



MPC106UM/AD
7/2003
Rev. 2

MPC106 PCI Bridge/Memory Controller User's Manual

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution
 P.O. Box 5405, Denver, Colorado 80217
 1-480-768-2130
 (800) 521-6274

JAPAN:

Motorola Japan Ltd.
 SPS, Technical Information Center
 3-20-1, Minami-Azabu Minato-ku
 Tokyo 106-8573 Japan
 81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.
 Silicon Harbour Centre, 2 Dai King Street
 Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
 852-26668334

TECHNICAL INFORMATION CENTER:

(800) 521-6274

HOME PAGE:

www.motorola.com/semiconductors

Information in this document is provided solely to enable system and software implementers to use Motorola products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Motorola reserves the right to make changes without further notice to any products herein.

Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.



Motorola and the Stylized M Logo are registered in the U.S. Patent and Trademark Office. digital dna is a trademark of Motorola, Inc. The described product is a PowerPC microprocessor. The PowerPC name is a trademark of IBM Corp. and is used under license. All other product or service names are the property of their respective owners. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

© Motorola, Inc. 2003

**For More Information On This Product,
 Go to: www.freescale.com**

Overview	1
Signal Descriptions	2
Device Programming	3
Processor Bus Interface	4
Secondary Cache Interface	5
Memory Interface	6
PCI Bus Interface	7
Internal Control	8
Error Handling	9
Performance Monitor	10
Power Management	A
Bit and Byte Ordering	B
JTAG/Testing Support	C
Initialization Example	D
Revision History	E
Glossary of Terms and Abbreviations	GLO
Index	IND

1	Overview
2	Signal Descriptions
3	Device Programming
4	Processor Bus Interface
5	Secondary Cache Interface
6	Memory Interface
7	PCI Bus Interface
8	Internal Control
9	Error Handling
10	Performance Monitor
A	Power Management
B	Bit and Byte Ordering
C	JTAG/Testing Support
D	Initialization Example
E	Revision History
GLO	Glossary of Terms and Abbreviations
IND	Index

Contents

Paragraph Number	Title	Page Number
------------------	-------	-------------

About This Book

	Audience	xxvii
	Organization.....	xxviii
	Suggested Reading.....	xxix
	General Information.....	xxix
	Documentation on PowerPC Architecture.....	xxix
	Conventions	xxx
	Acronyms and Abbreviations	xxx

Chapter 1 Overview

1.1	MPC106 PCIB/MC Features	1-1
1.2	MPC106 Major Functional Units	1-4
1.2.1	60x Processor Interface.....	1-4
1.2.2	Secondary (L2) Cache/Multiple Processor Interface.....	1-4
1.2.3	Memory Interface	1-5
1.2.4	PCI Interface	1-6
1.3	Performance Monitor	1-6
1.4	Power Management	1-7
1.4.1	Full-On Mode	1-7
1.4.2	Doze Mode.....	1-7
1.4.3	Nap Mode	1-7
1.4.4	Sleep Mode	1-7
1.4.5	Suspend Mode.....	1-8

Chapter 2 Signal Descriptions

2.1	Signal Configuration.....	2-1
2.2	Signal Descriptions	2-2
2.2.1	Signal States at Reset.....	2-8
2.2.2	60x Processor Interface Signals.....	2-9
2.2.2.1	Address Bus (A[0–31]).....	2-10

Contents

Paragraph Number	Title	Page Number
2.2.2.1.1	Address Bus (A[0–31])—Output.....	2-10
2.2.2.1.2	Address Bus (A[0–31])—Input	2-10
2.2.2.2	Address Acknowledge ($\overline{\text{AACK}}$).....	2-10
2.2.2.2.1	Address Acknowledge ($\overline{\text{AACK}}$)—Output.....	2-10
2.2.2.2.2	Address Acknowledge ($\overline{\text{AACK}}$)—Input	2-11
2.2.2.3	Address Retry ($\overline{\text{ARTRY}}$).....	2-11
2.2.2.3.1	Address Retry ($\overline{\text{ARTRY}}$)—Output	2-11
2.2.2.3.2	Address Retry ($\overline{\text{ARTRY}}$)—Input	2-12
2.2.2.4	Bus Grant 0 ($\overline{\text{BG0}}$)—Output.....	2-12
2.2.2.5	Bus Request 0 ($\overline{\text{BR0}}$)—Input.....	2-13
2.2.2.6	Caching-Inhibited ($\overline{\text{CI}}$)—Input/Output.....	2-13
2.2.2.7	Data Bus Grant 0 ($\overline{\text{DBG0}}$)—Output	2-13
2.2.2.8	Data Bus Grant Local Bus Slave ($\overline{\text{DBGLB}}$)—Output.....	2-14
2.2.2.9	Data Bus (DH[0–31], DL[0–31])	2-14
2.2.2.9.1	Data Bus (DH[0–31], DL[0–31])—Output	2-15
2.2.2.9.2	Data Bus (DH[0–31], DL[0–31])—Input.....	2-15
2.2.2.10	Global ($\overline{\text{GBL}}$)—Input/Output	2-16
2.2.2.11	Local Bus Slave Claim ($\overline{\text{LBCLAIM}}$)—Input	2-16
2.2.2.12	Machine Check ($\overline{\text{MCP}}$)—Output.....	2-16
2.2.2.13	Transfer Acknowledge ($\overline{\text{TA}}$).....	2-17
2.2.2.13.1	Transfer Acknowledge ($\overline{\text{TA}}$)—Output.....	2-17
2.2.2.13.2	Transfer Acknowledge ($\overline{\text{TA}}$)—Input	2-17
2.2.2.14	Transfer Burst ($\overline{\text{TBST}}$).....	2-18
2.2.2.14.1	Transfer Burst ($\overline{\text{TBST}}$)—Output.....	2-18
2.2.2.14.2	Transfer Burst ($\overline{\text{TBST}}$)—Input	2-18
2.2.2.15	Transfer Error Acknowledge ($\overline{\text{TEA}}$)—Output.....	2-18
2.2.2.16	Transfer Start ($\overline{\text{TS}}$).....	2-19
2.2.2.16.1	Transfer Start ($\overline{\text{TS}}$)—Output.....	2-19
2.2.2.16.2	Transfer Start ($\overline{\text{TS}}$)—Input.....	2-19
2.2.2.17	Transfer Size (TSIZ[0–2])	2-19
2.2.2.17.1	Transfer Size (TSIZ[0–2])—Output.....	2-19
2.2.2.17.2	Transfer Size (TSIZ[0–2])—Input.....	2-20
2.2.2.18	Transfer Type (TT[0–4]).....	2-20
2.2.2.18.1	Transfer Type (TT[0–4])—Output.....	2-20
2.2.2.18.2	Transfer Type (TT[0–4])—Input	2-20
2.2.2.19	Write-Through ($\overline{\text{WT}}$)—Input/Output.....	2-20
2.2.2.20	Extended Address Transfer Start ($\overline{\text{XATS}}$)—Input	2-21
2.2.3	L2 Cache/Multiple Processor Interface Signals.....	2-21
2.2.3.1	Internal L2 Controller Signals	2-22
2.2.3.1.1	Address Strobe ($\overline{\text{ADS}}$)—Output.....	2-22
2.2.3.1.2	Burst Address 0 ($\overline{\text{BA0}}$)—Output	2-22

Contents

Paragraph Number	Title	Page Number
2.2.3.1.3	Burst Address 1 (BA1)—Output	2-22
2.2.3.1.4	Bus Address Advance ($\overline{\text{BAA}}$)—Output	2-23
2.2.3.1.5	Data Address Latch Enable ($\overline{\text{DALE}}$)—Output	2-23
2.2.3.1.6	Data RAM Chip Select ($\overline{\text{DCS}}$)—Output	2-23
2.2.3.1.7	Dirty In ($\overline{\text{DIRTY_IN}}$)—Input	2-24
2.2.3.1.8	Dirty Out ($\overline{\text{DIRTY_OUT}}$)—Output	2-24
2.2.3.1.9	Data RAM Output Enable ($\overline{\text{DOE}}$)—Output	2-24
2.2.3.1.10	Data RAM Write Enable ($\overline{\text{DWE}}[0-2]$)—Output	2-25
2.2.3.1.11	Hit ($\overline{\text{HIT}}$)—Input	2-25
2.2.3.1.12	Tag Output Enable ($\overline{\text{TOE}}$)—Output	2-25
2.2.3.1.13	Tag Valid (TV)—Output	2-26
2.2.3.1.14	Tag Write Enable ($\overline{\text{TWE}}$)—Output	2-26
2.2.3.2	External L2 Controller Signals	2-26
2.2.3.2.1	External L2 Bus Grant ($\overline{\text{BGL2}}$)—Output	2-26
2.2.3.2.2	External L2 Bus Request ($\overline{\text{BRL2}}$)—Input	2-27
2.2.3.2.3	External L2 Data Bus Grant ($\overline{\text{DBGL2}}$)—Output	2-27
2.2.3.2.4	Hit ($\overline{\text{HIT}}$)—Input	2-27
2.2.3.3	Multiple Processor Interface Signals	2-28
2.2.3.3.1	Bus Grant 1–3 ($\overline{\text{BG}}[1-3]$)—Output	2-28
2.2.3.3.2	Bus Request 1–3 ($\overline{\text{BR}}[1-3]$)—Input	2-28
2.2.3.3.3	Data Bus Grant 1–3 ($\overline{\text{DBG}}[1-3]$)—Output	2-29
2.2.4	Memory Interface Signals	2-29
2.2.4.1	ROM Address 0 (AR0)—Output	2-29
2.2.4.2	ROM Address 1–8 (AR[1–8])—Output	2-30
2.2.4.3	ROM Address 9–20 (AR[9–20])—Output	2-30
2.2.4.4	Buffer Control ($\overline{\text{BCTL}}[0-1]$)—Output	2-30
2.2.4.5	Column Address Strobe ($\overline{\text{CAS}}[0-7]$)—Output	2-31
2.2.4.6	SDRAM Clock Enable (CKE)—Output	2-31
2.2.4.7	SDRAM Chip Select ($\overline{\text{CS}}[0-7]$)—Output	2-31
2.2.4.8	SDRAM Data Input/Output Mask (DQM[0–7])—Output	2-32
2.2.4.9	Flash Output Enable ($\overline{\text{FOE}}$)—Output	2-32
2.2.4.10	Memory Address (MA[0–12])—Output	2-32
2.2.4.11	Memory Data Latch Enable ($\overline{\text{MDLE}}$)—Output	2-33
2.2.4.12	Data Parity/ECC (PAR[0–7])	2-33
2.2.4.12.1	Data Parity (PAR[0–7])—Output	2-33
2.2.4.12.2	Data Parity (PAR[0–7])—Input	2-33
2.2.4.13	Parity Path Read Enable ($\overline{\text{PPEN}}$)—Output	2-33
2.2.4.14	Row Address Strobe ($\overline{\text{RAS}}[0-7]$)—Output	2-34
2.2.4.15	ROM Bank 0 Select ($\overline{\text{RCS0}}$)—Output	2-34
2.2.4.16	ROM Bank 1 Select ($\overline{\text{RCS1}}$)—Output	2-34
2.2.4.17	Real Time Clock (RTC)—Input	2-35

Contents

Paragraph Number	Title	Page Number
2.2.4.18	SDRAM Internal Bank Select 0–1 (SDBA[0–1])—Output	2-35
2.2.4.19	SDRAM Column Address Strobe ($\overline{\text{SDCAS}}$)—Output	2-35
2.2.4.20	SDRAM Address (SDMA[0–12])—Output	2-35
2.2.4.21	SDRAM Row Address Strobe ($\overline{\text{SDRAS}}$)—Output	2-36
2.2.4.22	Write Enable ($\overline{\text{WE}}$)—Output	2-36
2.2.5	PCI Interface Signals	2-36
2.2.5.1	PCI Address/Data Bus (AD[31–0])	2-37
2.2.5.1.1	Address/Data (AD[31–0])—Output	2-37
2.2.5.1.2	Address/Data (AD[31–0])—Input	2-37
2.2.5.2	Command/Byte Enable ($\overline{\text{C/BE}}[3–0]$)	2-37
2.2.5.2.1	Command/Byte Enable ($\overline{\text{C/BE}}[3–0]$)—Output	2-37
2.2.5.2.2	Command/Byte Enable ($\overline{\text{C/BE}}[3–0]$)—Input	2-38
2.2.5.3	Device Select ($\overline{\text{DEVSEL}}$)	2-38
2.2.5.3.1	Device Select ($\overline{\text{DEVSEL}}$)—Output	2-38
2.2.5.3.2	Device Select ($\overline{\text{DEVSEL}}$)—Input	2-38
2.2.5.4	Frame ($\overline{\text{FRAME}}$)	2-39
2.2.5.4.1	Frame ($\overline{\text{FRAME}}$)—Output	2-39
2.2.5.4.2	Frame ($\overline{\text{FRAME}}$)—Input	2-39
2.2.5.5	PCI Bus Grant ($\overline{\text{GNT}}$)—Input	2-39
2.2.5.6	Initiator Ready ($\overline{\text{IRDY}}$)	2-39
2.2.5.6.1	Initiator Ready ($\overline{\text{IRDY}}$)—Output	2-40
2.2.5.6.2	Initiator Ready ($\overline{\text{IRDY}}$)—Input	2-40
2.2.5.7	Lock ($\overline{\text{LOCK}}$)—Input	2-40
2.2.5.8	Parity ($\overline{\text{PAR}}$)	2-40
2.2.5.8.1	Parity ($\overline{\text{PAR}}$)—Output	2-40
2.2.5.8.2	Parity ($\overline{\text{PAR}}$)—Input	2-41
2.2.5.9	Parity Error ($\overline{\text{PERR}}$)	2-41
2.2.5.9.1	Parity Error ($\overline{\text{PERR}}$)—Output	2-41
2.2.5.9.2	Parity Error ($\overline{\text{PERR}}$)—Input	2-41
2.2.5.10	PCI Bus Request ($\overline{\text{REQ}}$)—Output	2-41
2.2.5.11	System Error ($\overline{\text{SERR}}$)	2-42
2.2.5.11.1	System Error ($\overline{\text{SERR}}$)—Output	2-42
2.2.5.11.2	System Error ($\overline{\text{SERR}}$)—Input	2-42
2.2.5.12	Stop ($\overline{\text{STOP}}$)	2-42
2.2.5.12.1	Stop ($\overline{\text{STOP}}$)—Output	2-42
2.2.5.12.2	Stop ($\overline{\text{STOP}}$)—Input	2-42
2.2.5.13	Target Ready ($\overline{\text{TRDY}}$)	2-42
2.2.5.13.1	Target Ready ($\overline{\text{TRDY}}$)—Output	2-42
2.2.5.13.2	Target Ready ($\overline{\text{TRDY}}$)—Input	2-43
2.2.5.14	PCI Sideband Signals	2-43
2.2.5.14.1	Flush Request ($\overline{\text{FLSHREQ}}$)—Input	2-43

Contents

Paragraph Number	Title	Page Number
2.2.5.14.2	ISA Master ($\overline{\text{ISA_MASTER}}$)—Input	2-43
2.2.5.14.3	Memory Acknowledge ($\overline{\text{MEMACK}}$)—Output	2-44
2.2.5.14.4	Modified Memory Interrupt Request ($\overline{\text{PIRQ}}$)—Output	2-44
2.2.6	Interrupt, Clock, and Power Management Signals	2-44
2.2.7	External ECM Error Detect ($\overline{\text{BERR}}$)—Input	2-44
2.2.7.1	Test Clock ($\overline{\text{CKO}}$)—Output	2-45
2.2.7.2	Hard Reset ($\overline{\text{HRST}}$)—Input	2-45
2.2.7.3	Nonmaskable Interrupt ($\overline{\text{NMI}}$)—Input	2-45
2.2.7.4	Quiesce Acknowledge ($\overline{\text{QACK}}$)—Output	2-46
2.2.7.5	Quiesce Request ($\overline{\text{QREQ}}$)—Input	2-46
2.2.7.6	Suspend ($\overline{\text{SUSPEND}}$)—Input	2-46
2.2.7.7	System Clock ($\overline{\text{SYSCLK}}$)—Input	2-47
2.2.8	IEEE 1149.1 Interface Signals	2-47
2.2.8.1	JTAG Test Clock ($\overline{\text{TCK}}$)—Input	2-47
2.2.8.2	JTAG Test Data Output ($\overline{\text{TDO}}$)—Output	2-47
2.2.8.3	JTAG Test Data Input ($\overline{\text{TDI}}$)—Input	2-47
2.2.8.4	JTAG Test Mode Select ($\overline{\text{TMS}}$)—Input	2-48
2.2.8.5	JTAG Test Reset ($\overline{\text{TRST}}$)—Input	2-48
2.2.9	Configuration Signals	2-48
2.2.9.1	501-Mode ($\overline{\text{BCTL0}}$)—Input	2-48
2.2.9.2	Address Map ($\overline{\text{DBG0}}$)—Input	2-49
2.2.9.3	ROM Bank 0 Data Path Width ($\overline{\text{FOE}}$)—Input	2-49
2.2.9.4	60x Clock Phase ($\overline{\text{LBCLAIM}}$)—Input	2-49
2.2.9.5	Clock Mode ($\overline{\text{PLL[0-3]}}$)—Input	2-49
2.2.9.6	ROM Location ($\overline{\text{RCS0}}$)—Input	2-50
2.3	Clocking	2-50

Chapter 3 Device Programming

3.1	Address Maps	3-1
3.1.1	Address Map A	3-1
3.1.2	Address Map B	3-8
3.1.3	Emulation Mode Address Map	3-11
3.2	Configuration Registers	3-14
3.2.1	Configuration Register Access	3-14
3.2.1.1	Configuration Register Access in Little-Endian Mode	3-14
3.2.1.2	Configuration Register Access in Big-Endian Mode	3-16
3.2.2	Configuration Register Summary	3-17
3.2.3	PCI Registers	3-21
3.2.3.1	PCI Command Register	3-22
3.2.3.2	PCI Status Register	3-24

Contents

Paragraph Number	Title	Page Number
3.2.4	Performance Monitor Registers	3-25
3.2.4.1	Performance Monitor Command Register (CMDR)—0x48	3-25
3.2.4.2	Performance Monitor Mode Control Register (MMCR)—0x4C	3-26
3.2.4.3	Performance Monitor Counters (PMC0, PMC1, PMC2, PMC3)— 0x50, 0x54, 0x58, 0x5C.....	3-27
3.2.5	Power Management Configuration Registers (PMCRs).....	3-28
3.2.6	Output Driver Control Register (ODCR)—0x73.....	3-31
3.2.7	Error Handling Registers	3-33
3.2.7.1	ECC Single-Bit Error Registers	3-33
3.2.7.2	Error Enabling Registers.....	3-34
3.2.7.3	Error Detection Registers	3-36
3.2.7.4	Error Status Registers	3-38
3.2.8	Memory Interface Configuration Registers	3-40
3.2.8.1	Memory Boundary Registers	3-40
3.2.8.2	Memory Bank Enable Register.....	3-44
3.2.8.3	Memory Page Mode Register	3-45
3.2.8.4	Memory Control Configuration Registers	3-46
3.2.9	Processor Interface Configuration Registers	3-56
3.2.10	Alternate OS-Visible Parameters Registers	3-66
3.2.11	Emulation Support Configuration Registers.....	3-68
3.2.11.1	Modified Memory Status Register.....	3-70
3.2.12	External Configuration Registers.....	3-71

Chapter 4 Processor Bus Interface

4.1	MPC106 Processor Bus Configuration.....	4-1
4.1.1	Single-Processor System Configuration	4-1
4.1.2	Multiprocessor System Configuration	4-1
4.1.3	Multiprocessor System Configuration with External L2 Cache.....	4-3
4.1.4	Processor Bus Interface Configuration Registers	4-5
4.2	Processor Bus Protocol Overview	4-5
4.2.1	MPC106 Arbitration	4-6
4.2.2	Address Pipelining and Split-Bus Transactions.....	4-7
4.3	Address Tenure Operations.....	4-8
4.3.1	Address Arbitration.....	4-8
4.3.2	Address Transfer Attribute Signals.....	4-10
4.3.2.1	Transfer Type Signal Encodings.....	4-10
4.3.2.2	$\overline{\text{TBST}}$ and $\text{TSIZ}[0-2]$ Signals and Size of Transfer	4-13
4.3.2.3	Burst Ordering During Data Transfers	4-13
4.3.2.4	Effect of Alignment on Data Transfers.....	4-14

Contents

Paragraph Number	Title	Page Number
4.3.3	Address Transfer Termination	4-16
4.3.3.1	MPC106 Snoop Response	4-16
4.3.3.2	Address Tenure Timing Configuration	4-17
4.4	Data Tenure Operations	4-18
4.4.1	Data Bus Arbitration	4-18
4.4.2	Data Bus Transfers and Normal Termination	4-18
4.4.3	Data Tenure Timing Configurations	4-19
4.4.4	Data Bus Termination by \overline{TEA}	4-19
4.4.5	60x Local Bus Slave Support.....	4-20
4.4.5.1	60x Local Bus Slave Timing	4-21

Chapter 5 Secondary Cache Interface

5.1	L2 Cache Configurations	5-2
5.1.1	Write-Back Cache Operation	5-2
5.1.2	Write-Through Cache Operation	5-2
5.1.3	Synchronous Burst SRAMs	5-2
5.1.4	Pipelined Burst SRAMs.....	5-4
5.1.5	Asynchronous SRAMs	5-6
5.1.6	Two-Bank Support.....	5-7
5.2	Internal L2 Cache Controller Operation	5-8
5.2.1	L2 Cache Addressing.....	5-8
5.2.2	L2 Cache Line Status	5-9
5.2.3	L2 Cache Tag Lookup.....	5-9
5.2.4	L2 Cache Castout Operations	5-10
5.2.5	L2 Cache Parity	5-10
5.2.6	L2 Cache Interface and Interrupt Vector Relocation	5-11
5.3	L2 Cache Response to Bus Operations.....	5-11
5.3.1	Write-Back L2 Cache Response	5-11
5.3.2	Write-Through L2 Cache Response.....	5-18
5.4	L2 Cache Interface Parameters	5-21
5.4.1	L2 Cache Interface Control Parameters	5-22
5.4.2	L2 Cache Interface Initialization Parameters.....	5-22
5.4.2.1	CF_L2_HIT_DELAY	5-24
5.4.2.2	CF_DOE	5-24
5.4.2.3	CF_WDATA	5-24
5.4.2.4	CF_WMODE.....	5-25
5.4.2.4.1	Normal Write Timing without Partial Update (CF_WMODE = 0).....	5-25
5.4.2.4.2	Normal Write Timing (CF_WMODE = 1)	5-26

Contents

Paragraph Number	Title	Page Number
5.4.2.4.3	Delayed Write Timing (CF_WMODE = 2)	5-27
5.4.2.4.4	Early Write Timing (CF_WMODE = 3).....	5-28
5.5	L2 Cache Interface Timing Examples	5-29
5.5.1	Synchronous Burst SRAM L2 Cache Timing	5-29
5.5.1.1	L2 Cache Read Hit Timing	5-29
5.5.1.2	L2 Cache Write Hit Timing	5-31
5.5.1.3	L2 Cache Line Update Timing	5-32
5.5.1.4	L2 Cache Line Castout Timing.....	5-33
5.5.1.5	L2 Cache Hit Timing Following PCI Read Snoop	5-35
5.5.1.6	L2 Cache Line Push Timing Following PCI Write Snoop	5-36
5.5.1.7	L2 Cache Line Invalidate Timing Following PCI Write-with-Invalidate Snoop	5-37
5.5.2	Asynchronous SRAM L2 Cache Timing.....	5-38
5.5.2.1	Burst Read Timing.....	5-38
5.5.2.2	L2 Cache Burst Read Line Update Timing	5-39
5.5.2.3	Burst Write Timing	5-40
5.6	External L2 Cache Controller Operation	5-41
5.6.1	External L2 Cache Operation	5-42
5.6.2	External L2 Cache Controller Interface Parameters	5-42

Chapter 6 Memory Interface

6.1	Overview.....	6-1
6.2	Memory Interface Signal Buffering.....	6-2
6.2.1	Flow-Through Buffers	6-3
6.2.2	Transparent Latch Buffers	6-4
6.2.3	Registered Buffers	6-5
6.2.4	Parity/ECC Path Read Control	6-8
6.3	Memory Interface Signal Connections	6-8
6.4	DRAM/EDO Interface Operation	6-10
6.4.1	Supported DRAM/EDO Organizations	6-12
6.4.2	DRAM/EDO Address Multiplexing	6-13
6.4.3	DRAM/EDO Power-On Initialization	6-15
6.4.3.1	Supported Memory Interface Configurations.....	6-16
6.4.4	DRAM/EDO Interface Timing	6-16
6.4.5	DRAM/EDO Burst Wrap.....	6-23
6.4.6	DRAM/EDO Latency	6-23
6.4.7	DRAM/EDO Page Mode Retention	6-24

Contents

Paragraph Number	Title	Page Number
6.4.8	DRAM/EDO Parity and RMW Parity	6-25
6.4.8.1	RMW Parity Latency Considerations	6-25
6.4.9	Error Checking and Correction—ECC	6-26
6.4.9.1	DRAM/EDO Interface Timing with ECC	6-26
6.4.10	DRAM/EDO Refresh.....	6-30
6.4.10.1	DRAM/EDO Refresh Timing	6-30
6.4.10.2	DRAM/EDO Refresh and Power Saving Modes.....	6-31
6.4.10.2.1	Self-Refresh in Sleep and Suspend Modes.....	6-32
6.4.10.2.2	RTC Refresh in Suspend Mode	6-33
6.4.11	Processor-to-DRAM Transaction Examples.....	6-33
6.4.12	PCI-to-DRAM Transaction Examples	6-37
6.5	SDRAM Interface Operation	6-44
6.5.1	Supported SDRAM Organizations	6-46
6.5.2	SDRAM Address Multiplexing	6-47
6.5.3	SDRAM Burst and Single-Beat Transactions.....	6-49
6.5.4	SDRAM Page Mode Retention.....	6-49
6.5.5	SDRAM Power-On Initialization	6-51
6.5.6	JEDEC Standard SDRAM Interface Commands.....	6-53
6.5.7	SDRAM Interface Timing	6-55
6.5.7.1	Processor-to-SDRAM Transaction Examples	6-55
6.5.7.2	SDRAM Mode-Set Command Timing	6-60
6.5.8	SDRAM Parity and RMW Parity	6-61
6.5.8.1	RMW Parity Latency Considerations.....	6-62
6.5.9	External Error Checking Module (ECM) Support.....	6-62
6.5.9.1	External ECM Timing Examples.....	6-64
6.5.10	SDRAM Refresh.....	6-66
6.5.10.1	SDRAM Refresh Timing	6-68
6.5.10.2	SDRAM Refresh and Power Saving Modes.....	6-68
6.6	ROM/Flash Interface Operation	6-70
6.6.1	ROM/Flash Cacheability	6-73
6.6.2	64-Bit ROM/Flash Interface Timing	6-73
6.6.3	Processor Read from 64-Bit ROM Transaction Examples.....	6-75
6.6.4	Eight-Bit ROM/Flash Interface Timing.....	6-78
6.6.5	Processor Read from 8-Bit ROM Transaction Examples	6-79
6.6.6	ROM/Flash Interface Write Operations.....	6-83
6.6.6.1	ROM/Flash Interface Write Timing.....	6-83
6.6.6.2	Processor Write to 8-Bit ROM Transaction Example	6-84

Contents

Paragraph Number	Title	Page Number
Chapter 7		
PCI Bus Interface		
7.1	PCI Interface Overview	7-1
7.1.1	MPC106 as a PCI Master.....	7-2
7.1.2	MPC106 as a PCI Target.....	7-2
7.2	PCI Bus Arbitration	7-3
7.3	PCI Bus Protocol	7-3
7.3.1	Basic Transfer Control.....	7-3
7.3.2	PCI Bus Commands.....	7-4
7.3.3	Addressing	7-6
7.3.3.1	Memory Space Addressing.....	7-6
7.3.3.2	I/O Space Addressing	7-7
7.3.3.3	Configuration Space Addressing	7-7
7.3.4	Device Selection	7-7
7.3.5	Byte Alignment.....	7-8
7.3.6	Bus Driving and Turnaround	7-8
7.4	PCI Bus Transactions.....	7-9
7.4.1	PCI Read Transactions.....	7-9
7.4.2	PCI Write Transactions	7-10
7.4.3	Transaction Termination	7-11
7.4.3.1	Master-Initiated Termination	7-11
7.4.3.2	Target-Initiated Termination	7-12
7.4.4	Fast Back-to-Back Transactions	7-14
7.4.5	Configuration Cycles	7-15
7.4.5.1	The PCI Configuration Space Header	7-15
7.4.5.2	Accessing the PCI Configuration Space.....	7-17
7.4.6	Other Bus Transactions.....	7-21
7.4.6.1	Interrupt Acknowledge Transactions.....	7-21
7.4.6.2	Special-Cycle Transactions.....	7-22
7.5	Exclusive Access	7-23
7.5.1	Starting an Exclusive Access	7-23
7.5.2	Continuing an Exclusive Access.....	7-24
7.5.3	Completing an Exclusive Access.....	7-24
7.5.4	Attempting to Access a Locked Target.....	7-24
7.5.5	Exclusive Access and the MPC106	7-24
7.6	PCI Error Functions	7-25
7.6.1	PCI Parity.....	7-25
7.6.2	Error Reporting.....	7-26
7.7	MPC106-Implemented PCI Sideband Signals.....	7-27
7.7.1	ISA_MASTER.....	7-27
7.7.2	FLSHREQ and MEMACK.....	7-27

Contents

Paragraph Number	Title	Page Number
7.8	Emulation Support	7-28
7.8.1	PCI Address Decoding.....	7-28
7.8.2	Interrupt Vector Relocation.....	7-28
7.8.3	Modified Memory Status Register.....	7-28
7.8.4	Curious Code Protection.....	7-31
7.9	Processor-to-PCI Transaction Examples	7-31

Chapter 8 Internal Control

8.1	Internal Buffers	8-1
8.1.1	60x Processor/System Memory Buffers	8-2
8.1.2	60x Processor/PCI Buffers.....	8-3
8.1.2.1	Processor-to-PCI-Read Buffer (PRPRB).....	8-4
8.1.2.2	Processor-to-PCI-Write Buffers (PRPWBs).....	8-5
8.1.3	PCI/System Memory Buffers.....	8-6
8.1.3.1	PCI-to-System-Memory-Read Buffer (PCMRB).....	8-7
8.1.3.1.1	Speculative PCI Reads from System Memory	8-8
8.1.3.2	PCI-to-System-Memory-Write Buffers (PCMWBs)	8-9
8.2	Internal Arbitration	8-9

Chapter 9 Error Handling

9.1	Priority of Externally-Generated Interrupts	9-2
9.2	Interrupt and Error Signals	9-3
9.2.1	System Reset.....	9-3
9.2.2	60x Processor Bus Error Signals	9-3
9.2.2.1	Machine Check (\overline{MCP}).....	9-3
9.2.2.2	Transfer Error Acknowledge (\overline{TEA}).....	9-4
9.2.3	PCI Bus Error Signals.....	9-5
9.2.3.1	System Error (\overline{SERR})	9-5
9.2.3.2	Parity Error (PERR).....	9-5
9.2.3.3	Nonmaskable Interrupt (NMI).....	9-5
9.3	Error Reporting	9-6
9.3.1	60x Processor Interface.....	9-7
9.3.1.1	Unsupported 60x Bus Transaction Error	9-7
9.3.1.2	Illegal L2 Copyback Error	9-7
9.3.1.3	Flash Write Error	9-7
9.3.2	Memory Interface	9-7
9.3.2.1	System Memory Read Data Parity Error	9-8
9.3.2.2	L2 Cache Read Data Parity Error	9-8

Contents

Paragraph Number	Title	Page Number
9.3.2.3	System Memory ECC Error.....	9-8
9.3.2.3.1	External ECM Errors	9-9
9.3.2.4	System Memory Select Error.....	9-9
9.3.2.5	System Memory Refresh Overflow Error.....	9-9
9.3.3	PCI Interface	9-9
9.3.3.1	Address Parity Error	9-10
9.3.3.2	Data Parity Error	9-10
9.3.3.3	Master-Abort Transaction Termination	9-10
9.3.3.4	Received Target-Abort Error	9-11
9.3.3.5	NMI (Nonmaskable Interrupt).....	9-12
9.4	Interrupt Latencies	9-12
9.5	Example Signal Connections	9-12

Chapter 10 Performance Monitor

10.1	Performance Monitor Operation	10-1
10.2	Performance Monitor Events	10-2
10.2.1	Command Type 0 Events	10-3
10.2.2	Command Type 1 Events	10-4
10.2.3	Threshold Events	10-8
10.2.4	Burstiness.....	10-8
10.2.5	Counter Overflow	10-9

Appendix A Power Management

A.1	MPC106 Power Modes	A-1
A.1.1	MPC106 Power Mode Transition	A-1
A.1.2	Full-On Mode	A-3
A.1.3	Doze Mode.....	A-3
A.1.4	Nap Mode	A-3
A.1.5	Sleep Mode	A-4
A.1.6	Suspend Mode.....	A-5
A.2	MPC106 Power Management Support	A-6
A.2.1	Power Management Configuration Registers	A-6
A.2.2	Clock Configuration	A-6
A.2.3	PCI Address Bus Decoding	A-7
A.2.4	PCI Bus Special-Cycle Operations	A-7
A.2.5	Processor Bus Request Monitoring.....	A-7

Contents

Paragraph Number	Title	Page Number
A.2.6	Memory Refresh Operations in Sleep/Suspend Mode.....	A-8
A.2.7	PCI Bus Parking in Sleep and Suspend Modes	A-8
A.2.8	Device Drivers	A-8

Appendix B Bit and Byte Ordering

B.1	Big-Endian Mode.....	B-1
B.2	Little-Endian Mode.....	B-5

Appendix C JTAG/Testing Support

C.1	JTAG Interface Description	C-1
C.1.1	JTAG Signals.....	C-2
C.1.2	JTAG Registers and Scan Chains	C-2
C.1.2.1	Bypass Register	C-2
C.1.2.2	Boundary-Scan Registers.....	C-2
C.1.2.3	Instruction Register.....	C-3
C.1.3	TAP Controller	C-3

Appendix D Initialization Example

Appendix E Revision History

E.1	Revision Changes From Revision 1 to Revision 2	D-1
-----	--	-----

Glossary of Terms and Abbreviations

Index



Contents

**Paragraph
Number**

Title

**Page
Number**

Freescale Semiconductor, Inc.

Figures

Figure Number	Title	Page Number
1-1	MPC106 Block Diagram.....	1-2
2-1	MPC106 Signal Groupings	2-3
2-2	SYSCLK Input with Internal Multiples	2-50
3-1	Address Map A (Contiguous Map).....	3-4
3-2	Address Map A (Discontiguous Map)	3-5
3-3	PCI Memory Map (Address Map A)	3-6
3-4	PCI I/O Map (Address Map A).....	3-7
3-5	Address Map B.....	3-10
3-6	Emulation Mode Address Map	3-13
3-7	MPC106 Configuration Space	3-21
3-8	PCI Command Register	3-22
3-9	PCI Status Register	3-24
3-10	Performance Monitor Command Register (CMDR)—0x48.....	3-25
3-11	Performance Monitor Mode Control Register (MMCR)—0x4C	3-26
3-12	Performance Monitor Counter (PMcN)	3-28
3-13	Power Management Configuration Register 1 (PMCR1)	3-28
3-14	Power Management Configuration Register 2 (PMCR2)	3-30
3-15	Output Driver Control Register (ODCR)—0x73.....	3-31
3-16	ECC Single-Bit Error Counter Register—0xB8	3-33
3-17	ECC Single-Bit Error Trigger Register—0xB9	3-33
3-18	Error Enabling Register 1 (ErrEnR1).....	3-34
3-19	Error Enabling Register 2 (ErrEnR2).....	3-35
3-20	Error Detection Register 1 (ErrDR1)—0xC1	3-36
3-21	Error Detection Register 2 (ErrDR2)—0xC5	3-37
3-22	60x Bus Error Status Register—0xC3	3-38
3-23	PCI Bus Error Status Register—0xC7	3-39
3-24	60x/PCI Error Address Register—0xC8.....	3-39
3-25	Memory Starting Address Register 1—0x80.....	3-40
3-26	Memory Starting Address Register 2—0x84.....	3-41
3-27	Extended Memory Starting Address Register 1—0x88.....	3-41
3-28	Extended Memory Starting Address Register 2—0x8C	3-41
3-29	Memory Ending Address Register 1—0x90.....	3-42
3-30	Memory Ending Address Register 2—0x94.....	3-42

Figures

Figure Number	Title	Page Number
3-31	Extended Memory Ending Address Register 1—0x98.....	3-43
3-32	Extended Memory Ending Address Register 2—0x9C.....	3-43
3-33	Memory Bank Enable Register—0xA0.....	3-44
3-34	Memory Page Mode Register—0xA3.....	3-45
3-35	Memory Control Configuration Register 1 (MCCR1)—0xF0.....	3-46
3-36	Memory Control Configuration Register 2 (MCCR2)—0xF4.....	3-49
3-37	Memory Control Configuration Register 3 (MCCR3)—0xF8.....	3-51
3-38	Memory Control Configuration Register 4 (MCCR4)—0xFC.....	3-54
3-39	Processor Interface Configuration Register 1 (PICR1)—0xA8.....	3-57
3-40	Processor Interface Configuration Register 2 (PICR2)—0xAC.....	3-61
3-41	Alternate OS-Visible Parameters Register 1—0xBA.....	3-66
3-42	Alternate OS-Visible Parameters Register 2—0xBB.....	3-67
3-43	Emulation Support Configuration Register 1 (ESCR1)—0xE0.....	3-68
3-44	Emulation Support Configuration Register 2 (ESCR2)—0xE8.....	3-69
3-45	Modified Memory Status Register—0xE4/0xEC.....	3-70
3-46	External Configuration Register 1—Port 0x092.....	3-71
3-47	External Configuration Register 2—Port 0x81C.....	3-72
3-48	External Configuration Register 3—Port 0x850.....	3-72
4-1	Single-Processor Configuration with Optional L2 Cache.....	4-2
4-2	Multiprocessor Configuration.....	4-3
4-3	Multiprocessor Configuration with External L2 Cache.....	4-4
4-4	Overlapping Tenures on the 60x Bus for a Single-Beat Transfer.....	4-6
4-5	Address Bus Arbitration with Dual Processors.....	4-9
4-6	Address Pipelining.....	4-10
4-7	Snooped Address Transaction with $\overline{\text{ARTRY}}$ and L1 Cache Copyback.....	4-17
4-8	Single-Beat and Burst Data Transfers.....	4-19
4-9	Data Tenure Terminated by Assertion of $\overline{\text{TEA}}$	4-20
4-10	Local Bus Slave Transaction.....	4-23
4-11	60x Bus State Diagram.....	4-24
5-1	Typical L2 Cache Using Burst SRAM (Write-Back).....	5-3
5-2	Typical L2 Cache Using Pipelined Burst SRAM (Write-Back, $\overline{\text{ADSC}}$ Only).....	5-4
5-3	Alternate L2 Cache Using Pipelined Burst SRAM (Write-Back Using $\overline{\text{ADSP}}$).....	5-5
5-4	Typical L2 Cache Using Asynchronous SRAM (Write-Back).....	5-6
5-5	512-Kbyte, Two-Bank, L2 Cache Using Synchronous Burst SRAM (Write-Back)...	5-7
5-6	1-Mbyte, Two-Bank, L2 Cache Using Pipelined Burst SRAM (Write-Back, $\overline{\text{ADSC}}$ Only).....	5-8
5-7	$\overline{\text{HIT}}$ and $\overline{\text{DIRTY_IN}}$ Delay Configuration.....	5-24
5-8	External Byte Decode Logic Requiring $\text{CF_WMODE} = 1$	5-26
5-9	Normal Write Timing ($\text{CF_WMODE} = 0$ or 1).....	5-26

Figures

Figure Number	Title	Page Number
5-10	External Byte Decode Logic Requiring CF_WMODE = 2	5-27
5-11	Delayed Write Timing (CF_WMODE = 2)	5-27
5-12	External Byte Decode Logic Requiring CF_WMODE = 3	5-28
5-13	Early Write Timing (CF_WMODE = 3)	5-28
5-14	Timing Diagram Legend	5-29
5-15	L2 Cache Read Hit Timing with CF_DOE = 0.....	5-29
5-16	L2 Cache Read Hit Timing with CF_DOE = 1	5-30
5-17	L2 Cache Write Hit Timing.....	5-31
5-18	L2 Cache Line Update Timing.....	5-32
5-19	L2 Cache Line Castout Timing	5-33
5-20	L2 Cache Line Castout Timing with No $\overline{\text{ARTRY}}$	5-34
5-21	L2 Cache Hit Timing Following PCI Read Snoop	5-35
5-22	Modified L2 Cache Line Push Timing Following PCI Write Snoop.....	5-36
5-23	L2 Cache Line Invalidate Timing Following PCI Write-with-Invalidate Snoop.....	5-37
5-24	L2 Cache Burst Read Timing with CF_DOE = 0	5-38
5-25	L2 Cache Burst Read Timing with CF_DOE = 1	5-39
5-26	L2 Cache Burst Read Line Update Timing with CF_WDATA = 0	5-39
5-27	L2 Cache Burst Read Line Update Timing with CF_WDATA = 1	5-40
5-28	L2 Cache Burst Write Timing with CF_WDATA = 0 or 1	5-40
5-29	Typical External L2 Cache Configuration	5-41
6-1	Flow-Through Buffer	6-4
6-2	Transparent Latch-Type Buffer	6-5
6-3	16952-Type Registered Buffer	6-6
6-4	16601-Type Registered Buffer	6-7
6-5	16501-Type Universal Transceiver/Buffer.....	6-8
6-6	16-Mbyte DRAM System with Parity	6-11
6-7	DRAM/EDO Address Multiplexing	6-14
6-8	DRAM Single-Beat Read Timing—No ECC	6-19
6-9	DRAM Burst Read Timing—No ECC	6-20
6-10	EDO Burst Read Timing—No ECC	6-21
6-11	DRAM Single-Beat Write Timing—No ECC.....	6-22
6-12	DRAM/EDO Burst Write Timing—No ECC, CPX = 1	6-23
6-13	DRAM Burst Read with ECC	6-28
6-14	EDO Burst Read Timing with ECC	6-29
6-15	DRAM Single-Beat Write Timing with RMW or ECC Enabled.....	6-29
6-16	DRAM/EDO Bank Staggered CBR Refresh Timing.....	6-31
6-17	DRAM/EDO Self-Refresh Timing in Sleep and Suspend Modes	6-33
6-18	Suspend Mode—Real Time Clock Refresh	6-33

Figures

Figure Number	Title	Page Number
6-19	Processor Burst Reads from Memory—60-ns DRAM with Flow-Through Buffers.....	6-34
6-19	(Continued) Processor Burst Reads from Memory—60-ns DRAM with Flow-Through Buffers.....	6-35
6-20	Processor Burst Write to Memory—60-ns DRAM with Flow-Through Buffers.....	6-36
6-21	PCI Reads from Memory—Speculative Reads Enabled—60-ns DRAM with Flow-Through Buffers.....	6-38
6-21	(Continued) PCI Reads from Memory—Speculative Reads Enabled—60-ns DRAM with Flow-Through Buffers.....	6-39
6-22	PCI Reads from Memory—Speculative Reads Disabled—60-ns DRAM with Flow-Through Buffers.....	6-40
6-22	(Continued) PCI Reads from Memory—Speculative Reads Disabled—60-ns DRAM with Flow-Through Buffers.....	6-41
6-23	PCI Writes to Memory—60-ns DRAM with Flow-Through Buffers.....	6-42
6-23	(Continued) PCI Writes to Memory—60-ns DRAM with Flow-Through	6-43
6-24	128-Mbyte SDRAM System with Parity	6-45
6-25	SDRAM Address Multiplexing	6-49
6-26	PGMAX Parameter Setting for SDRAM Interface.....	6-50
6-27	SDRAM Burst Read Timing.....	6-56
6-28	SDRAM Burst Write Timing	6-57
6-29	SDRAM Burst Read Followed By Burst Write Timing.....	6-58
6-30	SDRAM Single-Beat Read Timing.....	6-59
6-31	SDRAM Single-Beat Write Timing.....	6-60
6-32	SDRAM Mode-Set Command Timing	6-61
6-33	External ECM Block Diagram.....	6-63
6-34	Burst Read Timing with External ECM.....	6-64
6-35	Single-Beat Read Timing with External ECM.....	6-65
6-36	Burst Write Timing with External ECM	6-65
6-37	Single-Beat Write Timing with External ECM.....	6-65
6-38	SDRAM Refresh Period.....	6-66
6-39	SDRAM Bank-Staggered CBR Refresh Timing.....	6-68
6-40	SDRAM Self-Refresh Entry Timing.....	6-69
6-41	SDRAM Self-Refresh Exit Timing.....	6-70
6-42	16-Mbyte ROM System	6-71
6-43	1-Mbyte Flash System	6-72
6-44	64-Bit ROM/Flash Interface—Nonburst ROM Timing.....	6-74
6-45	64-Bit ROM/Flash Interface—Burst ROM Timing.....	6-75
6-46	64-Bit ROM, 32-Byte Read, Flow-Through Buffers	6-76
6-47	64-Bit ROM, 8-Byte Read, Flow-Through Buffers	6-77

Figures

Figure Number	Title	Page Number
6-48	8-Bit ROM/Flash Interface—Single-Byte Read Timing	6-78
6-49	8-Bit ROM/Flash Interface—Half-Word Read Timing	6-79
6-50	8-Bit ROM/Flash Interface—Burst Read Timing	6-79
6-51	8-Bit ROM, 1-Byte Read, Flow-Through Buffers	6-80
6-52	8-Bit ROM, 8-Byte Read, Flow-Through Buffers	6-81
6-52	(Continued) 8-Bit ROM, 8-Byte Read, Flow-Through Buffers.....	6-82
6-53	Flash Memory Write Timing.....	6-84
6-54	8-Bit Flash, 1-Byte Write Followed by 1-Byte Read, Flow-Through Buffers	6-85
7-1	PCI Single-Beat Read Transaction.....	7-10
7-2	PCI Burst Read Transaction.....	7-10
7-3	PCI Single-Beat Write Transaction.....	7-11
7-4	PCI Burst Write Transaction	7-11
7-5	PCI Target-Initiated Terminations.....	7-14
7-6	Standard PCI Configuration Header	7-16
7-7	CONFIG_ADDR Register Format.....	7-18
7-8	Type 0 Configuration Translation	7-19
7-9	Direct-Access PCI Configuration Transaction.....	7-21
7-10	PCI Parity Operation.....	7-26
7-11	Modified Memory Tracking States	7-29
7-12	Processor Burst Read to Device on PCI Bus	7-32
7-13	Processor Burst Write to Device on PCI Bus.....	7-33
7-14	Processor Read from PCI with Master-Abort	7-34
8-1	MPC106 Internal Buffer Organization.....	8-2
8-2	60x Processor/System Memory Buffers.....	8-2
8-3	60x Processor/PCI Buffers.....	8-4
8-4	PCI/System Memory Buffers.....	8-6
9-1	Internal Interrupt Management Block Diagram.....	9-2
9-2	Example Interrupt Signal Configuration—603-/604-/740-/750-Based Systems	9-12
9-3	Example Interrupt Signal Configuration—601-Based Systems	9-13
10-1	Pipelined Processor Transaction and End-of-Data (EOD).....	10-2
10-2	Processor Latency	10-3
10-3	PCI Latency.....	10-3
10-4	Command Type 0 Threshold Example.....	10-8
10-5	Processor Burstiness Example	10-9
10-6	Overflow Example	10-10
A-1	MPC106 Power Modes	A-2
B-1	Four-Byte Transfer to PCI Memory Space—Big-Endian Mode	B-3
B-2	Big-Endian Memory Image in System Memory.....	B-4
B-3	Big-Endian Memory Image in Big-Endian PCI Memory Space	B-4
B-4	Munged Memory Image in System Memory	B-6

Figures

Figure Number	Title	Page Number
B-5	Little-Endian Memory Image in Little-Endian PCI Memory Space.....	B-7
B-6	One-Byte Transfer to PCI Memory Space—Little-Endian Mode	B-8
B-7	Two-Byte Transfer to PCI Memory Space—Little-Endian Mode	B-9
B-8	Four-Byte Transfer to PCI Memory Space—Little-Endian Mode.....	B-10
B-9	One-Byte Transfer to PCI I/O Space—Little-Endian Mode	B-11
B-10	Two-Byte Transfer to PCI I/O Space—Little-Endian Mode	B-12
B-11	Four-Byte Transfer to PCI I/O Space—Little-Endian Mode	B-13
C-1	JTAG Interface Block Diagram	C-1

Tables

Table Number	Title	Page Number
i	Acronyms and Abbreviated Terms.....	xxxi
2-1	MPC106 Signal Cross Reference.....	2-4
2-2	Output Signal States During System Reset.....	2-9
2-3	Data Bus Byte Lane Assignments.....	2-14
2-4	PCI Command Encodings.....	2-37
3-1	Address Map A—Processor View.....	3-2
3-2	Address Map A—PCI Memory Master View.....	3-2
3-3	Address Map A—PCI I/O Master View.....	3-3
3-4	Address Map B—Processor View.....	3-8
3-5	Address Map B—PCI Memory Master View.....	3-8
3-6	Address Map B—PCI I/O Master View.....	3-9
3-7	Emulation Mode Address Map—Processor View.....	3-11
3-8	Emulation Mode Address Map—PCI Memory Master View.....	3-12
3-9	Emulation Mode Address Map—PCI I/O Master View.....	3-12
3-10	MPC106 Configuration Registers.....	3-17
3-11	PCI Configuration Space Header Summary.....	3-21
3-12	Bit Settings for PCI Command Register—0x04.....	3-23
3-13	Bit Settings for PCI Status Register—0x06.....	3-24
3-14	Bit Settings for CMDR—0x48.....	3-25
3-15	Bit Settings for MMCR—0x4C.....	3-27
3-16	Bit Settings for Power Management Configuration Register 1—0x70.....	3-29
3-17	Bit Settings for Power Management Configuration Register 2—0x72.....	3-31
3-18	Bit Settings for ODCR—0x73.....	3-31
3-19	Bit Settings for ECC Single-Bit Error Counter Register—0xB8.....	3-33
3-20	Bit Settings for ECC Single-Bit Error Trigger Register—0xB9.....	3-33
3-21	Bit Settings for Error Enabling Register 1 (ErrEnR1)—0xC0.....	3-34
3-22	Bit Settings for Error Enabling Register 2 (ErrEnR2)—0xC4.....	3-35
3-23	Bit Settings for Error Detection Register 1 (ErrDR1)—0xC1.....	3-37
3-24	Bit Settings for Error Detection Register 2 (ErrDR2)—0xC5.....	3-38
3-25	Bit Settings for 60x Bus Error Status Register—0xC3.....	3-39
3-26	Bit Settings for PCI Bus Error Status Register—0xC7.....	3-39
3-27	Bit Settings for 60x/PCI Error Address Register—0xC8.....	3-40
3-28	Bit Settings for Memory Starting Address Registers 1 and 2.....	3-41
3-29	Bit Settings for Extended Memory Starting Address Registers 1 and 2.....	3-42
3-30	Bit Settings for Memory Ending Address Registers 1 and 2.....	3-43

Tables

Table Number	Title	Page Number
3-31	Bit Settings for Extended Memory Ending Address Registers 1 and 2	3-44
3-32	Bit Settings for Memory Bank Enable Register—0xA0.....	3-45
3-33	Bit Settings for Memory Page Mode Register—0xA3	3-46
3-34	Bit Settings for MCCR1—0xF0	3-47
3-35	Bit Settings for MCCR2 —0xF4	3-49
3-36	Bit Settings for MCCR3—0xF8	3-51
3-37	Bit Settings for MCCR4—0xFC.....	3-54
3-38	Bit Settings for PICR1—0xA8	3-57
3-39	Processor/L2 Configurations.....	3-60
3-40	Bit Settings for PICR2—0xAC.....	3-62
3-41	Bit Settings for Alternate OS-Visible Parameters Register 1—0xBA.....	3-67
3-42	Bit Settings for Alternate OS-Visible Parameters Register 2—0xBB	3-67
3-43	Bit Settings for ESCR1—0xE0.....	3-68
3-44	Bit Settings for ESCR2—0xE8.....	3-70
3-45	Bit Settings for Modified Memory Status Register—0xE4/0xEC.....	3-70
3-46	Bit Settings for External Configuration Register 1—Port 0x092	3-71
3-47	Bit Settings for External Configuration Register 2—Port 0x81C.....	3-72
3-48	Bit Settings for External Configuration Register 3—Port 0x850	3-73
4-1	MPC106 Responses to 60x Transfer Type Signals	4-10
4-2	Transfer Type Encodings Generated by the MPC106.....	4-12
4-3	MPC106 Transfer Size Encodings	4-13
4-4	MPC106 Burst Ordering	4-13
4-5	Aligned Data Transfers	4-14
4-6	Misaligned Data Transfers (4-Byte Examples).....	4-15
5-1	60x to Tag and Data RAM Addressing for 4-Gbyte Cacheable Address Space.....	5-9
5-2	Write-Back L2 Cache Response	5-12
5-3	Write-Through L2 Cache Response.....	5-18
6-1	Buffer Configurations	6-2
6-2	Memory Address Signal Mappings.....	6-9
6-3	Supported DRAM/EDO Device Configurations.....	6-12
6-4	DRAM/EDO Address Multiplexing	6-13
6-5	Supported Memory Interface Configurations	6-16
6-6	Suggested DRAM Timing Configurations.....	6-17
6-7	DRAM/EDO Timing Parameters	6-17
6-8	Estimated DRAM Latency.....	6-24
6-9	Estimated EDO Latency.....	6-24
6-10	Suggested DRAM Refresh Timing Configurations	6-31
6-11	DRAM/EDO Power Saving Modes Refresh Configuration	6-32
6-12	Supported SDRAM Device Configurations.....	6-46
6-13	SDRAM Address Multiplexing	6-47
6-14	SDRAM Command Encodings.....	6-54

Tables

Table Number	Title	Page Number
6-15	MPC106 Parameter Values for External ECM	6-64
6-16	SDRAM Power Saving Modes Refresh Configuration	6-69
7-1	PCI Bus Commands	7-4
7-2	PCI Configuration Space Header Summary	7-16
7-3	CONFIG_ADDR Register Fields	7-18
7-4	Type 0 Configuration—Device Number to IDSEL Translation	7-20
7-5	Special-Cycle Message Encodings	7-23
8-1	Snooping Behavior Caused by a Hit in an Internal Buffer	8-7
8-2	Internal Arbitration Priorities	8-10
9-1	Externally-Generated Interrupt Priorities	9-2
10-1	Command Type 0—Processor Transactions	10-4
10-2	Command Type 1—Event Encodings	10-5
B-1	Byte Lane Translation in Big-Endian Mode	B-2
B-2	Processor Address Modification for Individual Aligned Scalars	B-5
B-3	MPC106 Address Modification for Individual Aligned Scalars	B-5
B-4	Byte Lane Translation in Little-Endian Mode	B-6



Tables

**Table
Number**

Title

**Page
Number**

Freescale Semiconductor, Inc.

About This Book

The primary objective of this user's manual is to describe the functionality of the MPC106 PCI bridge/memory controller (PCIB/MC) for use by systems designers and software developers. The MPC106 is one device in a family of products that provides system-level support for industry-standard interfaces to be used with PowerPC™ microprocessors. Note that this document describes Rev. 4.0 of the MPC106. Earlier revisions are slightly different.

In this manual, the term '60x' is used to denote a 32-bit microprocessor from the PowerPC architecture family that conforms to the bus interface of the MPC601, MPC603, MPC604, MPC740, or MPC750 microprocessors. Note that this does not include the MPC602 microprocessor which has a multiplexed address/data bus. The 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- and double-precision).

It must be kept in mind that each processor is a unique implementation of the PowerPC architecture. It is beyond the scope of this manual to provide a thorough description of the PowerPC architecture; refer to the *Programming Environments Manual* for more information about the architecture. It is also beyond the scope of this manual to provide a thorough description of the PCI local bus; refer to *PCI Local Bus Specification* for more information about the PCI bus.

To locate published errata or updates for this document, refer to the website at <http://www.motorola.com/semiconductors>.

Audience

This manual is intended for system software and hardware developers who want to develop products incorporating microprocessors that implement the PowerPC architecture and the PCI bus. It is assumed that the reader understands operating systems, microprocessor system design, and the basic principles of reduced instruction set computing (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 MPC106.
- Chapter 2, “Signal Descriptions,” provides descriptions of individual signals of the MPC106.
- Chapter 3, “Device Programming,” is useful for software engineers who need to understand the address space and functionality of the registers implemented in the MPC106.
- Chapter 4, “Processor Bus Interface,” describes the interaction between the MPC106 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 MPC106 to DRAM, EDO, SDRAM, ROM, and Flash ROM devices.
- Chapter 7, “PCI Bus Interface,” describes the MPC106 as a bridge from the 60x processor bus to the PCI bus and the MPC106 as a PCI agent.
- Chapter 8, “Internal Control,” describes the internal buffers between the interfaces of the MPC106.
- Chapter 9, “Error Handling,” describes how the MPC106 handles error detection and reporting on the three primary interfaces—processor interface, memory interface, and PCI interface.
- Chapter 10, “Performance Monitor,” describes the operation of the built-in performance monitor facility of the MPC106.
- Appendix A, “Power Management,” provides information about power saving modes for the MPC106.
- Appendix B, “Bit and Byte Ordering,” describes big- and little-endian byte ordering and the implications on systems using the MPC106.
- 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.
- Appendix E, “Revision History,” lists the major differences between Revision 1 and Revision 2 of the *MPC106 PCI Bridge/Memory Controller User’s Manual*.
- This manual also includes a glossary and an index.

In this document, the terms 601, 603, 604, and 750 are used as abbreviations for MPC601, MPC603, MPC604, and MPC750, respectively.

Suggested Reading

This section lists additional reading that provides background for the information in this manual as well as general information about the PowerPC architecture.

General Information

The following documentation provides useful information about the PowerPC architecture and computer architecture in general:

- The following books are available from the PCI Special Interest Group, P.O. Box 14070, Portland, OR 97214; Tel. (800) 433-5177 (U.S.A.), (503) 797-4207 (International).
 - *Local Bus Specification, Rev 2.1*
 - *PCI System Design Guide, Rev 1.0*
- The following books are available from the Morgan-Kaufmann Publishers, 340 Pine Street, Sixth Floor, San Francisco, CA 94104; Tel. (800) 745-7323 (U.S.A.), (415) 392-2665 (International); web site: www.mkp.com; internet address: mkp@mkp.com.
 - *The PowerPC Architecture: A Specification for a New Family of RISC Processors*, Second Edition, by International Business Machines, Inc.
Updates to the architecture specification are accessible via the world-wide web at <http://www.austin.ibm.com/tech/ppc-chg.html>.
 - *PowerPC Microprocessor Common Hardware Reference Platform: A System Architecture*, by Apple Computer, Inc., International Business Machines, Inc., and Motorola, Inc.
 - *Computer Architecture: A Quantitative Approach*, Second Edition, by John L. Hennessy and David A. Patterson
 - *Computer Organization and Design: The Hardware/Software Interface*, Second Edition, by David A. Patterson and John L. Hennessy.

