of bus accesses to external main memory.

Shared state. A cache state that indicates that more than one caching device contains unmodified data that is also in system memory.

Slave. The device addressed by a master device. The slave is identified in the address tenure/phase and is responsible for supplying or latching the requested data for the master during the data tenure/phase.

Snooping. Monitoring addresses driven by a bus master to detect the need for coherency actions.

Snoop push. Write-backs due to a snoop hit. The cache line will transition to an invalid or exclusive state.

Split-transaction. A transaction with independent request and response tenures.

Split-transaction Bus. A bus that allows address and data transactions from different processors to occur independently.

T

Tenure. The period of bus mastership. For the 60x, there can be separate address bus tenures and data bus tenures. A tenure consists of three phases: arbitration, transfer, termination

Timeout. A transaction termination due to exceeding a latency limit. A transaction is not necessarily concluded when a timeout occurs.

Transaction. A complete exchange between two bus devices. A transaction is minimally comprised of an address tenure/phase; one or more data tenures/phases may be involved in the exchange. There are two kinds of transactions: address/data and address-only.

Transfer termination. Signal that refers to both signals that acknowledge the transfer of individual beats (of both single-beat transfer and individual beats of a burst transfer) and to signals that mark the end of the tenure/phase.

W

Write-back.A memory update policy in which processor write cycles are only required to be written to the cache. The data in the cache does not necessarily stay consistent with that same location's data in memory. The data in the cache is copied to memory when a copy-back operation is required.

Write-through. A memory update policy in which all processor write cycles are written to both the cache and memory.

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

Q

R

S

INDEX

T

INDEX

W

X