Documentation on PowerPC Architecture

The documentation based on PowerPC architecture is available from the sources listed on the back cover of this manual; the document order numbers are included in parentheses for ease in ordering:


- User's manuals—These books provide details about individual implementations on PowerPC architecture and are intended to be used in conjunction with the *Programming Environments Manual*.
- Programming environments manuals—These books provide information about resources defined by the PowerPC architecture that are common to processors. The

32-bit architecture model is described in the *Programming Environments Manual for 32-Bit Implementations of the PowerPC Architecture*, Rev. 2: MPCFPE32B/AD (Motorola order #)

- *Implementation Variances Relative to Rev. 1 of The Programming Environments Manual* is available via the world-wide web at:
http://e-www.motorola.com/collateral/MPCFPE32B_CH.html.
- Addenda/errata to user's manuals—Because some processors have follow-on parts an addendum is provided that describes the additional features and changes to functionality of the follow-on part. These addenda are intended for use with the corresponding user's manuals.
- Hardware specifications—Hardware specifications provide specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations for each implementation based on PowerPC architecture.
- Technical summaries—Each implementation on PowerPC architecture has a technical summary that provides an overview of its features. This document is roughly the equivalent to the overview (Chapter 1) of an implementation's user's manual.
- *PowerPC Microprocessor Family: The Bus Interface for 32-Bit Microprocessors: MPCBUSIF/AD* (Motorola order #) provides a detailed functional description of the 60x bus interface, as implemented on the 601, 603, and 604 family of microprocessors. This document is intended to help system and chipset developers by providing a centralized reference source to identify the bus interface presented by the 60x family of microprocessors that implement the PowerPC architecture.
- *PowerPC Microprocessor Family: The Programmer's Reference Guide: MPCPRG/D* (Motorola order #) is a concise reference that includes the register summary, memory control model, exception vectors, and the PowerPC instruction set.
- *PowerPC Microprocessor Family: The Programmer's Pocket Reference Guide: MPCPRGREF/D* (Motorola order #). This foldout card provides an overview of the PowerPC registers, instructions, and exceptions for 32-bit implementations.
- Application notes—These documents contain useful information about specific design issues useful to programmers and engineers working with processors.
- Additional literature on PowerPC implementations is being released as new processors become available. For a current list of documentation on PowerPC architecture based microprocessors, refer to the world-wide web at <http://www.motorola.com/home> and search for 'MPC.'

Conventions

This document uses the following notational conventions:

mnemonics	Instruction mnemonics are shown in lowercase bold.
<i>italics</i>	Italics indicate variable command parameters, for example, bcctrx . Book titles in text are set in italics.
0x0	Prefix to denote hexadecimal number
0b0	Prefix to denote binary number
rA, rB	Instruction syntax used to identify a source GPR
rA 0	Contents of a specified GPR or the value 0
rD	Instruction syntax used to identify a destination GPR
frA, frB, frC	Instruction syntax used to identify a source FPR
frD	Instruction syntax used to identify a destination FPR
REG[FIELD]	Abbreviations or acronyms for registers are shown in uppercase text. Specific bits, fields, or ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In certain contexts, such as a signal encoding, this indicates a don't care.
n	Used to express an undefined numerical value
¬	NOT logical operator
&	AND logical operator
	OR logical operator
	Indicates reserved bits or bit fields in a register. Although these bits may be written to as either ones or zeros, they are always read as zeros.

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

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
CAS	Column address strobe
CBR	CAS before RAS
DIMM	Dual in-line memory module
DRAM	Dynamic random access memory
ECC	Error checking and correction
EDO	Extended data out DRAM
ErrDR	Error detection register
ErrEnR	Error enabling register
ESCR	Emulation support configuration register
IEEE	Institute of Electrical and Electronics Engineers
Int Ack	Interrupt acknowledge
ISA	Industry standard architecture
JTAG	Joint test action group interface
L2	Secondary cache
MICR	Memory interface configuration register
MCCR	Memory control configuration register
Mux	Multiplex
PCI	Peripheral component interconnect
PCIB/MC	PCI bridge/memory controller
PICR	Processor interface configuration register
PLL	Phase-locked loop
PMC	Power management controller
PMCR	Power management configuration register
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

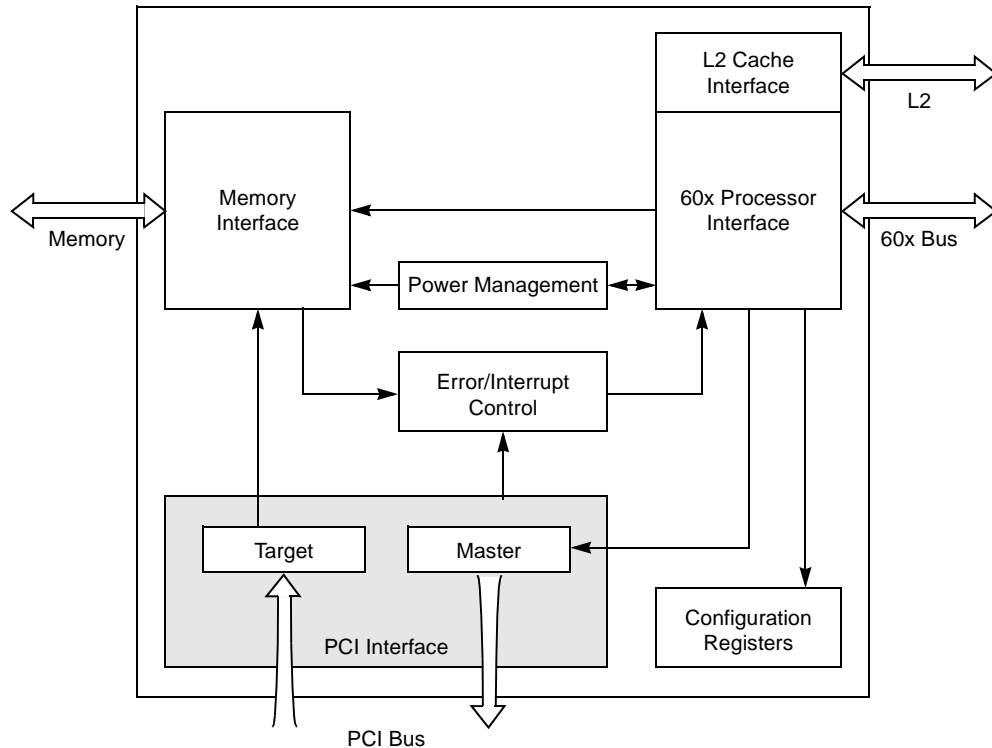
The MPC106 is one device in a family of products that provides system-level support for industry-standard interfaces to be used with PowerPC™ microprocessors. The MPC106 provides a common hardware reference platform (CHRP™) compliant bridge between the PowerPC architecture defined microprocessor family and the peripheral component interconnect (PCI) bus. CHRP is a set of specifications that defines a unified system architecture. PCI support allows system designers to design systems rapidly, using peripherals already designed for PCI and the other standard interfaces available in the personal computer hardware environment. These open specifications make it easier for system vendors to design systems capable of running multiple operating systems. The MPC106 integrates secondary cache control and a high-performance memory controller. The MPC106 uses an advanced, 3.3-V CMOS process technology and is fully compatible with TTL devices.

This manual describes the MPC106, its interfaces, and its signals.

1.1 MPC106 PCIB/MC Features

The MPC106 provides an integrated high-bandwidth, high-performance, TTL-compatible interface between a 60x processor, a secondary (L2) cache or additional (up to four total) 60x processors, the PCI bus, and main memory. This section summarizes the features of the MPC106.

Figure 1-1 shows the major functional units within the MPC106. 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. MPC106 Block Diagram

Major features of the MPC106 are as follows:

- 60x processor interface
 - Supports up to four 60x processors
 - Supports various operating frequencies and bus divider ratios
 - 32-bit address bus, 64-bit data bus
 - Supports full memory coherency
 - Supports optional 60x local bus slave
 - Decoupled address and data buses for pipelining of 60x accesses
 - Store gathering on 60x-to-PCI writes
- Secondary (L2) cache control
 - Configurable for write-through or write-back operation
 - Supports cache sizes of 256 Kbytes, 512 Kbytes, and 1 Mbyte
 - Up to 4 Gbytes of cacheable space
 - Direct-mapped
 - Supports byte parity
 - Supports partial update with external byte decode for write enables
 - Programmable interface timing

- Supports pipelined burst, synchronous burst, or asynchronous SRAMs
- Alternately supports an external L2 cache controller or integrated L2 cache module
- Memory interface
 - 1 Gbyte of RAM space, 16 Mbytes of ROM space
 - High-bandwidth, 64-bit data bus (72 bits including parity or ECC)
 - Supports fast page mode DRAMs, extended data out (EDO) DRAMs, or synchronous DRAMs (SDRAMs)
 - Supports 1 to 8 banks of DRAM/EDO/SDRAM with sizes ranging from 2 to 128 Mbytes per bank
 - SDRAM interface supports 64- and 128-Mbit, 4- and 2-bank SDRAMs with 2 open pages simultaneously and 16-Mbit, 2-bank SDRAMs with 2 open pages simultaneously
 - DRAM/EDO configurations support parity or error checking and correction (ECC); SDRAM configurations support parity or external, in-line, error checking module (ECM)/buffer
 - ROM space may be split between the PCI bus and the 60x/memory bus (8 Mbytes each)
 - Supports 8-bit asynchronous ROM or 64-bit burst-mode ROM
 - Supports writing to Flash ROM
 - Configurable external buffer control logic
 - Programmable interface timing
- PCI interface
 - Compliant with *PCI Local Bus Specification, Revision 2.1*
 - Supports PCI interlocked accesses to memory using $\overline{\text{LOCK}}$ signal and protocol
 - Supports accesses to all PCI address spaces
 - 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 MPC106 to the PCI bus
 - Word parity supported
 - 3.3-/5.0-V compatible
- Concurrent transactions on 60x and PCI buses supported
- Built-in performance monitor facility
- Programmable 60x, memory, and PCI bus drivers

- 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 MPC106 Major Functional Units

The MPC106 consists of the following major functional units:

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

This section describes each of these functional units.

1.2.1 60x Processor Interface

The MPC106 supports a programmable interface to a variety of microprocessors operating at select bus speeds. The 60x address bus is 32 bits wide and the data bus is 64 bits wide. The 60x processor interface of the MPC106 uses a subset of the 60x bus protocol, supporting single-beat and burst data transfers. The address and data buses are decoupled to support pipelined transactions.

Two signals on the MPC106, local bus slave claim ($\overline{\text{LBCLAIM}}$) and data bus grant local bus slave ($\overline{\text{DBGLB}}$), are provided for an optional local bus slave. However, the local bus slave must be capable of generating the transfer acknowledge ($\overline{\text{TA}}$) signal to interact with the 60x processor(s).

Depending on the system implementation, the processor bus may operate at the PCI bus clock rate (1x), or at 1.5x, 2x, 2.5x, or 3x the PCI bus clock rate. The 60x processor bus is synchronous, with all timing relative to the rising edge of the 60x bus clock.

1.2.2 Secondary (L2) Cache/Multiple Processor Interface

The MPC106 provides support for the following configurations of 60x processors and L2 cache:

- Up to four 60x processors with no L2 cache
- A single 60x processor plus a direct-mapped, lookaside, L2 cache using the internal L2 cache controller of the MPC106

- Up to four 60x processors plus an externally controlled L2 cache (such as the Motorola MPC2605 integrated L2 lookaside cache)

The internal L2 cache controller generates the arbitration and support signals necessary to maintain a write-through or write-back L2 cache. The internal L2 cache controller supports either asynchronous SRAMs, pipelined burst SRAMs, or synchronous burst SRAMs, using byte parity for data error detection.

When more than one 60x processor is used, nine signals of the L2 interface change their functions (to $\overline{BR}[1-3]$, $\overline{BG}[1-3]$, and $\overline{DBG}[1-3]$) to allow for arbitration between the 60x processors. The 60x processors share all 60x interface signals of the MPC106, except the bus request, bus grant, and data bus grant signals.

When an external L2 controller (or integrated L2 cache module) is used, three signals of the L2 interface change their functions (to $\overline{BRL2}$, $\overline{BGL2}$, and $\overline{DBGL2}$) to allow the MPC106 to arbitrate between the external cache and the 60x processor(s).

1.2.3 Memory Interface

The memory interface controls processor and PCI interactions to main memory and is capable of supporting a variety of DRAM, extended data-out (EDO) DRAM, or synchronous DRAM (SDRAM) and ROM or Flash ROM configurations as main memory. The maximum supported memory size is 1 Gbyte of DRAM/EDO/SDRAM, with 16 Mbytes of ROM/Flash. The MPC106 configures its memory controller to support the various memory sizes through software initialization of on-chip configuration registers. Parity (DRAM/EDO/SDRAM) or ECC (DRAM/EDO-only) is provided for error detection. For ECC with SDRAM configurations, the MPC106 supports an external, in-line, error checking module (ECM)/buffer. Using an external ECM requires one additional pin—an error indicator signal from the ECM connected to the $\overline{ISA_MASTER/BERR}$ input signal.

The MPC106 controls the 64-bit data path to main memory (72 bits with parity or ECC). To reduce loading on the data bus, system designers may need to add buffers between the 60x bus and memory. The MPC106 features configurable data/parity buffer control logic to accommodate several buffer types.

The MPC106 supports a variety of DRAM/EDO/SDRAM configurations. DRAM/EDO/SDRAM banks can be built using dual in-line memory modules (DIMMs), single in-line memory modules (SIMMs), or directly-attached memory devices. Eight row address strobe/chip select ($\overline{RAS/CS}[0-7]$) signals support up to eight banks of memory. The MPC106 supports bank sizes from 2 to 128 Mbytes. Eight column address strobe/data input/output mask ($\overline{CAS/DQM}[0-7]$) signals are used to provide byte selection for memory bank accesses.

The MPC106 supports parity checking and generation in two forms—normal parity and read-modify-write (RMW) parity. As an alternative to simple parity, the MPC106 supports

error checking and correction (ECC) for DRAM/EDO configurations. Using ECC, the MPC106 detects and corrects all single-bit errors and detects all double-bit errors and all errors within a nibble.

For ROM/Flash support, the MPC106 provides 20 address bits (21 address bits for the 8-bit wide ROM interface), two bank selects, one output enable, and one Flash write enable. The 16-Mbyte system ROM space is subdivided into two 8-Mbyte banks. Bank 0 (selected by $\overline{RCS0}$) is addressed from 0xFF80_0000 to 0xFFFF_FFFF. Bank 1 (selected by $\overline{RCS1}$) is addressed from 0xFF00_0000 to 0xFF7F_FFFF. A configuration signal (\overline{FOE}) sampled at reset, determines the bus width of the ROM or Flash device (8-bit or 64-bit) in bank 0. The data bus width for ROM bank 1 is always 64 bits. For systems using the 8-bit interface to bank 0, the ROM/Flash device must be connected to the most-significant byte lane of the data bus (DH[0–7]).

The MPC106 also supports a mixed ROM system configuration. That is, the system can have the upper 8 Mbytes (bank 0) of ROM space located on the PCI bus and the lower 8 Mbytes (bank 1) of ROM space located on the 60x/memory bus.

1.2.4 PCI Interface

The MPC106's PCI interface complies with the *PCI Local Bus Specification*, Revision 2.1, and follows the guidelines in the *PCI System Design Guide*, Revision 1.0 for host bridge architecture. 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 PCI interface of the MPC106 functions as both a master and target device. As a master, the MPC106 supports read and write operations to PCI memory space, PCI I/O space, and PCI configuration space. The MPC106 also supports PCI special-cycle and interrupt-acknowledge commands. As a target, the MPC106 supports read and write operations to system memory. Mode selectable big-endian to little-endian conversion is supplied at the PCI interface.

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.

1.3 Performance Monitor

The MPC106 includes a performance monitor facility that allows it to record/monitor selected system behaviors. Four 32-bit performance monitor counters in the MPC106 count the occurrence of software-selectable events. Up to four different events can be counted simultaneously using the four different counters.

1.4 Power Management

The MPC106 provides hardware support for four levels of power reduction; the doze, nap, and sleep modes are invoked by register programming, and the suspend mode is invoked by assertion of an external signal. The design of the MPC106 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 MPC106.

1.4.1 Full-On Mode

This is the default power state of the MPC106 following a hard reset, with all internal functional units fully powered and operating at full clock speed.

1.4.2 Doze Mode

In this power saving mode, all the MPC106 functional units are disabled except for PCI address decoding, system RAM refreshing, and 60x bus request monitoring (through $\overline{\text{BRx}}$). Once the doze mode is entered, a hard reset, a PCI transaction referenced to system memory, or a 60x bus request can bring the MPC106 out of the doze mode and into the full-on state. If the MPC106 is awakened for a processor or PCI bus access, the access is completed and the MPC106 returns to the doze mode. The MPC106's doze mode is totally independent of the power saving mode of the processor.

1.4.3 Nap Mode

Nap mode provides further power savings compared to doze mode. The greater power savings can be achieved by placing both the processor and the MPC106 in a power reduction mode. In this mode, only the PCI address decoding, system RAM refresh, and the processor bus request monitoring are still operating. A hard reset, a PCI bus transaction referenced to system memory, or a 60x bus request can bring the MPC106 out of the nap mode. If the MPC106 is awakened by a PCI access, the access is completed, and the MPC106 returns to the nap mode. If the MPC106 is awakened by a processor access, the access is completed, but the MPC106 remains in the full-on state. When in the nap mode, the PLL is required to be running and locked to the system clock (SYSCLK).

1.4.4 Sleep Mode

Sleep mode provides further power savings compared to the nap mode. As in nap mode, both the processor and the MPC106 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 MPC106 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 PLL[0–3] signals into the PLL-bypass mode. The external PMC must enable the PLL, turn on SYSCLK, and allow the PLL time to lock before waking the system from sleep mode.

1.4.5 Suspend Mode

Suspend mode is activated by asserting the $\overline{\text{SUSPEND}}$ signal. In suspend mode, the MPC106 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 released to a high-impedance state and all inputs (including $\overline{\text{HRST}}$) are ignored.

Chapter 2

Signal Descriptions

This chapter provides descriptions of the MPC106'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 TV (tag valid) and NMI (nonmaskable interrupt), are referred to as asserted when they are high and negated when they are low.

For multiple function signals, outlined signal names refer to the alternate function(s) of the signal being described. For example, the L2 controller signal $\overline{\text{TOE}}$ (tag output enable) has the alternate function $\overline{\text{DBG1}}$ (data bus grant 1) when the MPC106 is configured for a second 60x processor.

2.1 Signal Configuration

The MPC106's signals are grouped as follows:

- 60x processor interface signals
- L2 cache/multiple 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 MPC106, showing how the signals are grouped. A pinout diagram showing actual pin numbers is included in the MPC106 hardware specifications.

2.2 Signal Descriptions

This section describes individual MPC106 signals, grouped according to Figure 2-1. The following sections are intended to provide a quick summary of signal functions. Table 2-1 provides an alphabetical cross-reference to the signals of the MPC106. It details the signal name, interface, alternate functions, number of signals, whether the signal is an input, output, or bidirectional, and finally a pointer to the section in this chapter where the signal is described. (The MPC106 samples these signals at power-on reset or hard reset to determine the configuration. After they are sampled, they assume their normal functions. See Section 2.2.9, “Configuration Signals,” for more information.)

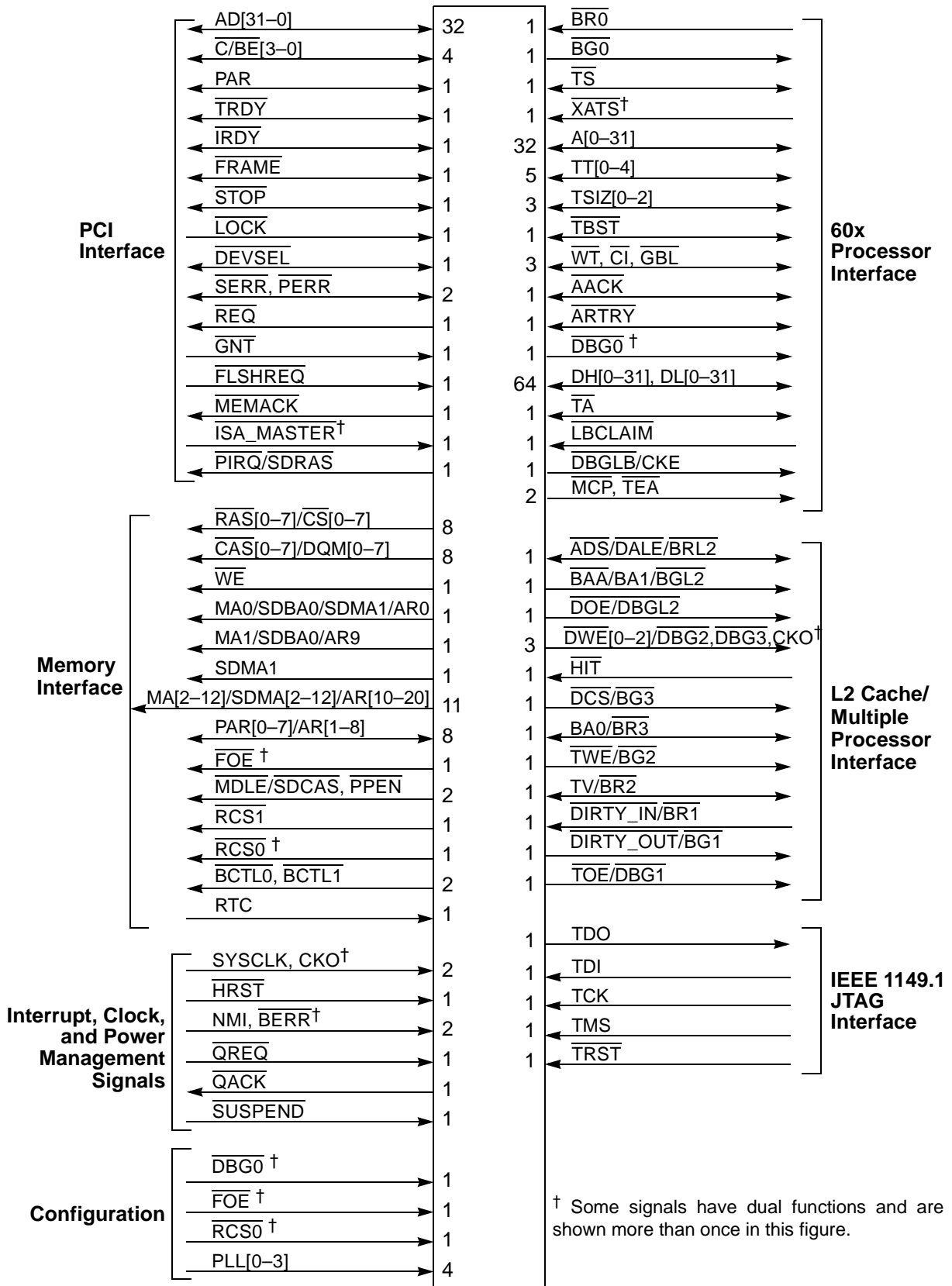


Figure 2-1. MPC106 Signal Groupings

Table 2-1. MPC106 Signal Cross Reference

Signal	Signal Name	Interface	Alternate Function(s)	Number of Pins	I/O	Section No.
A[0–31]	Address bus	60x processor	—	32	I/O	2.2.2.1
$\overline{\text{AACK}}$	Address acknowledge	60x processor	—	1	I/O	2.2.2.2
AD[31–0]	Address/data	PCI	—	32	I/O	2.2.5.1
$\overline{\text{ADS}}$	Address strobe	Internal L2 controller	$\overline{\text{DALE}}$ $\overline{\text{BRL2}}$	1	O	2.2.3.1.1
AR0	ROM address 0	Memory	MA0 SDBA1 SDMA0	1	O	2.2.4.1
AR[1–8]	ROM address 1–8	Memory	PAR[0–7]	8	O	2.2.4.2
AR[9]	ROM address 9	Memory	MA1 SDBA0	1	O	2.2.4.3
AR[10–20]	ROM address 10–20	Memory	MA[2–12] SDMA[2–12]	11	O	2.2.4.3
$\overline{\text{ARTRY}}$	Address retry	60x processor	—	1	I/O	2.2.2.3
BA0	Burst address 0	Internal L2 controller	$\overline{\text{BR3}}$	1	O	2.2.3.1.2
BA1	Burst address 1	Internal L2 controller	$\overline{\text{BAA}}$ $\overline{\text{BGL2}}$	1	O	2.2.3.1.3
$\overline{\text{BAA}}$	Bus address advance	Internal L2 controller	BA1 $\overline{\text{BGL2}}$	1	O	2.2.3.1.4
$\overline{\text{BCTL}}[0–1]$	Buffer control 0–1	Memory	—	2	O	2.2.4.4
$\overline{\text{BCTL0}}^1$	501-mode	Configuration	—	1	I	2.2.9.1
$\overline{\text{BG0}}$	Bus grant 0	60x processor	—	1	O	2.2.2.4
$\overline{\text{BERR}}$	External ECM error detect	Interrupt	$\overline{\text{ISA_MASTER}}$	1	I	2.2.7
$\overline{\text{BG1}}$	Bus grant 1	Multiple Processor	$\overline{\text{DIRTY_OUT}}$	1	O	2.2.3.3.1
$\overline{\text{BG2}}$	Bus grant 2	Multiple Processor	$\overline{\text{TWE}}$	1	O	2.2.3.3.1
$\overline{\text{BG3}}$	Bus grant 3	Multiple Processor	$\overline{\text{DCS}}$	1	O	2.2.3.3.1
$\overline{\text{BGL2}}$	External L2 bus grant	External L2 controller	BA1 BAA	1	O	2.2.3.2.1
$\overline{\text{BR0}}$	Bus request 0	60x processor	—	1	I	2.2.2.5
$\overline{\text{BR1}}$	Bus request 1	Multiple Processor	$\overline{\text{DIRTY_IN}}$	1	I	2.2.3.3.2
$\overline{\text{BR2}}$	Bus request 2	Multiple Processor	TV	1	I	2.2.3.3.2

Table 2-1. MPC106 Signal Cross Reference (continued)

Signal	Signal Name	Interface	Alternate Function(s)	Number of Pins	I/O	Section No.
$\overline{BR3}$	Bus request 3	Multiple Processor	$BA0$	1	I	2.2.3.3.2
$\overline{BRL2}$	External L2 bus request	External L2 controller	\overline{ADS} \overline{DALE}	1	I	2.2.3.2.2
$\overline{CAS}[0-7]$	Column address strobe 0-7	Memory	$\overline{DQM}[0-7]$	8	O	2.2.4.5
$\overline{C/BE}[3-0]$	Command/byte enable	PCI	—	4	I/O	2.2.5.2
\overline{CI}	Caching-inhibit	60x processor	—	1	I/O	2.2.2.6
\overline{CKE}	SDRAM clock enable	Memory	\overline{DBGLB}	1	O	2.2.4.6
\overline{CKO}	Test clock	Clock	$\overline{DWE2}$	1	O	2.2.7.1
$\overline{CS}[0-7]$	SDRAM chip select	Memory	$\overline{RAS}[0-7]$	8	O	2.2.4.7
\overline{DALE}	Data address latch enable	Internal L2 controller	\overline{ADS} $\overline{BRL2}$	1	O	2.2.3.1.5
$\overline{DBG0}$	Data bus grant 0	60x processor	—	1	O	2.2.2.7
$\overline{DBG0}^1$	Address map	Configuration	—	1	I	2.2.9.2
$\overline{DBG1}$	Data bus grant 1	Multiple Processor	\overline{TOE}	1	O	2.2.3.3.3
$\overline{DBG2}$	Data bus grant 2	Multiple Processor	$\overline{DWE0}$	1	O	2.2.3.3.3
$\overline{DBG3}$	Data bus grant 3	Multiple Processor	$\overline{DWE1}$	1	O	2.2.3.3.3
$\overline{DBGL2}$	External L2 data bus grant	External L2 controller	\overline{DOE}	1	O	2.2.3.2.3
\overline{DBGLB}	Data bus grant local bus slave	60x processor	\overline{CKE}	1	O	2.2.2.8
\overline{DCS}	Data RAM chip select	Internal L2 controller	$\overline{BG3}$	1	O	2.2.3.1.6
\overline{DEVSEL}	Device select	PCI	—	1	I/O	2.2.5.3
$\overline{DH}[0-31]$, $\overline{DL}[0-31]$	Data bus	60x processor	—	64	I/O	2.2.2.9
$\overline{DIRTY_IN}$	Dirty in	Internal L2 controller	$\overline{BR1}$	1	I	2.2.3.1.7
$\overline{DIRTY_OUT}$	Dirty out	Internal L2 controller	$\overline{BG1}$	1	O	2.2.3.1.8
\overline{DOE}	Data RAM output enable	Internal L2 controller	$\overline{DBGL2}$	1	O	2.2.3.1.9
$\overline{DQM}[0-7]$	SDRAM data input/output mask	Memory	$\overline{CAS}[0-7]$	8	O	2.2.4.8

Table 2-1. MPC106 Signal Cross Reference (continued)

Signal	Signal Name	Interface	Alternate Function(s)	Number of Pins	I/O	Section No.
$\overline{\text{DWE}}[0-2]$	Data RAM write enable 0-2	Internal L2 controller	$\overline{\text{DBG2}}$ $\overline{\text{DBG3}}$ CKO	3	O	2.2.3.1.10
$\overline{\text{FLSHREQ}}$	Flush request	PCI (sideband)	—	1	I	2.2.5.14.1
$\overline{\text{FOE}}$	Flash output enable	Memory	—	1	O	2.2.4.9
$\overline{\text{FOE}}^1$	ROM bank 0 data path width	Configuration	—	1	I	2.2.9.3
$\overline{\text{FRAME}}$	Frame	PCI	—	1	I/O	2.2.5.4
$\overline{\text{GBL}}$	Global	60x processor	—	1	I/O	2.2.2.10
$\overline{\text{GNT}}$	PCI bus grant	PCI	—	1	I	2.2.5.5
$\overline{\text{HIT}}$	Hit	Internal L2 controller	—	1	I	2.2.3.1.11
$\overline{\text{HIT}}$	External L2 hit	External L2 controller	—	1	I	2.2.3.2.4
$\overline{\text{HRST}}$	Hard reset	Interrupt	—	1	I	2.2.7.2
$\overline{\text{IRDY}}$	Initiator ready	PCI	—	1	I/O	2.2.5.6
$\overline{\text{ISA_MASTER}}$	ISA master	PCI (sideband)	$\overline{\text{BERR}}$	1	I	2.2.5.14.2
$\overline{\text{LBCLAIM}}$	Local bus slave claim	60x processor	—	1	I	2.2.2.11
$\overline{\text{LBCLAIM}}^1$	60x clock phase	Configuration	—	1	I	2.2.9.4
$\overline{\text{LOCK}}$	Lock	PCI	—	1	I	2.2.5.7
MA0	Memory address 0	Memory	AR0 SDBA1 SDMA0	1	O	2.2.4.10
MA1	Memory address 1	Memory	AR9 SDBA0	1	O	2.2.4.10
MA[2-12]	Memory address 2-12	Memory	AR[10-20] SDMA[2-12]	11	O	2.2.4.10
$\overline{\text{MCP}}$	Machine check	60x processor	—	1	O	2.2.2.12
$\overline{\text{MDLE}}$	Memory data latch enable	Memory	$\overline{\text{SDCAS}}$	1	O	2.2.4.11
$\overline{\text{MEMACK}}$	Flush acknowledge	PCI (sideband)	—	1	O	2.2.5.14.3
NMI	Nonmaskable interrupt	Interrupt	—	1	I	2.2.7.3
PAR	Parity	PCI	—	1	I/O	2.2.5.8
PAR[0-7]	Data parity 0-7	Memory	AR[1-8]	8	I/O	2.2.4.12
$\overline{\text{PERR}}$	Parity error	PCI	—	1	I/O	2.2.5.9

Table 2-1. MPC106 Signal Cross Reference (continued)

Signal	Signal Name	Interface	Alternate Function(s)	Number of Pins	I/O	Section No.
$\overline{\text{PIRQ}}$	Modified memory interrupt request	PCI (sideband)	$\overline{\text{SDRAS}}$	1	O	2.2.5.14.4
PLL[0–3]	Clock mode	Configuration	—	4	I	2.2.9.5
$\overline{\text{PPEN}}$	Parity path read enable	Memory	—	1	O	2.2.4.13
$\overline{\text{QACK}}$	Quiesce acknowledge	Power Management	—	1	O	2.2.7.4
$\overline{\text{QREQ}}$	Quiesce request	Power Management	—	1	I	2.2.7.5
$\overline{\text{RAS}}[0–7]$	Row address strobe 0–7	Memory	$\overline{\text{CS}}[0–7]$	8	O	2.2.4.14
$\overline{\text{RCS0}}$	ROM/bank 0 select	Memory	—	1	O	2.2.4.15
$\overline{\text{RCS0}}^1$	ROM location	Configuration	—	1	I	2.2.9.6
$\overline{\text{RCS1}}$	ROM/bank 1 select	Memory	—	1	O	2.2.4.16
$\overline{\text{REQ}}$	PCI bus request	PCI	—	1	O	2.2.5.10
RTC	Real time clock	Memory	—	1	I	2.2.4.17
SDBA0	SDRAM bank select 0	Memory	$\overline{\text{AR9}}$ $\overline{\text{MA1}}$	1	O	2.2.4.18
SDBA1	SDRAM bank select 1	Memory	$\overline{\text{AR0}}$ $\overline{\text{MA0}}$ $\overline{\text{SDMA0}}$	1	O	2.2.4.18
$\overline{\text{SDCAS}}$	SDRAM column address strobe	Memory	$\overline{\text{MDLE}}$	1	O	2.2.4.19
SDMA0	SDRAM address 0	Memory	$\overline{\text{AR0}}$ $\overline{\text{MA0}}$ $\overline{\text{SDBA1}}$	1	O	2.2.4.20
SDMA1	SDRAM address 1	Memory	$\overline{\text{XATS}}$	1	O	2.2.4.20
SDMA[2–12]	SDRAM address 2–12	Memory	$\overline{\text{AR}}[10–20]$ $\overline{\text{MA}}[2–12]$	11	O	2.2.4.20
$\overline{\text{SDRAS}}$	SDRAM row address strobe	Memory	$\overline{\text{PIRQ}}$	1	O	2.2.4.21
$\overline{\text{SERR}}$	System error	PCI	—	1	I/O	2.2.5.11
$\overline{\text{STOP}}$	Stop	PCI	—	1	I/O	2.2.5.12
$\overline{\text{SUSPEND}}$	Suspend	Power Management	—	1	I	2.2.7.6
SYCLK	System clock	Clock	—	1	I	2.2.7.7
$\overline{\text{TA}}$	Transfer acknowledge	60x processor	—	1	I/O	2.2.2.13
$\overline{\text{TBST}}$	Transfer burst	60x processor	—	1	I/O	2.2.2.14

Table 2-1. MPC106 Signal Cross Reference (continued)

Signal	Signal Name	Interface	Alternate Function(s)	Number of Pins	I/O	Section No.
TCK	JTAG test clock	IEEE 1149.1 JTAG	—	1	I	2.2.8.1
TDO	JTAG test data output	IEEE 1149.1 JTAG	—	1	O	2.2.8.2
TDI	JTAG test data Input	IEEE 1149.1 JTAG	—	1	I	2.2.8.3
$\overline{\text{TEA}}$	Transfer error acknowledge	60x processor	—	1	O	2.2.2.15
$\overline{\text{TOE}}$	Tag output enable	Internal L2 controller	$\overline{\text{DBG1}}$	1	O	2.2.3.1.12
TMS	JTAG test mode select	IEEE 1149.1 JTAG	—	1	I	2.2.8.4
$\overline{\text{TRDY}}$	Target ready	PCI	—	1	I/O	2.2.5.13
$\overline{\text{TRST}}$	JTAG test reset	IEEE 1149.1 JTAG	—	1	I	2.2.8.5
$\overline{\text{TS}}$	Transfer start	60x processor	—	1	I/O	2.2.2.16
TSIZ[0–2]	Transfer size	60x processor	—	3	I/O	2.2.2.17
TT[0–4]	Transfer type	60x processor	—	5	I/O	2.2.2.18
TV	Tag valid	Internal L2 controller	$\overline{\text{BR2}}$	1	O	2.2.3.1.13
$\overline{\text{TWE}}$	Tag write enable	Internal L2 controller	$\overline{\text{BG2}}$	1	O	2.2.3.1.14
$\overline{\text{WE}}$	Write enable	Memory	—	1	O	2.2.4.22
$\overline{\text{WT}}$	Write-through	60x processor	—	1	I/O	2.2.2.19
$\overline{\text{XATS}}$	Extended address transfer start	60x processor	SDMA1	1	I	2.2.2.20

2.2.1 Signal States at Reset

When a system reset request is recognized ($\overline{\text{HRST}}$ asserted), the MPC106 aborts all current internal and external transactions, releases all bidirectional I/O signals to a high-impedance state, ignores the input signals (except for SYSCLK , and the configuration signals BCTL0 , DBG0 , FOE , LBCLAIM , RCS0 , and $\text{PLL}[0–3]$), and drives most of the output signals to an inactive state. (Table 2-2 shows the states of the output-only signals during system reset.)

Table 2-2. Output Signal States During System Reset

Interface	Signal	State During System Reset
60x processor	$\overline{\text{BG0}}$ $\overline{\text{MCP}}$ $\overline{\text{TEA}}$	Negated
	$\overline{\text{DBGLB/CKE}}$	Driven, but with indeterminate state
	$\overline{\text{DBG0}}$ $\overline{\text{LBCLAIM}}$	Configuration input during system reset
L2	$\overline{\text{ADS/DALE}}$ $\overline{\text{DBGL2/DOE}}$ $\overline{\text{DCS}}$ $\overline{\text{DIRTY_OUT}}$ $\overline{\text{DWE[0-2]}}$ $\overline{\text{TOE}}$	Negated
	$\overline{\text{BAA/BA1/BGL2}}$ $\overline{\text{TWE}}$	Driven, but with indeterminate state
Memory	MA[0-12]/SDBA0, SDBA1/SDMA0, SDMA[2-12]	High-impedance
	$\overline{\text{BCTL1}}$ $\overline{\text{CAS/DQM[0-7]}}$ $\overline{\text{MDLE/SDCAS}}$ $\overline{\text{PPEN}}$ $\overline{\text{RAS/CS[0-7]}}$ $\overline{\text{RCS1}}$ $\overline{\text{WE}}$	Negated
	$\overline{\text{BCTL0}}$ $\overline{\text{FOE}}$ $\overline{\text{RCS0}}$	Configuration input during system reset
	$\overline{\text{SDMA1/XATS}}$	Functions as $\overline{\text{XATS}}$ input until $\overline{\text{MCCR1[RAM_TYPE]}}$ is cleared. Must be pulled up.
PCI	$\overline{\text{MEMACK}}$ $\overline{\text{REQ}}$ $\overline{\text{PIRQ/SDRAS}}$	High-impedance
Clock	$\overline{\text{CKO}}$	Internal (core) clock
Power management	$\overline{\text{QACK}}$	High-impedance
IEEE 1149.1 JTAG	$\overline{\text{TDO}}$	High-impedance

2.2.2 60x Processor Interface Signals

This section provides descriptions of the 60x processor interface signals on the MPC106. Note that with the exception of $\overline{\text{BGn}}$, $\overline{\text{BRn}}$, $\overline{\text{DBGn}}$, $\overline{\text{DBGLB}}$, and $\overline{\text{LBCLAIM}}$, all of the 60x processor interface signals are connected to all processors in a multiprocessor system. See Section 4.1.2, “Multiprocessor System Configuration,” for more information.

2.2.2.1 Address Bus (A[0–31])

The address bus (A[0–31]) consists of 32 signals that are both input and output signals.

2.2.2.1.1 Address Bus (A[0–31])—Output

Following are the state meaning and timing comments for A[0–31] as output signals.

State Meaning Asserted/Negated—Specifies the physical address for 60x bus snooping.

Timing Comments Assertion/Negation—Driven valid in the same clock cycle as the assertion of \overline{TS} . Once driven, these signals remain valid for the entire address tenure.

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

2.2.2.1.2 Address Bus (A[0–31])—Input

Following are the state meaning and timing comments for A[0–31] 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 in the same clock cycle as the assertion of \overline{TS} . Once driven, these signals must remain stable for the entire address tenure.

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

2.2.2.2 Address Acknowledge (\overline{AACK})

The address acknowledge (\overline{AACK}) signal is an input and output signal on the MPC106.

2.2.2.2.1 Address Acknowledge (\overline{AACK})—Output

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

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

Negated—Indicates that the address tenure must remain active, and all address-tenure-related signals must remain valid.

Timing Comments Assertion—Occurs a programmable number of clock cycles after \overline{TS} and whenever \overline{ARTRY} conditions are resolved. For pipelined transactions, \overline{AACK} is asserted in the same clock cycle or after the last \overline{TA} of the previous data tenure. When using the internal L2 controller, the assertion of \overline{AACK} is delayed to hold the address valid for the tag update.

Negation—Occurs one clock cycle after assertion.

High-impedance—Occurs one clock cycle after negation.

2.2.2.2.2 Address Acknowledge (\overline{AACK})—Input

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

State Meaning Asserted—Indicates that the externally-controlled L2 cache is terminating the address tenure. On the cycle following the assertion of \overline{AACK} , the bus master releases the address tenure related signals to the high-impedance state and samples \overline{ARTRY} .

Negated—Indicates that the address tenure must remain active, and all address-tenure-related signals must remain valid.

Timing Comments Assertion—Occurs anytime after the assertion of \overline{TS} when \overline{ARTRY} is valid (or expected to be valid in the next clock cycle). For pipelined transactions, the assertion of \overline{AACK} should not occur before the last \overline{TA} of the previous data tenure.

Negation—Occurs one clock cycle after assertion.

High-impedance—Occurs one clock cycle after negation.

2.2.2.3 Address Retry (\overline{ARTRY})

The address retry (\overline{ARTRY}) signal is both an input and output signal on the MPC106.

2.2.2.3.1 Address Retry (\overline{ARTRY})—Output

Following are the state meaning and timing comments for \overline{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 MPC106 does not require the address tenure to be retried.

Timing Comments Assertion—For processor to system memory accesses, occurs two clock cycles after \overline{TS} . For processor to PCI accesses, the MPC106 withholds \overline{ARTRY} and \overline{AACK} until the \overline{ARTRY} condition for the PCI access is resolved. See Section 4.3.3.1, “MPC106 Snoop

Response,” for more information. Once asserted, occurs one clock after the assertion of $\overline{\text{AACK}}$.

Negation/High-impedance— $\overline{\text{ARTRY}}$ is released to high-impedance for the first half of the second clock cycle after the assertion of $\overline{\text{AACK}}$, then it is negated for one clock, and then it is released to high-impedance.

2.2.2.3.2 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 copyback operation.

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/High-impedance— $\overline{\text{ARTRY}}$ is released to high-impedance for the first half of the second clock cycle after the assertion of $\overline{\text{AACK}}$, then it is negated for one clock, and then it is released to high-impedance.

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

The bus grant 0 ($\overline{\text{BG0}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{BG0}}$ signal.

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

Negated—Indicates that the primary 60x processor 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 bus is parked on the primary 60x processor and no other request is pending.

Negation—Occurs when other higher priority transactions are pending.

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

The bus request 0 ($\overline{BR0}$) signal is an input on the MPC106. Following are the state meaning and timing comments for the $\overline{BR0}$ signal.

State Meaning	Asserted—Indicates that the primary 60x processor requires mastership of the 60x bus for a transaction.
	Negated—Indicates that the primary 60x processor does not require mastership of the 60x bus.
Timing Comments	Assertion—May occur when bus grant 0 ($\overline{BG0}$) is negated and a bus transaction is needed by the 60x processor. This may occur even if the two possible pipeline accesses have already occurred.
	Negation—Occurs for at least one clock cycle after an accepted, qualified bus grant, even if another transaction is pending on the 60x processor. It is also negated for at least one clock cycle when the assertion of \overline{ARTRY} is detected on the 60x bus (except for assertions due to primary 60x snoop copyback operations).

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

The caching-inhibited (\overline{CI}) signal is both an input and output signal on the MPC106. 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. Note that \overline{CI} is always negated for snoop cycles initiated by the MPC106.
Timing Comments	Assertion/Negation—The same as A[0–31].
	High-impedance—The same as A[0–31].

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

The data bus grant 0 ($\overline{DBG0}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{DBG0}$ signal. Note that $\overline{DBG0}$ also serves as a configuration input at power-on reset (POR). See Section 2.2.9, “Configuration Signals,” for more information.

State Meaning	Asserted—Indicates that the primary 60x may, with the proper qualification, assume mastership of the data bus. A qualified data bus grant is defined as the assertion of $\overline{BG0}$ and negation of \overline{ARTRY} . The requirement for the \overline{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 primary 60x processor is not granted mastership of the data bus.

Timing Comments Assertion—Occurs on the first clock cycle in which the data bus is not busy and the primary 60x processor has the highest priority outstanding data transaction. If the data bus is parked on the primary 60x processor, $\overline{\text{DBG0}}$ is asserted one clock cycle after $\overline{\text{BG0}}$. In fast-L2 mode, $\overline{\text{DBG0}}$ may be asserted as early as the same clock cycle as the last $\overline{\text{TA}}$ of the previous data tenure.

Negation—Occurs one clock cycle after assertion.

2.2.2.8 Data Bus Grant Local Bus Slave ($\overline{\text{DBGLB}}$)—Output

The data bus grant local bus slave ($\overline{\text{DBGLB}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{DBGLB}}$ signal.

State Meaning Asserted—Indicates, to the local bus slave, that the MPC106 has granted a 60x processor the data bus. If the cycle is a local bus slave cycle, the local bus slave can use the data bus to transfer data to the 60x processor.

Negated—Indicates that a 60x processor has not been granted mastership of the data bus.

Timing Comments Assertion—The same as $\overline{\text{DBGn}}$ except if the previous transaction is an external L2 cache operation and the externally-controlled L2 asserted $\overline{\text{AACK}}$ before $\overline{\text{DBGn}}$. In that case, $\overline{\text{DBGLB}}$ is not asserted for the external L2 cache transaction (so as not to confuse the local bus slave). See Section 4.4.5, “60x Local Bus Slave Support,” for more information.

Negation—Occurs one clock cycle after assertion.

2.2.2.9 Data Bus (DH[0–31], DL[0–31])

The data bus (DH[0–31], DL[0–31]) consists of 64 signals that are both input and output signals on the MPC106. The data bus is comprised of two halves—data bus high (DH[0–31]) and data bus low (DL[0–31]). Table 2-3 specifies the byte lane assignments for the data bus.

Table 2-3. Data Bus Byte Lane Assignments

Data Bus Signals	Byte Lane
DH[0–7]	0 (MSB)
DH[8–15]	1
DH[16–23]	2
DH[24–31]	3
DL[0–7]	4

Table 2-3. Data Bus Byte Lane Assignments (continued)

Data Bus Signals	Byte Lane
DL[8–15]	5
DL[16–23]	6
DL[24–31]	7 (LSB)

2.2.2.9.1 Data Bus (DH[0–31], DL[0–31])—Output

Following are the state meaning and timing comments for the data bus as output signals.

State Meaning Asserted/Negated—Represents the value of data being driven by the MPC106.

Timing Comments Assertion/Negation—For a 60x processor read transaction, the data bus signals are valid one clock cycle after the \overline{DBGn} signal is asserted. For PCI-to-memory or internal buffer flush transactions, the data bus signals are valid at least one clock cycle after the data bus becomes idle.

High-impedance—For 60x processor read transactions, the data bus signals are released to high-impedance one clock cycle after the last assertion of \overline{TA} or one clock cycle after detecting a qualified \overline{ARTRY} . For PCI-to-memory or internal buffer flush transactions, the data bus signals are released to high-impedance when the transaction is complete.

2.2.2.9.2 Data Bus (DH[0–31], DL[0–31])—Input

Following are the state meaning and timing comments for the data bus as input signals.

State Meaning Asserted/Negated—Represents the state of data being driven by a 60x processor, the L2 cache, or the memory subsystem.

Timing Comments Assertion/Negation—For a 60x processor write transaction, the data bus signals are valid one clock cycle after the assertion of \overline{DBGn} . For an L2 copyback transaction, the data bus signals are valid when data RAM output enable (\overline{DOE}) is asserted. For a memory read transaction, the data bus signals are valid at a time dependent on the memory interface configuration. Refer to Chapter 6, “Memory Interface,” for more information.

High-impedance—the data bus signals are released to high-impedance one clock cycle after the last assertion of \overline{TA} . If the address tenure is \overline{ARTRYd} , the data bus signals go to a high-impedance state one clock cycle after the qualified \overline{ARTRY} .

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

The global ($\overline{\text{GBL}}$) signal is both an input and output signal on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{GBL}}$ signal.

State Meaning Asserted—Indicates that an access is global. Coherency needs to be enforced by hardware. Note that $\overline{\text{GBL}}$ is always asserted for snoop cycles initiated by the MPC106.

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

Timing Comments Assertion/Negation—The same as A[0–31].

High-impedance—The same as A[0–31].

2.2.2.11 Local Bus Slave Claim ($\overline{\text{LBCLAIM}}$)—Input

The local bus slave claim ($\overline{\text{LBCLAIM}}$) signal is an input on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{LBCLAIM}}$ signal. Note that when PLL[0:3] corresponds to one of the half-bus clock settings (5:2 and 3:2), $\overline{\text{LBCLAIM}}$ functionality is disabled. Thus, local bus slaves cannot be used in conjunction with half-clock bus modes. See Section 2.2.9, “Configuration Signals,” for more information.

State Meaning Asserted—Indicates that the local bus slave claims the transaction and is responsible for driving $\overline{\text{TA}}$ during the data tenure.

Negated—Indicates that the transaction is not claimed by the local bus slave.

Timing Comments Assertion—The MPC106 samples the $\overline{\text{LBCLAIM}}$ signal when PICR2[CF_L2_HIT_DELAY] expires.

Negation—Occurs one clock cycle after assertion.

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

The machine check ($\overline{\text{MCP}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{MCP}}$ output signal.

State Meaning Asserted—Indicates that the MPC106 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 the 60x bus clock.

Negation/high-impedance—Occurs after all error flags have been cleared by software and the machine check exception vector has been accessed. The configuration parameter, SHARED_MCP, in

power management configuration register 2 (PMCR2) controls whether the MPC106 negates $\overline{\text{MCP}}$ or releases $\overline{\text{MCP}}$ to high-impedance.

2.2.2.13 Transfer Acknowledge ($\overline{\text{TA}}$)

The transfer acknowledge ($\overline{\text{TA}}$) signal is both an input and output signal on the MPC106.

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

Following are the state meaning and timing comments for $\overline{\text{TA}}$ 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 MPC106.

Timing Comments Assertion—Occurs when the current data beat can be completed.

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

High-impedance—Occurs one-half clock cycle after negation.

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

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

State Meaning Asserted—Indicates that the external L2 cache or local bus slave has latched data for a write operation, or is indicating the data is valid for a read operation, thus terminating the current data beat. 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 external L2 cache or local bus slave.

Timing Comments Assertion—Occurs when the current data beat can be completed.

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

High-impedance—Occurs one-half clock cycle after negation.

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

The transfer burst ($\overline{\text{TBST}}$) signal is an input and output signal on the MPC106.

2.2.2.14.1 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 MPC106-generated snoop operations are 8-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 A[0–31].
 High-impedance—The same as A[0–31].

2.2.2.14.2 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 A[0–31].
 High-impedance—The same as A[0–31].

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

The transfer error acknowledge ($\overline{\text{TEA}}$) signal is an output on the MPC106. Note that the $\overline{\text{TEA}}$ 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 $\overline{\text{TEA}}$ signal.

State Meaning Asserted—Indicates that a bus error has occurred. Assertion of $\overline{\text{TEA}}$ terminates the data transaction in progress; that is, it is not necessary to assert $\overline{\text{TA}}$ because it is ignored by the target processor. An unsupported transaction will cause the assertion of $\overline{\text{TEA}}$ (provided $\overline{\text{TEA}}$ is enabled). Unsupported transactions include the following:

- A direct-store access
- A graphics read or write (**eciwx** or **ecowx**)
- A write to the PCI interrupt acknowledge space
- A write to Flash space, when Flash writes are disabled
- An aborted processor-to-PCI transaction

Negated—Indicates that no bus error has been detected.

Timing Comments Assertion—Occurs during the data tenure in which the bus error is detected.

Negation—Occurs one clock after assertion.

2.2.2.16 Transfer Start ($\overline{\text{TS}}$)

The transfer start ($\overline{\text{TS}}$) signal is both an input and an output signal on the MPC106.

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

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

State Meaning Asserted—Indicates that the MPC106 has started a bus transaction, and that the address and transfer attribute signals are valid. Note that the MPC106 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 two clock cycles after $\overline{\text{BGn}}$ is negated and the address bus is idle.

Negation—Occurs one clock cycle after assertion.

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

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

Following are the state meaning and timing comments for the $\overline{\text{TS}}$ 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 clock cycle after $\overline{\text{BGn}}$ is asserted.

Negation—Occurs one clock cycle after assertion.

2.2.2.17 Transfer Size (TSIZ[0–2])

The transfer size (TSIZ[0–2]) signals consist of three input and output signals on the MPC106.

2.2.2.17.1 Transfer Size (TSIZ[0–2])—Output

Following are the state meaning and timing comments for TSIZ[0–2] as output signals. Note that all MPC106-generated snoop operations are eight-word bursts; therefore TSIZ[0–2] are always 0b010 for snoop operations.

State Meaning Asserted/Negated—In conjunction with the transfer burst ($\overline{\text{TBST}}$) signal, TSIZ[0–2] specify the data transfer size for the 60x bus

transaction. Refer to Section 4.3.2.2, “TBST and TSIZ[0–2] Signals and Size of Transfer,” for transfer size encodings.

Timing Comments Assertion/Negation—The same as A[0–31].
High-impedance—The same as A[0–31].

2.2.2.17.2 Transfer Size (TSIZ[0–2])—Input

Following are the state meaning and timing comments for TSIZ[0–2] as input signals.

State Meaning Asserted/Negated—In conjunction with the transfer burst ($\overline{\text{TBST}}$) signal, TSIZ[0–2] specify the data transfer size for the 60x bus transaction. Refer to Section 4.3.2.2, “TBST and TSIZ[0–2] Signals and Size of Transfer,” for transfer size encodings.

Timing Comments Assertion/Negation—The same as A[0–31].
High-impedance—The same as A[0–31].

2.2.2.18 Transfer Type (TT[0–4])

The transfer type (TT[0–4]) signals consist of five input and output signals on the MPC106.

2.2.2.18.1 Transfer Type (TT[0–4])—Output

Following are the state meaning and timing comments for TT[0–4] 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 A[0–31].
High-impedance—The same as A[0–31].

2.2.2.18.2 Transfer Type (TT[0–4])—Input

Following are the state meaning and timing comments for TT[0–4] 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 A[0–31].
High-impedance—The same as A[0–31].

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

The write-through ($\overline{\text{WT}}$) signal is both an input and output signal on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{WT}}$ signal.

State Meaning	Asserted—Indicates that an access is write-through.
	Negated—Indicates that an access is write-back. Note that \overline{WT} is always negated for snoop cycles initiated by the MPC106.
Timing Comments	Assertion/Negation—The same as A[0–31].
	High-impedance—The same as A[0–31].

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

The \overline{XATS} signal is an input on the MPC106. 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 MPC106, the MPC106 automatically asserts the \overline{TEA} signal when \overline{XATS} is asserted (provided \overline{TEA} is enabled). If \overline{TEA} is disabled, the MPC106 terminates the direct-store access by asserting \overline{TA} ; however, no data is altered for write operations, and invalid data is returned on read operations.
	Negated—Has no special meaning.
Timing Comments	Assertion—May occur one clock cycle after \overline{BGn} is asserted. The \overline{XATS} signal can only be asserted by a processor (that is, the MPC106 cannot assert \overline{XATS}).
	Negation—Occurs one clock cycle after assertion.

2.2.3 L2 Cache/Multiple Processor Interface Signals

The MPC106 provides support for either an internal L2 controller or an external L2 controller and/or multiple 60x processors.

The signals $\overline{ADS}/\overline{DALE}/\overline{BRL2}$, $\overline{BA1}/\overline{BAA}/\overline{BGL2}$, $\overline{DOE}/\overline{DBGL2}$, and \overline{HIT} function differently depending on whether the MPC106 is in the internal L2 controller or external L2 controller mode.

The signals $\overline{BA0}/\overline{BR3}$, $\overline{DCS}/\overline{BG3}$, $\overline{DIRTY_IN}/\overline{BR1}$, $\overline{DIRTY_OUT}/\overline{BG1}$, $\overline{DWE0}/\overline{DBG2}$, $\overline{DWE1}/\overline{DBG3}$, $\overline{TOE}/\overline{DBG1}$, $\overline{TV}/\overline{BR2}$, and $\overline{TWE}/\overline{BG2}$ function differently depending on whether the MPC106 is in the internal L2 controller or multiple 60x processor mode.

Section 2.2.3.1, “Internal L2 Controller Signals,” describes the internal L2 controller configuration for these signals; Section 2.2.3.2, “External L2 Controller Signals,” describes the external L2 controller configuration for these signals; and Section 2.2.3.3, “Multiple Processor Interface Signals,” describes the multiple processor configuration for these signals.

2.2.3.1 Internal L2 Controller Signals

This section provides a brief description of the interface signals for the internal L2 controller. The internal L2 controller supports synchronous burst SRAMs, pipelined burst SRAMs, and asynchronous SRAMs. Some of the signals perform different functions depending on the SRAM configuration.

2.2.3.1.1 Address Strobe ($\overline{\text{ADS}}$)—Output

The address strobe ($\overline{\text{ADS}}$) signal is an output on the MPC106. It is used for burst SRAM configurations only. Asynchronous SRAM configurations use the alternate function $\overline{\text{DALE}}$. Following are the state meaning and timing comments for the $\overline{\text{ADS}}$ signal. Note that this signal has an on-chip pull-up resistor.

State Meaning	Asserted—Indicates that the address is valid to be latched by the burst SRAMs.
	Negated—Indicates that the burst SRAMs should use addresses from their internal counters.
Timing Comments	Assertion—The MPC106 asserts $\overline{\text{ADS}}$ during the 60x bus address phase. The MPC106 also asserts $\overline{\text{ADS}}$ when a write cycle needs to be aborted.
	Negation—The MPC106 negates $\overline{\text{ADS}}$ until the data access is completed.

2.2.3.1.2 Burst Address 0 (BA0)—Output

The burst address 0 (BA0) signal is an output on the MPC106. It is used for asynchronous SRAM configurations only. Following are the state meaning and timing comments for the BA0 signal.

State Meaning	Asserted/Negated—Indicates the most-significant bit of the burst counter address.
Timing Comments	Assertion/Negation—Valid when $\overline{\text{DALE}}$ is negated, after each $\overline{\text{TA}}$ assertion.

2.2.3.1.3 Burst Address 1 (BA1)—Output

The burst address 1 (BA1) signal is an output on the MPC106. It is used for asynchronous SRAM configurations only. Burst SRAM configurations use the alternate function $\overline{\text{BAA}}$. Following are the state meaning and timing comments for the BA1 signal.

State Meaning	Asserted/Negated—Indicates the least-significant bit (lsb) of the burst counter address.
Timing Comments	Assertion/Negation—Valid when $\overline{\text{DALE}}$ is negated, after each $\overline{\text{TA}}$ assertion.

2.2.3.1.4 Bus Address Advance ($\overline{\text{BAA}}$)—Output

The bus address advance ($\overline{\text{BAA}}$) signal is an output on the MPC106. It is used for burst SRAM configurations only. Asynchronous SRAM configurations use the alternate function BA1. Following are the state meaning and timing comments for the $\overline{\text{BAA}}$ signal.

- State Meaning** Asserted—Indicates that the burst SRAMs should increment their internal addresses.
- Negated—Indicates no change to the addresses.
- Timing Comments** Assertion/Negation—The MPC106 asserts $\overline{\text{BAA}}$ together with $\overline{\text{TA}}$ during a read access and one clock cycle after $\overline{\text{TA}}$ during write accesses (to advance the burst address).

2.2.3.1.5 Data Address Latch Enable ($\overline{\text{DALE}}$)—Output

The data address latch enable ($\overline{\text{DALE}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{DALE}}$ signal.

- State Meaning** Asserted—Indicates the external address latch should latch the current 60x bus address.
- Negated—Indicates the external address latch should be transparent.
- Timing Comments** Assertion—Occurs when the data SRAM access starts and the address is valid.
- Negation—Occurs when the data access is completed.

2.2.3.1.6 Data RAM Chip Select ($\overline{\text{DCS}}$)—Output

The data RAM chip select ($\overline{\text{DCS}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{DCS}}$ signal.

- State Meaning** Asserted—Enables the L2 data RAMs for read or write operations.
- Negated—Disables the L2 data RAMs.
- Timing Comments** Assertion/Negation—For a burst SRAM configuration, $\overline{\text{DCS}}$ is valid when $\overline{\text{ADS}}$ is asserted.
- or–
- For a pipelined burst SRAM configuration, $\overline{\text{DCS}}$ is valid when $\overline{\text{ADS}}$ or $\overline{\text{TS}}$ is asserted.
- or–
- For an asynchronous SRAM configuration, $\overline{\text{DCS}}$ is asserted when a data read or write is in progress. $\overline{\text{DCS}}$ is negated when the L2 cache is idle.

2.2.3.1.7 Dirty In ($\overline{\text{DIRTY_IN}}$)—Input

The dirty in ($\overline{\text{DIRTY_IN}}$) signal is an input on the MPC106. The polarity of the $\overline{\text{DIRTY_IN}}$ signal is programmable by using the PICR2[CF_MOD_HIGH] parameter; see Section 3.2.9, “Processor Interface Configuration Registers,” for more information. Following are the state meaning and timing comments for the $\overline{\text{DIRTY_IN}}$ 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 $\overline{\text{DIRTY_IN}}$ signal is valid when the L2 hit delay after $\overline{\text{TS}}$ expires. The $\overline{\text{DIRTY_IN}}$ signal is held valid until the end of the address phase.

2.2.3.1.8 Dirty Out ($\overline{\text{DIRTY_OUT}}$)—Output

The dirty out ($\overline{\text{DIRTY_OUT}}$) signal is an output on the MPC106. The polarity of the $\overline{\text{DIRTY_OUT}}$ signal is programmable by using the PICR2[CF_MOD_HIGH] parameter; see Section 3.2.9, “Processor Interface Configuration Registers,” for more information. Following are the state meaning and timing comments for the $\overline{\text{DIRTY_OUT}}$ 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 $\overline{\text{DIRTY_OUT}}$ signal is valid when the tag write enable ($\overline{\text{TWE}}$) signal is asserted to indicate a new line status. The $\overline{\text{DIRTY_OUT}}$ signal is held valid for one clock cycle after $\overline{\text{TWE}}$ is negated.

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

The data RAM output enable ($\overline{\text{DOE}}$) signal is an output on the MPC106. 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.

2.2.3.1.10 Data RAM Write Enable ($\overline{\text{DWE}}[0-2]$)—Output

The data RAM write enable ($\overline{\text{DWE}}[0-2]$) signals are outputs on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{DWE}}_n$ signals.

State Meaning Asserted—Indicates that a write to the L2 data RAMs is in progress.
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. Note that all the $\overline{\text{DWE}}$ signals have the same timing. They are not gated by byte enables. Multiple $\overline{\text{DWE}}$ s are used to reduce loading.

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

The hit ($\overline{\text{HIT}}$) signal is an input on the MPC106. The polarity of the $\overline{\text{HIT}}$ signal is programmable by using the PICR2[CF_HIT_HIGH] parameter; see Section 3.2.9, “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.

Timing Comments Assertion/Negation—The $\overline{\text{HIT}}$ signal is 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.

2.2.3.1.12 Tag Output Enable ($\overline{\text{TOE}}$)—Output

The tag output enable ($\overline{\text{TOE}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{TOE}}$ 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 copyback cycles (depending on PICR2[CF_TOE_WIDTH]); see Chapter 5, “Secondary Cache Interface,” for more detailed timing information.

2.2.3.1.13 Tag Valid (TV)—Output

The tag valid (TV) signal is an output on the MPC106. The polarity of the TV signal is programmable by using the PICR2[CF_MOD_HIGH] parameter; see Section 3.2.9, “Processor Interface Configuration Registers,” for more information. Also, note that this signal has an on-chip pull-up resistor. 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 that the current L2 cache line should be marked invalid.

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

High-impedance—The TV signal is either released to a high-impedance state during tag read operations or always driven depending upon the parameters PICR2[CF_FAST_CASTOUT] and PICR2[CF_HOLD]; see Section 3.2.9, “Processor Interface Configuration Registers,” for more information.

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

The tag write enable ($\overline{\text{TWE}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{TWE}}$ signal.

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

Negated—Indicates that updating the tag address, valid, and dirty bits is not currently necessary.

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

2.2.3.2 External L2 Controller Signals

When an external L2 cache controller is used instead of the internal L2 controller, four signals change their functions. This section provides a brief description of the signals used by the MPC106 to interface with the external L2 cache controller.

2.2.3.2.1 External L2 Bus Grant ($\overline{\text{BGL2}}$)—Output

The external L2 bus grant ($\overline{\text{BGL2}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{BGL2}}$ signal.

- State Meaning** Asserted—Indicates that the external L2 controller may assume mastership of the 60x address bus.
- Negated—Indicates that the external L2 controller is not granted mastership of the next 60x address bus tenure.
- Timing Comments** Assertion—May occur at any time when the external L2 bus request ($\overline{\text{BRL2}}$) signal is asserted and the 60x address bus is available.
- Negation—May occur at any time after assertion, or after $\overline{\text{BRL2}}$ is negated.

2.2.3.2.2 External L2 Bus Request ($\overline{\text{BRL2}}$)—Input

The external L2 bus request ($\overline{\text{BRL2}}$) signal is an input on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{BRL2}}$ signal. Note that this signal has an on-chip pull-up resistor.

- State Meaning** Asserted—Indicates that the external L2 controller requires mastership of the 60x bus for a transaction.
- Negated—Indicates that the external L2 controller does not require mastership of the 60x bus.
- Timing Comments** Assertion—May occur at any time.
- Negation—May occur at any time. However, $\overline{\text{BRL2}}$ must be negated for at least one clock cycle after an accepted, qualified external L2 bus grant.

2.2.3.2.3 External L2 Data Bus Grant ($\overline{\text{DBGL2}}$)—Output

The external L2 data bus grant ($\overline{\text{DBGL2}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{DBGL2}}$ signal.

- State Meaning** Asserted—Indicates that the external L2 controller may assume mastership of the 60x data bus.
- Negated—Indicates that the external L2 controller is not granted mastership of the data bus.
- Timing Comments** Assertion—Occurs on the first clock cycle in which the data bus is not busy and the external L2 controller has the highest priority outstanding data transaction.
- Negation—Occurs one clock cycle after assertion.

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

The hit ($\overline{\text{HIT}}$) signal is an input on the MPC106. For the external L2 controller, the polarity of the $\overline{\text{HIT}}$ signal is always active low. That is, it is not affected by the

PICR2[CF_HIT_HIGH] parameter. Following are the state meaning and timing comments for the HIT signal.

State Meaning Asserted—Indicates that the current transaction is claimed by the external L2 controller. The external L2 controller will assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$ for the transaction.

Negated—Indicates that the current transaction is not claimed by the external L2 controller. The MPC106 should handle the transaction and control $\overline{\text{AACK}}$ and $\overline{\text{TA}}$ as appropriate.

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

2.2.3.3 Multiple Processor Interface Signals

When a system implementation uses more than one 60x processor, nine of the internal L2 controller signals change their functions. This section provides a brief description of the multiple processor interface signals. Note that in a multiprocessor system, with the exception of bus request ($\overline{\text{BR}}n$), bus grant ($\overline{\text{BG}}n$), and data bus grant ($\overline{\text{DBG}}n$), all of the 60x processor interface signals are connected to each processor. See Section 4.1.2, “Multiprocessor System Configuration,” for more information.

2.2.3.3.1 Bus Grant 1–3 ($\overline{\text{BG}}[1-3]$)—Output

The bus grant ($\overline{\text{BG}}[1-3]$) signals are outputs on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{BG}}n$ signals.

State Meaning Asserted—Indicates that processor n (where n is 1, 2, or 3) may, with the proper qualification, begin a bus transaction and assume mastership of the address bus.

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

Timing Comments Assertion—Occurs when $\overline{\text{BR}}n$ is the highest-priority request that is asserted.

Negation—Occurs when other higher-priority transactions are pending.

2.2.3.3.2 Bus Request 1–3 ($\overline{\text{BR}}[1-3]$)—Input

The bus request ($\overline{\text{BR}}[1-3]$) signals are inputs on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{BR}}n$ signals.

State Meaning Asserted—Indicates that processor n (where n is 1, 2, or 3) requires mastership of the 60x bus for a transaction.

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

Timing Comments Assertion—May occur when $\overline{B}Gn$ is negated and a bus transaction is needed by processor n . 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 processor n . It is also negated for at least one bus cycle when the assertion of \overline{ARTRY} is detected on the 60x bus (except for assertions due to 60x snoop copyback operations).

2.2.3.3.3 Data Bus Grant 1–3 ($\overline{DBG}[1-3]$)—Output

The data bus grant ($\overline{DBG}[1-3]$) signals are outputs on the MPC106. Following are the state meaning and timing comments for the $\overline{DBG}n$ signals.

State Meaning Asserted—Indicates that processor n (where n is 1, 2, or 3) may, with the proper qualification, assume mastership of the data bus. A qualified data bus grant is defined as the assertion of $\overline{DBG}n$, negation of \overline{DBB} , and negation of \overline{ARTRY} . The \overline{ARTRY} signal requirement 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 processor n is not granted mastership of the data bus.

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

Negation—Occurs one clock after assertion.

2.2.4 Memory Interface Signals

The memory interface supports either standard DRAMs, extended data out DRAMs (EDO DRAMs), or synchronous DRAMs (SDRAMs) and either standard ROM or Flash devices. Some of the memory interface signals perform different functions depending on the RAM and ROM configurations. This section provides a brief description of the memory interface signals on the MPC106.

2.2.4.1 ROM Address 0 (AR0)—Output

The ROM address 0 (AR0) signal is an output signal on the MPC106.

Note that the AR0 signal is only supported for ROM bank 0 when configured for an 8-bit ROM data bus width.

Following are the state meaning and timing comments for the AR0 output signal.

State Meaning Asserted/Negated—Represents address bit 0 (the most-significant bit) of the 8-bit ROM/Flash. Bits 1–20 of the ROM address are provided by AR[1–8] and AR[9–20].

Timing Comments Assertion/Negation—The ROM address is valid on assertion of $\overline{\text{RCS0}}$ or $\overline{\text{RCS1}}$.

2.2.4.2 ROM Address 1–8 (AR[1–8])—Output

The ROM address 1–8 (AR[1–8]) signals are output signals only for the ROM address function. Note that these signals are both input and output signals for the memory parity function (PAR[0–7]). Following are the state meaning and timing comments for AR[1–8] as output signals.

State Meaning Asserted/Negated—Represents bits 1–8 of the ROM/Flash address. The other ROM address bits are provided by AR0 and AR[9–20].

Timing Comments Assertion/Negation—The ROM address is valid on assertion of $\overline{\text{RCS0}}$ or $\overline{\text{RCS1}}$.

2.2.4.3 ROM Address 9–20 (AR[9–20])—Output

The ROM address (AR[9–20]) signals consist of 12 output signals on the MPC106. Following are the state meaning and timing comments for the AR[9–20] output signals.

State Meaning Asserted/Negated—Represents bits 9–20 of the ROM/Flash address (the 12 lowest-order bits, with AR20 as the lsb). Bits 0–8 of the ROM address are provided by AR0 and AR[1–8].

Timing Comments Assertion/Negation—The ROM address is valid on assertion of $\overline{\text{RCS0}}$ or $\overline{\text{RCS1}}$.

2.2.4.4 Buffer Control ($\overline{\text{BCTL}}[0–1]$)—Output

The two buffer control ($\overline{\text{BCTL}}[0–1]$) signals are outputs on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{BCTL}}[0–1]$ output signals. Note that $\overline{\text{BCTL0}}$ also serves as a configuration input at power-on reset. See Section 2.2.9, “Configuration Signals,” for more information.

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 (on the 60x processor/memory bus) are both in the system or if ECC is enabled.

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

2.2.4.5 Column Address Strobe ($\overline{\text{CAS}}[0-7]$)—Output

The eight column address strobe ($\overline{\text{CAS}}[0-7]$) signals are outputs on the MPC106. $\overline{\text{CAS}}0$ connects to the most-significant byte select. $\overline{\text{CAS}}7$ connects to the least-significant byte select. Following are the state meaning and timing comments for the $\overline{\text{CAS}}n$ output signals.

State Meaning Asserted—Indicates that the DRAM (or EDO) column address is valid and selects one of the columns in the row.

Negated—For DRAMs, indicates $\overline{\text{CAS}}$ precharge; the current DRAM data transfer has completed.

—or—

For EDO DRAMs, indicates $\overline{\text{CAS}}$ precharge; the current data transfer completes in the first clock cycle of $\overline{\text{CAS}}$ precharge.

Timing Comments Assertion—The MPC106 asserts $\overline{\text{CAS}}n$ two to eight clock cycles after the assertion of $\overline{\text{RAS}}n$ (depending on the setting of the MCCR3[RCD₂] parameter). See Section 6.4.4, “DRAM/EDO Interface Timing,” for more information.

2.2.4.6 SDRAM Clock Enable (CKE)—Output

The SDRAM clock enable (CKE) signal is an output on the MPC106. Following are the state meaning and timing comments for the CKE output signal.

State Meaning Asserted—Enables the internal clock circuit of the SDRAM memory device. Also, CKE is part of the SDRAM command encoding.

Negated—Disables the internal clock circuit of the SDRAM memory device. Also, CKE is part of the SDRAM command encoding. Note that the MPC106 negates CKE during certain system power-down situations.

Timing Comments Assertion—CKE is valid on the rising edge of the 60x bus clock. See Section 6.5, “SDRAM Interface Operation,” for more information.

2.2.4.7 SDRAM Chip Select ($\overline{\text{CS}}[0-7]$)—Output

The eight SDRAM chip select ($\overline{\text{CS}}[0-7]$) signals are outputs on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{CS}}n$ output signals.

State Meaning Asserted—Selects an SDRAM bank to perform a memory operation.

Negated—Indicates no SDRAM action during the current cycle.

Timing Comments Assertion—The MPC106 asserts the \overline{CS}_n signal to begin a memory cycle. For SDRAM, \overline{CS}_n must be valid on the rising edge of the 60x bus clock.

2.2.4.8 SDRAM Data Input/Output Mask (DQM[0–7])—Output

The eight SDRAM data input/output mask (DQM[0–7]) signals are outputs on the MPC106. Following are the state meaning and timing comments for the DQM $_n$ output signals.

State Meaning Asserted—Prevents writing to SDRAM. (Note that the DQM $_n$ signals are active high for SDRAM.)

Negated—Allows a read or write operation to SDRAM.

DQM0 connects to the most significant byte select.

DQM7 connects to the least significant byte select.

Timing Comments Assertion—For SDRAM, DQM $_n$ must be valid on the rising edge of the 60x bus clock during read or write cycles.

2.2.4.9 Flash Output Enable (\overline{FOE})—Output

The Flash output enable (\overline{FOE}) signal is an output on the MPC106. Following are the state meaning and timing comments for the \overline{FOE} output signal. Note that \overline{FOE} also serves as a configuration input at power-on reset. See Section 2.2.9, “Configuration Signals,” for more information.

State Meaning Asserted—Enables Flash output for the current read access.

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

Note that the \overline{FOE} signal provides no indication of any write operation(s) to Flash.

Timing Comments Assertion—The MPC106 asserts \overline{FOE} at the start of the Flash read cycle.

2.2.4.10 Memory Address (MA[0–12])—Output

The memory address (MA[0–12]) signals consist of 13 output signals on the MPC106. Following are the state meaning and timing comments for the MA[0–12] output signals.

State Meaning Asserted/Negated—Represents the row/column multiplexed physical address for DRAMs or EDOs (MA0 is the most-significant address bit; MA12 is the least-significant address bit).

Timing Comments Assertion—The row address is valid on assertion of \overline{RAS}_n , and the column address is valid on assertion of \overline{CAS}_n .

2.2.4.11 Memory Data Latch Enable ($\overline{\text{MDLE}}$)—Output

The memory data latch enable ($\overline{\text{MDLE}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{MDLE}}$ output signal.

- State Meaning** Asserted— $\overline{\text{MDLE}}$ enables an external latched data buffer for read operations, if such a buffer is used in the system.
- Negated— $\overline{\text{MDLE}}$ disables the external latched data buffer, if such a buffer is used in the system.
- Timing Comments** Assertion— For systems that use an external data buffer, $\overline{\text{MDLE}}$ follows $\overline{\text{CAS}}$ timing for DRAM read operations, follows $\overline{\text{CAS}}$ precharge timing for EDO read operations, and follows $\overline{\text{RCS}}[0-1]$ timing for ROM/Flash read operations.

2.2.4.12 Data Parity/ECC (PAR[0-7])

The eight data parity/ECC (PAR[0-7]) signals are both input and output signals on the MPC106.

2.2.4.12.1 Data Parity (PAR[0-7])—Output

Following are the state meaning and timing comments for PAR[0-7] as output signals.

- State Meaning** Asserted/Negated—Represents the byte parity or ECC being written to memory (PAR0 is the most-significant parity bit and corresponds to byte lane 0 which is selected by $\overline{\text{CAS}}/\text{DQM}[0]$). The data parity signals are asserted or negated as appropriate to provide odd parity (including the parity bit) or ECC.
- Timing Comments** Assertion/Negation—PAR[0-7] are valid concurrent with DH[0-31] and DL[0-31].

2.2.4.12.2 Data Parity (PAR[0-7])—Input

Following are the state meaning and timing comments for PAR[0-7] as input signals.

- State Meaning** Asserted/Negated—Represents the byte parity or ECC being read from memory (PAR0 is the most-significant parity bit and corresponds to byte lane 0 which is selected by $\overline{\text{CAS}}/\text{DQM}[0]$).
- Timing Comments** Assertion/Negation—PAR[0-7] are valid concurrent with DH[0-31] and DL[0-31].

2.2.4.13 Parity Path Read Enable ($\overline{\text{PPEN}}$)—Output

The parity path read enable ($\overline{\text{PPEN}}$) signal is an output on the MPC106 that controls external parity path transceivers between the 60x bus and memory. See Section 6.2.4, “Parity/ECC Path Read Control,” for more information. Following are the state meaning

and timing comments for the $\overline{\text{PPEN}}$ output signal.

State Meaning Asserted— $\overline{\text{PPEN}}$ acts as a read enable ($\overline{\text{RE}}$) for the parity bus transceiver (for $\text{MCCR2}[\text{BUF_MODE}] = 0$) or act as an output enable ($\overline{\text{OE}}$) for the parity bus transceiver (for $\text{MCCR2}[\text{BUF_MODE}] = 1$).

Negated— $\overline{\text{PPEN}}$ disables the parity bus buffer.

Timing Comments Assertion/Negation—For normal parity or no parity, $\overline{\text{PPEN}}$ is asserted with the first read $\overline{\text{CAS}}/\text{DQM}n$ and held valid throughout the read burst. For ECC, $\overline{\text{PPEN}}$ is asserted with the first read $\overline{\text{CAS}}n$ and negated during the bus turnaround cycle.

2.2.4.14 Row Address Strobe ($\overline{\text{RAS}}[0-7]$)—Output

The eight row address strobe ($\overline{\text{RAS}}[0-7]$) signals are outputs on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{RAS}}n$ output signals.

State Meaning Asserted—Indicates that the memory row address is valid and selects one of the rows in the selected bank.

Negated—Indicates DRAM precharge period.

Timing Comments Assertion—The MPC106 asserts the $\overline{\text{RAS}}n$ signal to begin a memory cycle. All other memory interface signal timings are referenced to $\overline{\text{RAS}}n$.

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

The ROM bank 0 select ($\overline{\text{RCS0}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{RCS0}}$ output signal. Note that $\overline{\text{RCS0}}$ also serves as a configuration input at power-on reset. See Section 2.2.9, “Configuration Signals,” for more information.

State Meaning Asserted—Selects ROM bank 0 for a read access or Flash bank 0 for a read or write access.

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

Timing Comments Assertion—The MPC106 asserts $\overline{\text{RCS0}}$ at the start of a ROM/Flash access cycle.

2.2.4.16 ROM Bank 1 Select ($\overline{\text{RCS1}}$)—Output

The ROM bank 1 select ($\overline{\text{RCS1}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{RCS1}}$ output signal.

State Meaning Asserted—Selects ROM bank 1 for a read access or Flash bank 1 for a read or write access.

Negated—Deselects bank 1, indicating no pending memory access to ROM/Flash.

Timing Comments Assertion—The MPC106 asserts $\overline{\text{RCSI}}$ at the start of a ROM/Flash access cycle.

2.2.4.17 Real Time Clock (RTC)—Input

The real time clock (RTC) signal is an input on the MPC106. 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 MPC106 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.

2.2.4.18 SDRAM Internal Bank Select 0–1 (SDBA[0–1])—Output

The SDRAM internal bank select (SDBA[0–1]) signals are output signals on the MPC106. Following are the state meaning and timing comments for the SDBA[0–1] output signals. Note that SDBA1/SDMA0 is only valid with 64- and 128-Mbit devices.

State Meaning Asserted/negated—Selects the SDRAM internal bank during bank activate, read, write, or precharge bank commands.

Timing Comments Assertion/Negation—The same as SDMA[0–12].

2.2.4.19 SDRAM Column Address Strobe ($\overline{\text{SDCAS}}$)—Output

The SDRAM column address strobe ($\overline{\text{SDCAS}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{SDCAS}}$ output signal.

State Meaning Asserted— $\overline{\text{SDCAS}}$ is part of the SDRAM command encoding and is used for SDRAM column selection during read or write operations. See Section 6.5, “SDRAM Interface Operation,” for more information.

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

Timing Comments Assertion—For SDRAM, $\overline{\text{SDCAS}}$ is valid on the rising edge of the 60x bus clock when a $\overline{\text{CS}}_n$ signal is asserted.

2.2.4.20 SDRAM Address (SDMA[0–12])—Output

The SDRAM address (SDMA[0–12]) signals consist of 13 output signals on the MPC106.

Following are the state meaning and timing comments for the SDMA[0–12] output signals. Note that SDBA1/SDMA0 and SDMA1 are only valid with 64- and 128-Mbit devices.

State Meaning Asserted/Negated—Represents the row/column multiplexed physical address for SDRAMs (SDMA0 is the most-significant address bit; SDMA12 is the least-significant address bit).

Timing Comments Assertion/Negation—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 $\overline{\text{SDCAS}}$ is asserted.

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

The SDRAM row address strobe ($\overline{\text{SDRAS}}$) signal is an output on the MPC106. 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 6.5, “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 $\overline{\text{CS}}_n$ signal is asserted.

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

The write enable ($\overline{\text{WE}}$) signal is an output on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{WE}}$ output signal.

State Meaning Asserted—Enables writing to DRAM, EDO, or Flash.
–or–
For SDRAM, $\overline{\text{WE}}$ is part of the SDRAM command encoding. See Section 6.5, “SDRAM Interface Operation,” for more information.
Negated—No DRAM, EDO, or Flash write operation is pending.

Timing Comments Assertion—For DRAM, the MPC106 asserts $\overline{\text{WE}}$ concurrent with the column address and prior to $\overline{\text{CAS}}_n$. For SDRAM, the MPC106 asserts $\overline{\text{WE}}$ concurrent with $\overline{\text{SDCAS}}$ for write operations. For Flash, the MPC106 asserts $\overline{\text{WE}}$ one clock cycle after $\overline{\text{RCS}}_n$ is asserted and negates $\overline{\text{WE}}$ one clock cycle after $\overline{\text{RCS}}_n$ is negated.

2.2.5 PCI Interface Signals

This section provides descriptions of the PCI interface signals on the MPC106. Note that throughout this manual, signals and bits of the PCI interface are referenced in little-endian format.

2.2.5.1 PCI Address/Data Bus (AD[31–0])

The PCI address/data bus (AD[31–0]) consists of 32 signals that are both input and output signals on the MPC106.

2.2.5.1.1 Address/Data (AD[31–0])—Output

Following is the state meaning for AD[31–0] 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, AD[31–0] contain data being written.

The AD[7–0] signals define the least-significant byte and AD[31–24] the most-significant byte.

2.2.5.1.2 Address/Data (AD[31–0])—Input

Following is the state meaning for AD[31–0] as input signals.

State Meaning Asserted/Negated—Represents the address to be decoded as a 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.

2.2.5.2 Command/Byte Enable ($\overline{C/BE}$ [3–0])

The four command/byte enable ($\overline{C/BE}$ [3–0]) signals are both input and output signals on the MPC106.

2.2.5.2.1 Command/Byte Enable ($\overline{C/BE}$ [3–0])—Output

Following is the state meaning for $\overline{C/BE}$ [3–0] as output signals.

State Meaning Asserted/Negated—During the address phase, $\overline{C/BE}$ [3–0] define the bus command. Table 2-4 specifies the PCI bus command encodings. See Section 7.3.2, “PCI Bus Commands,” for more information. During the data phase, $\overline{C/BE}$ [3–0] are used as byte enables. Byte enables determine which byte lanes carry meaningful data. The $\overline{C/BE0}$ signal applies to the least-significant byte.

Table 2-4. PCI Command Encodings

$\overline{C/BE}$ [3–0]	PCI Command
0000	Interrupt acknowledge ¹
0001	Special cycle ¹
0010	I/O read ¹
0011	I/O write ¹

Table 2-4. PCI Command Encodings

$\overline{C/BE}[3-0]$	PCI Command
0100	Reserved
0101	Reserved
0110	Memory read
0111	Memory write
1000	Reserved
1001	Reserved
1010	Configuration read ¹
1011	Configuration write ¹
1100	Memory read multiple
1101	Dual access cycle ¹
1110	Memory read line
1111	Memory write and invalidate

¹The MPC106 does not respond to this command

2.2.5.2.2 Command/Byte Enable ($\overline{C/BE}[3-0]$)—Input

Following is the state meaning for $\overline{C/BE}[3-0]$ as input signals.

State Meaning Asserted/Negated—During the address phase, $\overline{C/BE}[3-0]$ indicate the command that another master is sending. Table 2-4 specifies the PCI bus command encodings. See Section 7.3.2, “PCI Bus Commands,” for more information. During the data phase, $\overline{C/BE}[3-0]$ indicate which byte lanes are valid.

2.2.5.3 Device Select (\overline{DEVSEL})

The device select (\overline{DEVSEL}) signal is both an input and output signal on the MPC106.

2.2.5.3.1 Device Select (\overline{DEVSEL})—Output

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

State Meaning Asserted—Indicates that the MPC106 has decoded the address and is the target of the current access.

Negated—Indicates that the MPC106 has decoded the address and is not the target of the current access.

2.2.5.3.2 Device Select (\overline{DEVSEL})—Input

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

State Meaning Asserted—Indicates that some PCI agent (other than the MPC106) has decoded its address as the target of the current access.
Negated—Indicates that no PCI agent has been selected.

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

The frame ($\overline{\text{FRAME}}$) signal is both an input and output signal on the MPC106.

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

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

State Meaning Asserted—Indicates that the MPC106, 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; if $\overline{\text{IRDY}}$ is negated, indicates that the PCI bus is idle.

2.2.5.4.2 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.

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

The PCI bus grant ($\overline{\text{GNT}}$) signal is an input signal on the MPC106. Note that $\overline{\text{GNT}}$ is a point-to-point signal. Every master has its own $\overline{\text{GNT}}$ signal. Following is the state meaning for the $\overline{\text{GNT}}$ input signal.

State Meaning Asserted—Indicates that the MPC106 has been granted control of the PCI bus. If $\overline{\text{GNT}}$ is asserted before the MPC106 has a transaction to perform (that is, the MPC106 is parked), the MPC106 drives AD[31–0], C/BE[3–0], and PAR to stable (but meaningless) states until they are needed for a legitimate transaction.

Negated—Indicates that the MPC106 has not been granted control of the PCI bus, and cannot initiate a PCI transaction.

2.2.5.6 Initiator Ready ($\overline{\text{IRDY}}$)

The initiator ready ($\overline{\text{IRDY}}$) signal is both an input and output signal on the MPC106.

2.2.5.6.1 Initiator Ready ($\overline{\text{IRDY}}$)—Output

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

State Meaning Asserted—Indicates that the MPC106, acting as a PCI master, can complete the current data phase of a PCI transaction. During a write, the MPC106 asserts $\overline{\text{IRDY}}$ to indicate that valid data is present on AD[31–0]. During a read, the MPC106 asserts $\overline{\text{IRDY}}$ to indicate that it is prepared to accept data.

Negated—Indicates that the PCI target needs to wait before the MPC106, acting as a PCI master, can complete the current data phase. During a write, the MPC106 negates $\overline{\text{IRDY}}$ to insert a wait cycle when it cannot provide valid data to the target. During a read, the MPC106 negates $\overline{\text{IRDY}}$ to insert a wait cycle when it cannot accept data from the target.

2.2.5.6.2 Initiator 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—If $\overline{\text{FRAME}}$ is asserted, indicates a wait cycle from another master. If $\overline{\text{FRAME}}$ is negated, indicates the PCI bus is idle.

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

The lock ($\overline{\text{LOCK}}$) signal is an input on the MPC106. See Section 7.5, “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.

2.2.5.8 Parity (PAR)

The PCI parity (PAR) signal is both an input and output signal on the MPC106. See Section 7.6.1, “PCI Parity,” for more information.

2.2.5.8.1 Parity (PAR)—Output

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

State Meaning Asserted—Indicates odd parity across the AD[31–0] and $\overline{\text{C/BE}}$ [3–0] signals during address and data phases.

Negated—Indicates even parity across the AD[31–0] and C/BE[3–0] signals during address and data phases.

2.2.5.8.2 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.

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

The PCI parity error ($\overline{\text{PERR}}$) signal is both an input and output signal on the MPC106.

2.2.5.9.1 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 MPC106, acting as a PCI agent, detected a data parity error. (The PCI initiator drives $\overline{\text{PERR}}$ on read operations; the PCI target drives $\overline{\text{PERR}}$ on write operations.)

Negated—Indicates no error.

2.2.5.9.2 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 MPC106 was sourcing data (the MPC106 was acting as the PCI initiator during a write, or was acting as the PCI target during a read).

Negated—Indicates no error.

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

The PCI bus request ($\overline{\text{REQ}}$) signal is an output signal on the MPC106. Note that $\overline{\text{REQ}}$ is a point-to-point signal. Every master has its own $\overline{\text{REQ}}$ signal. Following is the state meaning for the $\overline{\text{REQ}}$ output signal.

State Meaning Asserted—Indicates that the MPC106 is requesting control of the PCI bus to perform a transaction. If the PCI bus grant ($\overline{\text{GNT}}$) signal is asserted before the MPC106 has a transaction to perform (that is, the MPC106 is parked), then $\overline{\text{REQ}}$ is not asserted.

Negated—Indicates that the MPC106 does not require use of the PCI bus.

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

The PCI system error ($\overline{\text{SERR}}$) signal is both an input and output signal on the MPC106.

2.2.5.11.1 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, a target-abort (when the MPC106 is acting as the initiator), or some other system error (where the result is a catastrophic error) was detected.

 Negated—Indicates no error.

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

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

State Meaning Asserted—Indicates that a target (other than the MPC106) has detected a catastrophic error.

 Negated—Indicates no error.

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

The stop ($\overline{\text{STOP}}$) signal is both an input and output signal on the MPC106.

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

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

State Meaning Asserted—Indicates that the MPC106, acting as a PCI target, is requesting that the initiator stop the current transaction.

 Negated—Indicates that the current transaction can continue.

2.2.5.12.2 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 PCI initiator stop the current transaction.

 Negated—Indicates that the current transaction can continue.

2.2.5.13 Target Ready ($\overline{\text{TRDY}}$)

The target ready ($\overline{\text{TRDY}}$) signal is both an input and output signal on the MPC106.

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

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

State Meaning Asserted—Indicates that the MPC106, acting as a PCI target, can complete the current data phase of a PCI transaction. During a read, the MPC106 asserts $\overline{\text{TRDY}}$ to indicate that valid data is present on AD[31–0]. During a write, the MPC106 asserts $\overline{\text{TRDY}}$ to indicate that it is prepared to accept data.

Negated—Indicates that the PCI initiator needs to wait before the MPC106, acting as a PCI target, can complete the current data phase. During a read, the MPC106 negates $\overline{\text{TRDY}}$ to insert a wait cycle when it cannot provide valid data to the initiator. During a write, the MPC106 negates $\overline{\text{TRDY}}$ to insert a wait cycle when it cannot accept data from the initiator.

2.2.5.13.2 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.

2.2.5.14 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 MPC106 implements four PCI sideband signals— $\overline{\text{FLSHREQ}}$, $\overline{\text{ISA_MASTER}}$, $\overline{\text{MEMACK}}$, and $\overline{\text{PIRQ}}$.

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

The flush request ($\overline{\text{FLSHREQ}}$) signal is an input signal on the MPC106. Following is the state meaning for the $\overline{\text{FLSHREQ}}$ input signal.

State Meaning Asserted—Indicates that a device needs to have the MPC106 flush all of its current operations. $\overline{\text{FLSHREQ}}$ should be asserted when $\overline{\text{MEMACK}}$ is negated and before $\overline{\text{FRAME}}$ is asserted.

Negated—Indicates normal operation for the MPC106. $\overline{\text{FLSHREQ}}$ should be deasserted after $\overline{\text{FRAME}}$ is deasserted.

2.2.5.14.2 ISA Master ($\overline{\text{ISA_MASTER}}$)—Input

The ISA master ($\overline{\text{ISA_MASTER}}$) signal is an input signal on the MPC106. This signal is only valid for address map A; it has no meaning for address map B or for the emulation mode address map. 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. The $\overline{\text{ISA_MASTER}}$ signal is an implied address bit 31 for

ISA devices that cannot drive a full 32-bit address. Accordingly, when the MPC106 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 MPC106 (or system memory behind it).

Negated—Indicates that no ISA master requires system memory.

2.2.5.14.3 Memory Acknowledge ($\overline{\text{MEMACK}}$)—Output

The memory acknowledge ($\overline{\text{MEMACK}}$) signal is an output signal on the MPC106. Following is the state meaning for the $\overline{\text{MEMACK}}$ output signal.

State Meaning Asserted—Indicates that the MPC106 has flushed all of its current operations and has blocked all 60x transfers except snoop copyback operations. The MPC106 asserts $\overline{\text{MEMACK}}$ in response to the assertion of $\overline{\text{FLSHREQ}}$, after the flush is complete.

Negated—Indicates the MPC106 may still have operations in its queues. The MPC106 negates $\overline{\text{MEMACK}}$ two cycles after $\overline{\text{FLSHREQ}}$ is deasserted.

2.2.5.14.4 Modified Memory Interrupt Request ($\overline{\text{PIRQ}}$)—Output

The modified memory interrupt request ($\overline{\text{PIRQ}}$) signal is an output signal on the MPC106. The polarity of the $\overline{\text{PIRQ}}$ signal is programmable by using the $\text{ESCR1}[\text{PIRQ_ACTIVE_HIGH}]$ parameter; see Section 3.2.11, “Emulation Support Configuration Registers,” for more information. Note that the $\overline{\text{PIRQ}}$ signal is only meaningful in the emulation mode address map. See Section 7.8, “Emulation Support,” for more information. Following is the state meaning for the $\overline{\text{PIRQ}}$ output signal.

State Meaning Asserted—Indicates that software has not recorded a PCI to system memory write operation.

Negated—Indicates either that software has recorded all PCI to system memory writes or that no writes have occurred.

2.2.6 Interrupt, Clock, and Power Management Signals

The MPC106 coordinates interrupt, clocking, and power management signals across the memory bus, the PCI bus, and the 60x processor bus. This section provides a brief description of these signals.

2.2.7 External ECM Error Detect ($\overline{\text{BERR}}$)—Input

The external ECM error detect ($\overline{\text{BERR}}$) signal is an input on the MPC106 Rev. 4.0 and later. Following are the state meaning and timing comments for the $\overline{\text{BERR}}$ signal.

State Meaning Asserted—Indicates that the external ECM has detected a parity or ECC error.
Negated—Indicates that the external ECM has not detected any errors.

Timing Comments Assertion/Negation—Valid in the same clock cycle as data on the destination bus (that is, the memory bus for writes, and the 60x bus for reads).

Note that the $\overline{\text{BERR}}$ signal functions as $\overline{\text{BERR}}$ only if external ECM mode is enabled. See the bit descriptions for MCCR4[EXT_ECM_EN] in Table 3-37. for more information.

2.2.7.1 Test Clock (CKO)—Output

The test clock (CKO) signal is an output on the MPC106. This signal provides a means to test or monitor the internal PLL output or the bus clock frequency. The CKO clock should be used for testing purposes only. It is not intended as a reference clock signal.

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

The hard reset ($\overline{\text{HRST}}$) signal is an input on the MPC106. Following are the state meaning and timing comments for the $\overline{\text{HRST}}$ input signal.

State Meaning Asserted—Initiates a complete hard reset of the MPC106. 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.

2.2.7.3 Nonmaskable Interrupt (NMI)—Input

The nonmaskable interrupt (NMI) signal is an input on the MPC106. Following are the state meaning and timing comments for the NMI input signal.

State Meaning Asserted—Indicates that the MPC106 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.

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

The quiesce acknowledge ($\overline{\text{QACK}}$) signal is an output on the MPC106. See Section A.1.1, “MPC106 Power Mode Transition,” for more information about the power management signals. Following are the state meaning and timing comments for the $\overline{\text{QACK}}$ output signal.

State Meaning Asserted—Indicates that the MPC106 is in a low-power state. All bus activity that requires snooping has terminated, and the 60x processor may enter a low-power state.

Negated—Indicates that the 60x processor should not enter a low-power state. The MPC106 is in full-on state with normal bus activity.

Timing Comments Assertion—The MPC106 can assert $\overline{\text{QACK}}$ at any time, synchronous to the 60x bus clock when $\overline{\text{QREQ}}$ is asserted.

Negation—The MPC106 can negate $\overline{\text{QACK}}$ any time, synchronous to the 60x bus clock.

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

The quiesce request ($\overline{\text{QREQ}}$) signal is an input on the MPC106. See Section A.1.1, “MPC106 Power Mode Transition,” for more information about the power management signals. 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 MPC106 synchronizes $\overline{\text{QREQ}}$ internally.

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

The suspend ($\overline{\text{SUSPEND}}$) signal is an input on the MPC106. 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 MPC106 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.

2.2.7.7 System Clock (SYSCLK)—Input

The system clock (SYSCLK) signal is an input on the MPC106. The SYSCLK signal sets the frequency of operation for the PCI bus, and provides a reference clock for the phase-locked loops in the MPC106. SYSCLK is used to synchronize bus operations. See Section 2.3, “Clocking,” for more information.

2.2.8 IEEE 1149.1 Interface Signals

To facilitate system testing, the MPC106 provides a JTAG test access port (TAP) that complies with the IEEE 1149.1 boundary-scan specification. This section describes the JTAG test access port signals.

2.2.8.1 JTAG Test Clock (TCK)—Input

The JTAG test clock (TCK) signal is an input on the MPC106. 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 are clocked in on the rising edge of TCK. Changes to the test access port 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.

2.2.8.2 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.

2.2.8.3 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.

2.2.8.4 JTAG Test Mode Select (TMS)—Input

The test mode select (TMS) signal is an input on the MPC106. 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.

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

The test reset ($\overline{\text{TRST}}$) signal is an input on the MPC106. 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.

2.2.9 Configuration Signals

The MPC106 has several signals that are sampled during power-on reset to determine the memory data buffer, address map, ROM interface, and clock configuration. This section describes the signals sampled during power-on reset, and how they are configured. Pull-up or pull-down resistors should be used to avoid interference with normal signal operations.

2.2.9.1 501-Mode ($\overline{\text{BCTL0}}$)—Input

The 501-mode configuration signal uses $\overline{\text{BCTL0}}$ as a configuration input. See Section 6.2, “Memory Interface Signal Buffering,” for more information. Following is the state meaning for the $\overline{\text{BCTL0}}$ configuration signal.

State Meaning High—Configures the MPC106 Rev. 4.0 to support backward compatibility with MPC106 Rev. 3.0 memory interface buffer configurations.

Low—Configures the MPC106 Rev. 4.0 to support 16501-type universal bus transceivers between the 60x/memory data bus.

Note that this input has an internal pull-up resistor to ensure backward compatibility as the default configuration.

2.2.9.2 Address Map ($\overline{\text{DBG0}}$)—Input

The address map configuration signal uses $\overline{\text{DBG0}}$ as a configuration input. Following is the state meaning for the $\overline{\text{DBG0}}$ configuration signal.

State Meaning High—Configures the MPC106 for address map A.

Low—Configures the MPC106 for address map B.

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

2.2.9.3 ROM Bank 0 Data Path Width ($\overline{\text{FOE}}$)—Input

The ROM bank 0 data path width configuration signal uses $\overline{\text{FOE}}$ as a configuration input. Following is the state meaning for the $\overline{\text{FOE}}$ configuration signal.

State Meaning High—Configures ROM bank 0 for an 8-bit data path.

Low—Configures ROM bank 0 for a 64-bit data path.

2.2.9.4 60x Clock Phase ($\overline{\text{LBCLAIM}}$)—Input

The 60x clock phase configuration signal uses $\overline{\text{LBCLAIM}}$ as a configuration input. Following is the state meaning for the $\overline{\text{LBCLAIM}}$ configuration signal.

State Meaning High/Low—60x bus clock input to resolve clock phasing with the PCI bus clock (SYSCLK) in 3:2 and 5:2 clock modes. The 60x clock’s phase is only sampled during power-on reset, hard reset, and when coming out of the sleep and suspend power saving modes.

Note that this signal is only necessary when the PLL is configured for one of the half-bus clock modes (5:2 and 3:2). When PLL[0:3] corresponds to a half-bus clock mode, local bus slave functionality is disabled. Thus, local bus slaves cannot be used in conjunction with half-bus clock modes.

2.2.9.5 Clock Mode (PLL[0–3])—Input

The clock mode (PLL[0–3]) configuration signals are dedicated inputs on the MPC106. Following is the state meaning for the PLL[0–3] 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.

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

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

- State Meaning**
- High—Indicates that ROM is located on the 60x processor/memory bus.
 - Low—Indicates that ROM is located on the PCI bus. Note that the parameter, PICR2[CF_FF0_LOCAL], may be used to remap the lower half of ROM space (0xFF00_000–0xFF7F_FFFF) back to the 60x processor/memory bus. See Section 6.6, “ROM/Flash Interface Operation,” for more information.

2.3 Clocking

The MPC106 requires a single system clock input, SYSCLK. The SYSCLK frequency dictates the frequency of operation for the PCI bus. An internal PLL on the MPC106 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 MPC106 generates a core frequency either x1, x1.5, x2, x2.5, or x3 of the SYSCLK frequency (see Figure 2-2) depending on the clock mode configuration signals (PLL[0–3]). The core frequency is phase-locked to the rising edge of SYSCLK. Note that SYSCLK is not required to have a 50% duty cycle.

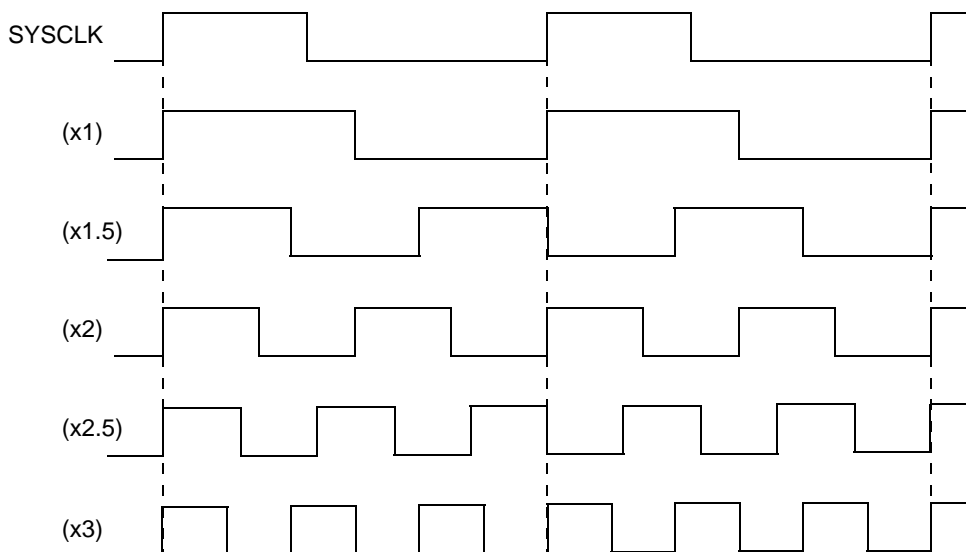


Figure 2-2. SYSCLK Input with Internal Multiples

The PLL is configured by the PLL[0–3] signals. For a given SYSCLK (PCI bus) frequency, the clock mode configuration signals (PLL[0–3]) set the core frequency (and the frequency of the VCO controlling the PLL lock).

In the 3:2 and 5:2 clock modes, the MPC106 needs to sample the 60x bus clock (on the $\overline{\text{LBCLAIM}}$ configuration input) to resolve clock phasing with the PCI bus clock (SYSCLK). Because the $\overline{\text{LBCLAIM}}$ signal is used for clock phasing in half-bus clock modes, the local bus slave functionality is disabled. Thus, local bus slaves cannot be used in conjunction with half-bus clock modes. Note that systems that use 3:2 or 5:2 clock modes should not use modified memory tracking. See Section 7.8.3, “Modified Memory Status Register,” for more information.

The supported core and VCO frequencies and the corresponding PLL[0–3] settings are provided in the *MPC106 PCI Bridge/Memory Controller Hardware Specifications*.



Chapter 3

Device Programming

The programmable aspects of the MPC106 are primarily for use by initialization and error handling software. This chapter describes the selectable address maps and the configuration registers on the MPC106.

3.1 Address Maps

The MPC106 supports three address mapping configurations designated address map A, address map B, and emulation mode map. Address map A conforms to the PowerPC reference platform specification. Address map B conforms to the common hardware reference platform (CHRP) of the processor that implement the PowerPC architecture. The emulation mode map is provided to support software emulation of x86 hardware.

The configuration signal $\overline{\text{DBG0}}$, sampled during power-on reset, selects between address map A and address map B. Map A is selected by using a pull-up resistor ($\overline{\text{DBG0}} = 1$); map B is selected by using a pull-down resistor ($\overline{\text{DBG0}} = 0$). After reset, the address map can be changed by programming PICR1[ADDRESS_MAP].

Emulation mode map can only be selected by software after reset by programming ESCR1[EMULATION_MODE_EN].

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—system memory, PCI I/O, PCI memory, and system ROM space. Table 3-1, Table 3-2, and Table 3-3 show separate views of address map A for the 60x processor, a PCI memory device, and a PCI I/O device, respectively. When configured for map A, the MPC106 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 discontinuous by PICR1[XIO_MODE]. The discontinuous map reserves a 4-Kbyte page for each 32-byte block addressed on the PCI bus allowing each 32-byte block from 0 to 64KB – 1 in PCI I/O space to have distinct page protection attributes. This can help when accessing PC-compatible I/O devices in the ISA/PCI I/O space address 0 to 64KB – 1. The contiguous map as seen from the processor

is shown in Figure 3-1. The discontinuous map as seen from the processor is shown in Figure 3-2.

Table 3-1. Address Map A—Processor View

60x Processor Address Range				PCI Address Range	Definition
Hex	Decimal				
00000000	3FFFFFFF	0	1G – 1	No PCI cycle	System memory space
40000000	7FFFFFFF	1G	2G – 1	No PCI cycle	Reserved ¹
80000000	807FFFFF	2G	2G + 8M – 1	00000000–007FFFFF	PCI/ISA I/O space (64 Kbytes or 8 Mbytes) ^{2,3}
80800000	80FFFFFF	2G + 8M	2G + 16M – 1	00800000–00FFFFFF	PCI configuration direct access ⁴
81000000	BF7FFFFF	2G + 16M	3G – 8M – 1	01000000–3F7FFFFF	PCI I/O space
BF800000	BFFFFFFE	3G – 8M	3G – 16 – 1	3F800000–3FFFFFFE	Reserved
BFFFFFF0	BFFFFFFF	3G – 16	3G – 1	3FFFFFF0–3FFFFFFF	PCI/ISA interrupt acknowledge ⁵
C0000000	FEFFFFFF	3G	4G – 16M – 1	00000000–3EFFFFFF	PCI memory space
FF000000	FFFFFFFF	4G – 16M	4G – 1	No PCI cycle ⁶	ROM space ⁶

Table 3-2. Address Map A—PCI Memory Master View

PCI Memory Transactions Address Range				60x Address Range	Definition
Hex	Decimal				
00000000	00FFFFFF	0	16M – 1	No system memory cycle ⁷	PCI/ISA memory space
01000000	7EFFFFFF	16M	2G – 16M – 1	No system memory cycle ⁷	PCI memory space
7F000000	7FFFFFFF	2G – 16M	2G – 1	No system memory cycle ⁷	Reserved
80000000	BFFFFFFF	2G	3G – 1	00000000–3FFFFFFF	System memory space
C0000000	FFFFFFFF	3G	4G – 1	Memory select error	Reserved

Table 3-3. Address Map A—PCI I/O Master View

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

Notes:

1. The MPC106 generates a memory select error (if enabled) for transactions in the address range 40000000–7FFFFFFF. If memory select errors are disabled, the MPC106 returns all 1s for read operations and no update for write operations.
2. 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.
3. Processor addresses are translated to PCI addresses as follows:
 In contiguous mode:
 PCI address (AD[31–0]) = 0b0 || A[1–31]. PCI configuration accesses use processor addresses 8000CF8 and 8000CFC–8000CFF.
 In discontinuous mode:
 PCI address (AD[31–0]) = 0x0000 || A[9–19] || A[27–31]. PCI configuration accesses use processor addresses 80067018 and 8006701C–8006701F.
4. IDSEL for direct-access method: 11=0x808008xx, 12=0x808010xx, ..., 18=0x808400xx.
5. Reads to this address generate PCI interrupt-acknowledge cycles; writes to this address generate \overline{TEA} (if enabled).
6. If the ROM is located on the PCI bus, these addresses are not reserved and PCI cycles will be generated.
7. If the $\overline{ISA_MASTER}$ signal is asserted, the PCI memory cycle is forwarded to system memory and the 60x bus address becomes 0b00 || AD[29–0].

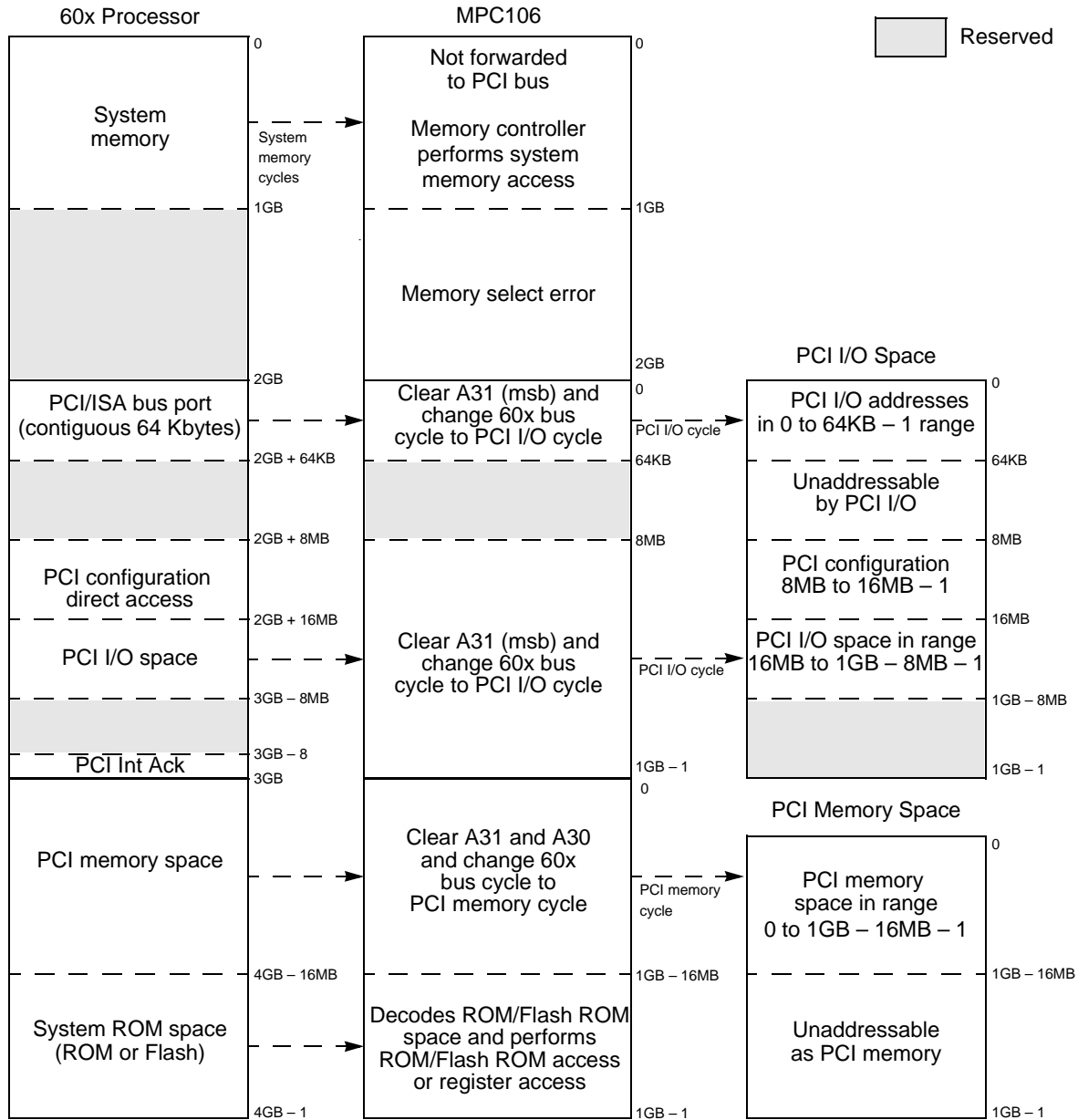


Figure 3-1. Address Map A (Contiguous Map)

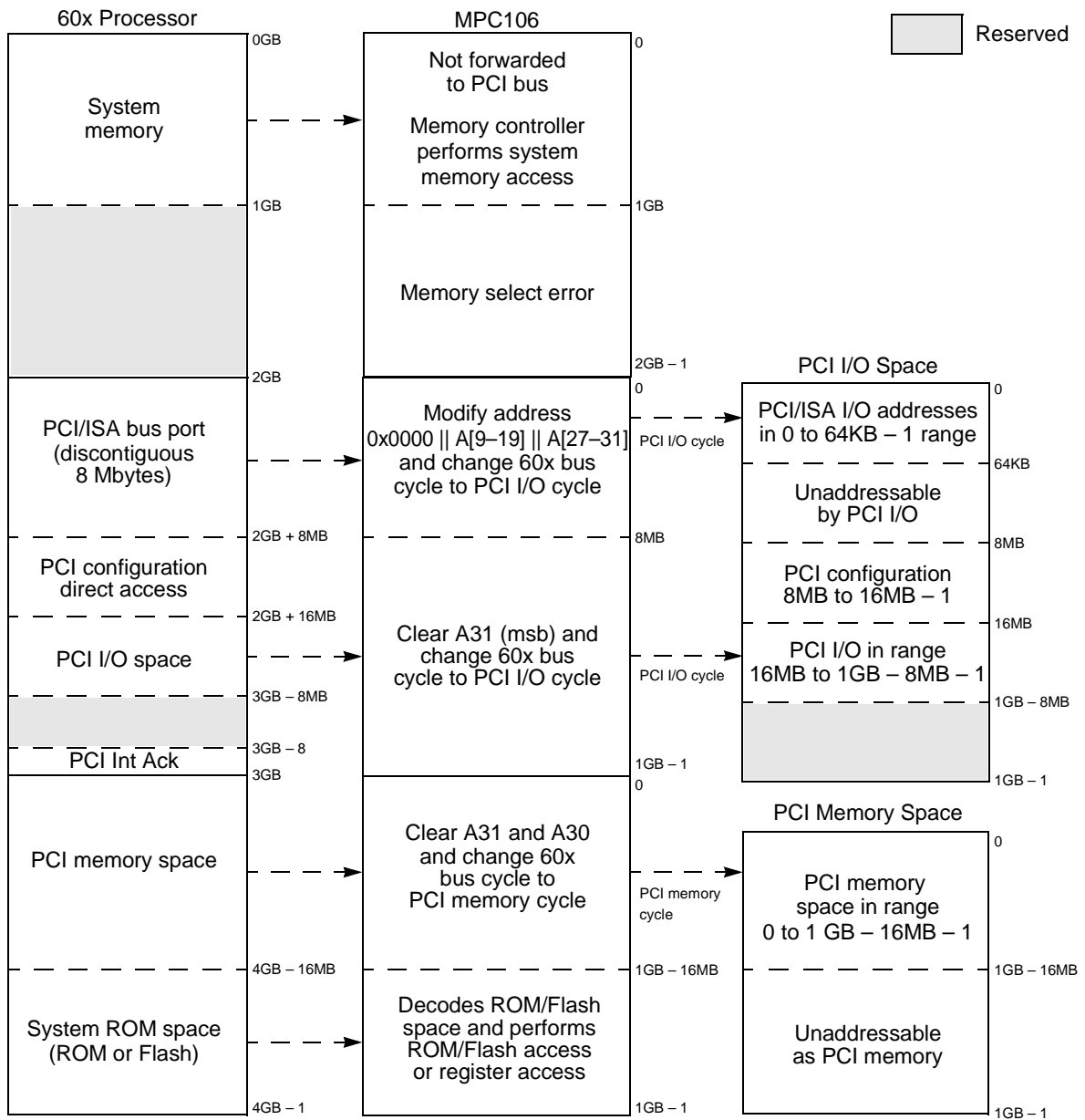


Figure 3-2. Address Map A (Discontiguous Map)

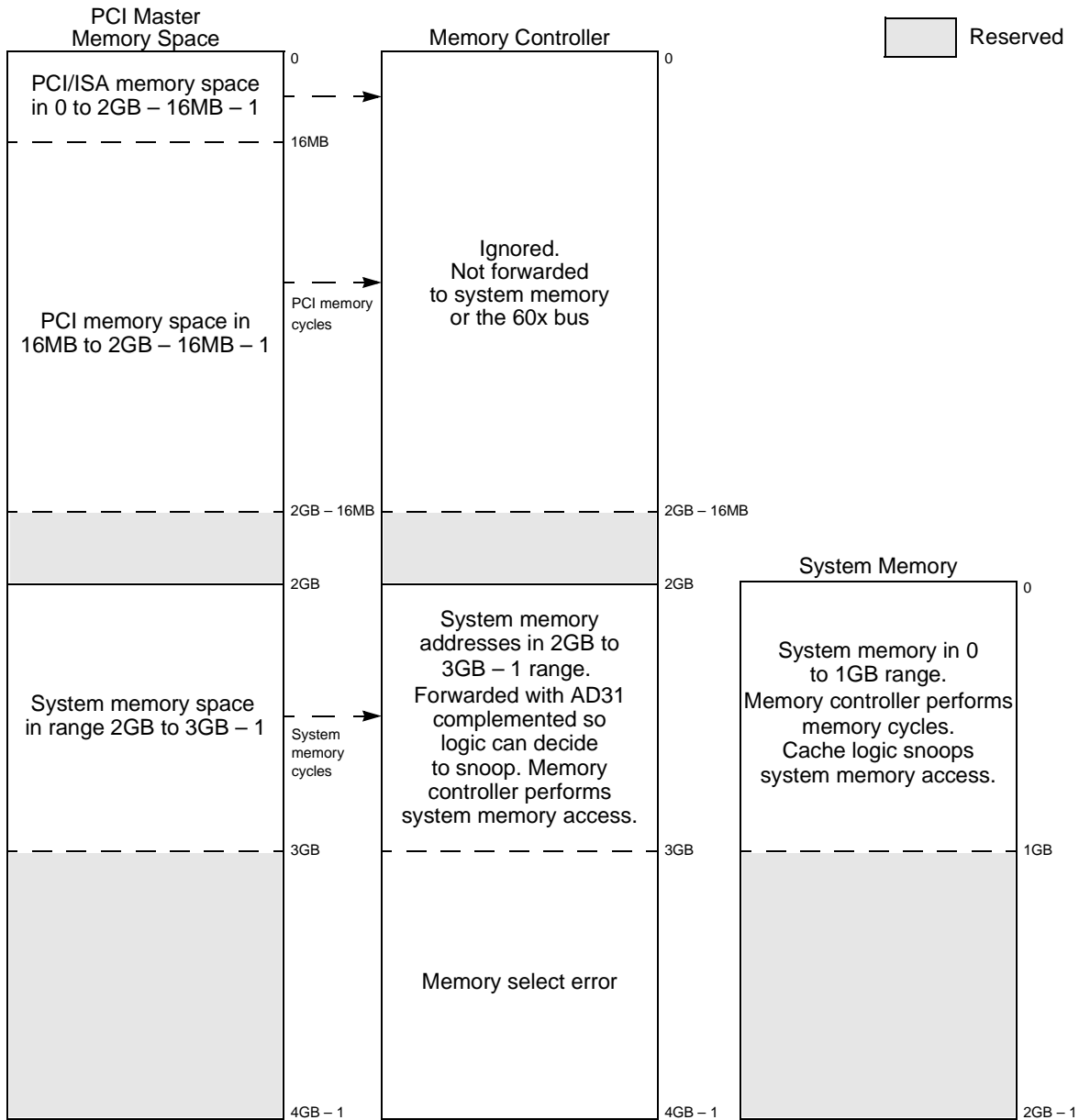


Figure 3-3. PCI Memory Map (Address Map A)

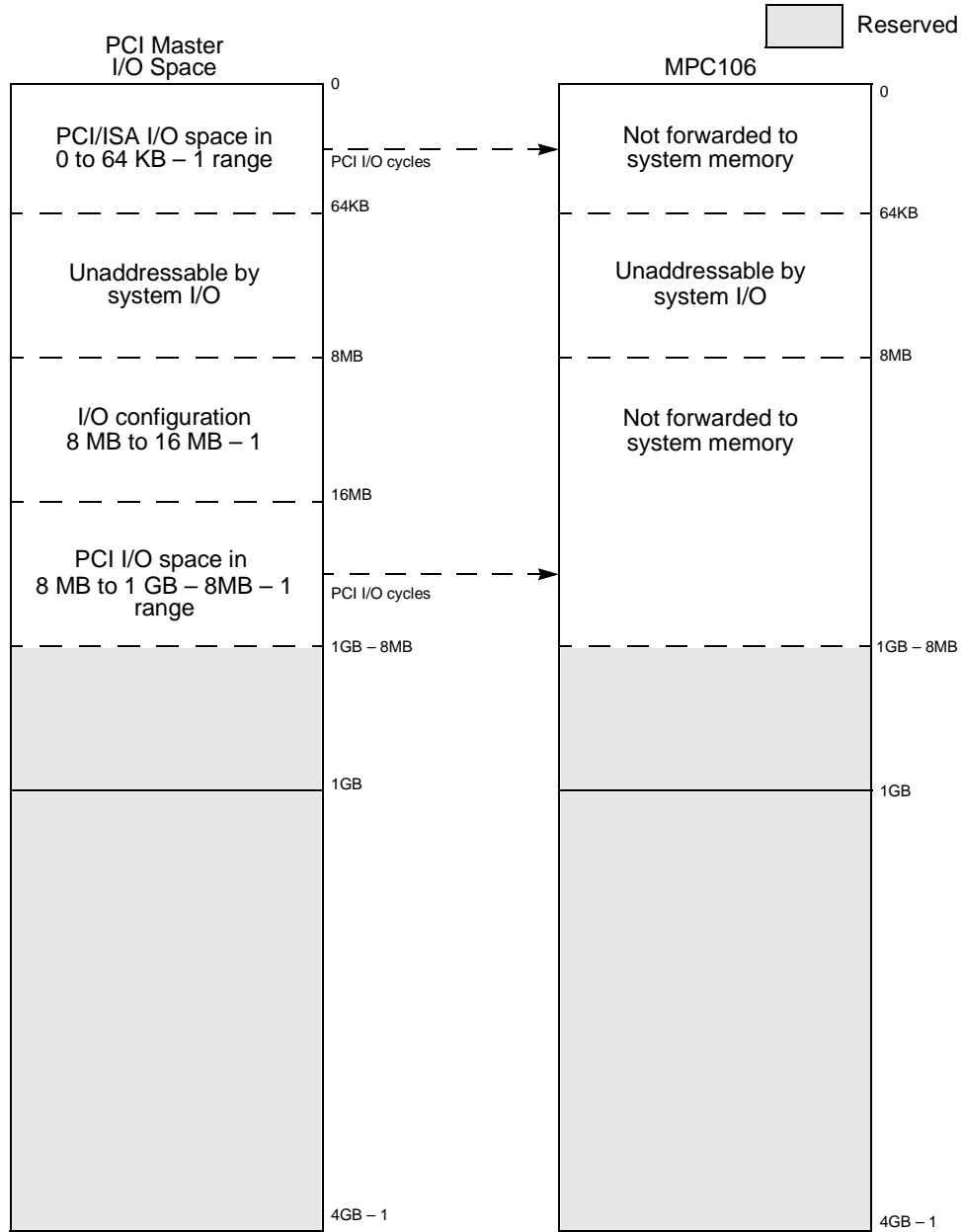


Figure 3-4. PCI I/O Map (Address Map A)

3.1.2 Address Map B

Address map B complies with the common hardware reference platform (CHRP) of the microprocessor that implement PowerPC architecture. As with address map A, the address space of map B is divided into four areas—system memory, PCI memory, PCI I/O, and system ROM space. When configured for map B, the MPC106 translates addresses across the 60x and PCI buses as shown in Figure 3-5. Table 3-4., Table 3-5, and Table 3-6 show separate views of address map B for the 60x processor, a PCI memory device, and a PCI I/O device, respectively.

Table 3-4. Address Map B—Processor View

60x Processor Address Range				PCI Address Range	Definition
Hex		Decimal			
00000000	0009FFFF	0	640K – 1	No PCI cycle	System memory space
000A0000	000BFFFF	640K	768K – 1	000A0000–000BFFFF	Compatibility hole ¹
000C0000	3FFFFFFF	768K	1G – 1	No PCI cycle	System memory space
40000000	7FFFFFFF	1G	2G – 1	No PCI cycle	Reserved ²
80000000	FCFFFFFF	2G	4G – 48M – 1	80000000–FCFFFFFF	PCI memory space
FD000000	FDFFFFFF	4G – 48M	4G – 32M – 1	00000000–00FFFFFF	PCI/ISA memory space ³
FE000000	FE7FFFFF	4G – 32M	4G – 24M – 1	00000000–0000FFFF	PCI/ISA I/O space (64 Kbytes or 8 Mbytes) ⁴
FE800000	FEBFFFFF	4G – 24M	4G – 20M – 1	00800000–00BFFFFF	PCI I/O space ⁵
FEC00000	FEDFFFFF	4G – 20M	4G – 18M – 1	CONFIG_ADDR	PCI configuration address register ⁶
FEE00000	FEEFFFFF	4G – 18M	4G – 17M – 1	CONFIG_DATA	PCI configuration data register ⁷
FEF00000	FEFFFFFF	4G – 17M	4G – 16M – 1	FEF00000–FEFFFFFF	PCI interrupt acknowledge ⁸
FF000000	FF7FFFFF	4G – 16M	4G – 8M – 1	FF000000–FF7FFFFF	64-bit system ROM space ⁹
FF800000	FFFFFFFF	4G – 8M	4G – 1	FF800000–FFFFFFFF	8- or 64-bit system ROM space ⁹

Table 3-5. Address Map B—PCI Memory Master View

PCI Memory Transaction Address Range				60x Address Range	Definition
Hex		Decimal			
00000000	0009FFFF	0	640K – 1	00000000–0009FFFF	System memory space
000A0000	000FFFFFFF	640K	1M – 1	000A0000–000FFFFFFF	Compatibility hole ¹
00100000	3FFFFFFF	1M	1G – 1	00100000–3FFFFFFF	System memory space
40000000	7FFFFFFF	1G	2G – 1	40000000–7FFFFFFF	Reserved ²
80000000	FCFFFFFF	2G	4G – 48M – 1	No system memory cycle	PCI memory space

Table 3-5. Address Map B—PCI Memory Master View (continued)

PCI Memory Transaction Address Range				60x Address Range	Definition
Hex		Decimal			
FD000000	FDFFFFFF	4G – 48M	4G – 32M – 1	00000000–00FFFFFF	System memory space ¹⁰
FE000000	FEFFFFFF	4G – 32M	4G – 16M – 1	No system memory cycle	Reserved
FF000000	FF7FFFFF	4G – 16M	4G – 8M – 1	FF000000–FF7FFFFF	64-bit system ROM space ⁹
FF800000	FFFFFFFF	4G – 8M	4G – 1	FF800000–FFFFFFFF	8- or 64-bit system ROM space ⁹

Table 3-6. Address Map B—PCI I/O Master View

PCI I/O Transaction Address Range				60x Address Range	Definition
Hex		Decimal			
00000000	0000FFFF	0	64K – 1	No system memory cycle	PCI/ISA I/O space
00010000	007FFFFF	64K	8M – 1	No system memory cycle	Reserved
00800000	00BFFFFF	8M	12M – 1	No system memory cycle	PCI I/O space
00C00000	FFFFFFF	12M	4G – 1	No system memory cycle	Reserved

Notes:

1. Transactions in the compatibility hole address range are controlled by the PCI_COMPATIBILITY_HOLE and PROC_COMPATIBILITY_HOLE parameters in emulation support configuration register 1 (ESCR1). The MPC106 directs the transaction to system memory or PCI memory, depending on these parameters. See Section 3.2.11, “Emulation Support Configuration Registers,” for more information.
2. The MPC106 generates a memory select error (if enabled) for transactions in the address range 40000000–7FFFFFFF. If memory select errors are disabled, the MPC106 returns all 1s for read operations and no update for write operations.
3. The MPC106 forwards 60x transactions in this range to the PCI memory space with the 8 most-significant bits cleared (that is, AD[31–0] = 0x00 || A[8–31]).
4. Processor addresses are translated to PCI addresses as follows:
 In contiguous mode:
 PCI address (AD[31–0]) = 0x00 || A[8–31]. Note that in contiguous mode, the processor range FE010000–FE7FFFFF is reserved.
 In discontinuous mode:
 PCI address (AD[31–0]) = 0x0000 || A[9–19] || A[27–31].
5. The MPC106 forwards 60x transactions in this range to the PCI I/O space with the 8 most-significant bits cleared (that is, AD[31–0] = 0x00 || A[8–31]).
6. Each word in this address range is aliased to the PCI CONFIG_ADDR register. See Section 7.4.5.2, “Accessing the PCI Configuration Space,” for more information.
7. Each word in this address range is aliased to the PCI CONFIG_DATA register. See Section 7.4.5.2, “Accessing the PCI Configuration Space,” for more information.
8. Reads from this address generate PCI interrupt-acknowledge cycles; writes to this address generate \overline{TEA} (if enabled).
9. The ROM/Flash space may be located on the 60x/memory bus, on the PCI bus, or on both. See Section 6.6, “ROM/Flash Interface Operation,” for more information.
10. If ESCR1[FD_ALIAS_EN] = 1, the MPC106 forwards PCI memory transactions in this range to system PCI memory with the 8 most-significant bits cleared (that is, 0x00 || AD[23–0]).

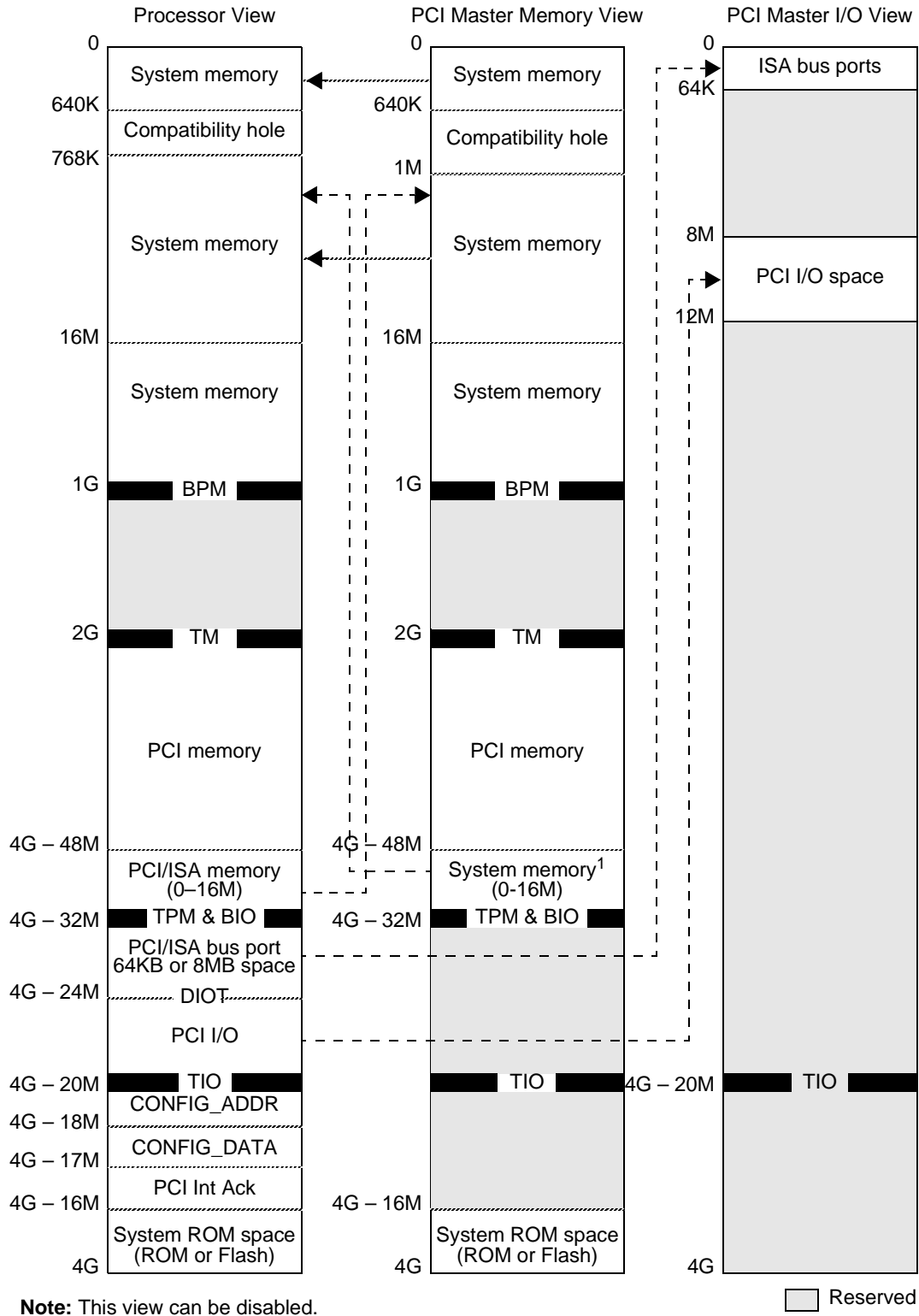


Figure 3-5. Address Map B

3.1.3 Emulation Mode Address Map

The emulation mode address map is actually a subset of address map B that has a programmable boundary, as seen by the PCI bus, between the system memory space and the PCI memory space. The emulation mode address map is fully compliant with the PC emulation option described in the CHRP specification.

When configured for the emulation mode address map, the MPC106 translates addresses across the 60x and PCI buses as shown in Figure 3-6. Table 3-7, Table 3-8, and Table 3-9 show separate views of the emulation mode address map for the 60x processor, a PCI memory device, and a PCI I/O device, respectively. See Section 7.8, “Emulation Support,” for more information about emulation mode.

Table 3-7. Emulation Mode Address Map—Processor View

60x Processor Address Range				PCI Address Range	Definition
Hex		Decimal			
00000000	0009FFFF	0	640K – 1	No PCI cycle	System memory space
000A0000	000BFFFF	640K	768K – 1	000A0000–000BFFFF	Compatibility hole ¹
000C0000	3FFFFFFF	768K	1G – 1	No PCI cycle	System memory space
40000000	7FFFFFFF	1G	2G – 1	No PCI cycle	Reserved ²
80000000	FCFFFFFF	2G	4G – 48M – 1	80000000–FCFFFFFF	PCI memory space
FD000000	FDFFFFFFF	4G – 48M	4G – 32M – 1	00000000–00FFFFFF	PCI/ISA memory space ³
FE000000	FE7FFFFFF	4G – 32M	4G – 24M – 1	00000000–0000FFFF	PCI/ISA I/O space (64 Kbytes or 8 Mbytes) ⁴
FE800000	FEBFFFFFF	4G – 24M	4G – 20M – 1	00800000–00BFFFFFF	PCI I/O space ⁵
FEC00000	FEDFFFFFF	4G – 20M	4G – 18M – 1	CONFIG_ADDR	PCI configuration address register ⁶
FEE00000	FEEFFFFFF	4G – 18M	4G – 17M – 1	CONFIG_DATA	PCI configuration data register ⁷
FEF00000	FEFFFFFF	4G – 17M	4G – 16M – 1	FEF00000–FEFFFFFF	PCI interrupt acknowledge ⁸
FF000000	FF7FFFFFF	4G – 16M	4G – 8M – 1	FF000000–FF7FFFFFF	64-bit system ROM space ⁹
FF800000	FFFFFFFF	4G – 8M	4G – 1	FF800000–FFFFFFFF	8- or 64-bit system ROM space ⁹

Table 3-8. Emulation Mode Address Map—PCI Memory Master View

PCI Memory Transaction Address Range				60x Address Range	Definition
Hex	Decimal				
00000000	0009FFFF	0	640K – 1	00000000–0009FFFF	System memory space
000A0000	000FFFFFFF	640K	1M – 1	000A0000–000FFFFFFF	Compatibility hole ¹
00100000	Programmable ¹⁰	1M	Programmable ¹⁰	00100000–Programmable ¹⁰	System memory space ¹⁰
Programmable ¹⁰	FFFFFFFF	Programmable ¹⁰	4G – 1	No system memory cycle	PCI memory space ¹⁰

Table 3-9. Emulation Mode Address Map—PCI I/O Master View

PCI I/O Transaction Address Range				60x Address Range	Definition
Hex	Decimal				
00000000	0000FFFF	0	64K – 1	No system memory cycle	PCI/ISA I/O space
00010000	007FFFFFFF	64K	8M – 1	No system memory cycle	Reserved
00800000	00BFFFFFFF	8M	12M – 1	No system memory cycle	PCI I/O space
00C00000	FFFFFFFF	12M	4G – 1	No system memory cycle	Reserved

Notes:

- Transactions in the compatibility hole address range are controlled by the PCI_COMPATIBILITY_HOLE and PROC_COMPATIBILITY_HOLE parameters in emulation support configuration register 1 (ESCR1). The MPC106 directs the transaction to system memory or PCI memory, depending on these parameters. See Section 3.2.11, “Emulation Support Configuration Registers,” for more information.
- The MPC106 generates a memory select error (if enabled) for transactions in the address range 40000000–7FFFFFFF. If memory select errors are disabled, the MPC106 returns all 1s for read operations and no update for write operations.
- The MPC106 forwards 60x transactions in this range to the PCI memory space with the 8 most-significant bits cleared (that is, AD[31–0] = 0x00 || A[8–31]).
- Processor addresses are translated to PCI addresses as follows:
 In contiguous mode:
 PCI address (AD[31–0]) = 0x00 || A[8–31]. Note that in contiguous mode, the processor range FE010000–FE7FFFFFFF is reserved.
 In discontinuous mode:
 PCI address (AD[31–0]) = 0x0000 || A[9–19] || A[27–31].
- The MPC106 forwards 60x transactions in this range to the PCI I/O space with the 8 most-significant bits cleared (that is, AD[31–0] = 0x00 || A[8–31]).
- Each word in this address range is aliased to the PCI CONFIG_ADDR register. See Section 7.4.5.2, “Accessing the PCI Configuration Space,” for more information.
- Each word in this address range is aliased to the PCI CONFIG_DATA register. See Section 7.4.5.2, “Accessing the PCI Configuration Space,” for more information.
- Reads from this address generate PCI interrupt-acknowledge cycles; writes to this address generate \overline{TEA} (if enabled).
- The ROM/Flash space may be located on the 60x/memory bus, on the PCI bus, or on both. See Section 6.6, “ROM/Flash Interface Operation,” for more information.
- The emulation mode allows system software to configure the PCI memory address space by programming the parameter ESCR1[TOP_OF_MEM]. The MPC106 claims PCI memory transactions addressed from 00100000 to 0x0 || TOP_OF_MEM || FFFFFF and forwards the transactions to system memory. The MPC106 does not claim PCI memory transactions addressed from 0x0 || TOP_OF_MEM + 1 || 00000 to FFFFFFFF; other PCI memory targets claim transactions in this range. 60x processor-to-PCI transactions are unaffected by the value in TOP_OF_MEM.

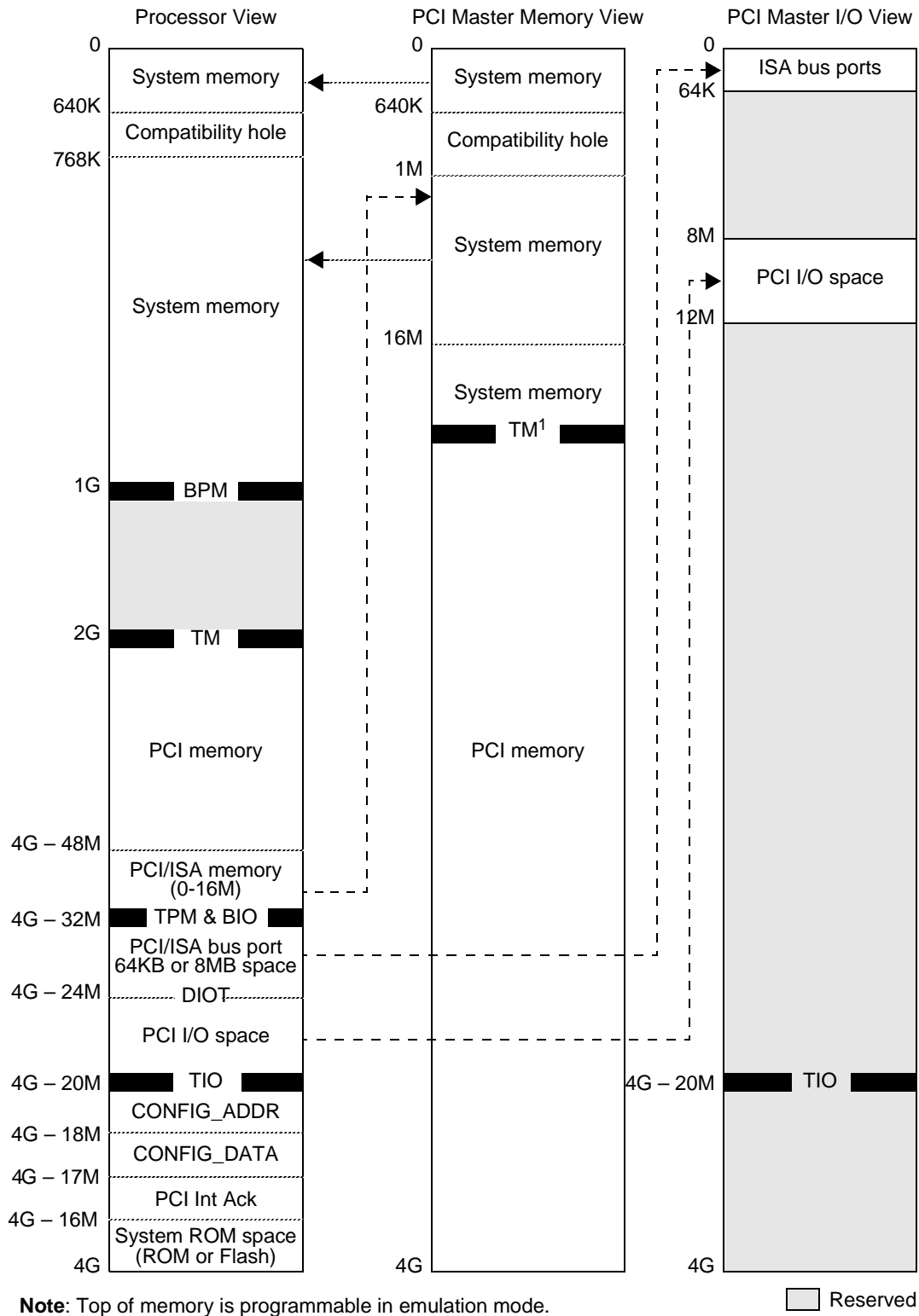


Figure 3-6. Emulation Mode Address Map

3.2 Configuration Registers

This section describes the programmable configuration registers of the MPC106. These registers are generally set up by initialization software following a power-on reset or hard reset, or by error handling routines. All the internal registers of the MPC106 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 MPC106 configuration registers are accessed by an indirect method similar to accessing PCI device configuration registers. The 32-bit register address ($0x8000_00nn$, where nn is the address offset of the desired configuration register—see Table 3-10 and Figure 3-7) is written to CONFIG_ADDR at $0x8000_0CF8$ ($0x8006_7018$ in discontinuous mode). Then, the data is accessed at CONFIG_DAT at addresses $0x8000_0CFC$ – $0x8000_0CFF$ ($0x8006_701C$ – $0x8006_701F$ in discontinuous mode).

When using address map A, certain configuration bits for the MPC106 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 3.2.12, “External Configuration Registers,” for more information.

When using address map B or emulation mode address map, the MPC106 uses a similar indirect method to access the internal configuration registers—the exception is that CONFIG_ADDR and CONFIG_DATA are found at different addresses. The 32-bit register address ($0x8000_00nn$, where nn is the address offset of the desired configuration register—see Table 3-10 and Figure 3-7) is written to CONFIG_ADDR at any word-aligned address in the range $0xFEC0_0000$ – $0xFEDF_FFFF$. Every word within this range is aliased to the same location. Then, the data is accessed at CONFIG_DAT at any address in the range $0xFEE0_0000$ – $0xFEEF_FFFF$. Every word within this range is aliased to the same location.

3.2.1.1 Configuration Register Access in Little-Endian Mode

In little-endian mode (both processor and the MPC106), the program should access the configuration registers using the methods described in Section 3.2.1, “Configuration Register Access.” The data appears in the 60x processor register in descending significance byte order (MSB to LSB) at the time it is stored to the MPC106. 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 MPC106.

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
 Register at 0xA8 contains 0xAABB_CCDD (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_00CC

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

```
Initial values: r0 contains 0x8000_00A8
                r1 contains 0xFEC0_0000
                r2 contains 0xFEE0_0000
                r3 contains 0xAABB_CCDD
                Register at 0xA8 contains 0xFFFF_FFFF (AB to A8)
```

```
Code sequence: stw    r0,0(r1)
                stw    r3,0(r2)
```

```
Results:       Address 0xFEC0_0000 contains 0x8000_00A8 (MSB to LSB)
                Register at 0xA8 contains 0xAABB_CCDD (AB to A8)
```

Example: Map B or emulation mode address map configuration sequence, 1-byte data write to register at address offset 0xAA

```
Initial values: r1 contains 0x8000_00A8
                r1 contains 0xFEC0_0000
                r2 contains 0xFEE0_0000
                r3 contains 0xAABB_CCDD
                Register at 0xA8 contains 0xFFFF_FFFF (AB to A8)
```

```
Code sequence: stw    r0,0(r1)
                stb    r3,2(r2)
```

```
Results:       Address 0xFEC0_0000 contains 0x8000_00A8 (MSB to LSB)
                Register at 0xA8 contains 0xFFDD_FFFF (AB to A8)
```

3.2.1.2 Configuration Register Access in Big-Endian Mode

In big-endian mode (both the processor and the MPC106), 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 either address map (A, B, or emulation mode), software loads the configuration register address and the configuration register data into the processor register 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 or emulation mode address map configuration sequence, 2-byte data write to register at address offset 0xAA

Initial values: r1 contains 0xA800_0080
 r1 contains 0xFEC0_0000
 r2 contains 0xFEE0_0000
 r3 contains 0xDDCC_BBAA
 Register at 0xA8 contains 0xFFFF_FFFF (AB to A8)

Code sequence: stw r0,0(r1)
 sth r3,2(r2)

Results: Address 0xFEC0_0000 contains 0x8000_00A8 (MSB to LSB)
 Register at 0xA8 contains 0xAABB_FFFF (AB to A8)

3.2.2 Configuration Register Summary

Table 3-10 describes the configuration registers provided by the MPC106. Note that any configuration addresses not defined in Table 3-10 are reserved.

Table 3-10. MPC106 Configuration Registers

Address Offset (in Hex)	Register Size	Program-Accessible Size	Register	Register Access	Default Value
00	2 bytes	2 bytes	Vendor ID =1057h	Read	0x1057
02	2 bytes	2 bytes	Device ID = 0002h	Read	0x0002
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	0xnn
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	0x08
0D	1 byte	1 byte	Latency timer	Read	0x00

Table 3-10. MPC106 Configuration Registers (continued)

Address Offset (in Hex)	Register Size	Program-Accessible Size	Register	Register Access	Default Value
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/write	0x00
42	1 byte	1 byte	Disconnect counter	Read	0x00
48	4 bytes	2 or 4 bytes	Performance monitor command	Write-only	0x0000
4C	2 bytes	1 or 2 bytes	Performance monitor mode control	Read/write	0x00
50	4 bytes	4 bytes	Performance monitor counter 0	Read/write	0x0000
54	4 bytes	4 bytes	Performance monitor counter 1	Read/write	0x0000
58	4 bytes	4 bytes	Performance monitor counter 2	Read/write	0x0000
5C	4 bytes	4 bytes	Performance monitor counter 3	Read/write	0x0000
70	2 bytes	1 or 2 bytes	Power management configuration 1	Read/write	0x0000
72	1 byte	1 byte	Power management configuration 2	Read/write	0x00
73	1 byte	1 byte	Output driver control	Read/write	0xCD
80–87	8 bytes	1, 2, or 4 bytes	Memory starting address	Read/write	0x0000_0000
88–8F	8 bytes	1, 2, or 4 bytes	Extended memory starting address	Read/write	0x0000_0000
90–97	8 bytes	4 bytes	Memory ending address	Read/write	0x0000_0000
98–9F	8 bytes	1, 2, or 4 bytes	Extended memory ending address	Read/write	0x0000_0000
A0	1 byte	1 byte	Memory bank enable	Read/write	0x00
A3	1 byte	1 byte	Memory page mode	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
B8	1 byte	1 byte	ECC single-bit error counter	Read/write	0x00
B9	1 byte	1 byte	ECC single-bit error trigger	Read/write	0x00
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

Table 3-10. MPC106 Configuration Registers (continued)

Address Offset (in Hex)	Register Size	Program-Accessible Size	Register	Register Access	Default Value
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	0x0000_0000
E0	4 bytes	1, 2, or 4 bytes	Emulation support configuration 1	Read/write	0x0FFF_0042
E4	4 bytes	1, 2, or 4 bytes	Modified memory status	Read (no clear)	0x0000_0000
E8	4 bytes	1, 2, or 4 bytes	Emulation support configuration 2	Read/write	0x0000_0020
EC	4 bytes	1, 2, or 4 bytes	Modified memory status	Read (clear)	0x0000_0000
F0	4 bytes	1, 2, or 4 bytes	Memory control configuration 1	Read/write	0xFFn2_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 3-7 shows the MPC106 configuration space.

Reserved

Device ID (0x0002)		Vendor ID (0x1057)		00
PCI Status		PCI Command		04
Class Code	Subclass Code	Standard Programming	Revision ID	08
BIST Control	Header Type	Latency Timer	Cache Line Size	0C
~				
MAX LAT	MIN GNT	Interrupt Pin	Interrupt Line	3C
	Disconnect Counter	Subordinate Bus Number	Bus Number	40
~				
~				
Performance Monitor Command				48
	Performance Monitor Mode Control			4C
Performance Monitor Counter 0				50
Performance Monitor Counter 1				54
Performance Monitor Counter 2				58
Performance Monitor Counter 3				5C
~				
~				
ODCR	PMCR2	Power Management Configuration 1		70
~				
~				
Memory Starting Address				80
Memory Starting Address				84
Extended Memory Starting Address				88
Extended Memory Starting Address				8C
Memory Ending Address				90
Memory Ending Address				94
Extended Memory Ending Address				98
Extended Memory Ending Address				9C
Memory Page Mode		Memory Bank Enable		A0
~				
~				
Processor Interface Configuration 1				A8
Processor Interface Configuration 2				AC
~				
~				
Alternate OS Visible Params 2	Alternate OS Visible Params 1	ECC Single-Bit Trigger	ECC Single-Bit Counter	B8
~				
~				
60x Bus Error Status		Error Detection 1	Error Enabling 1	C0
PCI Bus Error Status		Error Detection 2	Error Enabling 2	C4
60x/PCI Error Address				C8
~				
~				

<input type="checkbox"/> Reserved	Address Offset (Hex)
Emulation Support Configuration 1	E0
Modified Memory Status (No Clear)	E4
Emulation Support Configuration 2	E8
Modified Memory Status (Clear)	EC
Memory Control Configuration 1	F0
Memory Control Configuration 2	F4
Memory Control Configuration 3	F8
Memory Control Configuration 4	FC

Figure 3-7. MPC106 Configuration Space

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 MPC106. Table 3-11 summarizes the PCI configuration registers of the MPC106. Detailed descriptions of these registers are provided in the *PCI Local Bus Specification*.

Table 3-11. 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 (0x0002 = MPC106)
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 MPC106 (0x00)
0A	Subclass code	Identifies more specifically the function of the MPC106 (0x00 = host bridge)
0B	Base class code	Broadly classifies the type of function the MPC106 performs (0x06 = bridge device)
0C	Cache line size	Specifies the system cache line size

Table 3-11. PCI Configuration Space Header Summary (continued)

Address Offset	Register Name	Description
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 MPC106 uses the most common header type (0x00)
0F	BIST control	Optional register for control and status of built-in self test (BIST)
10–33	—	Reserved on the MPC106
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)
3E	MIN GNT	Specifies the length of the device’s burst period (0x00 indicates that the MPC106 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 MPC106 has no major requirements for the settings of latency timers)
40	Bus number	Identifies the assigned bus number of the MPC106 (0x00 = host bridge)
41	Subordinate bus number	Identifies the bus number beneath the MPC106 bridge
42	Disconnect counter	Specifies the timer for target-disconnect timeout (0x00 = the timer is disabled)
43	—	Reserved on the MPC106

3.2.3.1 PCI Command Register

The 2-byte PCI command register, shown in Figure 3-8, provides control over the ability to generate and respond to PCI cycles. Table 3-12 describes the bits of the PCI command register.

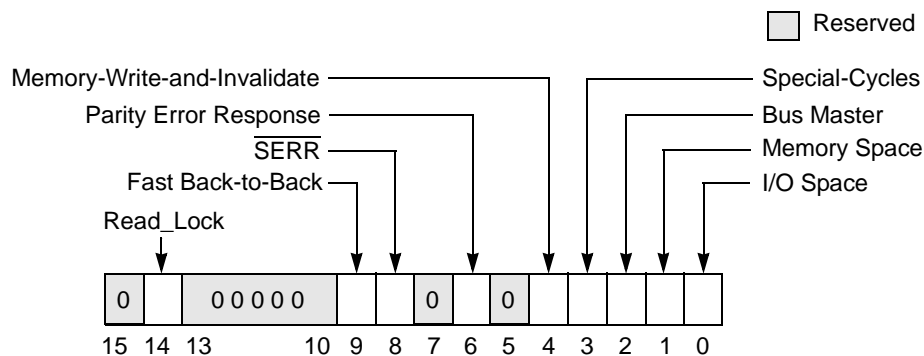


Figure 3-8. PCI Command Register

Table 3-12. Bit Settings for PCI Command Register—0x04

Bit	Name	Reset Value	Description
15	—	All 0s	This bit is reserved.
14	Read_Lock	0	<p>PCI force read lock. This bit controls how the MPC106 (Rev.4.0 and later), as a PCI target, treats PCI read transactions (memory-read, memory-read-multiple, and memory-read-line) to system memory. It is used to force a lock on PCI reads from system memory. Note that Read_Lock is always read back as a 0 even if it has been set by software.</p> <p>0 All PCI reads to system memory behave normally. 1 All PCI reads to system memory are treated internally as if the PCI master had requested an exclusive access (that is, as if the LOCK signal had been asserted). The MPC106 serializes all PCI read transactions to system memory, and all snoops are performed on the 60x bus as RWITM transfer types.</p> <p>This bit should only be set in cases where address collisions are known to cause erroneous transactions. Note that the Read_Lock parameter is not defined in the <i>PCI Local Bus Specification</i>.</p>
13–10	—	All 0s	These bits are reserved.
9	Fast back-to-back	0	This bit is hardwired to 0, indicating that the MPC106 (as a master) does not run fast back-to-back transactions.
8	SERR	0	<p>This bit controls the $\overline{\text{SERR}}$ driver of the MPC106. This bit (and bit 6) must be set to report address parity errors.</p> <p>0 Disables the $\overline{\text{SERR}}$ driver 1 Enables the $\overline{\text{SERR}}$ driver</p>
7	—	0	This bit is reserved.
6	Parity error response	0	<p>This bit controls whether the MPC106 responds to parity errors.</p> <p>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.</p>
5	—	0	This bit is reserved.
4	Memory-write-and-invalidate	0	This bit is hardwired to 0, indicating that the MPC106, acting as a master does not generate the memory-write-and-invalidate command. The MPC106 generates a memory-write command instead.
3	Special-cycles	0	This bit is hardwired to 0, indicating that the MPC106 (as a target) ignores all special-cycle commands.
2	Bus master	1	<p>This bit controls whether the MPC106 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 TEA (provided the TEA_EN bit in PICR1 is set).</p> <p>0 Disables the ability to generate PCI accesses 1 Enables the MPC106 to behave as a PCI bus master</p>
1	Memory space	1	<p>This bit controls whether the MPC106 (as a target) responds to memory accesses.</p> <p>0 The MPC106 does not respond to PCI memory space accesses. 1 The MPC106 responds to PCI memory space accesses.</p>
0	I/O space	0	This bit is hardwired to 0, indicating that the MPC106 (as a target) does not respond to PCI I/O space accesses.

3.2.3.2 PCI Status Register

The 2-byte PCI status register, shown in Figure 3-9, is used to record status information for PCI bus-related events. The definition of each bit is given in Table 3-13. Only 2-byte accesses to address offset 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. Note that attempts to clear the received master-abort bit (bit 13) may not succeed. See Section 9.3.3.3, “Master-Abort Transaction Termination,” for a procedure to ensure that the received master-abort bit is cleared properly.

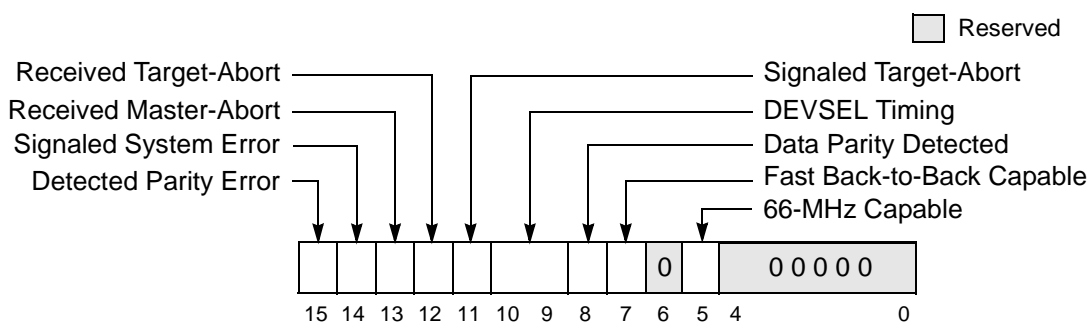


Figure 3-9. PCI Status Register

Table 3-13. Bit Settings for PCI Status Register—0x06

Bit	Name	Reset Value	Description
15	Detected parity error	0	This bit is set whenever the MPC106 detects a parity error, even if parity error handling is disabled (as controlled by bit 6 in the PCI command register).
14	Signaled system error	0	This bit is set whenever the MPC106 asserts \overline{SERR} .
13	Received master-abort	0	This bit is set whenever the MPC106, acting as the PCI master, terminates a transaction (except for a special-cycle) using master-abort.
12	Received target-abort	0	This bit is set whenever an MPC106-initiated transaction (excluding a special-cycle) is terminated by a target-abort.
11	Signaled target-abort	0	This bit is set whenever the MPC106, acting as the PCI target, issues a target-abort to a PCI master.
10–9	DEVSEL timing	00	These bits are hardwired to 0b00, indicating that the MPC106 uses fast device select timing.
8	Data parity detected	0	This bit is set upon detecting a data parity error. Three conditions must be met for this bit to be set: <ul style="list-style-type: none"> • The MPC106 detected a parity error. • MPC106 was acting as the bus master for the operation in which the error occurred. • Bit 6 in the PCI command register was set.

Table 3-13. Bit Settings for PCI Status Register—0x06 (continued)

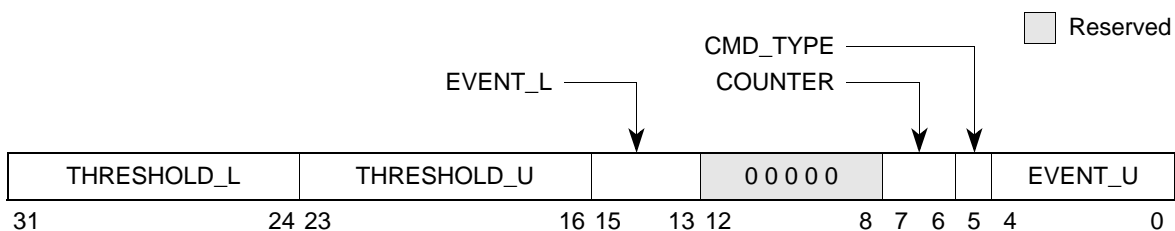
Bit	Name	Reset Value	Description
7	Fast back-to-back capable	1	This bit is hardwired to 1, indicating that the MPC106 (as a target) is capable of accepting fast back-to-back transactions.
6	—	0	This bit is reserved.
5	66-MHz capable	0	This bit is read-only and indicates that the MPC106 is not capable of 66-MHz PCI bus operation.
4-0	—	00000	These bits are reserved.

3.2.4 Performance Monitor Registers

The MPC106 (Rev. 4.0 and later) contains facilities to monitor selected system behavioral characteristics. See Chapter 10, “Performance Monitor,” for more information. This section describes the registers implemented to support the performance monitor facilities.

3.2.4.1 Performance Monitor Command Register (CMDR)—0x48

The performance monitor command register (CMDR) is used to select the counter and the events to be counted. The CMDR can be accessed as a two byte register at address offset 0x48 or as a four byte register at address offset 0x48. It cannot be accessed as a 2-byte register at address offset 0x4A. See Figure 3-10 and Table 3-14 for performance monitor command register (CMDR) bit settings.


Figure 3-10. Performance Monitor Command Register (CMDR)—0x48
Table 3-14. Bit Settings for CMDR—0x48

Bit	Name	Reset Value	Description
31–24	THRESHOLD_L	0x00	Threshold—lower byte. THRESHOLD_U THRESHOLD_L define the threshold value for PMC0 and PMC1. See Section 10.2.3, “Threshold Events,” for more information.
23–16	THRESHOLD_U	0x00	Threshold—upper byte. THRESHOLD_U THRESHOLD_L define the threshold value for PMC0 and PMC1. See Section 10.2.3, “Threshold Events,” for more information.

Table 3-14. Bit Settings for CMDR—0x48 (continued)

Bit	Name	Reset Value	Description
15–13	EVENT_L	000	Event—low order bits. EVENT_U EVENT_L determines which events to count. See Section 10.2, “Performance Monitor Events,” for more information.
12–8	—	00000	These bits are reserved
7–6	COUNTER	00	Counter select. These bits determine the counter to be programmed. 00 PMC0 01 PMC1 10 PMC2 11 PMC3
5	CMD_TYPE	0	Command type. This bit determines the encoding for EVENT_U and EVENT_L. 0 Command type 0. Events to count are selected according to Table 10-1. See Section 10.2.1, “Command Type 0 Events,” for more information. 1 Command type 1. Events to count are selected according to Table 10-2. See Section 10.2.2, “Command Type 1 Events,” for more information.
4–0	EVENT_U	00000	Event—high order bits. EVENT_U EVENT_L determines which events to count. See Section 10.2, “Performance Monitor Events,” for more information.

3.2.4.2 Performance Monitor Mode Control Register (MMCR)—0x4C

The performance monitor mode control register (MMCR) is used to control performance monitor operation. See Figure 3-11 and Table 3-15 for performance monitor mode control register (MMCR) bit settings.

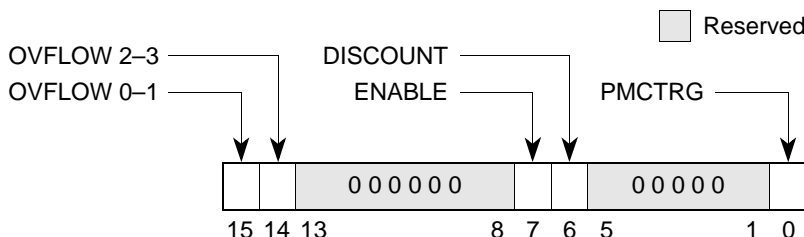

Figure 3-11. Performance Monitor Mode Control Register (MMCR)—0x4C

Table 3-15. Bit Settings for MMCR—0x4C

Bit	Name	Reset Value	Description
15	OVFLOW 0–1	0	Overflow PMC0 into PMC1. Setting this bit allows PMC0 and PMC1 to be linked to become a 64-bit counter. See Section 10.2.5, “Counter Overflow,” for more information. 0 Disable overflow from PMC0 into PMC1. 1 Enable overflow from PMC0 into PMC1. Note that using counter overflow requires DISCOUNT = 0 and PMCTRG = 0.
14	OVFLOW 2–3	0	Overflow PMC2 into PMC3. Setting this bit allows PMC2 and PMC3 to be linked to become a 64-bit counter. See Section 10.2.5, “Counter Overflow,” for more information. 0 Disable overflow from PMC2 into PMC3. 1 Enable overflow from PMC2 into PMC3. Note that using counter overflow requires DISCOUNT = 0 and PMCTRG = 0.
13–8	—	All 0s	These bits are reserved.
7	ENABLE	0	Count enable. This bit enables the performance monitor facility. 0 Disable performance monitor event counting. 1 Enable performance monitor event counting.
6	DISCOUNT	0	Disable counter for msb set. This bit controls the behavior of the counters when their msb (bit 7) changes from 0 to 1. 0 Counters continue to increment after their msb changes from 0 to 1. 1 PMC0 stops when its msb changes from 0 to 1. If PMCTRG = 0, PMC1, PMC2, and PMC3 stop when their msb changes from 0 to 1. If PMCTRG = 1, PMC1, PMC2, and PMC3 continue to increment after their msb changes from 0 to 1.
5–1	—	All 0s	These bits are reserved.
0	PMCTRG	0	Performance monitor counter trigger. This bit determines the behavior of PMC1, PMC2, and PMC3 when the msb (bit 7) of PMC0 changes from 0 to 1. 0 Enable PMC1, PMC2, and PMC3 counting regardless of the state of the msb of PMC0. 1 Disable PMC1, PMC2, and PMC3 counting until the msb of PMC0 changes from 0 to 1. PMCTRG provides a triggering mechanism to allow counting after a certain condition occurs or after a period of time has elapsed.

3.2.4.3 Performance Monitor Counters (PMC0, PMC1, PMC2, PMC3)—0x50, 0x54, 0x58, 0x5C

There are four performance monitor counter registers (PMC0, PMC1, PMC2, and PMC3) that can be used to count events selected by the CMDR. Figure 3-12 shows an example of one performance monitor counter (all four are similar with the exception of their respective addresses).

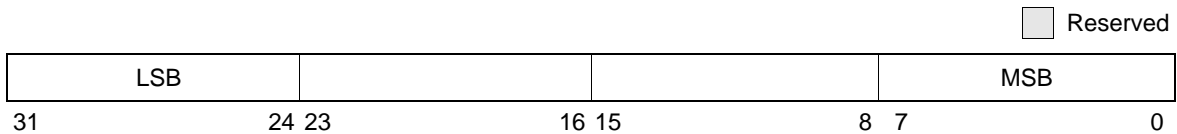


Figure 3-12. Performance Monitor Counter (PMCn)

Note that although the counters use little-endian bit-ordering, the byte ordering is actually big-endian (that is, the most-significant byte is located in bits 7–0, the next most-significant byte in bits 15–8, the third most-significant byte in bits 23–16, and the least-significant byte in bits 31–24).

3.2.5 Power Management Configuration Registers (PMCRs)

The power management configuration registers (PMCRs) control the power management functions of the MPC106.

Power management configuration register 1 (PMCR1), shown in Figure 3-13, is a 2-byte register located at offset 0x70. Some of the bits in PMCR1 configure the MPC106 to use the distinct power management features of different 60x processors. Table 3-16 describes the bits of PMCR1.

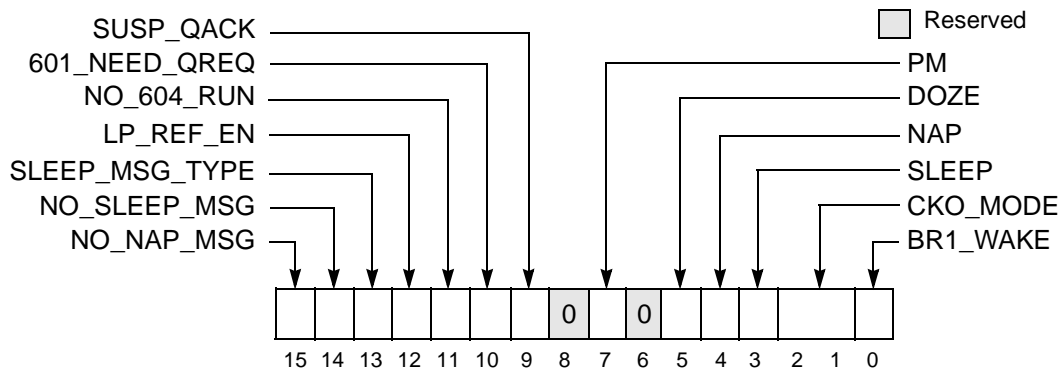


Figure 3-13. Power Management Configuration Register 1 (PMCR1)

**Table 3-16. Bit Settings for Power Management
Configuration Register 1—0x70**

Bit	Name	Reset Value	Description
15	NO_NAP_MSG	0	HALT command broadcast 0 Indicates that the MPC106 broadcasts a HALT command onto the PCI bus prior to entering the nap mode 1 Indicates that the MPC106 does 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 MPC106 broadcasts a sleep message command on the PCI bus prior to entering the sleep mode 1 Indicates that the MPC106 does 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 PMCR1[SLEEP_MSG_TYPE].
13	SLEEP_MSG_TYPE	0	Sleep message type 0 Indicates that the MPC106 broadcasts a HALT command onto the PCI bus prior to entering the sleep mode 1 Indicates that the MPC106 broadcasts a SHUTDOWN command onto the PCI bus prior to entering the sleep mode Note that the sleep message will be broadcast only if PMCR1[NO_SLEEP_MSG] = 0.
12	LP_REF_EN	0	Low-power refresh 0 Indicates that the MPC106 does not perform memory refresh cycles when it is in sleep or suspend mode 1 Indicates that the MPC106 continues to perform memory refresh cycles when in sleep or suspend mode
11	NO_604_RUN	0	When both the 604 and the MPC106 processors are in nap mode and the MPC106 is woken up by a PCI transaction that accesses system memory, this bit controls whether the MPC106 asserts the \overline{QACK} signal so the 604 can respond to the snoop (\overline{QACK} is connected to the RUN signal on the 604). Note that the MPC106 ignores NO_604_RUN unless PICR1[PROC_TYPE] = 0b11, indicating a 604. 0 Indicates that the MPC106 asserts the \overline{QACK} signal 1 Indicates that the MPC106 does not assert the \overline{QACK} signal
10	601_NEED_QREQ	0	Indicates whether the MPC106 should use the \overline{QREQ} signal as one of the conditions for entering the nap/sleep state when a 601 processor is used in the system. Note that the MPC106 ignores 601_NEED_QREQ unless PICR1[PROC_TYPE] = 0b00, indicating a 601. 0 Indicates that the \overline{QREQ} signal is not required 1 Indicates that the \overline{QREQ} signal is required
9	SUSP_QACK	0	Indicates whether the MPC106 asserts the \overline{QACK} signal when entering the suspend power saving mode. 0 Indicates that the MPC106 does not assert the \overline{QACK} signal when entering the suspend power saving mode 1 Indicates that the MPC106 asserts \overline{QACK} when entering the suspend power saving mode
8	—	0	This bit is reserved.

Table 3-16. Bit Settings for Power Management Configuration Register 1—0x70 (continued)

Bit	Name	Reset Value	Description
7	PM	0	Power management enable 0 Disables the power management logic within the MPC106 1 Enables the power management logic within the MPC106
6	—	0	This bit is reserved.
5	DOZE	0	Enables/disables the doze mode capability of the MPC106. Note that this bit is only valid if MPC106 power management is enabled (PMCR1[PM] = 1). 0 Disables the doze mode 1 Enables the doze mode
4	NAP	0	Enables/disables the nap mode capability of the MPC106. Note that this bit is only valid if MPC106 power management is enabled (PMCR1[PM] = 1). 0 Disables the nap mode 1 Enables the nap mode
3	SLEEP	0	Enables/disables the sleep mode capability of the MPC106. Note that this bit is only valid if MPC106 power management is enabled (PMCR1[PM] = 1). 0 Disables the sleep mode 1 Enables the sleep mode
2–1	CKO_MODE	00	Selects the clock source for the test clock output. 00 Disables the test clock output driver 01 Selects the internal (core) clock as the test clock output source 10 Selects one-half of the internal (core) clock as the test clock output source 11 Selects SYSCLK as the test clock output source
0	BR1_WAKE	0	Enables/disables awareness of a second, third, or fourth processor for nap and sleep modes. 0 Indicates the MPC106 will not awaken from nap or sleep mode when $\overline{BR1}$, $\overline{BR2}$, or $\overline{BR3}$ is asserted 1 Indicates the MPC106 will awaken from nap or sleep mode when $\overline{BR1}$, $\overline{BR2}$, or $\overline{BR3}$ is asserted in a multiprocessor configuration

Power management configuration register 2 (PMCR2), shown in Figure 3-14, is a 1-byte register located at offset 0x72. Table 3-17 describes the bits of PMCR2.

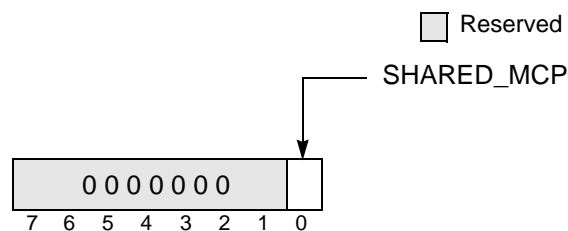
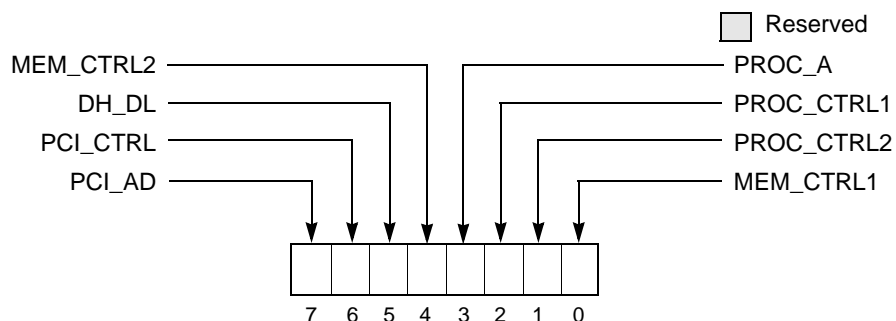

Figure 3-14. Power Management Configuration Register 2 (PMCR2)

Table 3-17. Bit Settings for Power Management Configuration Register 2—0x72

Bit	Name	Reset Value	Description
7-1	—	All 0s	These bits are reserved.
0	SHARED_MCP	0	This bit allows the MPC106 to share the $\overline{\text{MCP}}$ signal with another device that might drive $\overline{\text{MCP}}$. 0 The MPC106 always drives the $\overline{\text{MCP}}$ signal. 1 The MPC106 drives the $\overline{\text{MCP}}$ signal when there is an error to report. Otherwise, the $\overline{\text{MCP}}$ signal is released to a high-impedance state.

3.2.6 Output Driver Control Register (ODCR)—0x73

The output driver control register (ODCR) is an 8-bit register located at address offset 0x73. See Figure 3-15 and Table 3-18 for output driver control register bit settings.


Figure 3-15. Output Driver Control Register (ODCR)—0x73
Table 3-18. Bit Settings for ODCR—0x73

Bit	Name	Reset Value	Description
7	PCI_AD	1	PCI address/data signal driver control. This bit controls the drivers for the PCI bus signals AD[31-0], C/BE[3-0], and PAR. 0 Indicates that the MPC106 has been configured for 40 Ω PCI AD signal drivers. 1 Indicates that the MPC106 has been configured for 20 Ω PCI signal drivers.
6	PCI_CTRL	1	PCI control signal driver control. This bit controls the drivers for the PCI bus signals FRAME, IRDY, TRDY, DEVSEL, STOP, SERR, PERR, REQ, and MEMACK. 0 Indicates that the MPC106 has been configured for 40 Ω PCI control signal drivers. 1 Indicates that the MPC106 has been configured for 20 Ω PCI control signal drivers.

Table 3-18. Bit Settings for ODCR—0x73 (continued)

Bit	Name	Reset Value	Description																		
5	DH_DL	0	Data bus signal driver control. This bit controls the drivers for the processor/memory data bus signals DH[0–31] and DL[0–31]. 0 Indicates that the MPC106 has been configured for 40 Ω data bus signal drivers. 1 Indicates that the MPC106 has been configured for 20 Ω data bus signal drivers.																		
4	MEM_CTRL2	0	Memory signal driver control 2. This bit, along with MEM_CTRL1, controls the drivers for the memory control and address signals MA/SDMA[0–12], PAR/AR[0–7], RAS/CS[0–7], CAS/DQM[0–7], WE, PPEN, FOE, RCS[0–1], DBGLB/CKE, PIRQ/SDRAS, MDLE/SDCAS, and BCTL[0–1]. <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="2">MEM_CTRL</th> <th>Memory Bus Driver</th> </tr> <tr> <th>2</th> <th>1</th> <th></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Reserved</td> </tr> <tr> <td>0</td> <td>1</td> <td>20 Ω</td> </tr> <tr> <td>1</td> <td>0</td> <td>13 Ω</td> </tr> <tr> <td>1</td> <td>1</td> <td>8 Ω</td> </tr> </tbody> </table>	MEM_CTRL		Memory Bus Driver	2	1		0	0	Reserved	0	1	20 Ω	1	0	13 Ω	1	1	8 Ω
MEM_CTRL		Memory Bus Driver																			
2	1																				
0	0	Reserved																			
0	1	20 Ω																			
1	0	13 Ω																			
1	1	8 Ω																			
3	PROC_A	1	Processor address bus signal driver control. This bit controls the drivers for the processor address bus signals A[0–31]. 0 Indicates that the MPC106 has been configured for 40 Ω processor address bus signal drivers. 1 Indicates that the MPC106 has been configured for 20 Ω processor address bus signal drivers.																		
2	PROC_CTRL1	1	Processor/L2 control signal 1 driver control. This bit controls the drivers for the processor/L2 control signals CKO, TS, AACK, ARTRY, TA, TEA, BGN, DBGn, ADS, DOE, BAA, DCS, BA0, DWEn, LBCLAIM, QACK, and MCP. 0 Indicates that the MPC106 has been configured for 40 Ω processor/L2 control signal 1 drivers. 1 Indicates that the MPC106 has been configured for 20 Ω processor/L2 control signal 1 drivers.																		
1	PROC_CTRL2	0	Processor/L2 control signal 2 driver control. This bit controls the drivers for the processor/L2 control signals CI, WT, GBL, TBST, TSIZ[0–2], TT[0–4], TWE, and TV. 0 Indicates that the MPC106 has been configured for 40 Ω processor/L2 control signal 2 drivers. 1 Indicates that the MPC106 has been configured for 20 Ω processor/L2 control signal 2 drivers.																		
0	MEM_CTRL1	1	Memory signal driver control 1. This bit, along with MEM_CTRL2, controls the drivers for the memory control and address signals MA/SDMA[0–12], PAR/AR[0–7], RAS/CS[0–7], CAS/DQM[0–7], WE, PPEN, FOE, RCS[0–1], DBGLB/CKE, PIRQ/SDRAS, MDLE/SDCAS, and BCTL[0–1]. See the description for MEM_CTRL2 for the specific bit encodings.																		

3.2.7 Error Handling Registers

Chapter 9, “Error Handling,” describes specific error conditions and how the MPC106 responds to them. The registers at offsets 0xB8, 0xB9, and 0xC0 through 0xCB control the error handling and reporting for the MPC106. The following sections provide descriptions of these registers.

3.2.7.1 ECC Single-Bit Error Registers

The ECC single-bit error registers are two 8-bit registers used to control the reporting of ECC single-bit errors. See Chapter 9, “Error Handling,” for more information. The ECC single-bit error counter, shown in Figure 3-16, maintains a count of the number of single-bit errors that have been detected. Table 3-19 describes the bits of the ECC single-bit error counter.

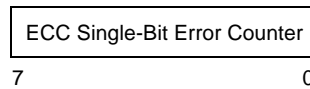


Figure 3-16. ECC Single-Bit Error Counter Register—0xB8

Table 3-19. Bit Settings for ECC Single-Bit Error Counter Register—0xB8

Bit	Name	Reset Value	Description
7–0	ECC single-bit error counter	All 0s	These bits maintain a count of the number of ECC single-bit errors that have been detected and corrected. If this value equals the value contained in the ECC single-bit error trigger register, then an error will be reported (provided ErrEnR1[2] = 1).

The ECC single-bit error trigger, shown in Figure 3-17, provides a threshold value, that, when equal to the single-bit error count, triggers the MPC106 error reporting logic. Table 3-20 describes the bits of the ECC single-bit error trigger.

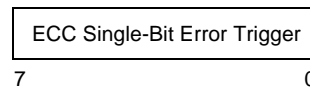


Figure 3-17. ECC Single-Bit Error Trigger Register—0xB9

Table 3-20. Bit Settings for ECC Single-Bit Error Trigger Register—0xB9

Bit	Name	Reset Value	Description
7–0	ECC single-bit error trigger	All 0s	These bits provide the threshold value for the number of ECC single-bit errors that are detected before reporting an error condition.

3.2.7.2 Error Enabling Registers

Error enabling registers 1 and 2 (ErrEnR1 and ErrEnR2), shown in Figure 3-18 and Figure 3-19, control whether the MPC106 recognizes and reports specific error conditions. Table 3-21 describes the bits of ErrEnR1 and Table 3-22 describes the bits of ErrEnR2.

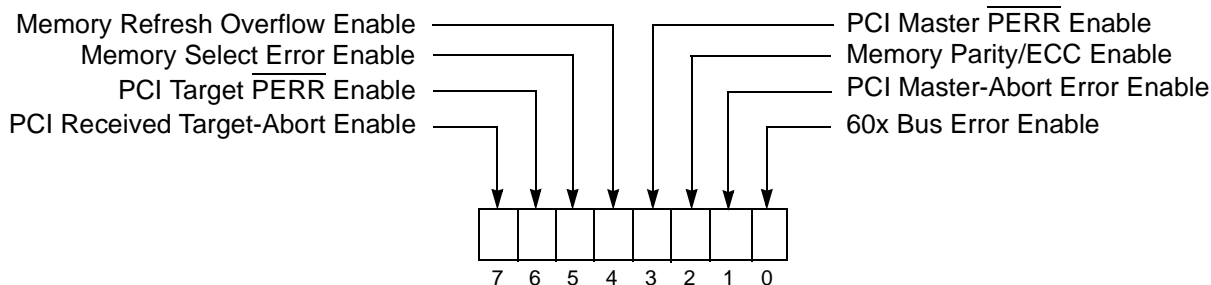


Figure 3-18. Error Enabling Register 1 (ErrEnR1)

Table 3-21. Bit Settings for Error Enabling Register 1 (ErrEnR1)—0xC0

Bit	Name	Reset Value	Description
7	PCI received target-abort enable	0	This bit enables the reporting of target-abort errors that occur on the PCI bus for transactions involving the MPC106 as a master. 0 Received PCI target-abort error disabled 1 Received PCI target-abort error enabled
6	PCI target $\overline{\text{PERR}}$ enable	0	This bit enables the reporting of data parity errors on the PCI bus for transactions involving the MPC106 as a target. 0 Target $\overline{\text{PERR}}$ disabled 1 Target $\overline{\text{PERR}}$ enabled
5	Memory select error enable	0	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	Memory refresh overflow enable	0	This bit enables the reporting of memory refresh overflow errors. 0 Memory refresh overflow disabled 1 Memory refresh overflow enabled
3	PCI master $\overline{\text{PERR}}$ enable	0	This bit enables the reporting of data parity errors on the PCI bus for transactions involving the MPC106 as a master. 0 Master $\overline{\text{PERR}}$ disabled 1 Master $\overline{\text{PERR}}$ enabled
2	Memory parity/ECC enable	0	This bit enables the reporting of system memory read parity errors that occur on accesses to system memory or exceeding the ECC single-bit error threshold. 0 Memory read parity/ECC single-bit threshold disabled 1 Memory read parity/ECC single-bit threshold enabled

Table 3-21. Bit Settings for Error Enabling Register 1 (ErrEnR1)—0xC0 (continued)

Bit	Name	Reset Value	Description
1	PCI master-abort error enable	0	This bit enables the reporting of master-abort errors that occurred on the PCI bus for transactions involving the MPC106 as a master. 0 PCI master-abort error disabled 1 PCI master-abort error enabled
0	60x bus error enable	1	This bit enables the reporting of 60x bus errors. 0 60x bus error disabled 1 60x bus error enabled

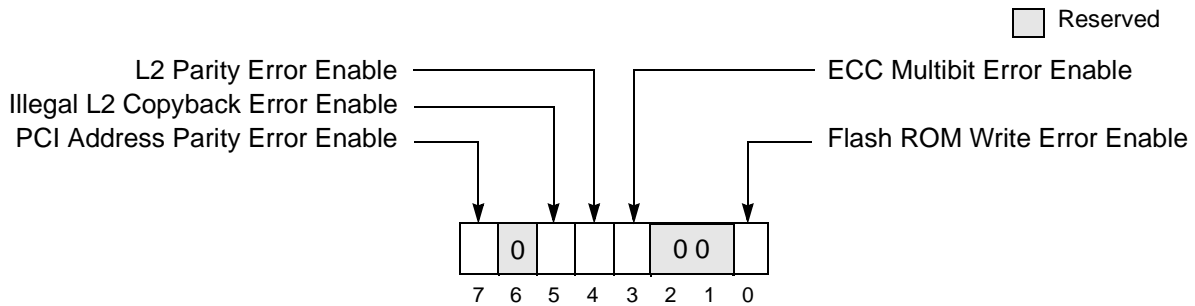


Figure 3-19. Error Enabling Register 2 (ErrEnR2)

Table 3-22. Bit Settings for Error Enabling Register 2 (ErrEnR2)—0xC4

Bit	Name	Reset Value	Description
7	PCI address parity error enable.	0	This bit controls whether the MPC106 asserts \overline{MCP} (provided \overline{MCP} is enabled) if an address parity error is detected by the MPC106 when acting as a target. 0 PCI address parity errors disabled 1 PCI address parity errors enabled
6	—	0	This bit is reserved.
5	Illegal L2 copyback error enable	0	This bit controls whether an error is reported when the L2 cache attempts a copyback to PCI memory space or ROM/Flash space. 0 Disabled 1 Enabled
4	L2 parity error enable	0	This bit enables parity generation and parity error detection for the internally-controlled L2 cache. Note that parity checking must be enabled (MICR1[PCKEN] = 1) to detect L2 parity errors. 0 L2 parity disabled 1 L2 parity enabled
3	ECC multibit error enable	0	This bit enables the detection of ECC multibit errors. 0 ECC multibit error detection disabled 1 ECC multibit error detection enabled

Table 3-22. Bit Settings for Error Enabling Register 2 (ErrEnR2)—0xC4 (continued)

Bit	Name	Reset Value	Description
2-1	—	00	These bits are reserved.
0	Flash ROM write error enable	0	This bit controls whether the MPC106 detects write Flash errors. 0 Disabled 1 Enabled

3.2.7.3 Error Detection Registers

Error detection registers 1 and 2 (ErrDR1 and ErrDR2), shown in Figure 3-20 and Figure 3-21, contain error flags that report when the MPC106 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 MPC106 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 3-23 describes the bits of error detection register 1 and Table 3-24 describes the bits of error detection register 2.

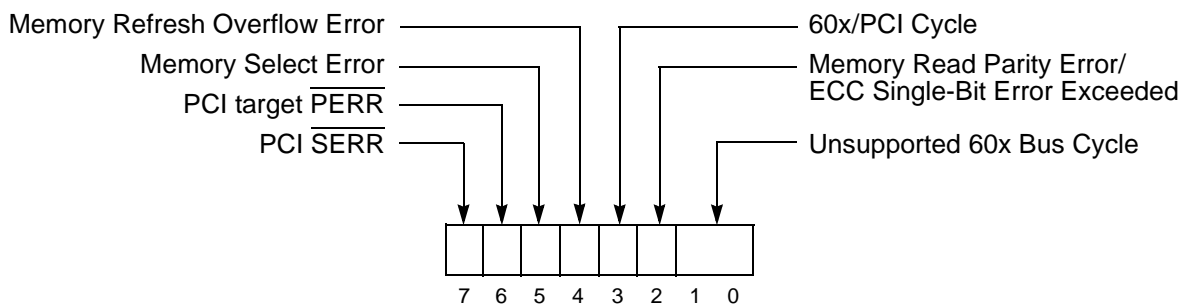


Figure 3-20. Error Detection Register 1 (ErrDR1)—0xC1

Table 3-23. Bit Settings for Error Detection Register 1 (ErrDR1)—0xC1

Bit	Name	Reset Value	Description
7	PCI $\overline{\text{SERR}}$	0	Note that for the MPC106 to recognize the assertion of $\overline{\text{SERR}}$ by another PCI agent, bit 5 (RX_SERR_EN) of the alternate OS-visible parameters register 1 must be set. See Table 3-46 for a description of the RX_SERR_EN bit. 0 No error signaled 1 Error signaled
6	PCI target $\overline{\text{PERR}}$	0	PCI target $\overline{\text{PERR}}$ 0 The MPC106, as a PCI target, has not detected a data parity error 1 The MPC106, as a PCI target, detected a data parity error
5	Memory select error	0	Memory select error 0 No error detected 1 Memory select error detected
4	Memory refresh overflow error	0	Memory refresh overflow error 0 No error detected 1 Memory refresh overflow has occurred
3	60x/PCI cycle	0	60x/PCI cycle 0 Error occurred on a 60x-initiated cycle 1 Error occurred on a PCI-initiated cycle
2	Memory read parity error/ECC single-bit error trigger exceeded	0	Memory read parity error/ECC single-bit error trigger exceeded 0 No error detected 1 Parity error detected or ECC single-bit error trigger exceeded
1–0	Unsupported 60x bus cycle	00	Unsupported 60x bus cycle 00 No error detected 01 Unsupported transfer attributes 10 XATS detected 11 Reserved

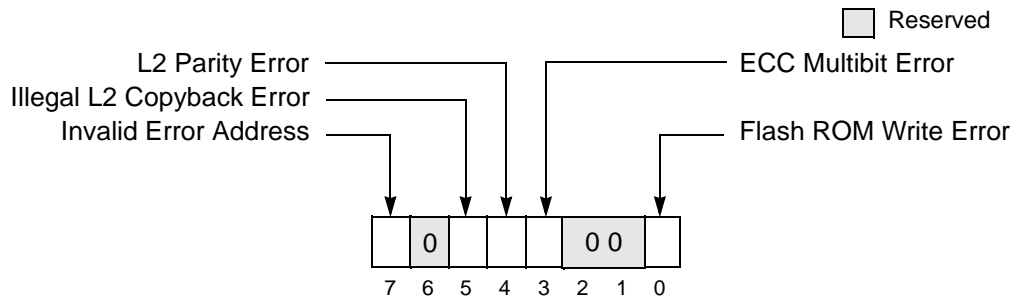


Figure 3-21. Error Detection Register 2 (ErrDR2)—0xC5

Table 3-24. Bit Settings for Error Detection Register 2 (ErrDR2)—0xC5

Bit	Name	Reset Value	Description
7	Invalid error address	0	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	—	0	This bit is reserved.
5	Illegal L2 copyback error	0	Illegal L2 copyback error 0 No error detected 1 Illegal L2 copyback (L2 copyback to PCI or ROM/Flash space) error detected
4	L2 parity error	0	L2 parity error 0 No error detected 1 L2 parity error detected
3	ECC multibit error	0	ECC multibit error 0 No error detected 1 ECC multibit error detected
2–1	—	00	These bits are reserved.
0	Flash ROM write error	0	Flash ROM write error 0 No error detected 1 The MPC106 detected a write to Flash ROM when writes to Flash ROM are disabled or a non-data-path-sized write to Flash ROM.

3.2.7.4 Error Status Registers

The error status registers latch the state of the 60x or PCI address bus when an error is detected; see Figure 3-22, Figure 3-23, and Figure 3-24. These registers provide system status for error handling software.

Table 3-25 describes the bits of the 60x bus error status register, Table 3-26 describes the bits of the PCI bus error status register, and Table 3-27 describes the bits of 60x/PCI error address register.

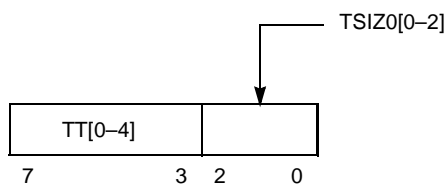
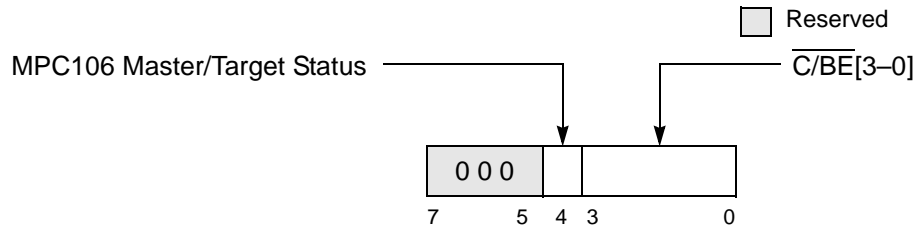


Figure 3-22. 60x Bus Error Status Register—0xC3

Table 3-25. Bit Settings for 60x Bus Error Status Register—0xC3

Bit	Name	Reset Value	Description
7–3	TT[0–4]	00000	These bits maintain a copy of TT[0–4]. When a 60x bus error is detected, these bits are latched until all error flags are cleared.
2–0	TSIZ[0–2]	000	These bits maintain a copy of TSIZ[0–2]. When a 60x bus error is detected, these bits are latched until all error flags are cleared.


Figure 3-23. PCI Bus Error Status Register—0xC7
Table 3-26. Bit Settings for PCI Bus Error Status Register—0xC7

Bit	Name	Reset Value	Description
7–5	—	000	These bits are reserved.
4	MPC106 master/target status	0	MPC106 master/target status 0 MPC106 is the PCI master 1 MPC106 is the PCI target
3–0	$\overline{C}/\overline{BE}[3-0]$	0000	These bits maintain a copy of $\overline{C}/\overline{BE}[3-0]$. When a PCI bus error is detected, these bits are latched until all error flags are cleared.

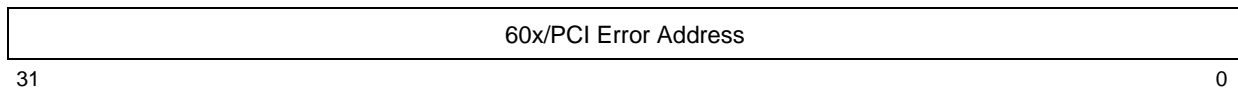

Figure 3-24. 60x/PCI Error Address Register—0xC8

Table 3-27. Bit Settings for 60x/PCI Error Address Register—0xC8

Bit	Name	Reset Value	Description
31–24	Error address	0x00	A[24–31] or AD[7–0] (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		0x00	A[16–23] or AD[15–8] (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		0x00	A[8–15] or AD[23–16] (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		0x00	A[0–7] or AD[31–24] (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.

3.2.8 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 MPC106 by setting the MEMGO bit in memory control configuration register 1 (MCCR1).

3.2.8.1 Memory Boundary Registers

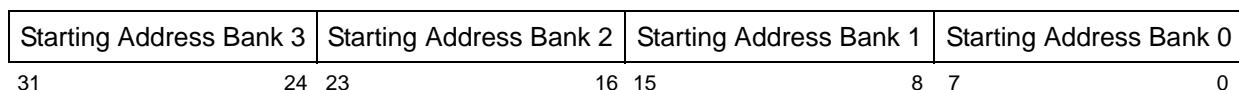
The extended starting address and the starting address registers are used to define the lower address boundary for each memory bank. The lower boundary is determined by the following formula:

$$\text{Lower boundary for bank } n = 0b00 \parallel \langle \text{extended starting address } n \rangle \parallel \langle \text{starting address } n \rangle \parallel 0x00000.$$

The extended ending address and the ending address registers are used to define the upper address boundary for each memory bank. The upper boundary is determined by the following formula:

$$\text{Upper boundary for bank } n = 0b00 \parallel \langle \text{extended ending address } n \rangle \parallel \langle \text{ending address } n \rangle \parallel 0xFFFFF.$$

See Figure 3-25, Figure 3-26, and Figure 3-28 for memory starting address register 1 and 2 bit settings.


Figure 3-25. Memory Starting Address Register 1—0x80

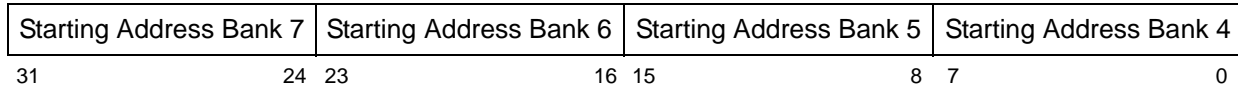


Figure 3-26. Memory Starting Address Register 2—0x84

Table 3-28. Bit Settings for Memory Starting Address Registers 1 and 2

Bit	Name	Reset Value	Description	Word Address
31–24	Starting address bank 3	0x00	Starting address for bank 3	0x80
23–16	Starting address bank 2	0x00	Starting address for bank 2	
15–8	Starting address bank 1	0x00	Starting address for bank 1	
7–0	Starting address bank 0	0x00	Starting address for bank 0	
31–24	Starting address bank 7	0x00	Starting address for bank 7	0x84
23–16	Starting address bank 6	0x00	Starting address for bank 6	
15–8	Starting address bank 5	0x00	Starting address for bank 5	
7–0	Starting address bank 4	0x00	Starting address for bank 4	

See Figure 3-27, Figure 3-28, and Table 3-29 for extended memory starting address register 1 and 2 bit settings.

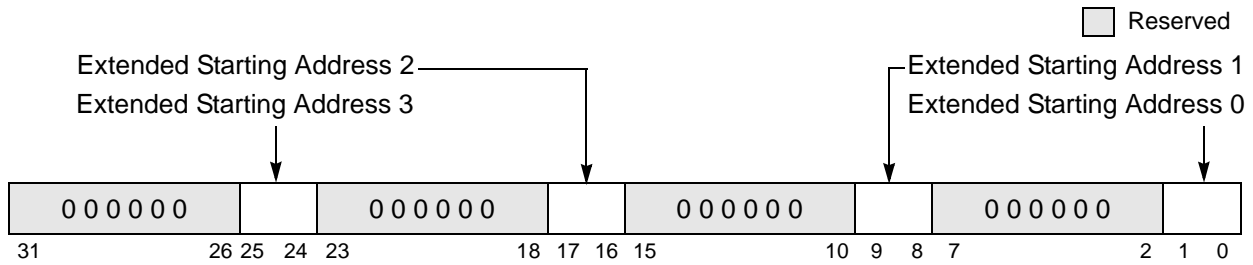


Figure 3-27. Extended Memory Starting Address Register 1—0x88

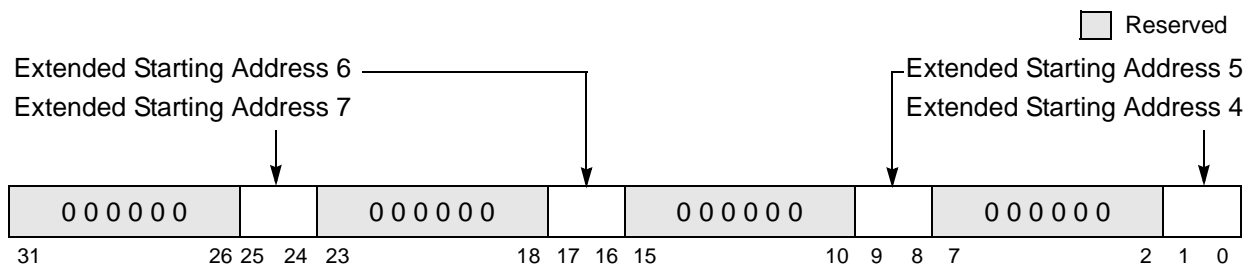


Figure 3-28. Extended Memory Starting Address Register 2—0x8C

Table 3-29. Bit Settings for Extended Memory Starting Address Registers 1 and 2

Bit	Name	Reset Value	Description	Byte Address
31–26	—	All 0s	These bits are reserved.	0x88
25–24	Extended starting address 3	0b00	Extended starting address for bank 3	
23–18	—	All 0s	These bits are reserved.	
17–16	Extended starting address 2	0b00	Extended starting address for bank 2	
15–10	—	All 0s	These bits are reserved.	
9–8	Extended starting address 1	0b00	Extended starting address for bank 1	
7–2	—	All 0s	These bits are reserved.	
1–0	Extended starting address 0	0b00	Extended starting address for bank 0	
31–26	—	All 0s	These bits are reserved.	0x8C
25–24	Extended starting address 7	0b00	Extended starting address for bank 7	
23–18	—	All 0s	These bits are reserved.	
17–16	Extended starting address 6	0b00	Extended starting address for bank 6	
15–10	—	All 0s	These bits are reserved.	
9–8	Extended starting address 5	0b00	Extended starting address for bank 5	
7–2	—	All 0s	These bits are reserved.	
1–0	Extended starting address 4	0b00	Extended starting address for bank 4	

See Figure 3-29, Figure 3-30, and Table 3-30 for memory ending address register 1 and 2 bit settings.

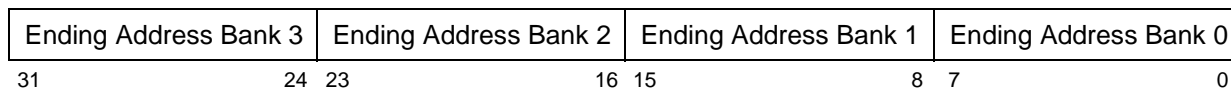


Figure 3-29. Memory Ending Address Register 1—0x90

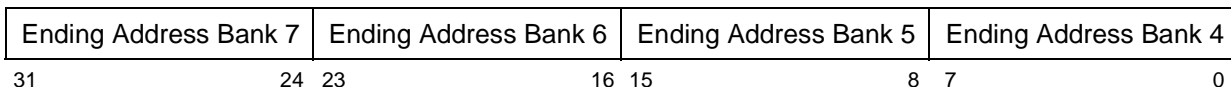


Figure 3-30. Memory Ending Address Register 2—0x94

Table 3-30. Bit Settings for Memory Ending Address Registers 1 and 2

Bit	Name	Reset Value	Description	Byte Address
31–24	Ending address bank 3	0x00	Ending address for bank 3	0x90
23–16	Ending address bank 2	0x00	Ending address for bank 2	
15–8	Ending address bank 1	0x00	Ending address for bank 1	
7–0	Ending address bank 0	0x00	Ending address for bank 0	
31–24	Ending address bank 7	0x00	Ending address for bank 7	0x94
23–16	Ending address bank 6	0x00	Ending address for bank 6	
15–8	Ending address bank 5	0x00	Ending address for bank 5	
7–0	Ending address bank 4	0x00	Ending address for bank 4	

See Figure 3-31, Figure 3-32, and Table 3-31 for extended memory ending address register 1 and 2 bit settings.

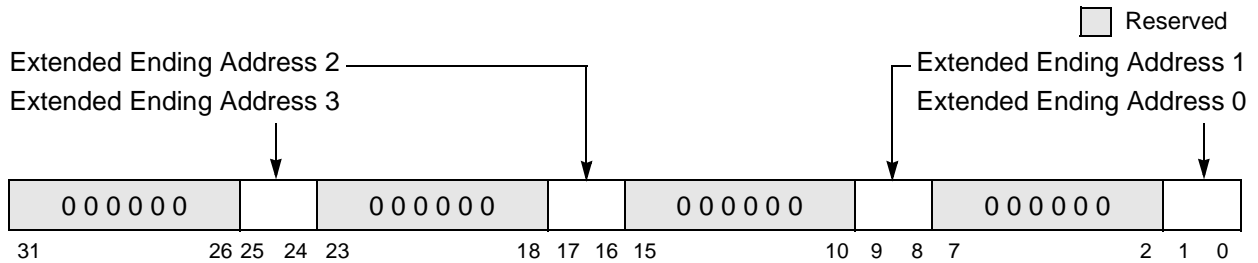


Figure 3-31. Extended Memory Ending Address Register 1—0x98

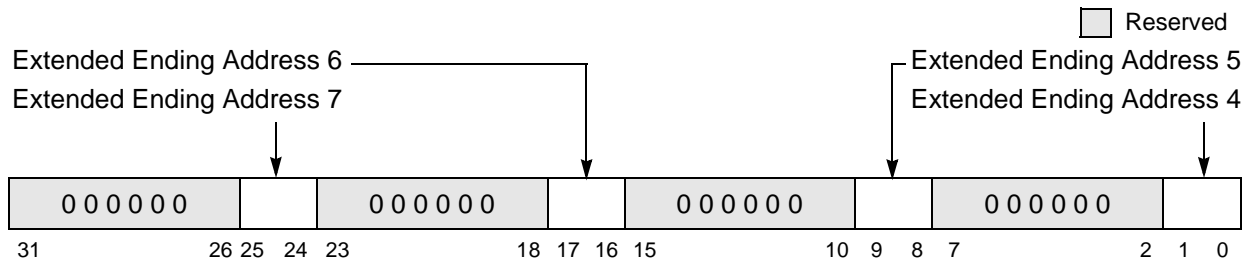


Figure 3-32. Extended Memory Ending Address Register 2—0x9C

Table 3-31. Bit Settings for Extended Memory Ending Address Registers 1 and 2

Bit	Name	Reset Value	Description	Byte Address
31–26	—	All 0s	These bits are reserved.	0x98
25–24	Extended ending address 3	0b00	Extended ending address for bank 3	
23–18	—	All 0s	These bits are reserved.	
17–16	Extended ending address 2	0b00	Extended ending address for bank 2	
15–10	—	All 0s	These bits are reserved.	
9–8	Extended ending address 1	0b00	Extended ending address for bank 1	
7–2	—	All 0s	These bits are reserved.	
1–0	Extended ending address 0	0b00	Extended ending address for bank 0	
31–26	—	All 0s	These bits are reserved.	0x9C
25–24	Extended ending address 7	0b00	Extended ending address for bank 7	
23–18	—	All 0s	These bits are reserved.	
17–16	Extended ending address 6	0b00	Extended ending address for bank 6	
15–10	—	All 0s	These bits are reserved.	
9–8	Extended ending address 5	0b00	Extended ending address for bank 5	
7–2	—	All 0s	These bits are reserved.	
1–0	Extended ending address 4	0b00	Extended ending address for bank 4	

3.2.8.2 Memory Bank Enable Register

Individual banks are enabled or disabled by using the 1-byte memory bank enable register, shown in Figure 3-33 and Table 3-32. 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.

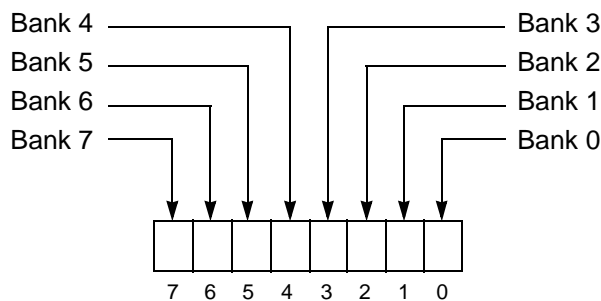


Figure 3-33. Memory Bank Enable Register—0xA0

Table 3-32. Bit Settings for Memory Bank Enable Register—0xA0

Bit	Name	Reset Value	Description
7	Bank 7	0	Bank 7 0 Disabled 1 Enabled
6	Bank 6	0	Bank 6 0 Disabled 1 Enabled
5	Bank 5	0	Bank 5 0 Disabled 1 Enabled
4	Bank 4	0	Bank 4 0 Disabled 1 Enabled
3	Bank 3	0	Bank 3 0 Disabled 1 Enabled
2	Bank 2	0	Bank 2 0 Disabled 1 Enabled
1	Bank 1	0	Bank 1 0 Disabled 1 Enabled
0	Bank 0	0	Bank 0 0 Disabled 1 Enabled

3.2.8.3 Memory Page Mode Register

The 1-byte memory page mode register, shown in Figure 3-34 and Table 3-33, contains the PGMAX parameter which controls how long the MPC106 retains the currently accessed page (row) in memory. See Section 6.4.7, “DRAM/EDO Page Mode Retention,” or Section 6.5.4, “SDRAM Page Mode Retention,” for more information.

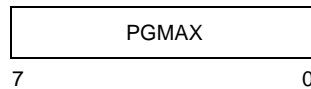

Figure 3-34. Memory Page Mode Register—0xA3

Table 3-33. Bit Settings for Memory Page Mode Register—0xA3

Bit	Name	Reset Value	Description
7-0	PGMAX	All 0s	For DRAM/EDO configurations, the value of PGMAX, multiplied by 64, determines the maximum $\overline{\text{RAS}}$ assertion interval for retained page mode. When programmed to 0x00, page mode is disabled. For SDRAM configurations, the value of PGMAX, multiplied by 64, determines the activate to precharge interval (sometimes called row active time or t_{RAS}) for retained page mode. When programmed to 0x00, page mode is disabled.

3.2.8.4 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 MPC106 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 Figure 3-35 and Table 3-34 for memory control configuration register 1 bit settings.

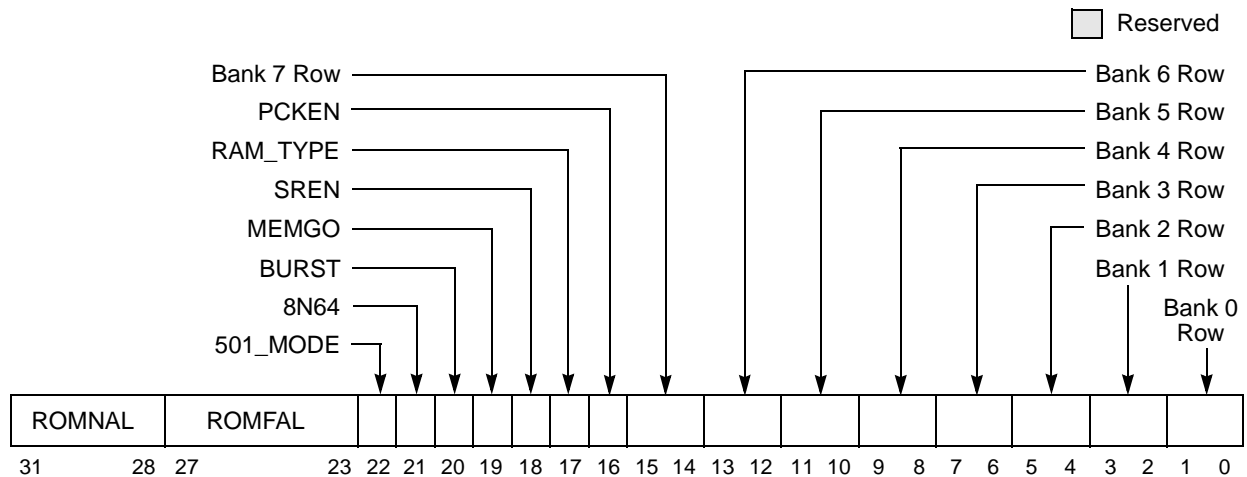


Figure 3-35. Memory Control Configuration Register 1 (MCCR1)—0xF0

Table 3-34. Bit Settings for MCCR1—0xF0

Bit	Name	Reset Value	Description
31–28	ROMNAL	All 1s	For burst-mode ROM and Flash reads, ROMNAL controls the next access time. The maximum value is 0b1111 (15). The actual cycle count will be three cycles more than the binary value of ROMNAL. For Flash writes, ROMNAL measures the write pulse recovery (high) time. The maximum value is 0b1111 (15). The actual cycle count will be four cycles more than the binary value of ROMNAL.
27–23	ROMFAL	All 1s	For nonburst ROM and Flash reads, ROMFAL controls the access time. For burst-mode ROMs, ROMFAL controls the first access time. The maximum value is 0b11111 (31). For the 64-bit configuration, the actual cycle count will be three cycles more than the binary value of ROMFAL. For the 8-bit configuration, the actual cycle count will be two cycles more than the binary value of ROMFAL. For Flash writes, ROMFAL measures the write pulse low time. The maximum value is 0b11111 (31). The actual cycle count will be two cycles more than the binary value of ROMFAL.
22	501_MODE	1	Read only. This bit indicates the state of the 501-mode configuration signal ($\overline{\text{BCTL0}}$) at power-on reset. See Section 2.2.9.1, “501-Mode ($\overline{\text{BCTL0}}$)—Input,” for a description of the 501-mode configuration signal. 0 Indicates that the MPC106 has been configured for 16501-type buffers 1 Indicates that the MPC106 has been configured for backward compatibility Note that the setting of 501_MODE and BUF_MODE (bit 1 of MCCR2) determine the behavior of the buffer control signals $\overline{\text{BCTL}}[0-1]$. See Section 6.2, “Memory Interface Signal Buffering,” for more information.
21	8N64	x	Read only. This bit indicates the state of the ROM bank 0 data path width configuration signal ($\overline{\text{FOE}}$) at power-on reset. 0 Indicates that the MPC106 has been configured for a 64-bit data path for ROM bank 0 1 Indicates that the MPC106 has been configured for an 8-bit data path for ROM bank 0
20	BURST	0	Burst mode ROM timing enable. 0 Indicates standard (nonburst) ROM access timing 1 Indicates burst-mode ROM access timing
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 MPC106 RAM interface logic disabled 1 MPC106 RAM interface logic enabled
18	SREN	0	Self-refresh enable. Note that if self refresh is disabled, the system is responsible for preserving the integrity of DRAM/EDO/SDRAM during sleep or suspend mode. 0 Disables the DRAM/EDO/SDRAM self refresh during sleep or suspend mode 1 Enables the DRAM/EDO/SDRAM self refresh during sleep or suspend mode

Table 3-34. Bit Settings for MCCR1—0xF0 (continued)

Bit	Name	Reset Value	Description
17	RAM_TYPE	1	RAM type 0 Indicates synchronous DRAM (SDRAM) 1 Indicates DRAM or EDO DRAM (depending on the setting for MCCR2[EDO])
16	PCKEN	0	Memory interface parity checking/generation enable 0 Disables parity checking and parity generation for transactions to DRAM/EDO/SDRAM memory. If ECC is enabled, disables L2 parity checking. 1 Enables parity checking and generation for all memory transactions to DRAM/EDO/SDRAM. If ECC is enabled, enables L2 parity checking.
15–14	Bank 7 row	00	RAM bank 7 row address bit count. These bits indicate the number of row address bits required by the RAM devices in bank 7. For DRAM/EDO configurations (RAM_TYPE = 1), the encoding is as follows: 00 9 row bits 01 10 row bits 10 11 row bits 11 12 or 13 row bits For SDRAM configurations (RAM_TYPE = 0), the encoding is as follows: 00 12 row bits + 2 bank selects—64- or 128-Mbit, 4 internal bank devices 01 13 row bits + 1 bank select—64- or 128-Mbit, 2 internal bank devices 10 Reserved 11 11 row bits + 1 bank select—16-Mbit, 2 internal bank devices
13–12	Bank 6 row	00	RAM bank 6 row address bit count. See the description for bits 15–14.
11–10	Bank 5 row	00	RAM bank 5 row address bit count. See the description for bits 15–14.
9–8	Bank 4 row	00	RAM bank 4 row address bit count. See the description for bits 15–14.
7–6	Bank 3 row	00	RAM bank 3 row address bit count. See the description for bits 15–14.
5–4	Bank 2 row	00	RAM bank 2 row address bit count. See the description for bits 15–14.
3–2	Bank 1 row	00	RAM bank 1 row address bit count. See the description for bits 15–14.
1–0	Bank 0 row	00	RAM bank 0 row address bit count. See the description for bits 15–14.

See Figure 3-36 and Table 3-35 for memory control configuration register 2 (MCCR2) bit settings.

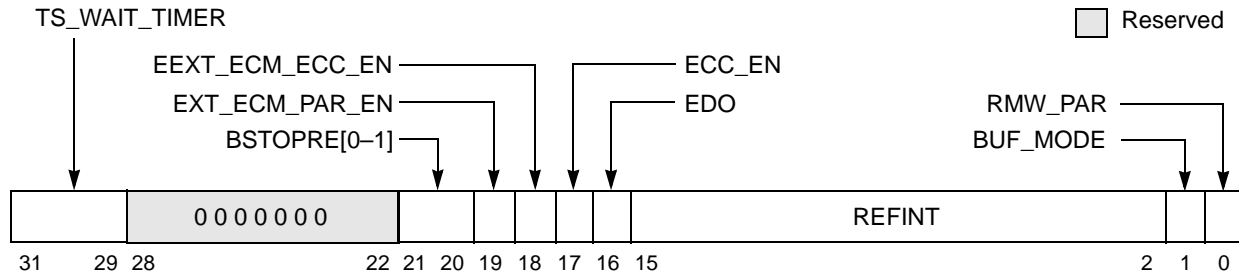


Figure 3-36. Memory Control Configuration Register 2 (MCCR2)—0xF4

Table 3-35. Bit Settings for MCCR2 —0xF4

Bit	Name	Reset Value	Description
31–29	TS_WAIT_TIMER	000	TS_WAIT_TIMER[0–2] adds a delay to the start of a new memory transaction after a ROM/Flash access. This allows the designer to compensate for slow ROM output disable timing. The minimum time allowed for ROM devices to enter high impedance is 2 clocks. TS_WAIT_TIMER adds (n–1) clocks to the minimum disable time. It has no effect on configurations that do not use buffers; therefore, putting a slow ROM directly on the 60x bus at higher clock frequencies is not recommended. Negative additions (that is, setting TS_WAIT_TIMER[0–2] = 1) do not reduce the minimum enforced disable times. This delay is enforced after all ROM and Flash accesses preventing any other memory access from starting (for example DRAM after ROM access, SDRAM after Flash access, ROM after Flash access, etc.). Note that TS_WAIT_TIMER[0–2] must be 0b000 for caching-allowed 8-bit ROM space. 000 2 clocks 001 2 clocks 010 3 clocks 011 4 clocks 100 5 clocks 101 6 clocks 110 7 clocks 111 8 clocks
28–22	—	All 0s	These bits are reserved.
21–20	BSTOPRE[0–1]	00	Burst to precharge—bits 0–1. For SDRAM only. These bits, together with BSTOPRE[2–5] (bits 31–28 of MCCR3), and BSTOPRE[6–9] (bits 3–0 of MCCR4), control the open page interval. The page open duration counter is reloaded with BSTOPRE[0–9] every time the page is accessed (including page hits). When the counter expires, the open page is closed with an SDRAM-precharge bank command. See Section 6.5.4, “SDRAM Page Mode Retention,” for more information.

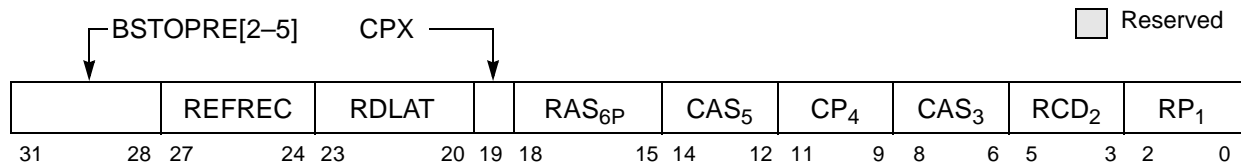
Table 3-35. Bit Settings for MCCR2 —0xF4 (continued)

Bit	Name	Reset Value	Description
19	EXT_ECM_PAR_EN	0	External error checking module parity enable. This bit controls whether the MPC106 uses an external error checking module to generate, check, and report parity errors. Note that EXT_ECM_EN (bits 23–22 of MCCR4) must be set to 0b11 to enable the external error checking module. 0 External module parity checking disabled 1 External module parity checking enabled
18	EXT_ECM_ECC_EN	0	External error checking module ECC enable. This bit controls whether the MPC106 uses an external error checking module to generate, check, correct, and report ECC. Note that EXT_ECM_EN (bits 23–22 of MCCR4) must be set to 0b11 to enable the external error checking module. 0 External module ECC disabled 1 External module ECC enabled
17	ECC_EN	0	ECC enable. This bit controls whether the MPC106 uses ECC for transactions to system memory. Note that the ECC_EN parameter overrides the PCKEN parameter. Also note that this bit and RMW_PAR cannot both be set to 1. This bit only applies to EDO and DRAM memory and not to SDRAM memory. See Section 6.4, “DRAM/EDO Interface Operation,” for more information. 0 ECC disabled 1 ECC enabled
16	EDO	0	EDO enable. This bit indicates the type of DRAMs for the MPC106 memory interface. See Section 6.4, “DRAM/EDO Interface Operation,” for more information. 0 Indicates standard DRAMs 1 Indicates EDO DRAMs
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 MPC106. See Section 6.4.10, “DRAM/EDO Refresh,” or Section 6.5.10, “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.

Table 3-35. Bit Settings for MCCR2 —0xF4 (continued)

Bit	Name	Reset Value	Description
1	BUF_MODE	1	Buffer mode. This bit controls how $\overline{BCTL0}$ and $\overline{BCTL1}$ operate. See Section 6.2, “Memory Interface Signal Buffering,” for more information. 0 $\overline{BCTL0}$ enables the buffer for write operations; $\overline{BCTL1}$ enables the buffer for read operations. 1 $\overline{BCTL0}$ controls the buffer direction (R/ \overline{W}); $\overline{BCTL1}$ acts as buffer enable.
0	RMW_PAR	0	Read-modify-write (RMW) parity enable. This bit controls how the MPC106 writes parity bits to DRAM/EDO/SDRAM. Note that this bit does not enable parity checking and generation. PCKEN must be set to enable parity checking. Also note that this bit and ECC_EN cannot both be set to 1. See Section 6.4.8, “DRAM/EDO Parity and RMW Parity,” or Section 6.5.8, “SDRAM Parity and RMW Parity,” for more information. 0 RMW parity disabled 1 RMW parity enabled

See Figure 3-37 and Table 3-36 for memory control configuration register 3 (MCCR3) bit settings.


Figure 3-37. Memory Control Configuration Register 3 (MCCR3)—0xF8
Table 3-36. Bit Settings for MCCR3—0xF8

Bit	Name	Reset Value	Description
31–28	BSTOPRE_U	0000	Burst to precharge—bits 2–5. For SDRAM only. These bits, together with BSTOPRE[0–1] (bits 21–20 of MCCR2), and BSTOPRE[6–9] (bits 3–0 of MCCR4), control the open page interval. The page open duration counter is reloaded with BSTOPRE[0–9] every time the page is accessed (including page hits). When the counter expires, the open page is closed with an SDRAM-precharge bank command. See Section 6.5.4, “SDRAM Page Mode Retention,” for more information.
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 6.5.10, “SDRAM Refresh,” for more information. 0001 1 clock 0010 2 clocks 0011 3 clocks 1111 15 clocks 0000 16 clocks

Table 3-36. Bit Settings for MCCR3—0xF8 (continued)

Bit	Name	Reset Value	Description
23–20	RDLAT	0000	<p>Data latency from read command. For SDRAM only. These bits control the number of clock cycles from an SDRAM-read command until the first data beat is available on the 60x data bus. RDLAT values greater than 5 clocks are not supported. See Section 6.5.5, “SDRAM Power-On Initialization,” for more information.</p> <p>0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 Reserved (not supported) 1111 Reserved (not supported)</p>
19	CPX	0	<p>$\overline{\text{CAS}}$ write timing modifier. For DRAM/EDO only. This bit, when set, adds one clock cycle to the $\overline{\text{CAS}}$ precharge interval ($\text{CP}_4 + 1$) and subtracts one clock cycle from the $\overline{\text{CAS}}$ assertion interval ($\text{CAS}_5 - 1$) for write operations to DRAM/EDO. Read operations are unmodified. See Section 6.4.4, “DRAM/EDO Interface Timing,” for more information.</p> <p>0 $\overline{\text{CAS}}$ write timing is unmodified 1 $\overline{\text{CAS}}$ write timing is modified as described above</p>
18–15	RAS_{6P}	0000	<p>$\overline{\text{RAS}}$ assertion interval for CBR refresh. For DRAM/EDO only. These bits control the number of clock cycles $\overline{\text{RAS}}$ is held asserted during CBR refresh. The value for RAS_{6P} depends on the specific DRAMs used and the 60x bus frequency. See Section 6.4.10, “DRAM/EDO Refresh,” for more information.</p> <p>0001 1 clock 0010 2 clocks 0011 3 clocks 1111 15 clocks 0000 16 clocks</p>
14–12	CAS_5	000	<p>$\overline{\text{CAS}}$ assertion interval for page mode access. For DRAM/EDO only. These bits control the number of clock cycles $\overline{\text{CAS}}$ is held asserted during page mode accesses. The value for CAS_5 depends on the specific DRAMs used and the 60x bus frequency. Note that when ECC is enabled, $\text{CAS}_5 + \text{CP}_4$ must equal four clock cycles. See Section 6.4.4, “DRAM/EDO Interface Timing,” for more information.</p> <p>001 1 clock 010 2 clocks 011 3 clocks 111 7 clocks 000 8 clocks</p>

Table 3-36. Bit Settings for MCCR3—0xF8 (continued)

Bit	Name	Reset Value	Description
11–9	CP ₄	000	<p>$\overline{\text{CAS}}$ precharge interval. For DRAM/EDO only. These bits control the number of clock cycles that $\overline{\text{CAS}}$ must be held negated in page mode (to allow for column precharge) before the next assertion of $\overline{\text{CAS}}$. Note that EDO DRAMs (MCCR2[EDO] = 1) must use CP₄ = 0b001. Also note that when ECC is enabled, CAS₅ + CP₄ must equal four clock cycles. See Section 6.4.4, “DRAM/EDO Interface Timing,” for more information.</p> <p>001 1 clock 010 2 clocks 011 3 clocks 111 7 clocks 000 8 clocks</p>
8–6	CAS ₃	000	<p>$\overline{\text{CAS}}$ assertion interval for the first access. For DRAM/EDO only. These bits control the number of clock cycles $\overline{\text{CAS}}$ is held asserted during a single beat or during the first access in a burst. The value for CAS₃ depends on the specific DRAMs used and the 60x bus frequency. See Section 6.4.4, “DRAM/EDO Interface Timing,” for more information.</p> <p>001 1 clock 010 2 clocks 011 3 clocks 111 7 clocks 000 8 clocks</p>
5–3	RCD ₂	000	<p>RAS to $\overline{\text{CAS}}$ delay interval. For DRAM/EDO only. These bits control the number of clock cycles between the assertion of RAS and the first assertion of $\overline{\text{CAS}}$. The value for RCD₂ depends on the specific DRAMs used and the 60x bus frequency. However, RCD₂ must be at least two clock cycles. See Section 6.4.4, “DRAM/EDO Interface Timing,” for more information.</p> <p>001 Reserved 010 2 clocks 011 3 clocks 111 7 clocks 000 8 clocks</p>
2–0	RP ₁	000	<p>$\overline{\text{RAS}}$ precharge interval. For DRAM/EDO only. These bits control the number of clock cycles that RAS must be held negated (to allow for row precharge) before the next assertion of RAS. Note that RP₁ must be at least two clock cycles. See Section 6.4.4, “DRAM/EDO Interface Timing,” for more information.</p> <p>001 Reserved 010 2 clocks 011 3 clocks 111 7 clocks 000 8 clocks</p>

See Figure 3-38 and Table 3-37 for memory control configuration register 4 (MCCR4) bit settings.

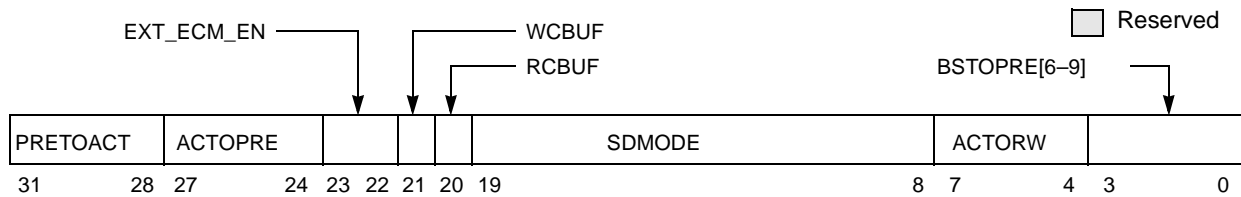


Figure 3-38. Memory Control Configuration Register 4 (MCCR4)—0xFC

Table 3-37. Bit Settings for MCCR4—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. Note that PRETOACT must be at least two clock cycles. See Section 6.5.5, “SDRAM Power-On Initialization,” for more information. 0001 Reserved 0010 2 clocks 0011 3 clocks 1111 15 clocks 0000 16 clocks
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 6.5.5, “SDRAM Power-On Initialization,” for more information. 0001 1 clock 0010 2 clocks 0011 3 clocks 1111 15 clocks 0000 16 clocks
23–22	EXT_ECM_EN	00	External error checking module enable. These bits enable the external error checking module interface of the MPC106. 00 External error checking module disabled 01 Reserved 10 Reserved 11 External error checking module enabled Note that both WCBUF and RCBUF must also be set to 1s to enable the external error checking module. Note also that EXT_ECM_PAR_EN (bit 19 of MCCR2) and EXT_ECM_ECC_EN (bit 18 of MCCR2) should be used to enable parity and ECC checking respectively when using an external ECM. PCKEN and ECC_EN should be cleared to 0 when using an external ECM.

Table 3-37. Bit Settings for MCCR4—0xFC (continued)

Bit	Name	Reset Value	Description
21	WCBUF	0	Memory write buffer type. This bit configures the MPC106 for one of two buffer types, and controls the timing and operation of the buffer control signals (BCTL0 and BCTL1). See Section 6.2, “Memory Interface Signal Buffering,” for more information. 0 Flow through or transparent latch type buffer. Also, registered buffer for DRAM/EDO. 1 Registered type buffer for SDRAM
20	RCBUF	1	Memory read buffer type. This bit configures the MPC106 for one of two buffer types, and controls the timing and operation of the buffer control signals (BCTL0 and BCTL1). 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. Note that for 64- and 128-Mbit SDRAMs, the SDMODE is actually a 14-bit field. The 2 most significant bits are forced to 0 and concatenated to the SDMODE bits in this register. 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 3-37. Bit Settings for MCCR4—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 or SDRAM-write command is allowed. See Section 6.5.5, “SDRAM Power-On Initialization,” for more information. 0001 1 clock 0010 2 clocks 0011 3 clocks 1111 15 clocks 0000 16 clocks
3–0	BSTOPRE[6–9]	0000	Burst to precharge—bits 6–9. For SDRAM only. These bits, together with BSTOPRE[0–1] (bits 21–20 of MCCR2) and BSTOPRE[2–5] (bits 31–28 of MCCR3) control the open page interval. The page open duration counter is reloaded with BSTOPRE[0–9] every time the page is accessed (including page hits). When the counter expires, the open page is closed with an SDRAM-precharge bank command. See Section 6.5.4, “SDRAM Page Mode Retention,” for more information.

3.2.9 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 3-39 and Table 3-38 for PICR1 bit settings.

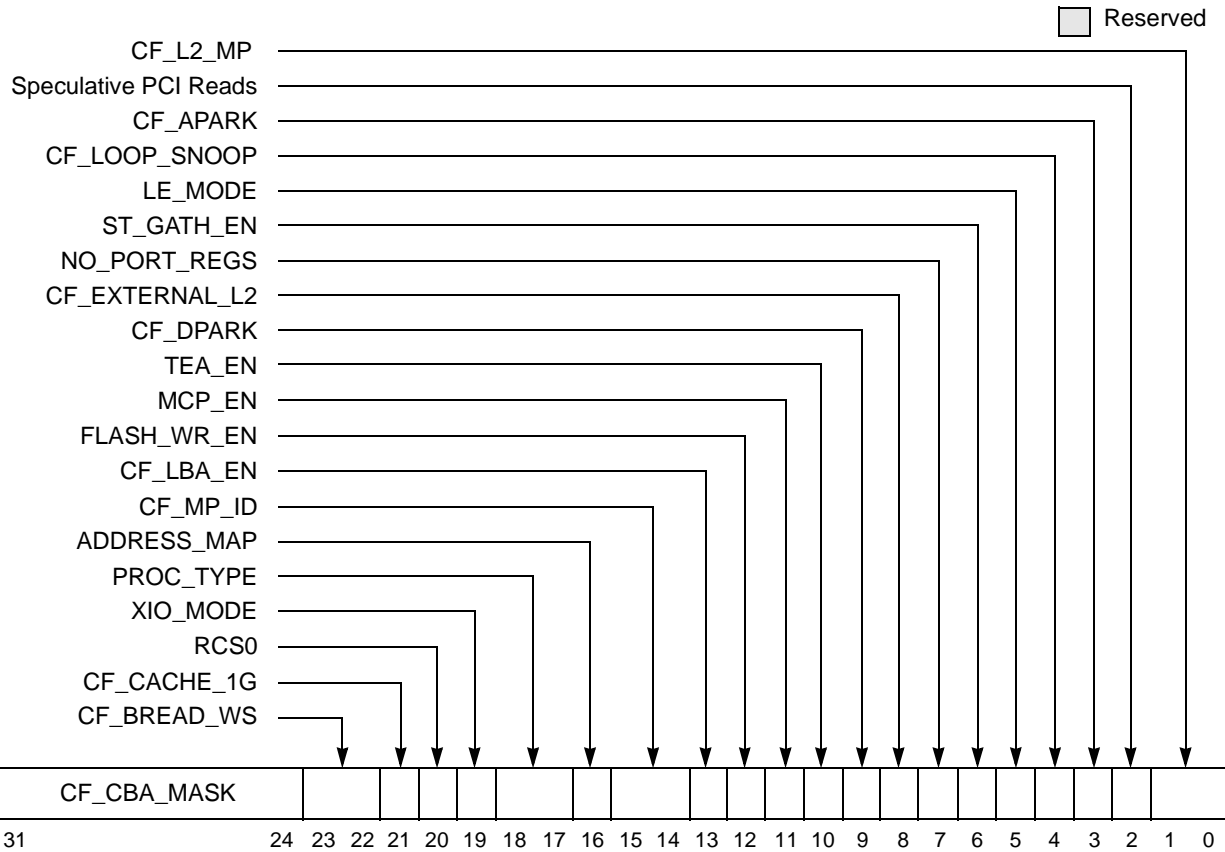


Figure 3-39. Processor Interface Configuration Register 1 (PICR1)—0xA8

Table 3-38. Bit Settings for PICR1—0xA8

Bit	Name	Reset Value	Description
31–24	CF_CBA_MASK	All 1s	L2 copyback address mask. The MPC106 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 (0b0), the corresponding address bit read from the tag RAM will be treated as 0b0 by the MPC106 (it is masked internally), regardless of its actual state. If a bit in CF_CBA_MASK is set (0b1), 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 minimum number of wait states from \overline{TS} to the first \overline{TA} for burst reads. 00 0 wait states (601, 603 with 2:1 or greater clock ratio, and 604) 01 1 wait state (603 with 1:1 clock ratio in \overline{DRTRY} mode) 10 2 wait states (603 with 1:1 clock ratio in no- \overline{DRTRY} mode) 11 3 wait states (not recommended)

Table 3-38. Bit Settings for PICR1—0xA8 (continued)

Bit	Name	Reset Value	Description
21	CF_CACHE_1G	0	L2 cache 0–1 Gbyte only. This bit controls whether the internally-controlled L2 cache caches addresses from 0 to 1 Gbytes or from 0 to 2 Gbytes and the ROM address space. 0 The internally-controlled L2 may cache addresses from 0 to 2 Gbytes and ROM addresses. 1 The internally-controlled L2 may only cache addresses from 0 to 1 Gbyte. No check for hit or miss is performed for addresses from 1 to 2 Gbytes or for ROM addresses.
20	RCS0	x	ROM Location. Read only. This bit indicates the state of the ROM location (RCS0) configuration signal at power-on reset. 0 ROM is located on PCI bus 1 ROM is located on 60x/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/O 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 MPC106 uses PROC_TYPE to control ARTRY timing (due to differences between the processors), and the power saving modes (for the 603 or 604). Note that the MPC106 does not support multiprocessing systems with 601 processors. 00 601 01 Reserved 10 603/740/750 11 604
16	ADDRESS_MAP	x	Address map. This bit controls which address map is used by the MPC106. The initial state is determined by the state of the address map (DBG0) configuration signal at power-on reset. Note that software that dynamically changes this bit must ensure that there are no pending PCI transactions and that there is a sync instruction following the address map change to allow the update to take effect. See Section 3.1, “Address Maps,” for more information. 0 The MPC106 is configured for address map B. 1 The MPC106 is configured for address map A.
15–14	CF_MP_ID	00	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. 00 Processor 0 is reading PICR1[CF_MP_ID]. 01 Processor 1 is reading PICR1[CF_MP_ID]. 10 Processor 2 is reading PICR1[CF_MP_ID]. 11 Processor 3 is reading PICR1[CF_MP_ID].

Table 3-38. Bit Settings for PICR1—0xA8 (continued)

Bit	Name	Reset Value	Description
13	CF_LBA_EN	0	Local bus slave access enable. This bit controls whether the MPC106 responds to the $\overline{\text{LBCLAIM}}$ signal (and therefore local bus slave accesses). When this bit is cleared, the MPC106 ignores the $\overline{\text{LBCLAIM}}$ signal. Note that local bus slaves must not claim any transaction in the range 0x8000_0000–0xFFFF_FFFF. See Section 4.4.5, “60x Local Bus Slave Support,” for more information. 0 Local bus slave access is disabled. 1 Local bus slave access is enabled.
12	FLASH_WR_EN	0	Flash write enable. This bit controls whether the MPC106 allows write operations to Flash ROM. 0 Flash write is disabled. 1 Flash write is enabled.
11	MCP_EN	0	Machine check enable. This bit controls whether the MPC106 asserts $\overline{\text{MCP}}$ upon detecting an error. See Chapter 9, “Error Handling,” for more information. 0 Machine check is disabled 1 Machine check is enabled
10	TEA_EN	0	Transfer error enable. This bit controls whether the MPC106 asserts $\overline{\text{TEA}}$ upon detecting an error. See Chapter 9, “Error Handling,” for more information. 0 Transfer error is disabled 1 Transfer error is 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	CF_EXTERNAL_L2	0	External L2 cache enable. This bit, in conjunction with CF_L2_MP, indicate the processor and L2 configuration of the system. See Table 3-39 for the specific bit encodings. See Section 5.6.2, “External L2 Cache Controller Interface Parameters,” for more information. 0 External L2 disabled. 1 External L2 enabled.
7	NO_PORT_REGS	0	When configured for address map A, this bit indicates the presence or absence of the external configuration registers. See Section 3.2.12, “External Configuration Registers,” for more information. 0 The system implements the external configuration registers. The MPC106 treats accesses to the external registers as PCI I/O cycles. 1 There are no physical registers for the external configuration registers. The MPC106 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 is disabled 1 Store gathering is enabled

Table 3-38. Bit Settings for PICR1—0xA8 (continued)

Bit	Name	Reset Value	Description
5	LE_MODE	0	This bit controls the endian mode of the MPC106. 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
4	CF_LOOP_SNOOP	1	This bit causes the MPC106 to repeat a snoop operation (due to a PCI-to-memory transaction) until it is not retried (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 (HP_SNP_REQ) signal on the 601. 0 Snoop looping is disabled 1 Snoop looping is 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 the last processor that used the 60x address bus is parked on the 60x address bus
2	Speculative PCI Reads	0	This bit controls speculative PCI reads from memory. Note that the MPC106 performs a speculative read in response to a PCI read-multiple command, even if this bit is cleared. 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, in conjunction with CF_EXTERNAL_L2, indicate the processor and L2 configuration of the system. See Table 3-39 for the specific bit encodings. Note that the MPC106 does not support multiprocessing systems with 601 processors. See Section 5.4, “L2 Cache Interface Parameters,” for more information.

Table 3-39 shows the processor/L2 configuration encodings for the CF_EXTERNAL_L2 and CF_L2_MP parameters in PICR1.

Table 3-39. Processor/L2 Configurations

CF_EXTERNAL_L2	CF_L2_MP	Configuration
0	00	Uniprocessor without L2 cache
0	01	Uniprocessor with internally-controlled, write-through L2 cache
0	10	Uniprocessor with internally-controlled, write-back L2 cache
0	11	Multiprocessor (2–4 60x processors on the 60x bus) without L2 cache
1	00	Uniprocessor with externally-controlled L2 cache
1	01	Reserved
1	10	Reserved
1	11	Multiprocessor with externally-controlled L2 cache

See Figure 3-40 and Table 3-40 for processor interface configuration register 2 (PICR2) bit settings.

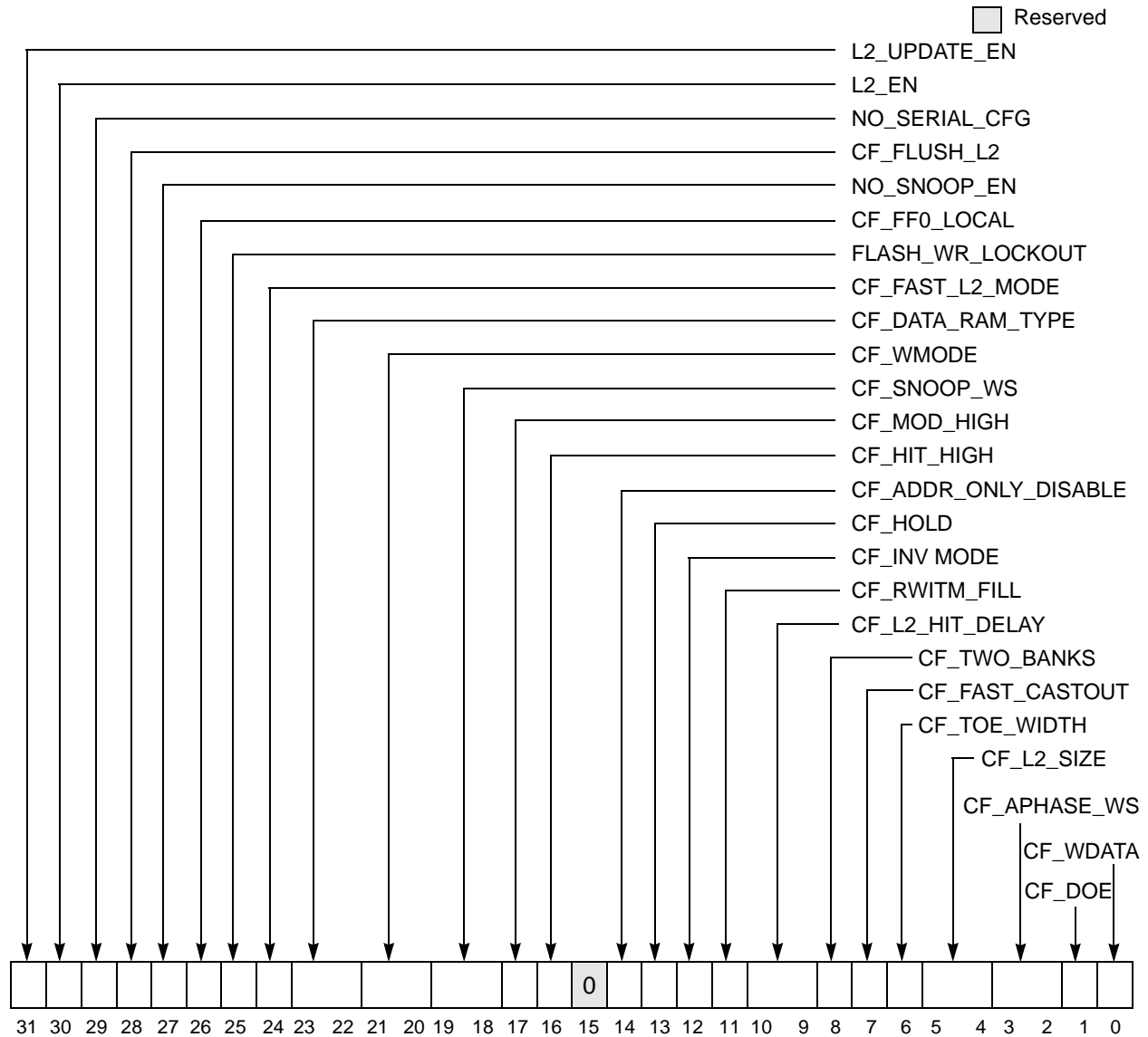


Figure 3-40. Processor Interface Configuration Register 2 (PICR2)—0xA

Table 3-40. Bit Settings for PICR2—0xAC

Bit	Name	Reset Value	Description
31	L2_UPDATE_EN	0	<p>L2 update enable. This bit controls whether the internally-controlled L2 cache can be updated with new data. Note that this bit has no effect on the external L2 cache controller operation. Also, note that L2_UPDATE_EN is accessible at port 0x81C.</p> <p>0 The L2 cache can only be read or invalidated. The L2 cache cannot be updated. Snoops are serviced to maintain coherency.</p> <p>1 The L2 cache can be updated with new data.</p>
30	L2_EN	0	<p>This bit enables/disables the internally-controlled L2 cache. The L2 cache is only enabled if both this bit and CF_L2_MP signify that there is an internally-controlled L2 cache in the system. Note that this bit has no effect on the external L2 cache controller operation. Also, note that L2_EN is accessible at port 0x81C.</p> <p>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.</p> <p>1 The L2 cache is enabled.</p>
29	NO_SERIAL_CFG	0	<p>This bit controls whether the MPC106 serializes configuration writes to PCI devices from the 60x bus.</p> <p>0 Configuration writes to PCI devices from the 60x bus cause the MPC106 to serialize and flush the internal buffers.</p> <p>1 Configuration writes to PCI devices from the 60x bus do not cause serialization. The internal buffers are not flushed.</p>
28	CF_FLUSH_L2	0	<p>L2 cache flush. The transition on this bit from 0 to 1 initiates an internally-controlled L2 cache flush and invalidate operation, provided L2_EN = 0b1. Note that this bit has no effect on the external L2 cache controller operation. Also, note that CF_FLUSH_L2 is accessible at port 0x81C.</p> <p>0 Normal cache operation.</p> <p>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.</p>
27	NO_SNOOP_EN	0	<p>This bit controls whether the MPC106 generates snoop transactions on the 60x bus for PCI-to-system-memory transactions. This is provided as a performance enhancement for systems that do not need to maintain coherency on system memory accesses by PCI.</p> <p>0 Snooping is enabled.</p> <p>1 Snooping is disabled.</p>

Table 3-40. Bit Settings for PICR2—0xAC (continued)

Bit	Name	Reset Value	Description
26	CF_FF0_LOCAL	0	<p>ROM remapping enable. This bit allows the lower 8 Mbytes of the ROM/Flash address range to be remapped from the PCI bus to the 60x/memory bus. Note that this bit is meaningful only if the ROM location parameter indicates that ROM is located on PCI bus (PICR1[RCS0] = 0).</p> <p>0 ROM/Flash remapping disabled. The lower 8 Mbytes of the ROM/Flash address space are not remapped. All ROM/Flash accesses are directed to the PCI bus.</p> <p>1 ROM/Flash remapping enabled. The lower 8 Mbytes of the ROM/Flash address space are remapped to the 60x/memory bus. ROM/Flash accesses in the range 0xFF00_0000–0xFF7F_FFFF are directed to the 60x/memory bus. ROM/Flash accesses in the range 0xFF80_0000–0xFFFF_FFFF are directed to the PCI bus.</p>
25	FLASH_WR_LOCKOUT	0	<p>Flash write lock-out. This bit, once set, prevents writing to Flash. Once set, this bit can only be cleared by a hard reset.</p> <p>0 Write operations to Flash are enabled, provided FLASH_WR_EN = 1.</p> <p>1 Write operations to Flash are disabled until the MPC106 is reset.</p>
24	CF_FAST_L2_MODE	0	<p>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.</p> <p>0 Disable fast L2 mode timing</p> <p>1 Enable fast L2 mode timing</p>
23–22	CF_DATA_RAM_TYPE	00	<p>L2 data RAM type. These bits indicate the type of data RAM used for the L2 cache.</p> <p>00 Synchronous burst SRAM</p> <p>01 Pipelined burst SRAM</p> <p>10 Asynchronous SRAM</p> <p>11 Reserved</p>
21–20	CF_WMODE	00	<p>SRAM write timing and partial update disable. These bits control L2 data RAM write timing. For an asynchronous SRAM cache configuration, only mode 00 is valid. See Section 5.4.2.4, “CF_WMODE,” for more information.</p> <p>00 Normal write timing without partial update.</p> <p>01 Normal write timing with partial update using external byte write decoding. Not valid for asynchronous SRAMs.</p> <p>10 Delayed write timing with partial update using external byte write decoding. When performing an L2 cache write, the MPC106 issues the L2 cache control signals, but delays the assertion of TA by one cycle to allow for external byte write decoding. Not valid for asynchronous SRAMs.</p> <p>11 Early write timing with partial update using external byte write decoding. The MPC106 speculatively asserts DWE one cycle earlier than the other L2 data RAM control signals for better write performance. Not valid for asynchronous SRAMs.</p>

Table 3-40. Bit Settings for PICR2—0xAC (continued)

Bit	Name	Reset Value	Description
19–18	CF_SNOOP_WS	11	<p>Snoop wait states. These bits control the minimum number of wait states for the address phase in a snoop cycle. See Section 4.3.3.2, “Address Tenure Timing Configuration,” for more information.</p> <p>00 0 wait states (2-clock address phase) 01 1 wait state (3-clock address phase) 10 2 wait states (4-clock address phase) 11 3 wait states (5-clock address phase)</p>
17	CF_MOD_HIGH	0	<p>Cache-modified signal polarity. This bit controls the active state of the $\overline{\text{DIRTY_IN}}$, $\overline{\text{DIRTY_OUT}}$, and TV L2 cache signals.</p> <p>0 The input signals $\overline{\text{TV}}$ and $\overline{\text{DIRTY_IN}}$ are active low and the output signals $\overline{\text{TV}}$ and $\overline{\text{DIRTY_OUT}}$ are active low. 1 The input signals TV and DIRTY_IN are active high and the output signals TV and DIRTY_OUT are active high.</p>
16	CF_HIT_HIGH	0	<p>L2 cache $\overline{\text{HIT}}$ signal polarity. This bit controls the active state of the $\overline{\text{HIT}}$ signal for the internally-controlled L2 cache. Note that this bit has no effect on the external L2 cache controller operation. $\overline{\text{HIT}}$ is always active low for the external L2 cache controller interface.</p> <p>0 $\overline{\text{HIT}}$ is active low. 1 HIT is active high.</p>
15	—	0	This bit is reserved.
14	CF_ADDR_ONLY_DISABLE	0	<p>This bit specifies whether the internally-controlled L2 cache responds to address-only transactions (clean, flush, and kill). This bit is set when the L2 is enabled for normal L2 operation. Note that this bit has no effect on the external L2 cache controller operation.</p> <p>0 The internally-controlled L2 cache responds to clean, flush, and kill transactions. 1 The internally-controlled L2 cache ignores clean, flush, and kill transactions.</p>
13	CF_HOLD	0	<p>L2 tag address hold. This bit controls the hold time of the address, TV, and $\overline{\text{DIRTY_OUT}}$ signals with respect to the rising edge (negation) of $\overline{\text{TWE}}$.</p> <p>0 Synchronous tag RAM configurations. No hold time (0 clock cycles). TV is always driven during tag reads. 1 Asynchronous tag RAM configurations. Tag address, TV, and $\overline{\text{DIRTY_OUT}}$ are held valid for one clock cycle after $\overline{\text{TWE}}$ is negated. TV is released to high-impedance during tag reads.</p>
12	CF_INV_MODE	0	<p>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. Note that this bit has no effect on the external L2 cache controller operation. See Section 5.2.2, “L2 Cache Line Status,” for more information.</p> <p>0 L2 invalidate mode is disabled. 1 L2 invalidate mode is enabled.</p>

Table 3-40. Bit Settings for PICR2—0xAC (continued)

Bit	Name	Reset Value	Description
11	CF_RWITM_FILL	0	<p>L2 read-with-intent-to-modify line-fill disable. This bit controls the response of the internally-controlled L2 cache to read-with-intent-to-modify (RWITM) misses. Note that this bit has no effect on the external L2 cache controller operation. See Section 5.3, “L2 Cache Response to Bus Operations,” for more information.</p> <p>0 The internally-controlled L2 cache performs a line-fill when a RWITM miss occurs. 1 The internally-controlled L2 cache does not perform a line-fill when a RWITM miss occurs.</p>
10–9	CF_L2_HIT_DELAY	11	<p>L2 cache hit delay. These bits control the number of clock cycles from the assertion of TS until HIT is valid. See Section 5.4.2.1, “CF_L2_HIT_DELAY,” for more information.</p> <p>00 Reserved 01 1 clock cycle 10 2 clock cycles 11 3 clock cycles</p>
8	CF_TWO_BANKS	0	<p>L2 cache banks. This bit specifies the number of banks of L2 data RAM. See Section 5.1.6, “Two-Bank Support,” for more information.</p> <p>0 1 SRAM bank 1 2 SRAM banks</p>
7	CF_FAST_CASTOUT	0	<p>Fast L2 castout timing</p> <p>0 Normal L2 castout timing. TV is released to high-impedance during tag reads. 1 Fast L2 castout timing for improved performance when using synchronous write tag RAMs. TV is always driven during tag reads.</p>
6	CF_TOE_WIDTH	0	<p>TOE active pulse width. This bit controls the number of clock cycles that TOE is held asserted during L2 tag castout/copyback operations.</p> <p>0 2 clock cycles 1 3 clock cycles</p>
5–4	CF_L2_SIZE	00	<p>L2 cache size. These bits indicate the size of the L2 cache.</p> <p>00 256 Kbytes 01 512 Kbytes 10 1 Mbyte 11 Reserved</p>
3–2	CF_APHASE_WS	11	<p>Address phase wait states. These bits control the minimum number of address phase wait states (in clock cycles) for processor-initiated operations.</p> <p>00 0 wait states 01 1 wait state 10 2 wait states 11 3 wait states</p>

Table 3-40. Bit Settings for PICR2—0xAC (continued)

Bit	Name	Reset Value	Description
1	CF_DOE	0	L2 first data read access timing. For synchronous burst SRAM configurations, this bit controls the number of clock cycles from \overline{DOE} assertion to valid data on the first read access. Note that this bit has no effect on the external L2 cache controller operation. See Section 5.4.2.2, "CF_DOE," for more information. 0 1 clock cycle 1 2 clock cycles For asynchronous SRAM 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)
0	CF_WDATA	0	This bit has different functions depending on the L2 data RAM configuration. See Section 5.4.2.3, "CF_WDATA," for more information. For synchronous burst SRAM configurations, this bit is reserved and must be cleared to 0. For pipelined burst SRAMs, this bit indicates \overline{ADSC} -only or \overline{ADSP} mode. See Section 5.1.4, "Pipelined Burst SRAMs," for more information. 0 \overline{ADSC} -only mode 1 \overline{ADSP} mode (\overline{TS} is connected to \overline{ADSP} on the L2 data RAM) For asynchronous SRAMs, this bit indicates the $\overline{DWE}n$ timing: 0 $\overline{DWE}n$ is negated on the falling clock edge of the cycle when \overline{TA} is asserted. 1 $\overline{DWE}n$ is negated on the rising clock edge when \overline{TA} is negated.

3.2.10 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 3-41 and Table 3-41 for alternate OS-visible parameters register 1 bit settings.

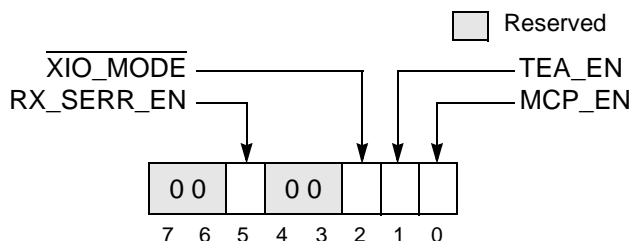
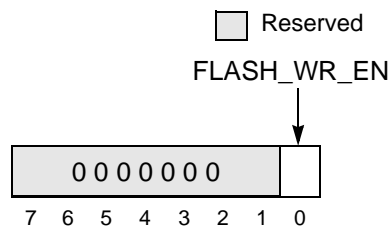


Figure 3-41. Alternate OS-Visible Parameters Register 1—0xBA

Table 3-41. 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 MPC106 recognizes the assertion of $\overline{\text{SERR}}$ by another PCI device. 0 The MPC106 ignores the assertion of $\overline{\text{SERR}}$ by another PCI device. 1 The MPC106 recognizes the assertion of $\overline{\text{SERR}}$ by another PCI device.
4–3	—	00	These bits are reserved.
2	$\overline{\text{XIO_MODE}}$	1	Address map A discontinuous/contiguous mode. This bit controls whether address map A uses the discontinuous or contiguous I/O 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 Discontinuous mode 1 Contiguous mode
1	TEA_EN	0	Transfer error acknowledge enable. This bit controls whether the MPC106 asserts $\overline{\text{TEA}}$ upon detecting an error. Note that this bit is the same as bit 10 of PICR1. 0 Transfer error acknowledge is disabled. 1 Transfer error acknowledge is enabled.
0	MCP_EN	0	Machine check enable. This bit controls whether the MPC106 asserts MCP upon detecting an error. Note that this bit is the same as bit 11 of PICR1. 0 Machine check is disabled. 1 Machine check is enabled.

See Figure 3-42 and Table 3-42 for alternate OS-visible parameters register 2 bit settings.


Figure 3-42. Alternate OS-Visible Parameters Register 2—0xBB
Table 3-42. Bit Settings for Alternate OS-Visible Parameters Register 2—0xBB

Bit	Name	Reset Value	Description
7–1	—	00	These bits are reserved.
0	FLASH_WR_EN	0	Flash write enable. This bit controls whether the MPC106 allows write operations to Flash ROM. Note that this bit is the same as bit 12 of PICR1. 0 Flash writes are disabled. 1 Flash writes are enabled.

3.2.11 Emulation Support Configuration Registers

The 32-bit emulation support configuration registers (ESCRs) control the behavior of the MPC106 when operating in emulation mode. The emulation mode is fully compliant with the PC emulation option described in Section 7.8, “Emulation Support.”

See Figure 3-43 and Table 3-43 for emulation support configuration register 1 (ESCR1) bit settings.

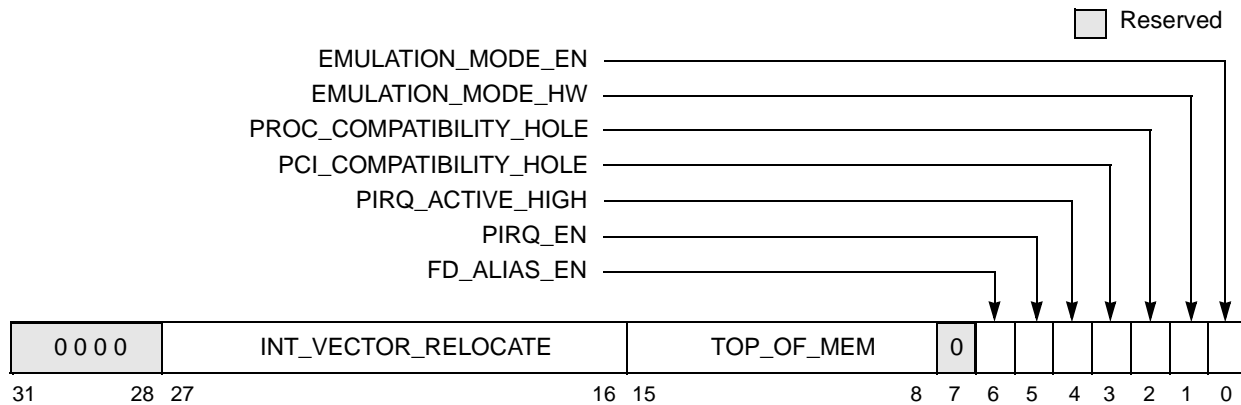


Figure 3-43. Emulation Support Configuration Register 1 (ESCR1)—0xE0

Table 3-43. Bit Settings for ESCR1—0xE0

Bit	Name	Reset Value	Description
31–28	—	All 0s	These bits are reserved.
27–16	INT_VECTOR_RELOCATE	All 1s	These bits represent the 1-Mbyte block of system memory that is accessed when emulation mode is enabled (EMULATION_MODE_EN = 1) and a 60x transaction to the address range 0xFFFF0_0000–0xFFFF_FFFF occurs. The 12 most-significant bits of the address (0xFFF) are replaced by these bits, and the 20 least-significant bits are left unchanged. These bits have no effect if emulation mode is disabled (EMULATION_MODE_EN = 0).
15–8	TOP_OF_MEM	All 0s	These bits represent the block address of a 1-Mbyte block that is the upper address boundary to which the MPC106, as a PCI target, will respond. These bits have no effect if emulation mode is disabled (EMULATION_MODE_EN = 0).
7	—	0	This bit is reserved.
6	FD_ALIAS_EN	1	This bit is used in address map B only; it is not used for map A or the emulation mode map. 0 No response 1 The MPC106, as a PCI target, responds to addresses in the range 0xFD00_0000–0xFDFF_FFFF, and forwards the transaction to system memory as 0x0000_0000–0x00FF_FFFF

Table 3-43. Bit Settings for ESCR1—0xE0 (continued)

Bit	Name	Reset Value	Description
5	PIRQ_EN	0	When emulation mode is enabled, $\overline{\text{PIRQ}}$ will be asserted if a PCI-write-to-system-memory occurs when the modified memory status is 0b00. See Section 7.8, “Emulation Support,” for more information. 0 $\overline{\text{PIRQ}}$ is disabled. 1 $\overline{\text{PIRQ}}$ is enabled. $\overline{\text{PIRQ}}$ is asserted when a PCI write to memory occurs and MOD_MEM_STATUS is 0b00.
4	PIRQ_ACTIVE_HIGH	0	$\overline{\text{PIRQ}}$ signal polarity. This bit controls the active state of the $\overline{\text{PIRQ}}$ signal. 0 $\overline{\text{PIRQ}}$ is active low. 1 PIRQ is active high.
3	PCI_COMPATIBILITY_HOLE	0	This bit is used for address map B and the emulation mode map only; it is not used for address map A. 0 The MPC106, as a PCI target, responds to PCI addresses in the range 0x000A_0000–0x000F_FFFF, and forwards the transaction to system memory. 1 The MPC106, as a PCI target, does not respond to PCI addresses in the range 0x000A_0000–0x000F_FFFF.
2	PROC_COMPATIBILITY_HOLE	0	This bit is used for address map B and the emulation mode map only; it is not used for address map A. 0 The MPC106 forwards 60x processor-initiated transactions in the address range 0x000A_0000–0x000B_FFFF to system memory. 1 The MPC106 forwards 60x processor-initiated transactions in the address range 0x000A_0000–0x000B_FFFF to the PCI memory space.
1	EMULATION_MODE_HW	1	This bit is read-only and indicates that the MPC106 supports the emulation mode address map.
0	EMULATION_MODE_EN	0	Emulation mode address map enable. This bit, in conjunction with PICR1[DBG0], controls which address map is used by the MPC106. See Section 3.1.3, “Emulation Mode Address Map,” for more information. 0 Emulation mode address map disabled. The MPC106 is configured for address map A or address map B, depending on PICR1[ADDRESS_MAP]. 1 Emulation mode address map enabled. The MPC106 is configured for emulation mode address map.

Figure 3-44 and Table 3-44 describe the bit settings for emulation support configuration register 2 (ESCR2).

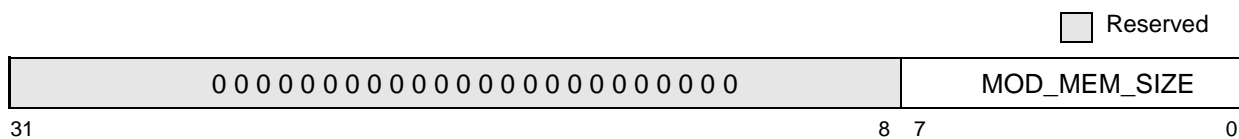


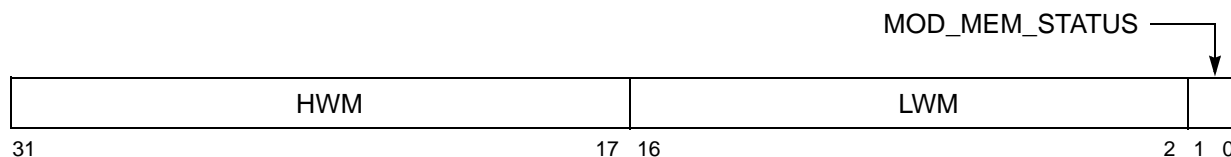
Figure 3-44. Emulation Support Configuration Register 2 (ESCR2)—0xE8

Table 3-44. Bit Settings for ESCR2—0xE8

Bit	Name	Reset Value	Description
31–8	—	All 0s	These bits are reserved.
7–0	MOD_MEM_SIZE	0x20	These bits configure the size of the modified memory regions in system memory for the emulation mode address map. Note that the MPC106 only supports sizes of 4 and 8 Kbytes. 0b1000_0000 16 Kbytes (not supported) 0b0100_0000 8 Kbytes 0b0010_0000 4 Kbytes 0b0001_0000 2 Kbytes (not supported) 0b0000_1000 1 Kbyte (not supported) 0b0000_0100 512 bytes (not supported) 0b0000_0010 256 bytes (not supported) 0b0000_0001 128 bytes (not supported)

3.2.11.1 Modified Memory Status Register

When the MPC106 is operating in emulation mode, the modified memory status register tracks the state of system memory that has been modified by PCI masters. See Section 7.8, “Emulation Support,” for more information. Figure 3-43 and Table 3-43 describe the bit settings for the modified memory status register.


Figure 3-45. Modified Memory Status Register—0xE4/0xEC
Table 3-45. Bit Settings for Modified Memory Status Register—0xE4/0xEC

Bit	Name	Reset Value	Description
31–17	HWM	All 0s	These bits indicate the address of a modified memory block or boundary depending on the value in MOD_MEM_STATUS. The block size is controlled by ESCR2[MOD_MEM_SIZE].
16–2	LWM	All 0s	These bits indicate the address of a modified memory block or boundary depending on the value in MOD_MEM_STATUS. The block size is controlled by ESCR2[MOD_MEM_SIZE].
1–0	MOD_MEM_STATUS	00	These bits encode the status of the HWM and LWM addresses. 00 Both HWM and LWM are invalid. 01 Both HWM and LWM are valid. HWM and LWM contain the same address of a block of memory that has been modified. 10 Both HWM and LWM are valid. HWM contains the address of a block of memory that has been modified, and LWM contains the address of another block of memory that has been modified. 11 Both HWM and LWM are valid. LWM contains the lower boundary and HWM contains the upper boundary of an area of memory that may have been modified.

3.2.12 External Configuration Registers

When using address map A, certain configuration bits can be accessed by reading or writing to the ports at addresses 0x8000_0092 (referred to as port 0x092), 0x8000_081C (referred to as port 0x81C), or 0x8000_0850 (referred to as port 0x850). 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, the MPC106 handles all accesses to the external configuration registers internally. If NO_PORT_REGS is cleared, the MPC106 treats accesses to the external configuration registers as PCI I/O accesses. 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 port 0x092 is set, then when the I/O write access completes on the PCI bus, the MPC106 enters little-endian mode and PICR1[LE_MODE] is set. Note that if external port registers are implemented as PCI I/O targets (that is, NO_PORT_REG = 0), then all writes to the external port registers must be performed with PICR2[NO_SERIAL_CFG] = 0.

See Figure 3-46 and Table 3-46 for external configuration register 1 bit settings.

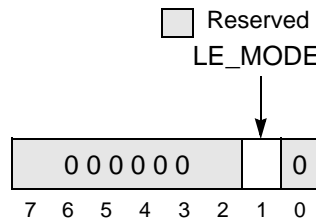


Figure 3-46. External Configuration Register 1—Port 0x092

Table 3-46. Bit Settings for External Configuration Register 1—Port 0x092

Bit	Name	Reset Value	Description
7–2	—	All 0s	These bits are reserved.
1	LE_MODE	0	This bit controls the endian mode of the MPC106. 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	This bit is reserved.

See Figure 3-47 and Table 3-44 for external configuration register 2 bit settings.

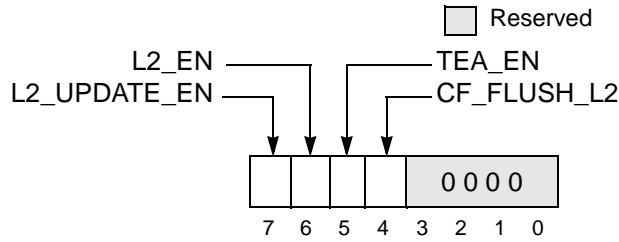


Figure 3-47. External Configuration Register 2—Port 0x81C

Table 3-47. Bit Settings for External Configuration Register 2—Port 0x81C

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 acknowledge enable. This bit controls whether the MPC106 will assert TEA upon detecting an error. Note that this bit corresponds to bit 10 of PICR1. 0 Transfer error acknowledge is disabled. 1 Transfer error acknowledge is enabled.
4	CF_FLUSH_L2	0	L2 cache flush. The transition on this bit from 0 to 1 initiates an internally-controlled L2 cache flush and invalidate operation, provided PICR2[L2_EN] = 0b1. 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	These bits are reserved.

See Figure 3-48 and Table 3-45 for external configuration register 3 bit settings.

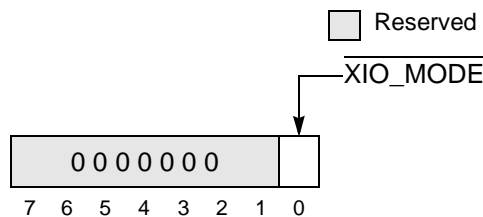


Figure 3-48. External Configuration Register 3—Port 0x850

Table 3-48. Bit Settings for External Configuration Register 3—Port 0x850

Bit	Name	Reset Value	Description
7-1	—	All 0s	These bits are reserved.
0	XIO_MODE	1	Address map A discontinuous/contiguous mode. This bit controls whether address map A uses the discontinuous or contiguous I/O 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 Discontinuous mode 1 Contiguous mode



Chapter 4

Processor Bus Interface

The MPC106 provides flexible support for system designs using the MPC601, MPC603, MPC604, MPC740™, and MPC750 microprocessors via the processor (60x) bus interface. The MPC106's 60x bus interface provides a 32-bit address bus and a 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 MPC106's 60x bus interface can be configured to support a single processor with a write-through or write-back L2 cache controlled by the MPC106, or up to four processors with or without an independently controlled external L2 cache.

4.1 MPC106 Processor Bus Configuration

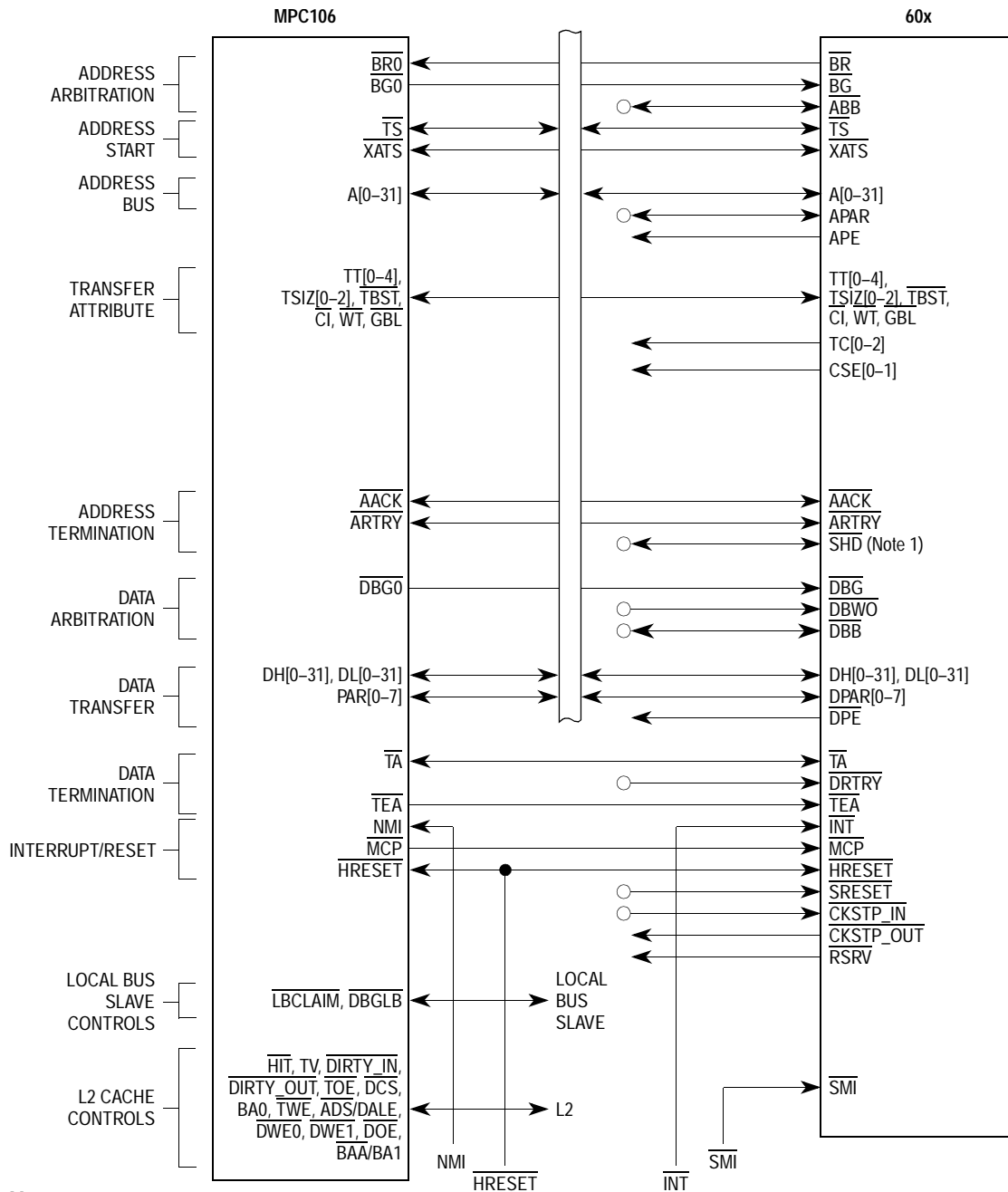
The figures in the following sections show how the MPC106 can be connected to support a single processor with an optional L2 cache, multiple processors, or multiple processors with an external lookaside L2 cache. The term 'alternate bus master' is used to refer to the L2 cache or another processor attached to the MPC106 in the following sections. Additional L2 cache configuration information is provided in Chapter 5, "Secondary Cache Interface."

4.1.1 Single-Processor System Configuration

The MPC106 can be connected to a single 601, 603, 604, 740, or 750 as shown in Figure 4-1. This configuration supports the addition of an L2 cache controlled by the MPC106, with the MPC106 snooping bus operations to maintain coherency between the primary cache in the processor, the L2 cache, and main memory.

4.1.2 Multiprocessor System Configuration

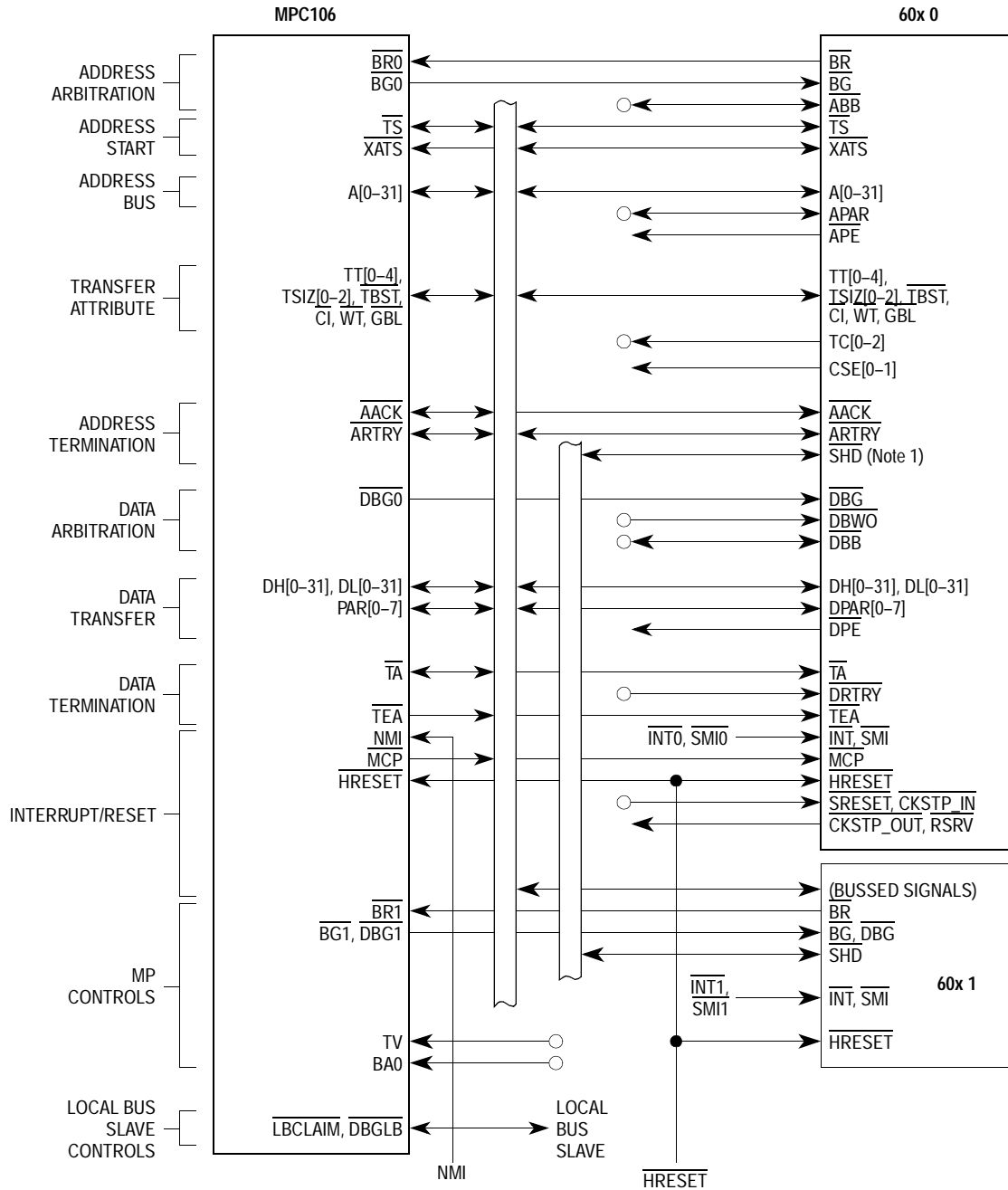
The MPC106 can also be configured to support up to four 603s, 604s, 740s, or 750s. When operating in a multiprocessor configuration, the MPC106 will snoop bus operations and maintain coherency between the primary caches and main memory. Note that the MPC106 does not support 601 processors in multiprocessor configurations. Figure 4-2 shows how two processors are attached to the MPC106.



Notes:

- 1. 603e/750 processors do not have a $\overline{\text{SHD}}$ pin. ○ Tied to V_{DD} through a resistor.
- All bidirectional control signals should have a pull-up resistor.

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



Note:

- 1. 603e/750 processors do not have a $\overline{\text{SHD}}$ pin.
- All bidirectional control signals should have a pull-up resistor.

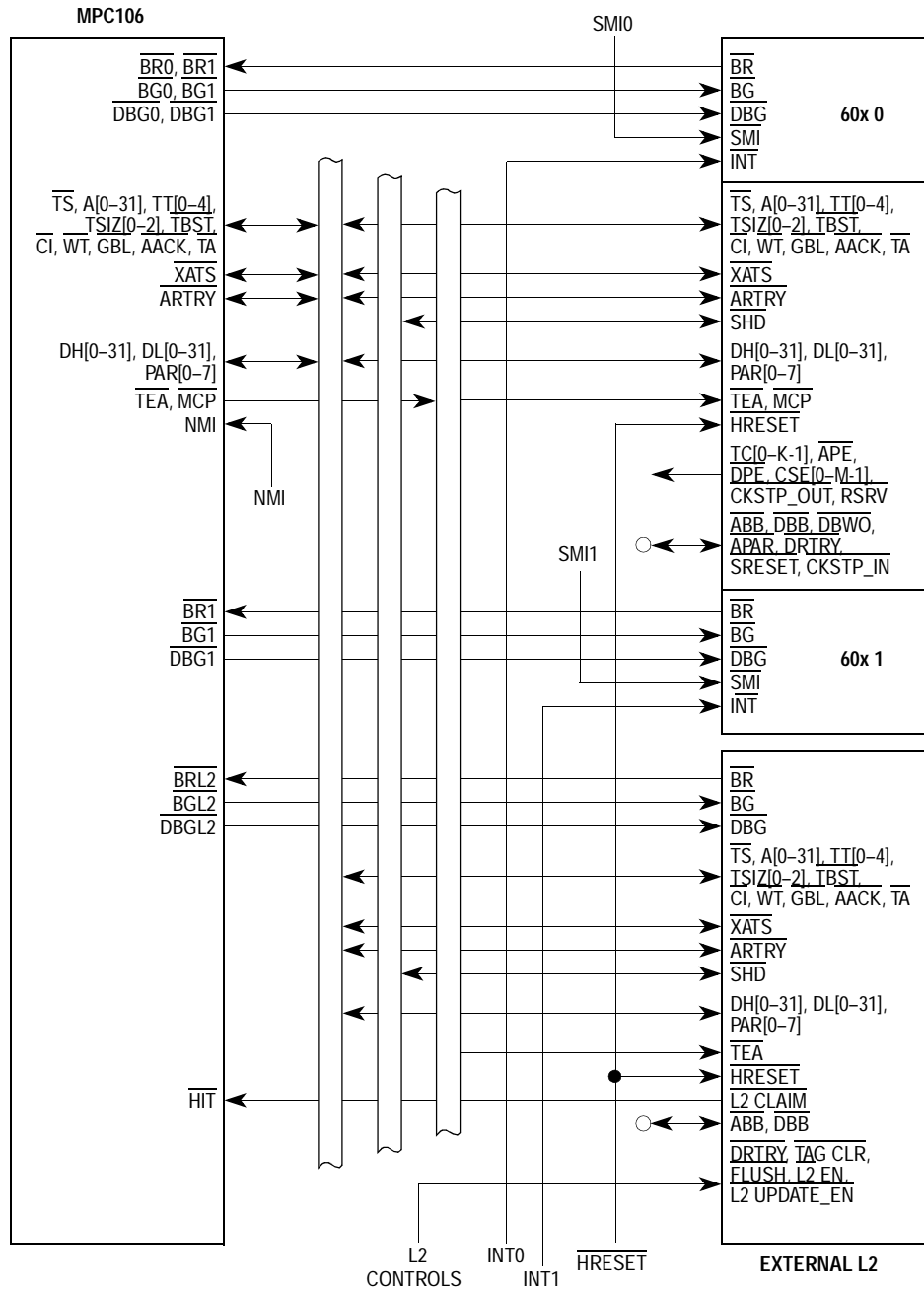
○ Tied to V_{DD} through a resistor.

Figure 4-2. Multiprocessor Configuration

4.1.3 Multiprocessor System Configuration with External L2 Cache

The MPC106 can be configured to support up to four 603s, 604s, 740s, or 750s, and an external L2 cache. Note that the MPC106 does not support 601 processors in

multiprocessor configurations. Figure 4-3 shows a typical system configuration with an MPC106, two 60x processors, and an external L2 cache. Note that signal connections to the external L2 cache may vary depending on the L2 cache implementation.



- Tied to V_{DD} through a resistor individually.
- All bidirectional signals should have a pull-up resistor.

Figure 4-3. Multiprocessor Configuration with External L2 Cache

4.1.4 Processor Bus Interface Configuration Registers

Following system reset, initialization software must set up the programmable parameters for the processor bus interface, located in processor interface configuration register 1 and 2 (PICR1 and PICR2). These programmable parameters control address and data bus parking, enable recognition of local bus slaves, and determine the configuration of multiprocessor systems and attached L2 caches, and sets the number of wait states required until the \overline{AACK} , \overline{ARTRY} , and L2 \overline{HIT} signals are sampled. See Section 3.2.9, “Processor Interface Configuration Registers,” for more detail about programming the PICR1 and PICR2 registers.

The following programmable parameters are relevant to the operation of the processor bus interface:

- PICR1[CF_L2_MP]. Sets the system for uniprocessor, L2 cache, or multiprocessor configuration.
- PICR1[CF_APARK]. The setting of this bit enables processor address bus parking when the address bus is idle.
- PICR1[CF_LOOP_SNOOP]. Setting this bit causes a PCI-to-memory transaction to be repeated until it is not retried.
- PICR1[CF_EXTERNAL_L2]. Setting this bit configures the system for external L2 cache.
- PICR1[CF_DPARK]. When this bit is set the processor data bus is parked when the data bus is idle.
- PICR1[CF_LBA_EN]. The setting of this bit enables local bus slave bus accesses.
- PICR1[CF_BREAD_WS]. These bits determine the minimum number of wait states from \overline{TS} to the first \overline{TA} for burst reads.
- PICR2[CF_APHASE_WS]. These bits determine the minimum number of wait states for address phase of processor-initiated bus transactions.
- PICR2[CF_L2_HIT_DELAY]. These bits determine the number of wait states from assertion of \overline{TS} until the \overline{HIT} signal is valid.
- PICR2[CF_SNOOP_WS]. These bits determine the number of wait states until \overline{ARTRY} is sampled.

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 4-4 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 4-4 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 4-4 demonstrates a data transfer that consists of a single-beat transfer of as many as 64 bits.

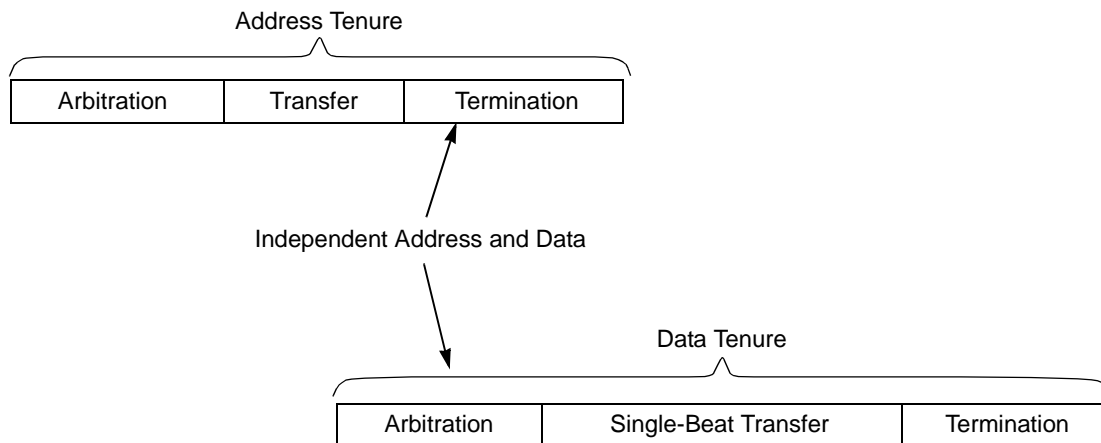


Figure 4-4. Overlapping Tenures on the 60x Bus for a Single-Beat Transfer

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 mastership of the address bus, it transfers the address. The address signals and the transfer attribute signals control 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 mastership of 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 MPC106 Arbitration

Arbitration for both address and data bus mastership is performed by the MPC106 through the use of the following signals. Note that the MPC106 controls bus access through the use of bus request and bus grant signals, and determines the busy state of the address and data

bus by monitoring the address and data bus request and bus grant signals, and the \overline{TS} , \overline{AACK} , and \overline{TA} signals.

The following signals are used for address bus arbitration:

- $\overline{BR0}$, $\overline{BR1}$, $\overline{BR2}$, and $\overline{BR3}$ (bus request)—Assertion indicates that a bus master is requesting mastership of the address bus.
- $\overline{BRL2}$ (bus request L2)—Assertion indicates that an external L2 cache is requesting mastership of the address bus.
- $\overline{BG0}$, $\overline{BG1}$, $\overline{BG2}$, and $\overline{BG3}$ (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{BGn} is asserted, \overline{ARTRY} is negated, and there is no current address tenure.
- $\overline{BGL2}$ (bus grant L2)—Assertion indicates that an external L2 cache may, with the proper qualification, assume mastership of the address bus.

The following signals are used for data bus arbitration:

- $\overline{DBG0}$, $\overline{DBG1}$, $\overline{DBG2}$, and $\overline{DBG3}$ (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{DBGn} is asserted while \overline{ARTRY} for the current data tenure is negated.
- $\overline{DBGL2}$ (data bus grant L2)—Indicates that an external L2 cache may, with proper qualification, assume mastership of the data bus.

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 MPC106) is required in systems in which multiple devices must compete for the system bus. The MPC106 affects pipelining by regulating address bus grants ($\overline{BG0}$, $\overline{BG1}$, $\overline{BG2}$, $\overline{BG3}$ and $\overline{BGL2}$), data bus grants ($\overline{DBG0}$, $\overline{DBG1}$, $\overline{DBG2}$, $\overline{DBG3}$, and $\overline{DBGL2}$), and the address acknowledge (\overline{AACK}) signal. One-level pipelining is implemented by the MPC106 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 MPC106 provides arbitration for the processor address bus. The external input signals to the arbiter are $\overline{BR0}$ in a single processor configuration, $\overline{BR0}$, $\overline{BR1}$, $\overline{BR2}$, $\overline{BR3}$ in a multiprocessor configuration, and $\overline{BRL2}$ if there is an external L2 cache. In addition to the external signals, there are internal bus request and bus grant signals for snoop broadcast and internal L2 castout operations. If the MPC106 needs to perform a snoop broadcast or internal L2 castout 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 copyback operations having the highest priority. External L2 snoop push operations have the next highest priority, followed by internal L2 snoop pushes and castout operations, snoop operations, external L2 cache castout operations (not including snoop pushes), and 60x bus requests. Processor bus requests when the MPC106 is in a multiprocessor configuration have rotating priority, unless an L1 cache copyback operation is required. In these cases, the MPC106 grants higher priority to the processor requesting the cache copyback. In multiprocessor systems where a 60x read from PCI operation has been \overline{ARTRYd} , the MPC106 will return the address bus grant to the same processor following the snoop operation.

Address bus parking is supported by the MPC106 through the use of the PICR2[CF_APARK] bit. When this bit is set, the MPC106 parks the address bus (asserts the address bus grant signal in anticipation of another processor address bus request) to the 60x processor that most recently had mastership of the bus.

The processor and alternate bus masters qualify \overline{BGn} by sampling \overline{TS} , \overline{AACK} , and \overline{ARTRY} in the negated state prior to assuming 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 alternate bus masters 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 mastership of the bus for the cache block push when it detects that bus grant is asserted. Note that if a 60x processor transaction hits in an internal buffer and is also claimed by an externally-controlled L2 cache, the MPC106 does not assert \overline{ARTRY} . However, if a 60x

processor transaction hits in an internal buffer but is not claimed by an externally-controlled L2 cache, the MPC106 does assert $\overline{\text{ARTRY}}$.

Figure 4-5 shows a series of address transfers to illustrate the transfer protocol when the MPC106 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 $\overline{\text{TS}}$) without first having asserted address bus request. During the same clock cycle, the MPC106's internal bus request is asserted to request access to the 60x bus, thereby causing the negation of $\overline{\text{BG0}}$. Following processor 0's address tenure, the MPC106 takes the bus and initiates its address transaction. At the completion of the MPC106's address transaction, both $\overline{\text{BR0}}$ and $\overline{\text{BR1}}$ are asserted. Because processor 0 was the last processor to have mastership of the bus, the MPC106's arbiter grants mastership to processor 1.

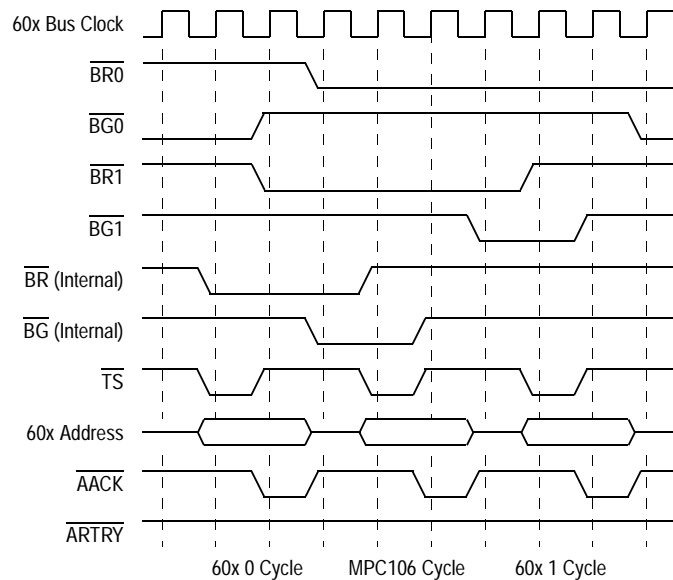
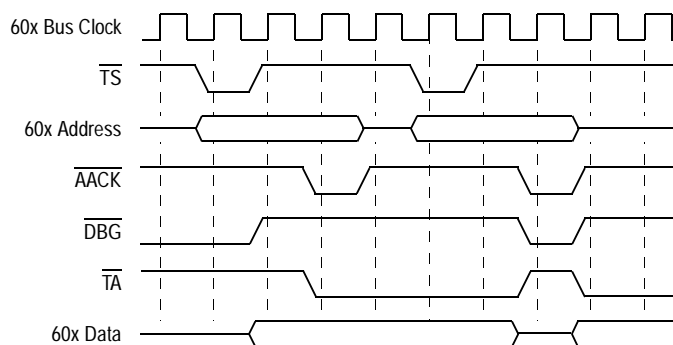


Figure 4-5. Address Bus Arbitration with Dual Processors

The MPC106 supports one level of address pipelining by asserting the $\overline{\text{AACK}}$ 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. In cycles controlled by the MPC106, the MPC106 never asserts $\overline{\text{AACK}}$ prior to the assertion of the corresponding data bus grant unless the cycle is aborted by the early assertion of $\overline{\text{ARTRY}}$. In bus transactions initiated by an external L2 cache controller, $\overline{\text{AACK}}$ may be asserted before the assertion of data bus grant for the transaction.

The MPC106 performs pipelined data bus operations strictly in order with the associated address operations. Figure 4-6 shows how address pipelining allows address tenures to overlap the associated data tenures.


Figure 4-6. Address Pipeling

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 (A[0–31]), along with the appropriate parity bits (however, the MPC106 does not check address bus parity). 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 (TT[0–4]), transfer size (TSIZ[0–2]), and transfer burst (\overline{TBST}) signals. These signals are discussed in the following sections.

4.3.2.1 Transfer Type Signal Encodings

The TT[0–4] 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 either the address bus master or the MPC106, depending on the nature of the bus transaction. Transfer type signals originating from the MPC106 occur due to snoop operations caused by PCI bus accesses to memory. Table 4-1 describes the MPC106's response to transfer type signals driven by an address bus master on the 60x bus.

Table 4-1. MPC106 Responses to 60x Transfer Type Signals

TT[0–4]	Bus Operation	Class of Operation	\overline{TEA}	MPC106 Response
01010	Read	Normal	—	Read, assert \overline{AACK} and \overline{TA} .
01110	Read-with-intent-to-modify	Normal	—	Read, assert \overline{AACK} and \overline{TA} .
11010	Read atomic	Normal	—	Read, assert \overline{AACK} and \overline{TA} .
11110	Read-with-intent-to-modify-atomic	Normal	—	Read, assert \overline{AACK} and \overline{TA} .
00010	Write-with-flush	Normal	—	Write, assert \overline{AACK} and \overline{TA} .
00110	Write-with-kill	Normal	—	Write, assert \overline{AACK} and \overline{TA} .

Table 4-1. MPC106 Responses to 60x Transfer Type Signals (continued)

TT[0-4]	Bus Operation	Class of Operation	$\overline{\text{TEA}}$	MPC106 Response
10010	Write-with-flush-atomic	Normal	—	Write, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.
01000	sync	Normal	—	Address only, asserts $\overline{\text{AACK}}$. Bus grant is negated until MPC106 buffers are flushed.
10000	eieio	Normal	—	Address only, asserts $\overline{\text{AACK}}$. Bus grant is negated until MPC106 buffers are flushed.
01100	Kill	Normal	—	Address only operation, $\overline{\text{AACK}}$ is asserted. MPC106 buffers are snooped.
01101	icbi	Normal	—	Address only operation, $\overline{\text{AACK}}$ is asserted. MPC106 buffers are snooped.
01011	Read-with-no-intent-to-cache (RWNITC)	Not supported	—	The MPC106 does not support RWNITC transactions. 60x bus masters must not generate this transfer type or the system may hang.
00000	Clean	Normal	—	Address only operation. $\overline{\text{AACK}}$ is asserted, and MPC106 takes no further action.
00100	Flush	Normal	—	Address only operation. $\overline{\text{AACK}}$ is asserted, and MPC106 takes no further action.
11000	tlbi	Normal	—	Address only operation. $\overline{\text{AACK}}$ is asserted, and MPC106 takes no further action.
00001	lwarx , reservation set	Normal	—	Address only operation. $\overline{\text{AACK}}$ is asserted, and MPC106 takes no further action. (The MPC106 does not support atomic references in PCI memory space.)
00101	stwcx. , reservation set	Normal	—	Address only operation. $\overline{\text{AACK}}$ is asserted, and MPC106 takes no further action. (The MPC106 does not support atomic references in PCI memory space.)
01001	tlbsync	Normal	—	Address only operation. $\overline{\text{AACK}}$ is asserted, and MPC106 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.
00101	<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.
0100x	Direct-store load request	Error	—	Illegal operation; signals error. Address only operation. $\overline{\text{AACK}}$ is asserted.

Table 4-1. MPC106 Responses to 60x Transfer Type Signals (continued)

TT[0-4]	Bus Operation	Class of Operation	$\overline{\text{TEA}}$	MPC106 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 is 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 is terminated by $\overline{\text{TA}}$.
0101x	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 is terminated by $\overline{\text{TA}}$.
0111x	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 is terminated by $\overline{\text{TA}}$.
0001x	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 is terminated by $\overline{\text{TA}}$.
0011x	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 is terminated by $\overline{\text{TA}}$.

The MPC106 propagates snoop broadcast operations to the 60x bus in response to PCI bus-initiated memory accesses. All snoop broadcasts generated by the MPC106 are caused by PCI bus operations and are identified as burst, cacheable, write-back, and global accesses. The transaction types driven by the MPC106 for snoop operations are not encoded as address-only; however, all MPC106-initiated snoop operations should be treated as address-only transactions, and no $\overline{\text{DBGn}}$ or $\overline{\text{TA}}$ signal assertions should be expected following the address tenure. Table 4-2 describes the transfer type encodings generated by the MPC106.

Table 4-2. Transfer Type Encodings Generated by the MPC106

TT[0-4] (Driven by MPC106)	60x Bus Operation	Condition
00010	Burst-write-with-flush	Generated in response to nonlocked 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 $\text{TSIZ}[0-2]$ Signals and Size of Transfer

The transfer size ($\text{TSIZ}[0-2]$) 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 $A[29-31]$ 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 the transfer of cache blocks. For these transactions, the $\text{TSIZ}[0-2]$ signals are encoded as 0b010, and address bits $A[27-28]$ determine which double-word transfer should occur first.

The MPC106 supports critical-word-first burst transactions (double-word-aligned) from the 60x processor. The MPC106 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.

Table 4-3. MPC106 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 block) 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 block, and the cache-block-fill operation may wrap around the end of the cache block. Table 4-4 describes MPC106 burst ordering.

Table 4-4. MPC106 Burst Ordering

Data Transfer	For Starting Address:			
	$A[27-28] = 00$	$A[27-28] = 01$	$A[27-28] = 10$	$A[27-28] = 11$
First data beat	DW0	DW1	DW2	DW3
Second data beat	DW1	DW2	DW3	DW0

Table 4-4. MPC106 Burst Ordering (continued)

Data Transfer	For Starting Address:			
	A[27–28] = 00	A[27–28] = 01	A[27–28] = 10	A[27–28] = 11
Third data beat	DW2	DW3	DW0	DW1
Fourth data beat	DW3	DW0	DW1	DW2

Note: The A[29–31] signals are always 0b000 for burst transfers initiated by the MPC106.

4.3.2.4 Effect of Alignment on Data Transfers

Table 4-5 lists the aligned transfers that can occur to and from the MPC106. 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 4-5 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 4-5. Aligned Data Transfers

Transfer Size	TSIZ0	TSIZ1	TSIZ2	A[29–31]	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 DH[0–7], byte lane 7 corresponds to DL[24–31].

The MPC106 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 from an odd byte address). The MPC106's processor bus interface supports misaligned transfers within a word (32-bit aligned) boundary, as shown in Table 4-6. Note that the 4-byte transfer in Table 4-6 is only one example of misalignment. As long as the attempted transfer does not cross a word boundary, the MPC106 can transfer the data to the misaligned address within a single bus transfer (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 should be 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 4-6. Misaligned Data Transfers (4-Byte Examples)

Transfer Size (Four Bytes)	TSIZ[0-2]	A[29-31]	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	—	—	—	—
	0 0 1	1 0 0	—	—	—	—	A	—	—	—
Misaligned first access	0 1 0	0 1 0	—	—	A	A	—	—	—	—
	0 1 0	1 0 0	—	—	—	—	A	A	—	—
Misaligned first access	0 0 1	0 1 1	—	—	—	A	—	—	—	—
	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
	0 0 1	0 0 0	A	—	—	—	—	—	—	—
Misaligned first access	0 1 0	1 1 0	—	—	—	—	—	—	A	A
	0 1 0	0 0 0	A	A	—	—	—	—	—	—
Misaligned first access	0 0 1	1 1 1	—	—	—	—	—	—	—	A
	0 1 1	0 0 0	A	A	A	—	—	—	—	—

Notes:

- A: Byte lane used
- : Byte lane not used

4.3.3 Address Transfer Termination

Address transfer termination occurs 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 until one clock after $\overline{\text{AACK}}$; the bus clock cycle after $\overline{\text{AACK}}$ is referred to as the $\overline{\text{ARTRY}}$ window. The MPC106 controls assertion of $\overline{\text{AACK}}$ unless the cycle is claimed by the external L2 cache controller (as indicated by the assertion of the $\overline{\text{HIT}}$ signal by the L2 cache controller). Following assertion of $\overline{\text{HIT}}$, the L2 cache controller is responsible for assertion of $\overline{\text{AACK}}$. When $\overline{\text{AACK}}$ is asserted by the L2 cache controller, it should be asserted for one clock cycle, and then negated for one clock cycle prior to entering a high-impedance state. The MPC106 holds the $\overline{\text{AACK}}$ signal in a high-impedance state until assertion of $\overline{\text{AACK}}$ by the MPC106 is required for the termination of the address cycle. For address bus transactions initiated by a processor, the snoop response originates from either the MPC106 or an alternate bus master (the other processors or an external L2 cache controller). For transactions initiated by the MPC106, the snoop response originates from an alternate bus master.

The following sections describe how the MPC106 can be configured through its register settings to accommodate a variety of snoop responses and snoop timing requirements.

4.3.3.1 MPC106 Snoop Response

Processors may assert $\overline{\text{ARTRY}}$ because of pipeline collisions or because an address snoop hits a modified block in the processor's L1 cache. When a processor detects a snoop hit due to a modified block in the cache, it will assert its bus request in the window of opportunity (the clock after the $\overline{\text{ARTRY}}$ window) to obtain mastership of the bus for its L1 copyback cycle. Note that the L1 copyback is a non-global ($\overline{\text{GBL}}$ negated) transaction. External devices on the 60x bus must not assert $\overline{\text{ARTRY}}$ for non-global transactions.

The MPC106 can be configured to repeat the snoop for a PCI-to-memory transaction that has been terminated by the assertion of $\overline{\text{ARTRY}}$ by a processor or by the L2 cache through the use of the PICR1[CF_LOOP_SNOOP] bit. If PICR1[CF_LOOP_SNOOP] is set, the MPC106 repeats snooping until $\overline{\text{ARTRY}}$ is not asserted. If PICR1[CF_LOOP_SNOOP] is cleared, the MPC106 repeats the snoop until either $\overline{\text{ARTRY}}$ is not asserted, or a snoop push occurs.

The MPC106 may assert $\overline{\text{ARTRY}}$ because of an L2 castout operation (when the MPC106's internal L2 cache controller is being used), an address collision with an MPC106 internal buffer, 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 while the PCI bus is occupied by another PCI bus master, or when a 60x processor performs a write operation to the PCI bus, but the MPC106'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. Note that the MPC106 may assert $\overline{\text{ARTRY}}$ on any clock cycle after the assertion of $\overline{\text{TS}}$ and the clock cycle following $\overline{\text{AACK}}$.

Figure 4-7 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 MPC106 to grant the bus to the 60x processor which then proceeds with a copyback to main memory of the modified cache block.

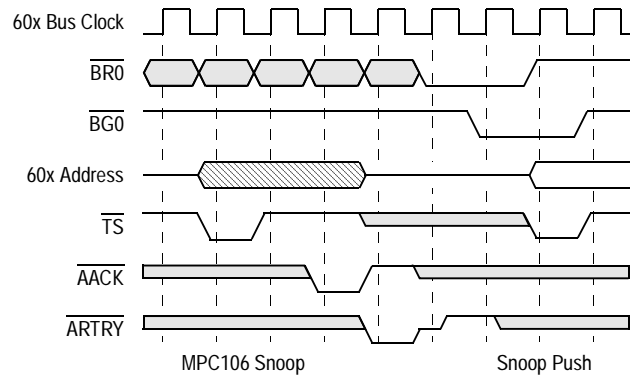


Figure 4-7. Snoop Address Transaction with $\overline{\text{ARTRY}}$ and L1 Cache Copyback

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 MPC106 is determined by the $\text{PICR2}[\text{CF_APHASE_WS}]$ and $\text{PICR2}[\text{CF_L2_HIT_DELAY}]$ bits, and the pipeline status of the 60x bus. Since the MPC106 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 MPC106 also withholds the assertion of $\overline{\text{AACK}}$ until no more $\overline{\text{ARTRY}}$ conditions can occur. Note that the earliest opportunity the MPC106 can assert the $\overline{\text{AACK}}$ signal is the clock cycle when the wait-state values set by both $\text{PICR2}[\text{CF_L2_HIT_DELAY}]$ and $\text{PICR2}[\text{CF_APHASE_WS}]$ have expired.

The $\text{PICR2}[\text{CF_APHASE_WS}]$ bits specify the minimum number of address tenure wait states for 60x processor-initiated address operations. Extra wait states may occur because of other MPC106 configuration parameters. Note that in a system implementing an L2 cache, the number of wait states configured by the $\text{PICR2}[\text{CF_APHASE_WS}]$ bits should be equal to or greater than the value configured in $\text{PICR2}[\text{CF_L2_HIT_DELAY}]$. In systems with multiple processors, the number of wait states configured by the $\text{PICR2}[\text{CF_APHASE_WS}]$ bits should be equal to or greater than the number of wait states selected by the $\text{PICR2}[\text{CF_SNOOP_WS}]$ bits, since alternate bus masters need to snoop the 60x access. If the MPC106 is configured to support the compatibility hole in memory map B or the emulation memory map, $\text{PICR2}[\text{CF_APHASE_WS}]$ should be set to a value of 1 or greater. The $\text{PICR2}[\text{CF_SNOOP_WS}]$ 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 MPC106-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 MPC106 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 TT[0–4] signals) indicates the transaction is not address-only.

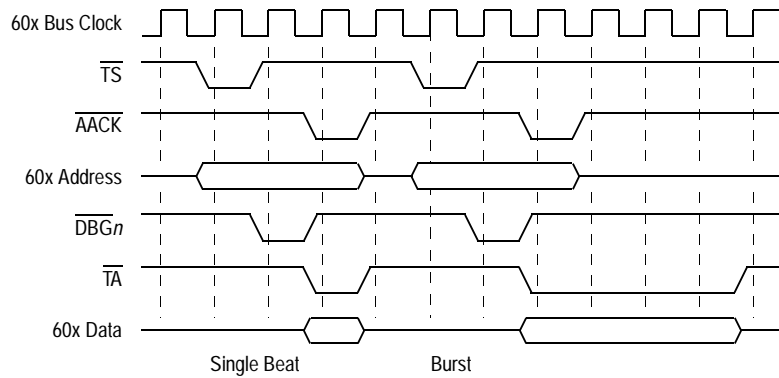
The MPC106 implements one data bus grant signal ($\overline{\text{DBGn}}$) for each potential master on the 60x interface. The $\overline{\text{DBGn}}$ signals are not asserted if the data bus, which is shared with the memory, is busy with a transaction. The internal buffer control circuitry arbitrates 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 MPC106 should park the 60x data bus. If the CF_DPARK bit is set, the data bus is parked to the processor that had most recently taken mastership of the 60x address bus.

4.4.2 Data Bus Transfers and Normal Termination

The MPC106 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. 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 4-8 illustrates a sample of both a single-beat and burst data transfer. The $\overline{\text{TA}}$ signal is asserted by the MPC106 to mark the cycle in which data is accepted. In a normal burst transfer, the fourth assertion of $\overline{\text{TA}}$ signals the end of a transfer.


Figure 4-8. Single-Beat and Burst Data Transfers

4.4.3 Data Tenure Timing Configurations

The MPC106 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 and alternate bus masters. For example, a 603 running in 1:1 mode requires a minimum of one wait state, and at least two wait states if no- \overline{DRTRY} mode is also selected.

4.4.4 Data Bus Termination by TEA

If a processor initiates a transaction that is not supported by the MPC106, the MPC106 signals an error by asserting either the transfer error acknowledge (\overline{TEA}) or machine check (\overline{MCP}) signals to the 60x processor(s). If the transaction is not address only, the MPC106 asserts \overline{TEA} to terminate the transaction provided \overline{TEA} is enabled by setting the PICR1[TEA_EN] bit. The MPC106 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 MPC106 ensures that the 60x processor receives a qualified data bus grant by asserting the \overline{DBGn} signal before asserting \overline{TEA} . The data tenure is terminated by a single assertion of \overline{TEA} regardless of whether the data tenure is single-beat or burst. This sequence is shown in Figure 4-9. In Figure 4-9 the data bus is busy at the beginning of the transaction, thus delaying the assertion of \overline{DBGn} . Note that although \overline{DBB} is not an input to the MPC106, the state of the bus is always known because the MPC106 controls the data bus grants and either drives or monitors the termination signals.

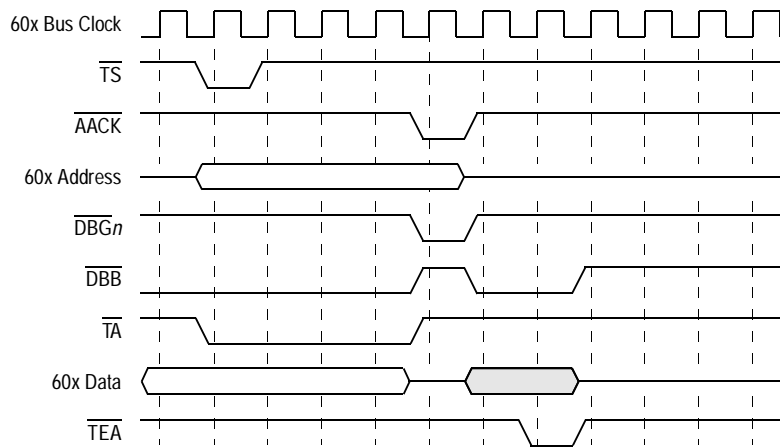


Figure 4-9. Data Tenure Terminated by Assertion of \overline{TEA}

The bus transactions interpreted by the MPC106 as bus errors are as follows:

- Direct-store transactions, as indicated by the assertion of \overline{XATS} and $TT[0-4]$
- Any graphics read/write transactions (caused by **eciwx** or **ecowx** instructions)
- Write operations into INTA space, 0xBFFF_FFF0 in address map A, or 0xFEFX_XXXX in address map B.
- Write operations into ROM or write to Flash ROM if Flash ROM is not enabled, or if the transfer bus width does not match the Flash ROM data width, or transaction is not caching-inhibited or write-through. Caching-inhibited or write-through writes to Flash ROM are the only transactions allowed.
- Processor read operation from **PCI** transaction which is target-aborted by the **PCI** target, or if the target asserts \overline{PERR} .

4.4.5 60x Local Bus Slave Support

The MPC106 provides support for a local bus slave that can handle 60x transactions by generating its own \overline{TA} responses. Note that local bus slaves cannot be used in conjunction with half-bus clock modes (3:2 and 5:2). See Section 2.3, “Clocking,” for more information. Following the assertion of the local bus claim ($\overline{LBCLAIM}$) signal, the MPC106 will assert \overline{AACK} for local bus slave address tenures. The local bus slave should monitor the \overline{ARTRY} and \overline{TEA} signals, and abort the associated data tenure if required, but should never assert \overline{ARTRY} . In addition, the local bus slave should only respond to bus transactions originated by processors, and not respond to L2 cache castout operations, or L1 snoop transactions. System designers should ensure that the local bus slave does not respond to any bus transaction by the MPC106’s internal L2 controller, or an external L2 cache controller, as the L2 controllers have no logic to sense the assertion of the $\overline{LBCLAIM}$ signal by the local bus slave.

The local bus slave can claim any address in the address space from 0x0000_0000 to 0x7FFF_FFFF (0–2 Gbytes). Under no circumstances can the local bus slave reside in the 2-Gbyte to 4-Gbyte address range. In addition, the location that the local bus slave may exist is limited as follows:

- If memory select errors are enabled ($\text{ErrEnR1}[5] = 1$), then the local bus slave must reside in a memory space that is mapped by the memory boundary registers and memory bank enable registers within the 0–1 Gbyte address space.
- If memory select errors are disabled ($\text{ErrEnR1}[5] = 0$), then the local bus slave can reside inside the 0–(2GB-1) address range.

The memory boundary registers and the MICRs should be programmed and the MEMGO bit should be set before the LBS claims any transaction. To prevent conflicts with the L2 cache, the local bus slave should use an address range that is not cacheable in the L2.

4.4.5.1 60x Local Bus Slave Timing

The MPC106's response to a local bus slave is controlled through the configuration of the $\text{PICR1}[\text{CF_LBA_EN}]$ bit. If the $\text{PICR1}[\text{CF_LBA_EN}]$ bit is cleared, the MPC106 ignores the $\overline{\text{LBCLAIM}}$ signal. If the $\text{PICR1}[\text{CF_LBA_EN}]$ bit is set, the MPC106 samples the $\overline{\text{LBCLAIM}}$ signal when the wait-state value set in $\text{PICR2}[\text{CF_L2_HIT_DELAY}]$ expires. If $\overline{\text{LBCLAIM}}$ is asserted for a transaction, and $\text{CF_LBA_EN} = 1$, the MPC106 asserts $\overline{\text{AACK}}$, and waits for the local bus slave to assert the appropriate number of $\overline{\text{TA}}$ s for the transaction (that is, one for a single-beat transaction, or four for a burst transaction). Note that the MPC106 does not drive $\overline{\text{TA}}$ for transactions claimed by the local bus slave; the local bus slave must drive $\overline{\text{TA}}$ for the transactions that it claims.

The $\overline{\text{DBGLB}}$ signal is derived from and has the same timing as $\overline{\text{DBGn}}$. The timing of $\overline{\text{DBGn}}$ assertion is dependent upon the data bus being idle. If the MPC106 is servicing a PCI-to-system-memory access, a copyback buffer flush to system memory, a memory refresh cycle, an L2 copyback, or a previous 60x data tenure, it will delay the assertion of $\overline{\text{DBGn}}$ until the data bus is idle. Note that the 106 is the 60x bus arbiter and sometimes uses the 60x bus for its own purposes by withholding the bus grants (both $\overline{\text{BGn}}$, and $\overline{\text{DBGn}}$) from the processor(s).

$\overline{\text{DBGLB}}$ is not a data bus grant to the local bus slave (LBS), but rather it is a signal to the local bus slave that the MPC106 has granted the data bus to one of the processors. The local bus slave cannot initiate a transaction on the 60x bus—it can only respond to transactions initiated by the 60x processor. $\overline{\text{DBGLB}}$ is only an indication that the data bus has been granted to the 60x processor. If the local bus slave claims the transaction (by asserting $\overline{\text{LBCLAIM}}$), the MPC106 stops driving the data bus and waits for the data tenure between the 60x processor and the local bus slave to complete before it resumes control of the data bus.

Because the MPC106 multiplexes the SDRAM clock enable signal (CKE) with the $\overline{\text{DBGLB}}$ signal, SDRAM-based systems that use local bus slaves must provide a data bus grant signal to the local bus slave by alternate means. In a uniprocessor system, the $\overline{\text{DBG0}}$ signal can be used for $\overline{\text{DBGLB}}$. In a multiprocessor system, $\overline{\text{DBG}}[0-3]$ can be logically ANDed to create a suitable $\overline{\text{DBGLB}}$ signal. Note that using these methods to provide a data bus grant signal for the local bus slave is incompatible with the external L2 interface of the MPC106. Therefore, SDRAM-based systems that use local bus slaves cannot use an external L2 cache.

The local bus slave needs to sample $\overline{\text{DBGLB}}$ continuously. If the local bus slave claims the transaction (by asserting $\overline{\text{LBCLAIM}}$) and $\overline{\text{DBGLB}}$ has been asserted for that address tenure, then the local bus slave can drive $\overline{\text{TA}}$. If $\overline{\text{DBGLB}}$ has not been asserted when the local bus slave claims a transaction, then it must wait for the MPC106 to grant the data bus to the processor before the local bus slave can drive $\overline{\text{TA}}$. This way, the MPC106 can maintain the pipeline and the previous data tenure is allowed to complete before the MPC106 relinquishes the data bus to the processor and the local bus slave.

Note that the local bus slave should only drive the $\overline{\text{TA}}$ signal when a data tenure is in progress. The local bus slave is responsible for precharging the $\overline{\text{TA}}$ signal following negation by driving $\overline{\text{TA}}$ high for one half clock cycle prior to entering a high-impedance state following the last $\overline{\text{TA}}$ assertion. If the system is running in fast-L2 mode, $\overline{\text{DBGLB}}$ may be asserted at the same time as the last assertion of $\overline{\text{TA}}$ in an L2 cache operation. If the following data bus tenure is a local bus slave transaction, the local bus slave should delay driving data and asserting $\overline{\text{TA}}$ to allow time for $\overline{\text{TA}}$ negation and precharging between the L2 bus operation and the subsequent local bus slave bus operation.

The local bus slave can drive $\overline{\text{TA}}$ the clock after the assertion of $\overline{\text{LBCLAIM}}$, but not earlier, thereby providing a fastest local bus slave access of 3-1-1-1 bus cycles. A local bus slave should not assert $\overline{\text{TA}}$ before the $\overline{\text{ARTRY}}$ window when the system is configured for no- $\overline{\text{DRTRY}}$ mode operation, and should not assert $\overline{\text{TA}}$ more than one clock before the $\overline{\text{ARTRY}}$ window when operating in $\overline{\text{DRTRY}}$ mode.

The MPC106 will not assert $\overline{\text{ARTRY}}$ for any bus operation claimed by the local bus slave. External L2 cache controllers should not assert $\overline{\text{ARTRY}}$ for any local bus slave operation, as data from the local bus slave should be treated as noncacheable. If the MPC106 is configured for multiprocessor operation, another processor can assert $\overline{\text{ARTRY}}$ to retry the bus operation in the clock cycle after $\text{PICR2}[\text{CF_SNOOP_WS}]$ expires.

Figure 4-10 shows an example of a fast local bus slave transaction.

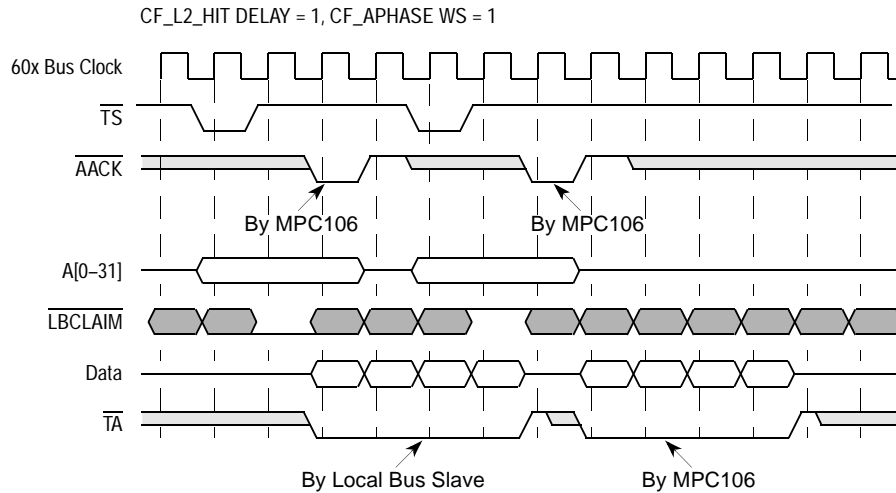
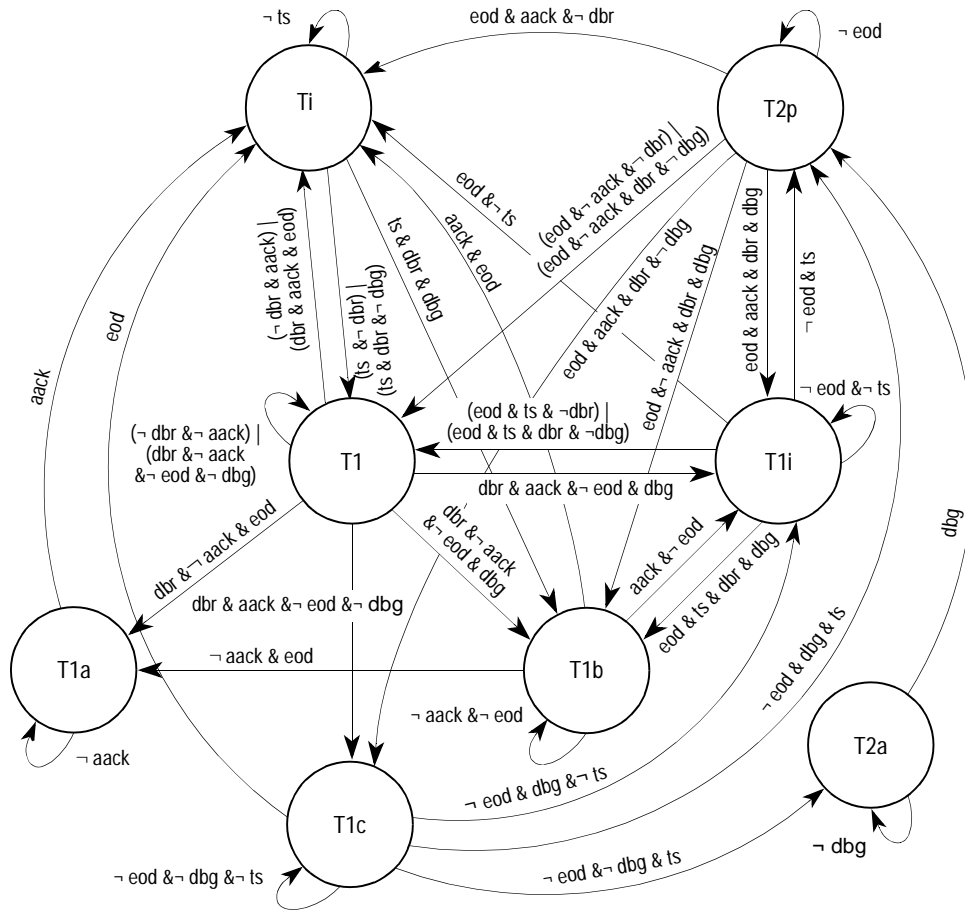


Figure 4-10. Local Bus Slave Transaction

When tracking 60x bus status, note that the MPC106 internal L2 cache controller does not use standard 60x bus signaling during L2 cache castout operations (including L2 flush operations generated by the L2 flush command.) The \overline{TS} , \overline{AACK} , or \overline{DBGn} will not be asserted on the 60x bus for L2 cache castout operations; under some conditions \overline{TA} may be asserted. Since the MPC106 tracks the L2 cache castout cycle as a normal bus operation internally, a following bus operation will be treated as a pipelined operation by the MPC106. The assertion of \overline{AACK} for the following pipelined bus operation will be delayed until the end of the L2 cache castout data tenure.

The encoding of the TT[0-4] bits of the MPC106 do not reflect address-only transactions during MPC106-initiated snoop operations; however, all snoop operations initiated by the MPC106 should be considered as address-only operations. There will be no \overline{DBGn} or \overline{TA} signal assertions by the MPC106 for snoop operations initiated by the MPC106.

The 60x bus state diagram shown in Figure 4-11 is provided to facilitate the design of local bus slave devices.



Legend:

- \neg —Condition shown is false
- aack—Address acknowledge
- dbr—Data bus request as indicated by TT[0–4] that the cycle is not address only.
- dbg—Data bus grant
- eod—End of data phase
- ts—Transfer start
- T_i —Idle
- T_1 —Address phase in progress, data phase not started yet.
- T_{1a} —Address phase in progress, data phase completed.
- T_{1b} —Address phase and data phase in progress.
- T_{1c} —Address phase completed, data phase not started yet.
- T_{1i} —Address phase completed, data phase in progress.
- T_{2a} —Address pipelined, data phase not started yet.
- T_{2p} —Address pipelined, data phase in progress.

Figure 4-11. 60x Bus State Diagram

Chapter 5

Secondary Cache Interface

A secondary (L2) cache provides the processor with faster access to instructions and data by maintaining a subset of system memory in high-speed static RAM devices (SRAMs). The MPC106 provides support for two L2 cache options—an internally-controlled L2 cache and an external L2 cache controller (or integrated L2 cache module).

The internal L2 cache controller allows the system designer to implement a direct-mapped, lookaside L2 cache in a write-back or write-through configuration with a cacheable address space of up to 4 Gbytes. The MPC106 supports L2 cache sizes of 256 Kbytes, 512 Kbytes, and 1 Mbyte made up of either synchronous burst, pipelined burst, or asynchronous SRAMs. The data path to the L2 cache is 64 bits wide. The L2 cache line size and coherence granularity is 32 bytes.

The MPC106 can perform fast nonpipelined bursts of 3-1-1-1 bus cycles and pipelined bursts of 2-1-1-1 bus cycles to the internally-controlled L2 cache. When used with the MPC604 microprocessor in fast L2 mode, the MPC106 can perform pipelined bursts of 1-1-1-1 bus cycles to the internally-controlled L2 cache.

This chapter describes the L2 cache interface of the MPC106—the various configurations, operation, programmable parameters, and response to bus operations. Design and timing examples are provided for several configurations.

Throughout this chapter, the following details should also be considered:

- The MPC106 does not support caching PCI space with the internally-controlled L2.
- L2 write-back operations to PCI space are illegal.
- When configured for an externally-controlled L2, PCI space may be cached if no write-back operations to PCI space occur.
- L2 write-back operations to the system ROM space are illegal.
- In emulation mode, L2 write-back operations to the interrupt vector relocation space are illegal.

Chapter 2, “Signal Descriptions,” contains the signal definitions for the L2 cache interface and Chapter 3, “Device Programming,” details the configurable parameters used to initialize the L2 cache interface. In addition, Chapter 8, “Internal Control,” provides

information about the internal buffers that permit the MPC106 to coordinate memory accesses between the L2 cache, the 60x processor(s), and devices on the PCI bus.

5.1 L2 Cache Configurations

The following sections describe the various L2 cache configurations that the MPC106 supports.

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.

5.1.2 Write-Through Cache Operation

Write-through L2 caches reduce read latency in the same way write-back L2 caches do, but write operations from the primary (L1) cache are written to both the L2 cache and the memory system, thereby exhibiting the same latency as an ordinary memory write. A 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.

5.1.3 Synchronous Burst SRAMs

The internal L2 cache controller of the MPC106 supports using synchronous burst SRAMs as the L2 data RAMs. A typical implementation is shown in Figure 5-1.

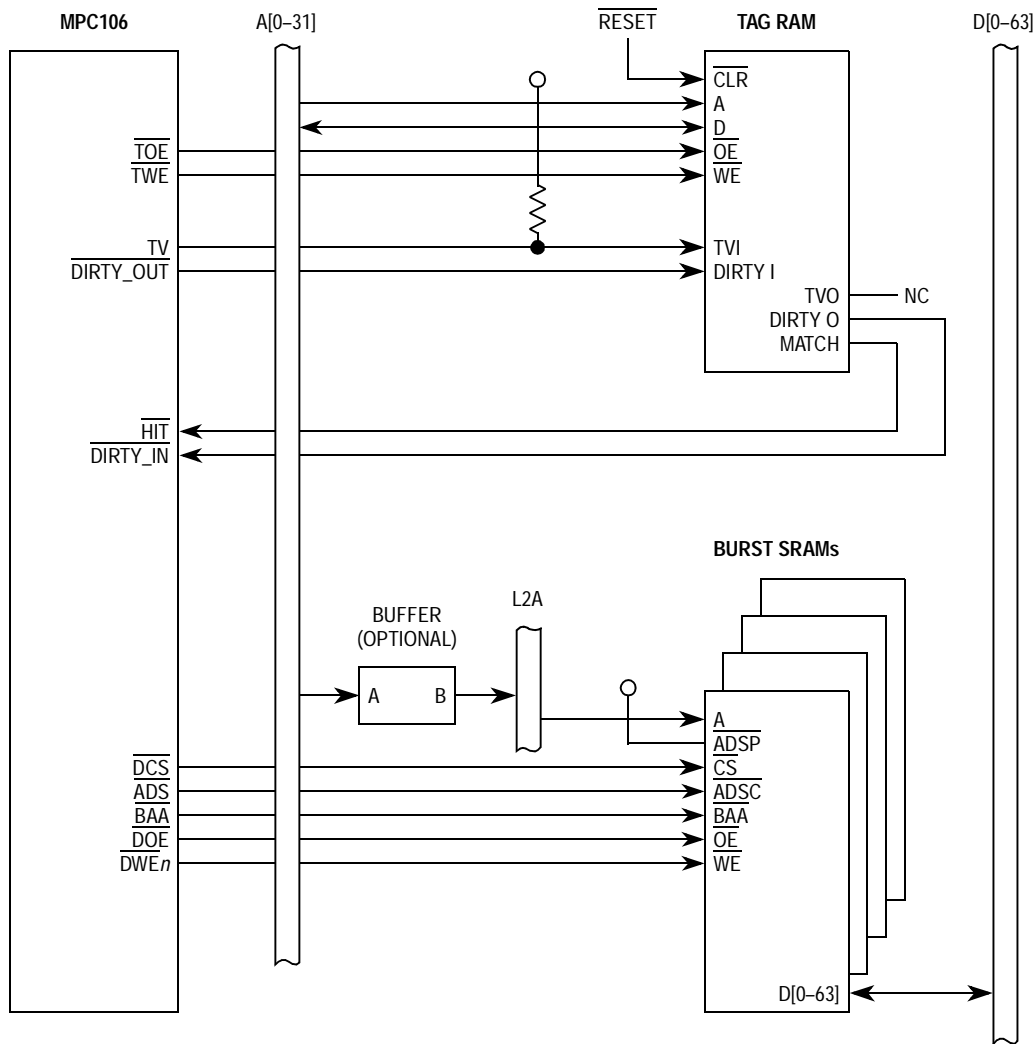


Figure 5-1. Typical L2 Cache Using Burst SRAM (Write-Back)

The MPC106 provides five signals for interfacing to synchronous burst SRAMs. Burst transactions consist of four beats, with 64 bits of data per beat. The data RAM latches the address and \overline{DCS} (data RAM chip select) inputs at the beginning of the access when \overline{ADS} (address strobe) is asserted. The \overline{DOE} (data RAM output enable) 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 $\overline{DWE_n}$ signals, when asserted, indicate that a write operation to the burst RAM is required. If the MPC106 is configured for partial update write timing ($CF_WMODE = 0b01, 0b10, \text{ or } 0b11$), then external logic is required to decode which bytes of the double word should be selected for the write to the L2 data RAM. The byte write enables can be decoded from the $\overline{DWE_n}$, $A[29-31]$, \overline{TBST} , and $TSIZ[0-2]$ signals.

5.1.4 Pipelined Burst SRAMs

The internal L2 cache controller of the MPC106 supports using pipelined burst SRAMs as the L2 data RAMs. Timing for pipelined burst SRAMs is very similar to burst SRAMs, except \overline{BAA} and \overline{TA} are delayed to account for the extra pipeline delay for the pipelined burst SRAM read accesses.

Typical applications only use the SRAM's \overline{ADSC} input as shown in Figure 5-2. The fastest possible cycle times are 4-1-1-1 for nonpipelined accesses and 1-1-1-1 for pipelined accesses.

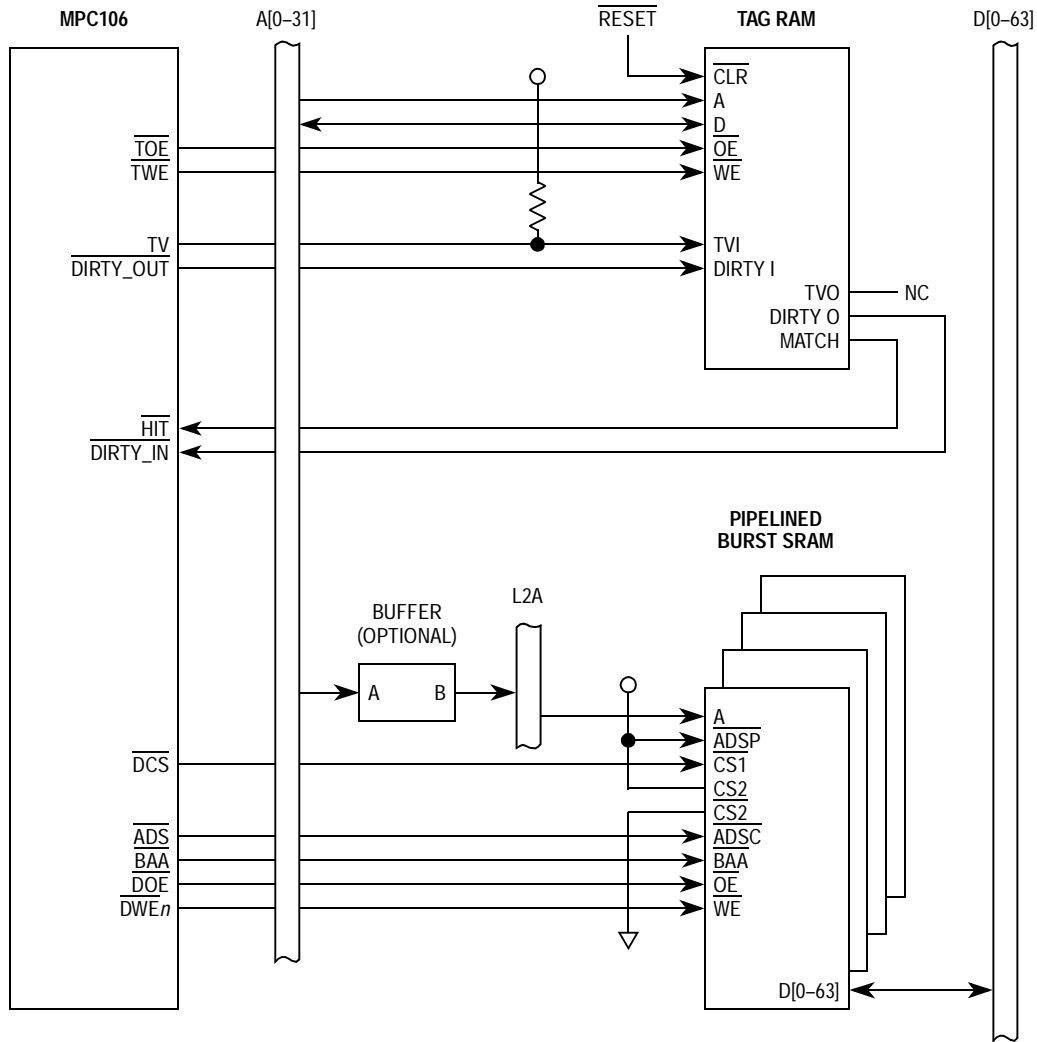


Figure 5-2. Typical L2 Cache Using Pipelined Burst SRAM (Write-Back, ADSC Only)

Alternately, an application may use both the \overline{ADSC} and \overline{ADSP} inputs to the SRAM to control addressing as shown in Figure 5-3. The fastest possible cycle times for this configuration are 3-1-1-1 for nonpipelined accesses and 1-1-1-1 for pipelined accesses.

Note that the use of $\overline{\text{ADSP}}$ is only supported for pipelined burst SRAMs; synchronous burst SRAMs cannot use $\overline{\text{ADSP}}$ in this manner.

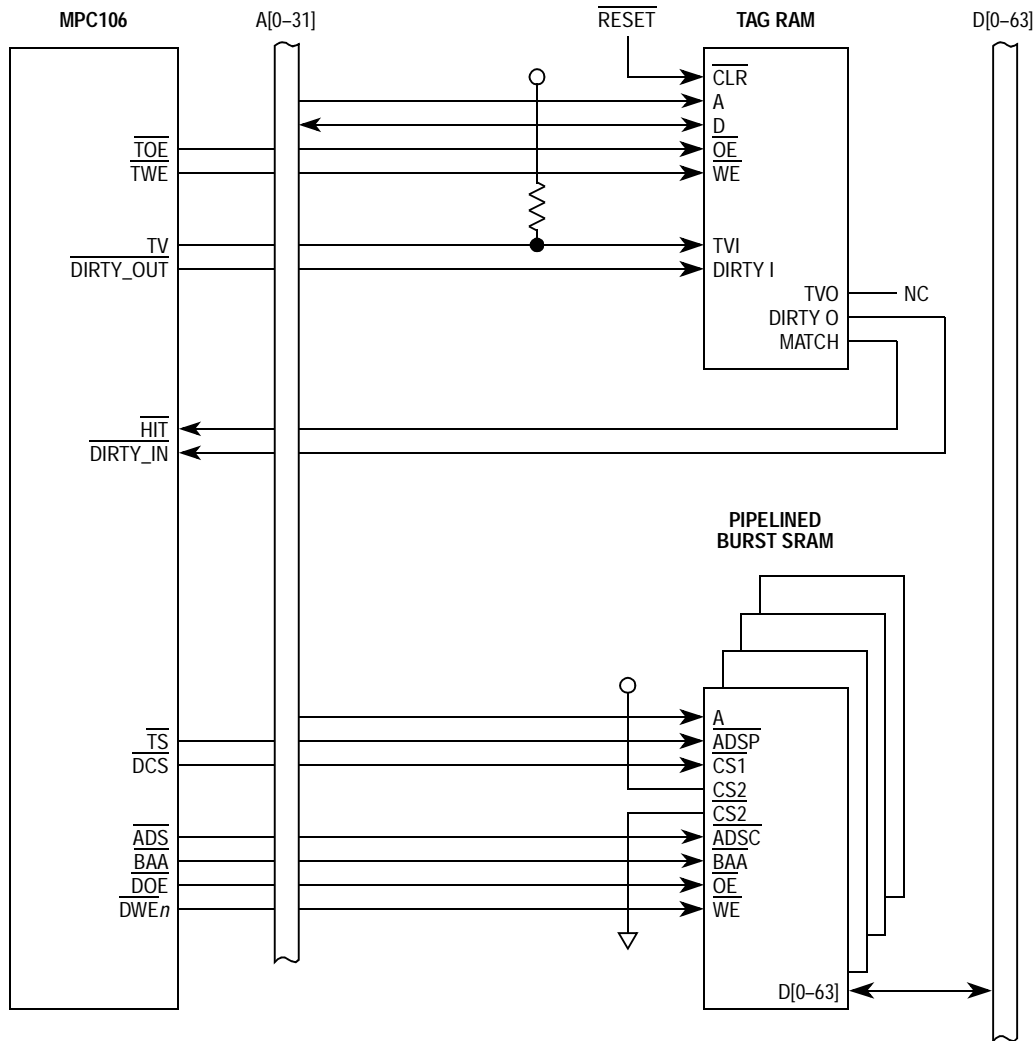


Figure 5-3. Alternate L2 Cache Using Pipelined Burst SRAM (Write-Back Using ADSP)

When $\overline{\text{ADSP}}$ is used, there are several constraints on the design. First, the $\overline{\text{DCS}}$ output of the MPC106 must be connected to the qualifying $\overline{\text{CS}}$ input (that is, the chip select that qualifies the $\overline{\text{ADSP}}$ signal) on the pipelined burst SRAM for proper pipeline operations. Second, early write timing ($\text{CF_WMODE} = 0b11$) and two-bank operation are not supported when $\overline{\text{ADSP}}$ is used. Finally, in this configuration, $\overline{\text{TS}}$ is connected directly to $\overline{\text{ADSP}}$ and the 60x address signals are connected to the L2 data RAM address inputs. Loading on these signals must be considered. There must be sufficient setup time for the MPC106 and the L2 data RAMs.

5.1.5 Asynchronous SRAMs

The internal L2 cache controller of the MPC106 supports using asynchronous SRAMs as L2 data RAM. A typical implementation is shown in Figure 5-4.

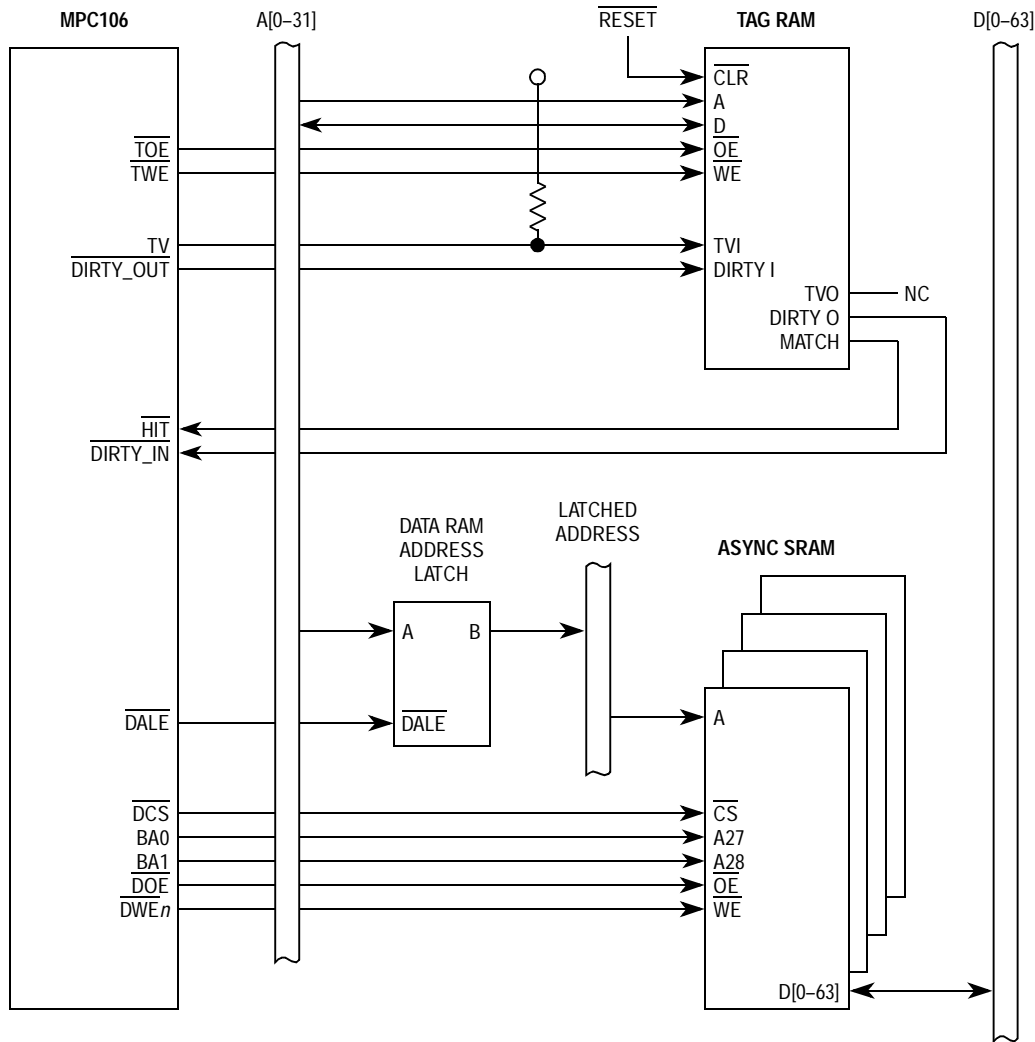


Figure 5-4. Typical L2 Cache Using Asynchronous SRAM (Write-Back)

When the MPC106 is configured for asynchronous SRAM, the signals \overline{ADS} and \overline{BAA} are redefined to \overline{DALE} and BA1. The two low-order address bits for the SRAMs are provided by a burst output counter (BA0 and BA1) in the MPC106. The remaining SRAM address inputs are provided by an external address latch that preserves the low-order 60x bus address bits when \overline{DALE} is asserted.

The MPC106 asserts \overline{DCS} when a data transfer is in progress and \overline{DOE} and \overline{DWE}_n for read and write transactions. In order to meet write timing requirements, \overline{DWE}_n is typically connected directly to the asynchronous SRAMs. Therefore, the MPC106 must be set to no partial update write timing (CF_WMODE = 0b00) when using asynchronous SRAMs.

5.1.6 Two-Bank Support

For cache sizes of 512 Kbytes and 1 Mbyte, the MPC106 supports operations using two banks of L2 data RAMs. Note that the MPC106 only supports two banks for caches using synchronous burst SRAMs or pipelined burst SRAMs (ADSP only). Two-bank operation is not supported for asynchronous SRAMs or pipelined burst SRAMs that use ADSP.

When configured for two banks (CF_TWO_BANKS = 1), the MPC106 uses bank detection logic to add turnaround cycles when switching between the L2 banks. For 512-Kbyte, two-bank caches, A13 is used to select the appropriate bank; for 1-Mbyte, two-bank caches, A12 is used to select the appropriate bank.

A typical 512-Kbyte, two-bank L2 cache implementation using synchronous burst SRAMs is shown in Figure 5-5. A 1-Mbyte, two-bank L2 cache implementation using pipelined burst SRAMs is shown in Figure 5-6.

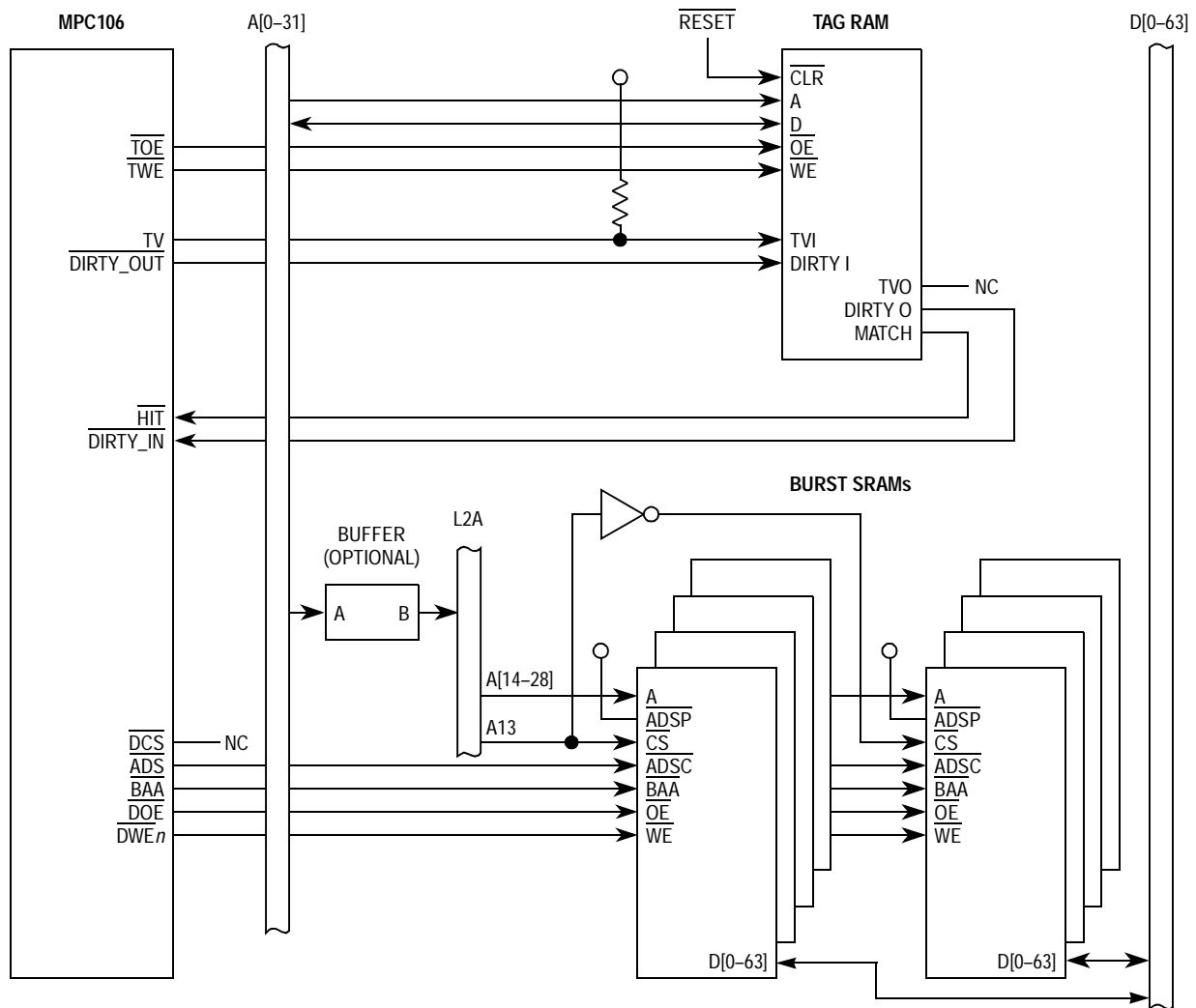


Figure 5-5. 512-Kbyte, Two-Bank, L2 Cache Using Synchronous Burst SRAM (Write-Back)

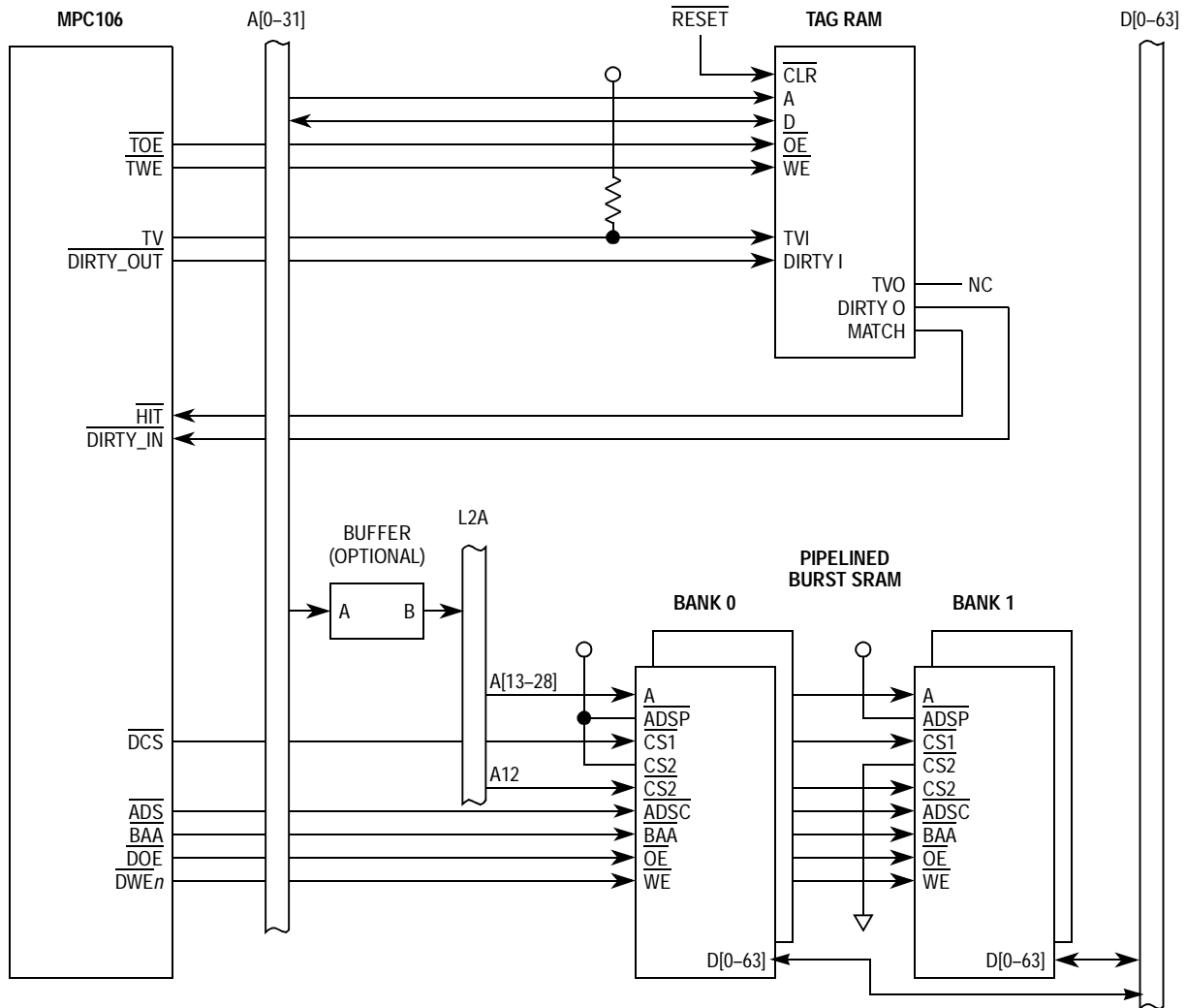


Figure 5-6. 1-Mbyte, Two-Bank, L2 Cache Using Pipelined Burst SRAM (Write-Back, ADSC Only)

5.2 Internal L2 Cache Controller Operation

This section describes the operation of the internal L2 cache controller. Section 5.6, “External L2 Cache Controller Operation,” describes the external L2 controller operation.

5.2.1 L2 Cache Addressing

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. 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 castout address mask (CF_CBA_MASK). For example, with a cache size of

512 Kbytes, an 8-bit tag RAM (plus the TV bit) can be used by masking off the five high-order address bits to provide a cacheable space of 128 Mbytes. See Section 3.2.9, “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 Kbytes	A[14–26] (13 bits)	A[0–13] (14 bits)	A[14–28] (15 bits)
512 Kbytes	A[13–26] (14 bits)	A[0–12] (13 bits)	A[13–28] (16 bits)
1 Mbyte	A[12–26] (15 bits)	A[0–11] (12 bits)	A[12–28] (17 bits)

5.2.2 L2 Cache Line Status

The high-order address bits, TV, and $\overline{\text{DIRTY_OUT}}$ signals are used to update the tag RAM and dirty RAM with new line status. The tag RAM and dirty RAM are updated at the assertion of $\overline{\text{TWE}}$.

The $\overline{\text{RESET}}$ 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. When invalidate mode is enabled, any 60x transaction causes the L2 controller to invalidate the tag entry indexed by the 60x address. Note that invalidate occurs regardless of the state of the CF_L2_MP and CF_L2_EN parameters, as well as the state of the $\overline{\text{HIT}}$ and $\overline{\text{DIRTY_IN}}$ signals. However, the L2 cache interface parameters, CF_L2_SIZE, CF_HIT_HIGH, CF_MOD_HIGH, CF_L2_HIT_DELAY, and CF_HOLD, must be programmed for proper tag write timings and proper valid and dirty bit polarity before using the invalidate mode.

5.2.3 L2 Cache Tag Lookup

When both the $\overline{\text{TWE}}$ and $\overline{\text{TOE}}$ signals are negated, the tag RAM is in tag lookup mode. During 60x bus operations, L1 copy-back operations, and MPC106-initiated snoop operations, the MPC106 uses the status of the $\overline{\text{HIT}}$ and $\overline{\text{DIRTY_IN}}$ signals to determine the current L2 line status and responds accordingly. The polarity of the $\overline{\text{HIT}}$ and $\overline{\text{DIRTY_IN}}$ signals is programmable.

The TV signal can be used with tag RAMs with separate I/O valid bits or one bidirectional valid bit. Note that the TV signal is either released to a high-impedance state or always driven during tag read operations depending on the parameters CF_HOLD and CF_FAST_CASTOUT in PICR2. For tag RAMs with separate I/O valid bits, the TV signal from the MPC106 is connected to the valid input of the tag RAM. The MPC106 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

MPC106 to work with tag RAMs that use the TV signal input for the lookup comparison. The TV signal is in a high-impedance state when the MPC106 is configured for a uniprocessor system without an L2 cache, or a multiprocessor system.

5.2.4 L2 Cache Castout Operations

For L2 castouts, the MPC106 normally drives $\overline{\text{ARTRY}}$ to retry the 60x processor transaction that caused the L2 castout and performs a tag RAM and data RAM read to retrieve the dirty address and data. After the tag RAM read completes, the processor restarts the memory transaction that caused the castout. Since the tag RAM read takes a considerable number of clock cycles, it delays the memory access. In the case of a processor burst read that causes a castout, the MPC106 does not retry the processor read transaction. Instead, the MPC106 asserts $\overline{\text{AACK}}$ for the processor transaction, and starts the L2 castout (tag RAM and data RAM read) and the memory read access in parallel. The memory read output enable ($\overline{\text{CAS}}_n$) signals are delayed until the L2 castout is completed. This effectively overlaps the $\overline{\text{RAS}}_n$ access of the memory transaction with the L2 castout, so the processor read data is ready as soon as the L2 castout is completed.

To get the address for L2 castouts, the MPC106 drives the low-order bits and asserts $\overline{\text{TOE}}$. The MPC106 latches only the address bits from the tag data signals during tag read transactions. The L2 cache line status is not used. Once the address is known for the line being written to memory, the line status in the L2 tag must be updated. When configured for normal castout timing ($\text{CF_FAST_CASTOUT} = 0$), the MPC106 drives all the address bits, TV, and $\overline{\text{DIRTY_OUT}}$, and asserts $\overline{\text{TWE}}$ to perform the tag update. When configured for fast castout timing ($\text{CF_FAST_CASTOUT} = 1$), the MPC106 performs the tag update during the last clock of the tag read. In the fast castout mode, the TV and $\overline{\text{DIRTY_OUT}}$ signals are driven when $\overline{\text{TOE}}$ is asserted, and $\overline{\text{TWE}}$ is asserted during the last clock cycle of $\overline{\text{TOE}}$ to update the tag RAM status.

5.2.5 L2 Cache Parity

The MPC106 internal L2 cache controller supports parity generation and checking for data in the L2 cache. The parity signals from the L2 data RAMs connect to the 60x/memory parity signals. Parity generation and checking is controlled through the $\text{L2_PARITY_ERROR_ENABLE}$ parameter in error enabling register 2. Note that memory parity checking/generation (MCCR1[PCKEN]) must also be enabled.

The parity signals are valid with the data on writes to the L2 data RAMs. When reading from the L2 data RAMs, the MPC106 checks parity at the assertion of $\overline{\text{TA}}$.

Since the MPC106 uses the parity signals for ROM address bits during ROM accesses, L2 cache line fills from ROM cannot reflect correct parity. Therefore, the MPC106 does not perform parity checking during L2 cache read operations within the ROM address space. Processor parity checking should be disabled when 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 MPC106.

5.2.6 L2 Cache Interface and Interrupt Vector Relocation

When the MPC106 is configured for interrupt vector relocation, all memory accesses from 0xFFFF0_0000 to 0xFFFF_FFFF are translated to a 1-Mbyte memory block lower in the memory address space. The 12 most-significant bits of the address (0xFFFF) of the 60x memory access are replaced by the contents of ESCR1[INT_VECTOR_RELOCATE], and the remaining 20 address bits are unchanged. Interrupt vector relocation presents special problems with respect to the L2 cache and snooping. The L2 tag RAM sees only the untranslated address (0xFFFFx_xxxx) and uses the 60x bus address (prior to translation by the MPC106) for lookups and tag updates. Software must handle any coherency issues regarding accesses to the original vector addresses and their translated locations.

Snooping always uses the original 60x interrupt vector address rather than the translated (relocated) address. Software must maintain cache coherency if PCI devices access the memory region used for vector relocation. Because the L2 cache uses the original vector address, it treats the access as a normal system ROM space access with respect to determining whether the data should be cached. If the system ROM space is located on the PCI bus, the RAM region is not cached.

5.3 L2 Cache Response to Bus Operations

The MPC106 samples the \overline{WT} , \overline{CI} , \overline{GBL} , \overline{ARTRY} , \overline{HIT} , and $\overline{DIRTY_IN}$ signals and responds with the activity required by the bus operation. The internal L2 cache controller only supports operations mapped in the system memory address space, and ignores memory operations mapped in the PCI space. The internal L2 cache controller supports the following four types of bus operations:

- Normal 60x bus operations (any 60x-initiated operations, with the exception of L1 copy-back operations)
- 60x L1 copy-back operations
- L2 castout operations (when configured as a write-back cache)
- Snoop operations due to PCI-to-system-memory transactions, provided snooping is enabled (PICR2[NO_SNOOP_EN] = 0)

The following sections describe the internal L2 cache controller responses to bus operations in both write-back and write-through configurations.

5.3.1 Write-Back L2 Cache Response

When the MPC106 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 write hits (depending on partial update configuration), or burst write hits, and burst write misses. Table 5-2 describes the internal L2 cache controller 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	ARTRY	L2 Hit	L2 Line Status	New L2 Line Status	L2 Controller Response	MPC106 Operation
Normal 60x Bus Operations							
—,R/RWITM	x0x	—	Hit	—	—	L2 → CPU	Stop MEM access.
B,R/RWITM	x0x	—	Miss	inv/um	um	MEM → L2 → um	MEM → CPU
B,R/RWITM with L2_UPDATE_EN = 0	x0x	—	Miss	inv/um	—	—	MEM → CPU
B,R/RWITM	x0x	A (106)	Miss	mod	—	If copy-back buffer is available, L2 → CBB, MEM → L2 If copy-back buffer is not available, set next state as invalid. Wait for L2 copy-back mode.	If copy-back buffer is available, L2 → CBB, MEM → CPU/L2 If copy-back buffer is not available, stop MEM access. Save low-order bit address. ARTRY the 60x read transaction. Goes to L2 copy-back mode.
B,R/RWITM with L2_UPDATE_EN = 0	x0x	—	Miss	mod	—	—	MEM → CPU
B,RWITM with CF_RWITM_FILL = 1	x0x	—	Miss	—	—	—	MEM → CPU
SB,R/RWITM	x0x	—	Miss	—	—	—	MEM → CPU
—,R/RWITM	x1x	—	Hit	um	inv	→ invalid	MEM/PCI → CPU
—,R/RWITM	x1x	A (106)	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 transaction. 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	00x	—	Hit	—	inv	→ invalid	CPU → MEM
B,W	10x	—	Hit	—	um	CPU → L2 → um	CPU → MEM
B,W with L2_UPDATE_EN = 0	10x	—	Hit	—	inv	→ invalid	CPU → MEM

Table 5-2. Write-Back L2 Cache Response (continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	ARTRY	L2 Hit	L2 Line Status	New L2 Line Status	L2 Controller Response	MPC106 Operation
B,W	00x	—	Miss	inv/um	mod	CPU → L2 → mod	Stop MEM access.
B,W with L2_UPDATE_EN = 0	00x	—	Miss	inv/um	—	—	CPU → MEM
B,W	10x	—	Miss	inv/um	um	CPU → L2 → um	CPU → MEM
B,W with L2_UPDATE_EN = 0	10x	—	Miss	inv/um	—	—	CPU → MEM
B,W	x0x	A (106)	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 transaction. Goes to L2 copy-back mode.
B,W with L2_UPDATE_EN = 0	x0x	—	Miss	mod	—	—	CPU → MEM
SB, W with partial update	00x	—	Hit	—	mod	CPU → L2 → mod	Stop MEM access.
SB, W with partial update and L2_UPDATE_EN = 0	00x	—	Hit	um	inv	→ invalid	CPU → MEM
SB, W with partial update and L2_UPDATE_EN = 0	00x	A (106)	Hit	mod	—	Set next state as invalid. Wait for L2 copy-back mode.	Stop MEM access. Save low-order bit address. ARTRY the 60x write transaction. Goes to L2 copy-back mode
SB, W with partial update	10x	—	Hit	um	—	CPU → L2	CPU → MEM
SB, W with partial update and L2_UPDATE_EN = 0	10x	—	Hit	um	inv	→ invalid	CPU → MEM
SB, W with partial update	10x	A (106)	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 transaction. Goes to L2 copy-back mode.
SB, W with partial update	x0x	—	Miss	—	—	—	CPU → MEM
SB, W without partial update	x0x	—	Hit	um	inv	→ invalid	CPU → MEM

Table 5-2. Write-Back L2 Cache Response (continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	ARTRY	L2 Hit	L2 Line Status	New L2 Line Status	L2 Controller Response	MPC106 Operation
SB, W without partial update	x0x	A (106)	Hit	mod	—	Set next state as invalid. Wait for L2 copy-back mode.	Stop MEM access. Save low-order bit address. ARTRY the 60x write transaction. Goes to L2 copy-back mode
—,W	x1x	—	Hit	um	inv	—> invalid	CPU —> MEM/PCI
B,W	x1x	—	Hit	mod	inv	—> invalid	CPU —> MEM/PCI
SB,W	x1x	A (106)	Hit	mod	—	Set next state as invalid. Wait for L2 copy-back mode	Stop MEM access. Save low-order bit address. ARTRY the 60x write transaction. Goes to L2 copy-back mode.
—,W	x1x	—	Miss	—	—	—	CPU —> MEM/PCI
—,Clean	x0x	—	Hit	um	—	—	—
—,Clean	x0x	A (106)	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 with CF_ADDR_ONLY_ DISABLE = 1	x0x	—	Hit	mod	—	—	—
—,Clean	x0x	—	Miss	—	—	—	—
—,Clean	x1x	—	Hit	um	inv	—	—
—,Clean with CF_ADDR_ONLY_ DISABLE = 1	x1x	—	Hit	um	—	—	—
—,Clean	x1x	A (106)	Hit	mod	—	Set next state as invalid. Wait for L2 copy-back mode	Save low-order bit address. ARTRY the 60x transaction. Goes to L2 copy-back mode
—,Clean with CF_ADDR_ONLY_ DISABLE = 1	x1x	—	Hit	mod	—	—	—
—,Clean	x1x	—	Miss	—	—	—	—
—,Flush	x0x	—	Hit	um	inv	—> invalid	—

Table 5-2. Write-Back L2 Cache Response (continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	ARTRY	L2 Hit	L2 Line Status	New L2 Line Status	L2 Controller Response	MPC106 Operation
—, Flush with CF_ADDR_ONLY_ DISABLE = 1	x0x	—	Hit	um	—	—	—
—, Flush	x0x	A (106)	Hit	mod	—	Set next state as invalid. Wait for L2 copy-back mode.	Save low-order bit address. ARTRY the 60x transaction. Goes to L2 copy-back mode.
—, Flush with CF_ADDR_ONLY_ DISABLE = 1	x0x	—	Hit	mod	—	—	—
—, Flush	x0x	—	Miss	—	—	—	—
—, Flush	x1x	—	Hit	um	inv	—> invalid	—
—, Flush with CF_ADDR_ONLY_ DISABLE = 1	x1x	—	Hit	um	—	—	—
—, Flush	x1x	A (106)	Hit	mod	—	Set next state as invalid. Wait for L2 copy-back mode	Save low-order bit address. ARTRY the 60x transaction. Goes to L2 copy-back mode
—, Flush with CF_ADDR_ONLY_ DISABLE = 1	x1x	—	Hit	mod	—	—	—
—, Flush	x1x	—	Miss	—	—	—	—
—, Kill/ ICBI	x0x	—	Hit	—	inv	—> invalid	Invalidate internal buffers if hit.
—, Kill/ ICBI with CF_ADDR_ONLY_ DISABLE = 1	x0x	—	Hit	—	—	—	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 with CF_ADDR_ONLY_ DISABLE = 1	x0x	—	Hit	—	—	—	Invalidate internal buffers if hit.
—, Kill/ ICBI	x1x	—	Miss	—	—	—	Invalidate internal buffers if hit.

Table 5-2. Write-Back L2 Cache Response (continued)

Bus Operation (Single-Beat or Burst, Read or Write)	WIM	ARTRY	L2 Hit	L2 Line Status	New L2 Line Status	L2 Controller Response	MPC106 Operation
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	MPC106 detects a 60x snoop-push to the snooped address and does not assert ARTRY. If PCI read or PCI read w/lock, run 60x transaction and send data to memory and PCI. If PCI write, run 60x transaction and merge 60x data with PCI write data and write to memory. Exit L1 copy-back mode. If in loop-snoop mode, repeat snoop.
— with L2_UPDATE_EN = 0	xxx	—	Hit	—	PCI read, PCI read w/ lock, PCI write: inv	—> invalid	MPC106 detects a 60x snoop-push to the snooped address and does not assert ARTRY. If PCI read or PCI read w/ lock, run 60x transaction and send data to memory and PCI. If PCI write, run 60x transaction 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	—	—	—	MPC106 detects a 60x snoop-push to the snooped address and does not assert ARTRY. Run 60x transaction. If PCI read or PCI read w/ lock snoop, send data to memory and PCI. If PCI write, send PCI data 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	MPC106 Operation
—	xxx	A (106)	—	—	—	—	MPC106 detects a 60x snoop-push that does not match the snooped address. MPC106 asserts $\overline{\text{ARTRY}}$ to retry 60x transaction. Exit L1 copy-back mode. If in loop-snoop mode, repeat snoop.
L2 Copy-Back Operations							
—	—	—	—	—	um / inv	Send dirty address onto address bus. Burst data onto data bus. \rightarrow next state.	Grant address bus to L2. Capture copy-back address. Grant data bus to L2. Send L2 data to memory. Enter normal mode.
PCI Bus Snoop Operations							
—,—	xx1	A (60x)	—	—	—	—	If 60x asserts $\overline{\text{BR}}$ in the window of opportunity, then save snoop type (PCI read, read w/ lock, or write). Enter L1 copy-back mode. Grant bus to 60x. Else if no $\overline{\text{BR}}$ assertion, repeat snoop.
—,R	xx1	N (60x)	Hit	—	—	L2 \rightarrow MPC106	L2 \rightarrow PCI.
—,R	xx1	N (60x)	Miss	—	—	—	MEM \rightarrow PCI.
—,W	xx1	N (60x)	Hit	um	inv	\rightarrow invalid	PCI \rightarrow MEM.
—,W NK	xx1	N (60x)	Hit	mod	inv	L2 \rightarrow MPC106, \rightarrow invalid	L2 data merge with PCI data \rightarrow MEM.
—,W K	xx1	N (60x)	Hit	mod	inv	\rightarrow invalid	PCI \rightarrow 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	MPC106 Operation
—, W	xx1	N (60x)	Miss	—	—	—	PCI → MEM.

Key:

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

5.3.2 Write-Through L2 Cache Response

When the MPC106 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 write hits (depending on partial update configuration), or burst write hits, and burst write misses. Table 5-3 describes the internal L2 cache controller 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	MPC106 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	x0x	—	Miss	—	—	MEM → CPU
B, RWITM with CF_RWITM_FILL = 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	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	MPC106 Operation
B, W with L2_UPDATE_EN = 0	x0x	—	Miss	—	—	CPU → MEM
SB, W with partial update	x0x	—	Hit	—	CPU → L2	CPU → MEM
SB, W with partial update and L2_UPDATE_EN = 0	x0x	—	Hit	inv	→ invalid	CPU → MEM
SB, W without partial update	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 with CF_ADDR_ONLY_ DISABLE = 1	x1x	—	Hit	—	—	—
—, Clean	x1x	—	Miss	—	—	—
—, Flush	x0x	—	Hit	inv	→ invalid	—
—, Flush with CF_ADDR_ONLY_ DISABLE = 1	x0x	—	Hit	—	—	—
—, Flush	x0x	—	Miss	—	—	—
—, Flush	x1x	—	Hit	inv	→ invalid	—
—, Flush with CF_ADDR_ONLY_ DISABLE = 1	x1x	—	Hit	—	—	—
—, Flush	x1x	—	Miss	—	—	—
—, Kill/ICBI	x0x	—	Hit	inv	→ invalid	(Invalidate buffer)
—, Kill/ICBI with CF_ADDR_ONLY_ DISABLE = 1	x0x	—	Hit	—	—	(Invalidate buffer)
—, Kill/ICBI	x0x	—	Miss	—	—	(Invalidate buffer)
—, Kill/ICBI	x1x	—	Hit	inv	→ invalid	(Invalidate buffer)
—, Kill/ICBI with CF_ADDR_ONLY_ DISABLE = 1	x1x	—	Hit	—	—	(Invalidate buffer)
—, Kill/ICBI	x1x	—	Miss	—	—	(Invalidate buffer)

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	MPC106 Operation
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	MPC106 detects a 60x snoop-push to the snooped address. If PCI read, or PCI read w/ lock, send CPU data to memory and PCI. If 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.
—, — with L2_UPDATE_EN = 0	xxx	—	Hit	PCI read, PCI read w/ lock, or PCI write snoop: inv.	→ invalid	MPC106 detects a 60x snoop-push to the snooped address. If PCI read, or PCI read w/ lock, send CPU data to memory and PCI. If 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	—	Miss	—	—	MPC106 detects a 60x snoop-push to the snooped address. If PCI read, or PCI read w/ lock, send CPU data to memory and PCI. If 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 (106)	—	—	—	MPC106 detects a 60x snoop-push that does not match the snooped address. MPC106 asserts ARTRY to retry the 60x transaction. 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	MPC106 Operation
PCI Bus Snoop Operations						
—, —	xx1	A (60x)	—	—	—	If the 60x asserts \overline{BR} in the window of opportunity, then save snoop-type (PCI read, PCI read w/ lock, or PCI write); Enter L1 copy-back mode; grant bus to 60x. Else repeat snoop.
—, R	xx1	N (60x)	Hit	—	L2 → MPC106	L2 → PCI
—, R	xx1	N (60x)	Miss	—	—	MEM → PCI
—, W	xx1	N (60x)	Hit	inv	→ invalid	PCI → MEM
—, W	xx1	N (60x)	Miss	—	—	PCI → MEM

Key:

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

5.4 L2 Cache Interface Parameters

The L2 cache interface parameters, located in PICR1 and PICR2, allow the MPC106 to support a variety of L2 cache configurations and timings. The MPC106's default power-up configuration holds the L2 cache disabled so that system software can program all the L2 cache interface parameters prior to enabling the L2 interface. Some of the parameters only affect the internal L2 cache controller interface, while others affect both the internal L2 cache controller interface and external L2 cache controller interface.

The following sections describe the L2 cache interface parameters. Refer to Chapter 3, "Device Programming," for additional information about the specific programming of the L2 cache interface parameters.

5.4.1 L2 Cache Interface Control Parameters

The L2 cache interface control parameters control specific operations of the L2 cache interface, and can be modified whether the internal L2 cache interface is enabled or disabled. The L2 cache interface control parameters are:

- **CF_EXTERNAL_L2**—Specifies if the external L2 cache interface is enabled.
- **CF_L2_MP**—Specifies single or multiprocessor configuration, and write-through or write-back L2 cache interface configuration.
- **L2_UPDATE_EN**—Specifies if the internally-controlled L2 cache can be updated. This L2 parameter can also be set through port 0x81C.
- **L2_EN**—Specifies if the internal L2 cache interface is enabled. Can also be configured through port 0x81C. Note that the **CF_L2_MP** parameter must also indicate a single processor with write-back or write-through L2 cache configuration to enable the internal L2 cache controller.
- **CF_FLUSH_L2**—Setting this configuration parameter causes the internal L2 cache controller to flush all modified lines to memory, and to invalidate all L2 cache lines. This configuration parameter can also be accessed through port 0x81C. Note that an L2 flush can only occur if the internal L2 cache controller is enabled.

Note that **L2_UPDATE_EN**, **CF_FLUSH_L2**, and **L2_EN** have no effect on the external L2 cache controller operation. However, it is possible for the external L2 cache controller to monitor these parameters at port 0x81C and use them to perform similar functions.

5.4.2 L2 Cache Interface Initialization Parameters

The L2 cache interface initialization parameters control the configuration and operational behavior of the L2 cache interface, and can only be changed when the L2 cache interface is disabled. These parameters must be set properly before the L2 cache is enabled and the L2 cache interface must be disabled before modifying these parameters.

The L2 cache interface initialization parameters are as follows:

- **CF_CBA_MASK**—Specifies which bits of the dirty address read from the tag RAM are valid. For systems requiring less than 4 Gbytes of cacheable space, this parameter allows the tag RAM width to be reduced. See Section 5.2.1, “L2 Cache Addressing,” for more information on this parameter.
- **CF_CACHE_1G**—Specifies the memory space cached by L2 cache. Note that this parameter also affects the external L2 cache controller interface.
- **CF_FAST_L2_MODE**—Specifies if fast L2 mode timing is enabled. The use of fast L2 mode is supported only by the 604. Note that this parameter also affects the external L2 cache controller interface.
- **CF_DATA_RAM_TYPE**—Specifies the type of SRAM used by the L2 cache.

- CF_WMODE—Specifies L2 data RAM write timing and partial update mode. See Section 5.4.2.4, “CF_WMODE,” for more information on this parameter.
- CF_MOD_HIGH—Specifies the polarity of the $\overline{\text{DIRTY_IN}}$, $\overline{\text{DIRTY_OUT}}$, and TV signals.
- CF_HIT_HIGH—Specifies the polarity of the $\overline{\text{HIT}}$ signal for the internal L2 cache controller interface only. Note that the $\overline{\text{HIT}}$ signal is always active low for the external L2 cache controller interface, regardless of the state of CF_HIT_HIGH.
- CF_ADDR_ONLY_DISABLE—Specifies whether the internal L2 controller responds to address-only transactions (Clean, Flush, and Kill).
- CF_HOLD—Specifies the hold time of the address, TV, and $\overline{\text{DIRTY_OUT}}$ signals with respect to the rising edge of the $\overline{\text{TWE}}$ signal.
- CF_INV_MODE—The L2 cache invalidate enable mode is used to initialize the tag contents before enabling the L2 cache in cases where hardware initialization of the tag and dirty RAM is not available. To flush the L2 cache, the CF_FLUSH_L2 (or port 0x81C[CF_FLUSH_L2]) configuration parameter can be set. See Section 5.2.2, “L2 Cache Line Status,” for more information on this parameter.
- CF_RWITM_FILL—Controls whether the internally-controlled L2 cache performs a line-fill when an RWITM miss occurs. See Section 5.3, “L2 Cache Response to Bus Operations,” for more information.
- CF_L2_HIT_DELAY—Specifies the earliest valid sampling point for the $\overline{\text{HIT}}$ and $\overline{\text{DIRTY_IN}}$ signals. Note that this parameter also affects the external L2 cache controller interface. See Section 5.4.2.1, “CF_L2_HIT_DELAY,” for more information on this parameter.
- CF_TWO_BANKS—Specifies the number of banks of L2 data RAM. See Section 5.1.6, “Two-Bank Support,” for more information on using two banks of L2 data RAM.
- CF_FAST_CASTOUT—Specifies timing of L2 castout operation. See Section 5.2.4, “L2 Cache Castout Operations,” for more information on fast castout timing.
- CF_TOE_WIDTH—Specifies the width of the active $\overline{\text{TOE}}$ pulse during L2 castout tag read operations.
- CF_L2_SIZE—Specifies L2 cache size.
- CF_DOE—Specifies the timing relation between the assertion of $\overline{\text{DOE}}$ and valid L2 data. See Section 5.4.2.2, “CF_DOE,” for more information on this parameter.
- CF_WDATA—Specifies $\overline{\text{ADSC}}$ -only or $\overline{\text{ADSP}}$ mode for pipelined burst SRAM configurations or the write pulse timing for asynchronous SRAM configurations. See Section 5.4.2.3, “CF_WDATA,” for more information on this parameter.

5.4.2.1 CF_L2_HIT_DELAY

CF_L2_HIT_DELAY specifies the earliest valid sampling point of the $\overline{\text{HIT}}$ and $\overline{\text{DIRTY_IN}}$ 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 2-1-1-1/1-1-1-1 pipelined), CF_L2_HIT_DELAY should be configured for a delay of one clock cycle. Note that the MPC106 may not sample the $\overline{\text{HIT}}$ and $\overline{\text{DIRTY_IN}}$ signals at the earliest sampling point, so these signals should be held valid as long as the address is valid. Note that CF_L2_HIT_DELAY controls the sampling point of the $\overline{\text{HIT}}$ signal for external L2 cache configurations. Figure 5-7 shows the earliest sampling points selected by the configuration of CF_L2_HIT_DELAY.

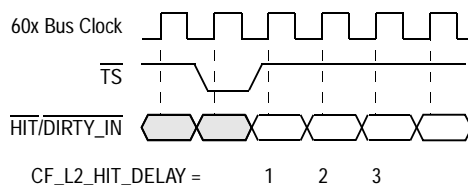


Figure 5-7. $\overline{\text{HIT}}$ and $\overline{\text{DIRTY_IN}}$ Delay Configuration

5.4.2.2 CF_DOE

CF_DOE specifies the time from $\overline{\text{DOE}}$ assertion to the data valid access time of the L2 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. If CF_DOE is set to 1, then the MPC106 will try to assert $\overline{\text{DOE}}$ speculatively at the end of the assertion of $\overline{\text{TS}}$ as long as $\overline{\text{DBGn}}$ is asserted in order to minimize the effect of the extra clock delay on read hits. Figure 5-15 and Figure 5-16 in Section 5.5, “L2 Cache Interface Timing Examples,” show synchronous burst SRAM read timing with CF_DOE = 0 and 1, respectively.

When using asynchronous SRAM, CF_DOE controls the first data beat access time of pipelined read transactions. 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 5-24 and Figure 5-25 in Section 5.5, “L2 Cache Interface Timing Examples,” show asynchronous SRAM read timing with CF_DOE = 0 and 1, respectively.

5.4.2.3 CF_WDATA

When using synchronous burst SRAMs, the CF_WDATA parameter is reserved and must be cleared to 0.

When using pipelined burst SRAMs, the CF_WDATA parameter configures the internal L2 cache controller for $\overline{\text{ADSC}}$ -only or $\overline{\text{ADSP}}$ mode. See Section 5.1.4, “Pipelined Burst SRAMs,” for more information on the two pipelined burst SRAM configurations.

When using asynchronous SRAMs, CF_WDATA controls the write pulse timing. If CF_WDATA is cleared to 0, $\overline{\text{DWEn}}$ is negated on the falling edge of the clock cycle in

which $\overline{\text{TA}}$ is asserted. If CF_WDATA is set to 1, $\overline{\text{DWE}n}$ and $\overline{\text{TA}}$ are negated on the same rising edge of the clock during cache line fills from memory. If CF_WDATA is set to 1, and the transaction is not a cache line fill from memory, the $\overline{\text{DWE}n}$ signals are negated on the falling edge of the clock in which $\overline{\text{TA}}$ is asserted (that is, the same timing as when CF_WDATA is cleared to 0).

Figure 5-26 and Figure 5-27 in Section 5.5, “L2 Cache Interface Timing Examples,” show asynchronous SRAM write data set-up timing with CF_WDATA = 0 and 1, respectively.

Note that when using asynchronous SRAMs, there must be sufficient data hold time when CF_WDATA is set to 1. It is recommended that the memory system employ a latched memory buffer that holds data valid for one clock after $\overline{\text{TA}}$ when CF_WDATA = 1. Also, the fastest cache line fill from memory is 3-3-3-3 when CF_WDATA = 1. Memory timing must be programmed accordingly.

5.4.2.4 CF_WMODE

CF_WMODE selects one of three L2 data RAM write timings or normal write timing without partial update. The following sections describe each of these settings.

5.4.2.4.1 Normal Write Timing without Partial Update (CF_WMODE = 0)

When CF_WMODE is set to 0 (0b00), normal write timing without partial update is selected. In this mode, the MPC106 can only update an entire double word at a time; it cannot issue single-byte writes to the L2 cache because there is no external byte decode logic present. Figure 5-9 shows normal write mode timing for pipelined and nonpipelined bus transactions with CF_WMODE set to 0, and CF_L2_HIT_DELAY set to 1.

Note that asynchronous SRAM configurations must use CF_WMODE = 0.

5.4.2.4.2 Normal Write Timing (CF_WMODE = 1)

When CF_WMODE is set to 1 (0b01), normal write timing with partial update is selected. The MPC106 assumes external byte decode logic is implemented with no external delays on the \overline{DWE}_n signals.

Figure 5-8 shows the logic required for external byte decode that requires CF_WMODE = 1.

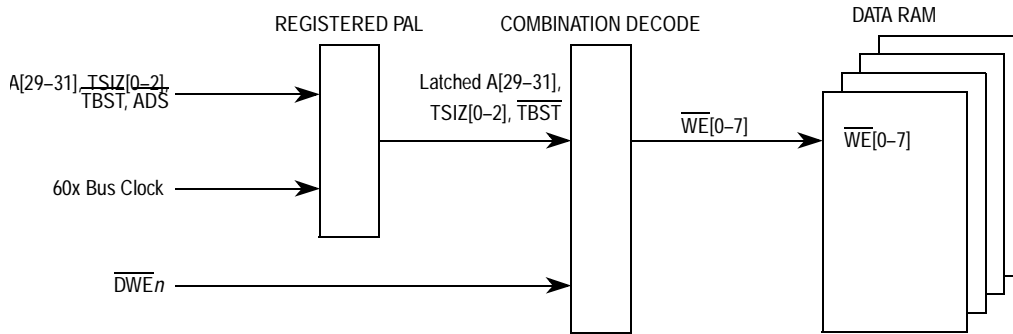


Figure 5-8. External Byte Decode Logic Requiring CF_WMODE = 1

Figure 5-9 shows the normal write mode timing associated with Figure 5-8 for pipelined and nonpipelined bus transactions with CF_WMODE set to 0 or 1, and CF_L2_HIT_DELAY set to 1.

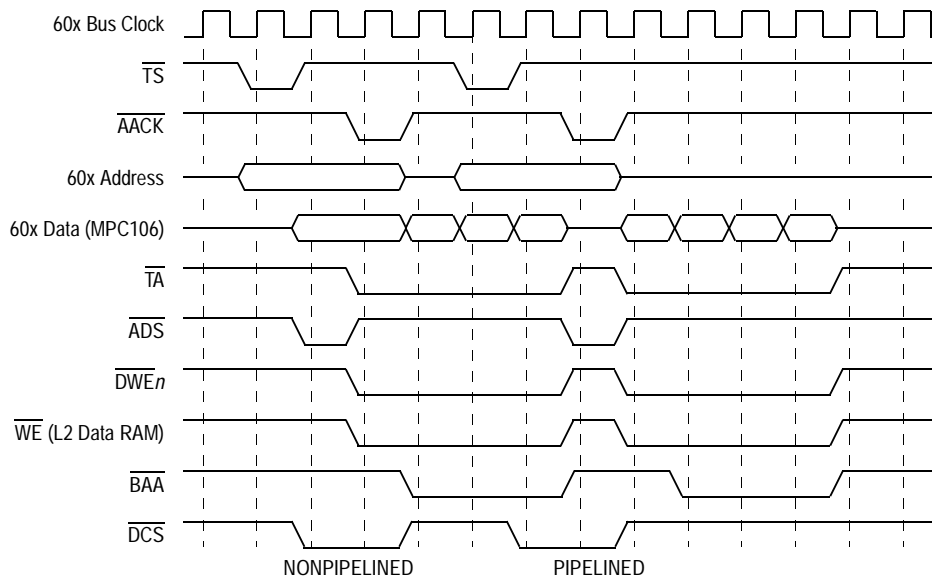


Figure 5-9. Normal Write Timing (CF_WMODE = 0 or 1)

5.4.2.4.3 Delayed Write Timing (CF_WMODE = 2)

When CF_WMODE is set to 2 (0b10), one clock cycle of delayed write timing is provided, allowing for one level of external logic for byte selection. Figure 5-10 shows the logic required for external byte decode that requires CF_WMODE to be set to 2.

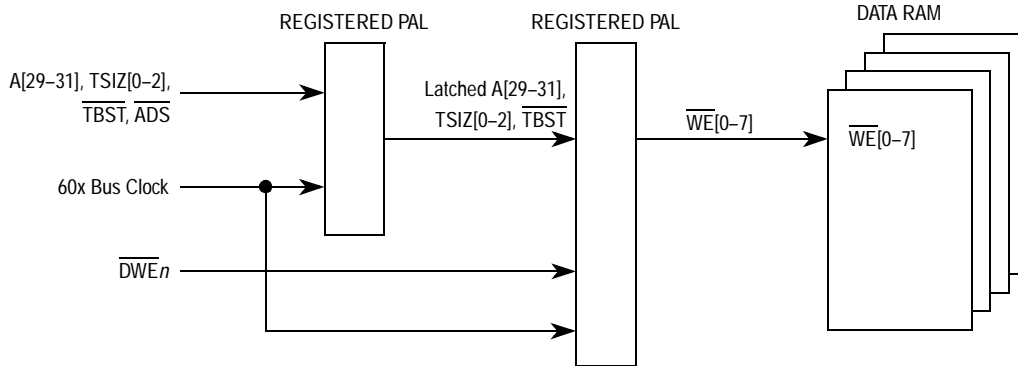


Figure 5-10. External Byte Decode Logic Requiring CF_WMODE = 2

Figure 5-11 shows the write mode timing associated with Figure 5-10 for pipelined and nonpipelined bus transactions with CF_WMODE set to 2, and CF_L2_HIT_DELAY set to 1.

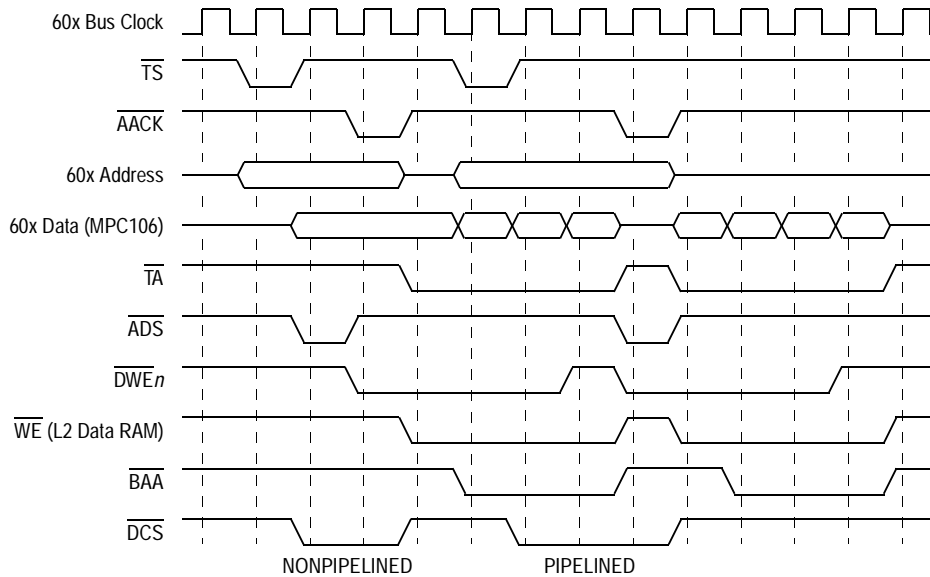


Figure 5-11. Delayed Write Timing (CF_WMODE = 2)

5.4.2.4.4 Early Write Timing (CF_WMODE = 3)

When CF_WMODE is set to 3 (0b11), early write timing is provided, and \overline{DWE} is asserted speculatively one clock cycle early to provide better write performance. Note that early write timing is not supported for synchronous pipelined SRAMs when \overline{ADSP} is used. Figure 5-12 shows the logic required for external byte decode that requires CF_WMODE to be set to 3.

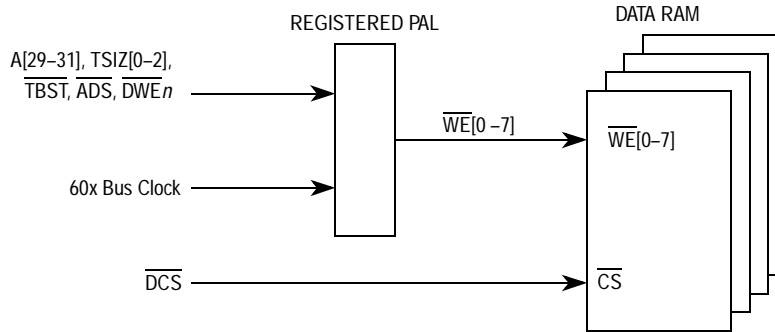


Figure 5-12. External Byte Decode Logic Requiring CF_WMODE = 3

Figure 5-13 shows the early write mode timing associated with Figure 5-12 for pipelined and nonpipelined bus transactions with CF_WMODE set to 3, and CF_L2_HIT_DELAY set to 1.

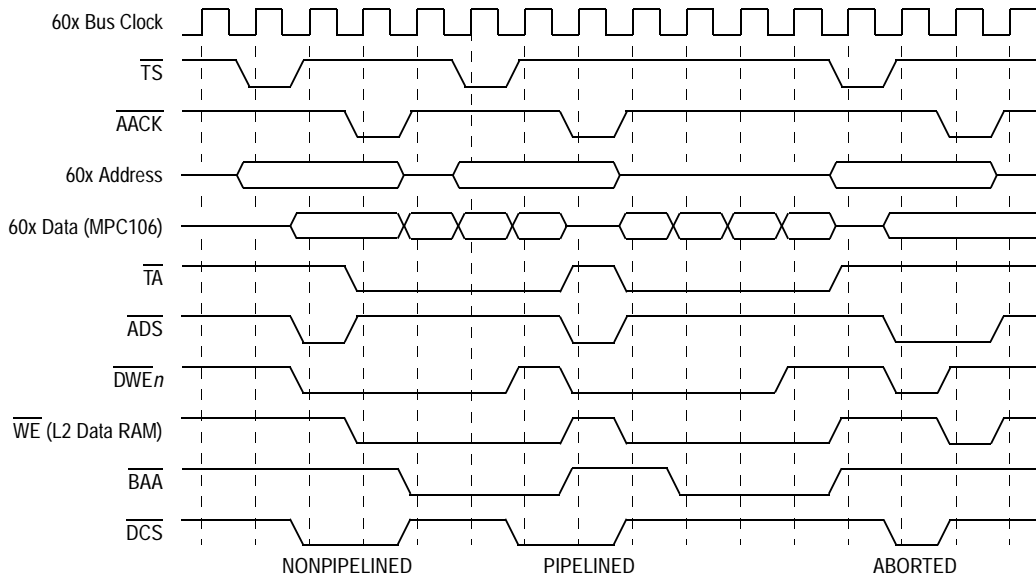


Figure 5-13. Early Write Timing (CF_WMODE = 3)

5.5 L2 Cache Interface Timing Examples

The figures in the following sections provide examples of the internal L2 cache controller interface signal timing in the course of cache read hits, cache write hits, cache line updates, L2 cache castout operations, and snoop operations. L2 cache timing examples are provided for a cache implemented with synchronous burst SRAMs and asynchronous SRAMs.

The symbols shown in Figure 5-14 are applicable to all the figures in the following sections.

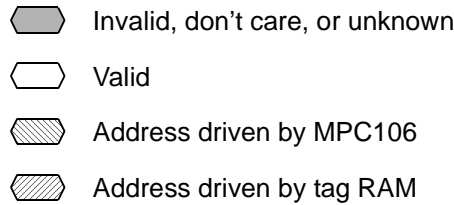


Figure 5-14. Timing Diagram Legend

5.5.1 Synchronous Burst SRAM L2 Cache Timing

The following sections provide timing examples for L2 cache operations in systems implemented using synchronous burst SRAM.

5.5.1.1 L2 Cache Read Hit Timing

Figure 5-15 shows the L2 interface timing for a read hit in the L2 cache. The L2 cache configuration parameters CF_APHASE_WS, CF_L2_HIT_DELAY, and CF_DPARK are set to 1, and CF_DOE is cleared to 0.

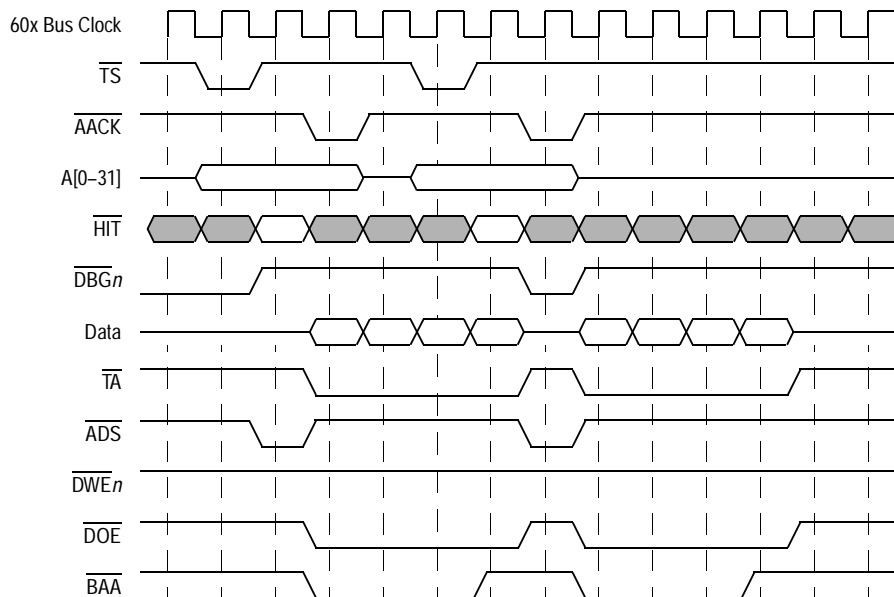


Figure 5-15. L2 Cache Read Hit Timing with CF_DOE = 0

Figure 5-16 shows the L2 interface timing for a read hit with configuration parameters, CF_APHASE_WS, CF_L2_HIT_DELAY, CF_DPARK, and CF_DOE set to 1.

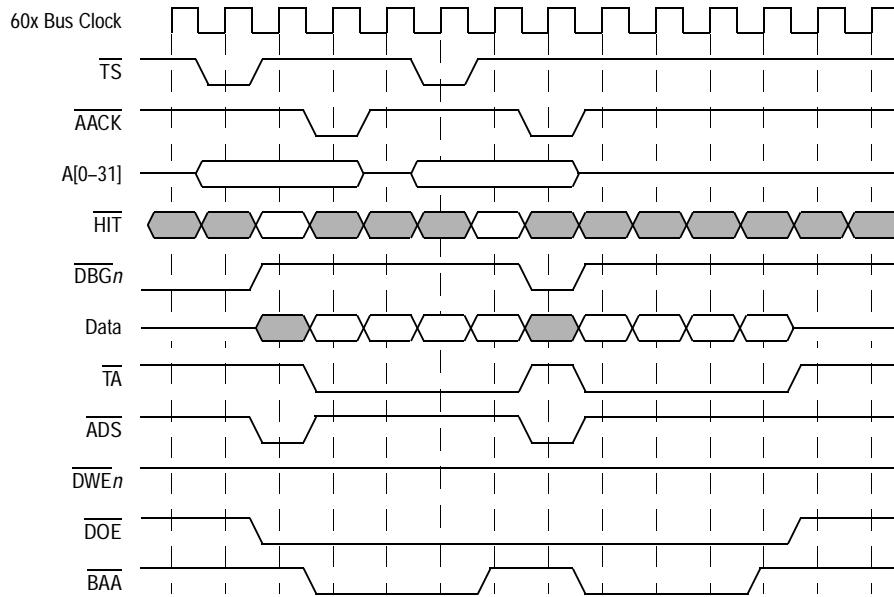


Figure 5-16. L2 Cache Read Hit Timing with CF_DOE = 1

5.5.1.2 L2 Cache Write Hit Timing

Figure 5-17 shows the L2 interface timing for a write hit in the L2 cache. The L2 cache configuration parameters 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 is set to 3.

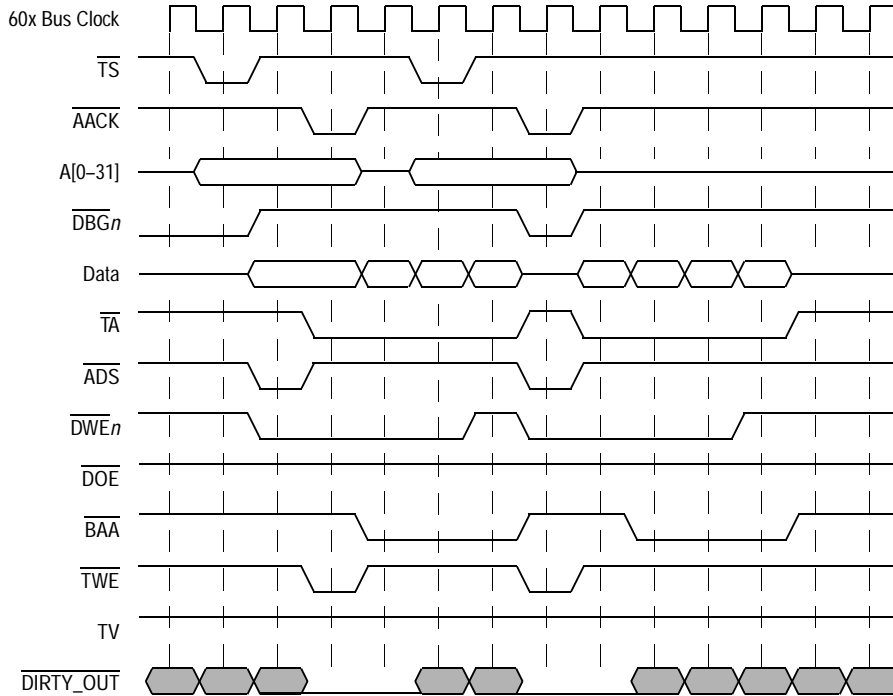


Figure 5-17. L2 Cache Write Hit Timing

5.5.1.3 L2 Cache Line Update Timing

Figure 5-18 shows the L2 interface timing for a cache line update (following a read miss). The L2 cache configuration parameters 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 is set to 3.

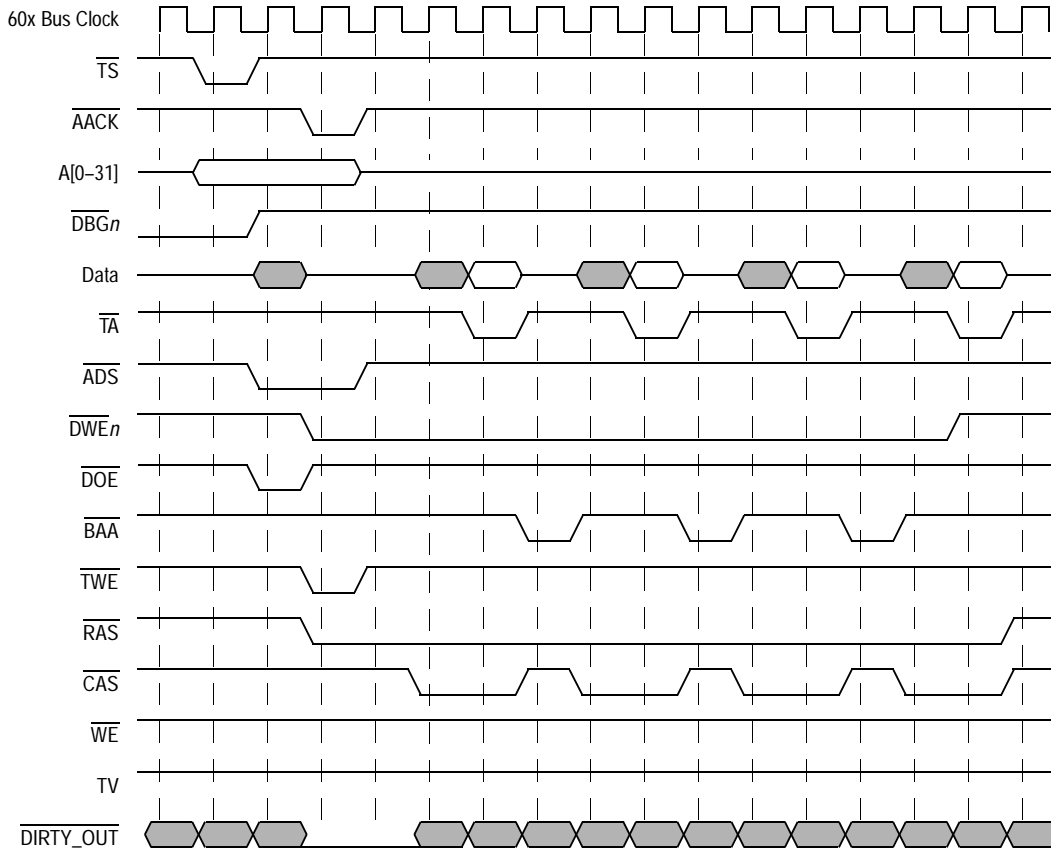


Figure 5-18. L2 Cache Line Update Timing

5.5.1.4 L2 Cache Line Castout Timing

Figure 5-19 shows the L2 interface timing for an L2 cache line castout. The L2 cache configuration parameters CF_APHASE_WS, CF_L2_HIT_DELAY, and CF_MOD_HIGH are set to 1, CF_DOE, CF_HOLD, CF_TOE_WIDTH, and CF_FAST_CASTOUT are cleared to 0.

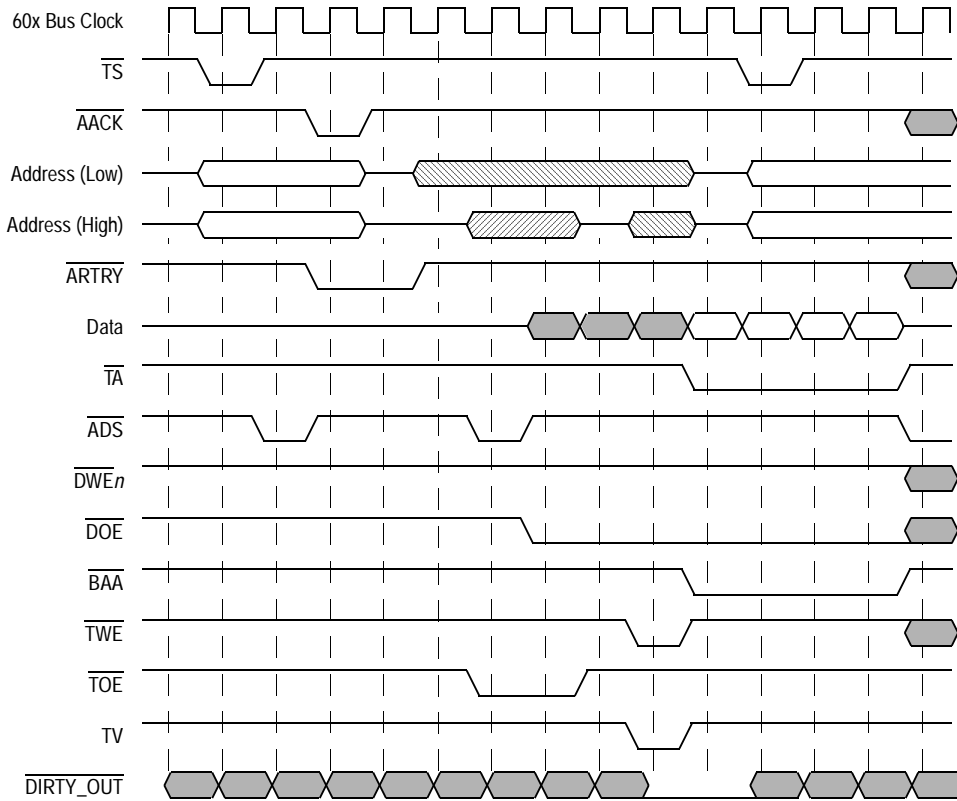


Figure 5-19. L2 Cache Line Castout Timing

Figure 5-20 shows the L2 interface timing for an L2 cache line castout. The L2 cache configuration parameters CF_APHASE_WS, CF_L2_HIT_DELAY, and CF_MOD_HIGH are set to 1, CF_DOE, CF_HOLD, and CF_TOE_WIDTH are cleared to 0.

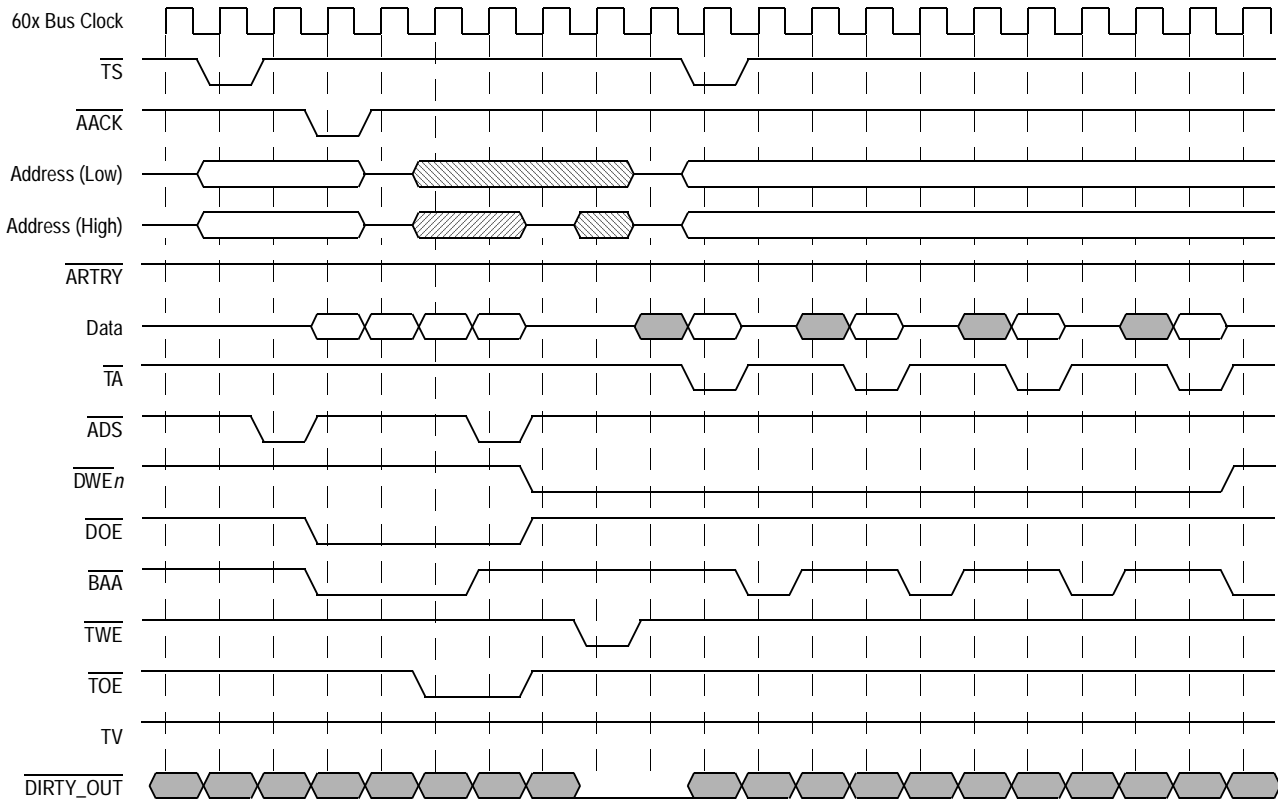


Figure 5-20. L2 Cache Line Castout Timing with No ARTRY

5.5.1.5 L2 Cache Hit Timing Following PCI Read Snoop

Figure 5-21 shows a read snoop (from a PCI-to-memory read operation) that misses in the L1 cache and then hits in the L2 cache. The L2 cache configuration parameters CF_APHASE_WS, CF_L2_HIT_DELAY, and CF_SNOOP_WS are set to 1, and CF_DOE is cleared to 0.

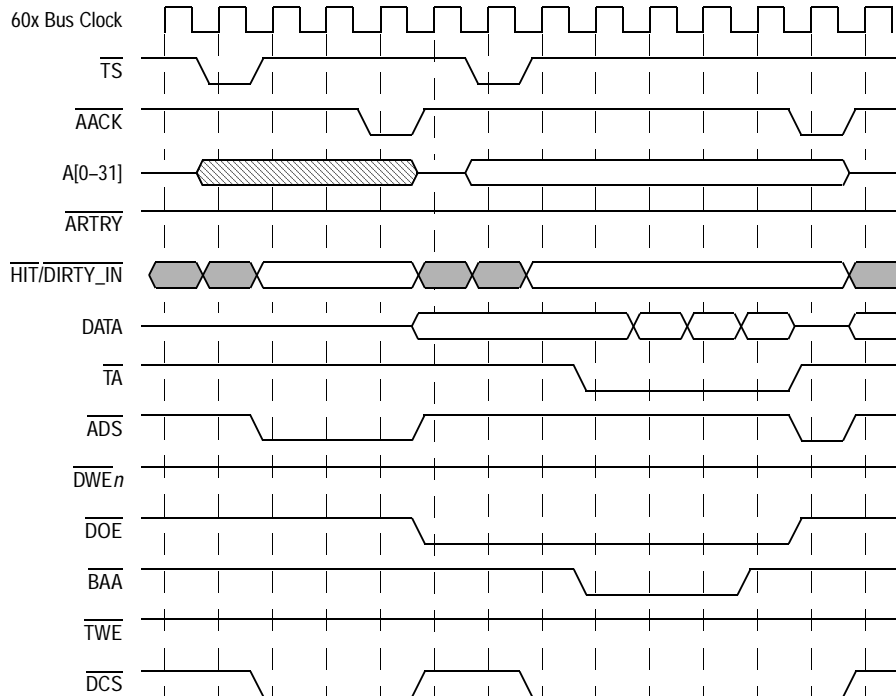


Figure 5-21. L2 Cache Hit Timing Following PCI Read Snoop

5.5.1.6 L2 Cache Line Push Timing Following PCI Write Snoop

Figure 5-22 shows a write snoop that misses in the L1 cache and then hits a modified line in the L2 cache. The L2 cache line is invalidated following the snoop push. The L2 cache configuration parameters 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.

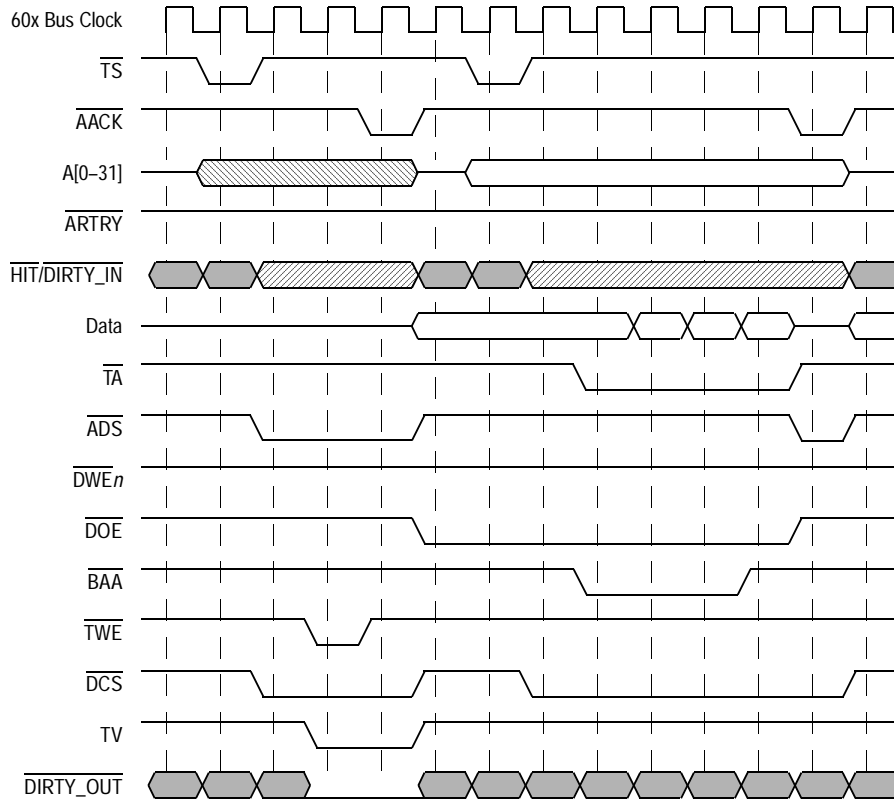


Figure 5-22. Modified L2 Cache Line Push Timing Following PCI Write Snoop

5.5.1.7 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 parameters 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.

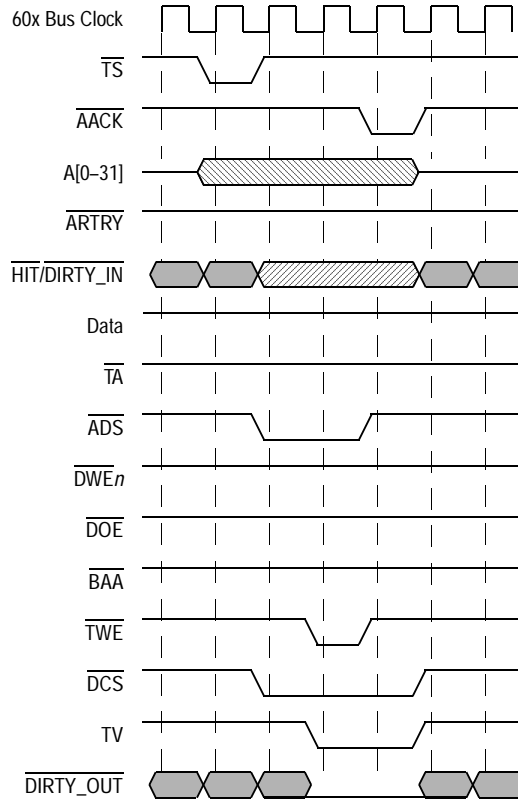


Figure 5-23. L2 Cache Line Invalidate Timing Following PCI Write-with-Invalidate Snoop

5.5.2 Asynchronous SRAM L2 Cache Timing

The following sections provide timing examples for L2 cache operations in systems implemented using asynchronous SRAM.

5.5.2.1 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 parameter CF_DOE is cleared to 0.

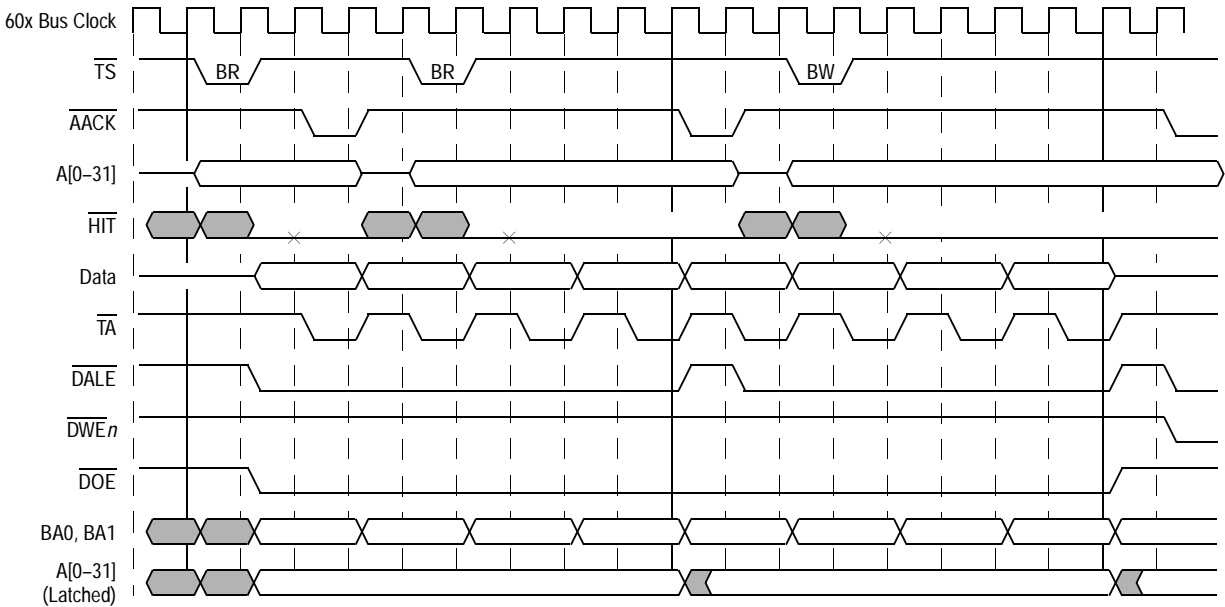


Figure 5-24. L2 Cache Burst Read Timing with CF_DOE = 0

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 parameter CF_DOE is set to 1.

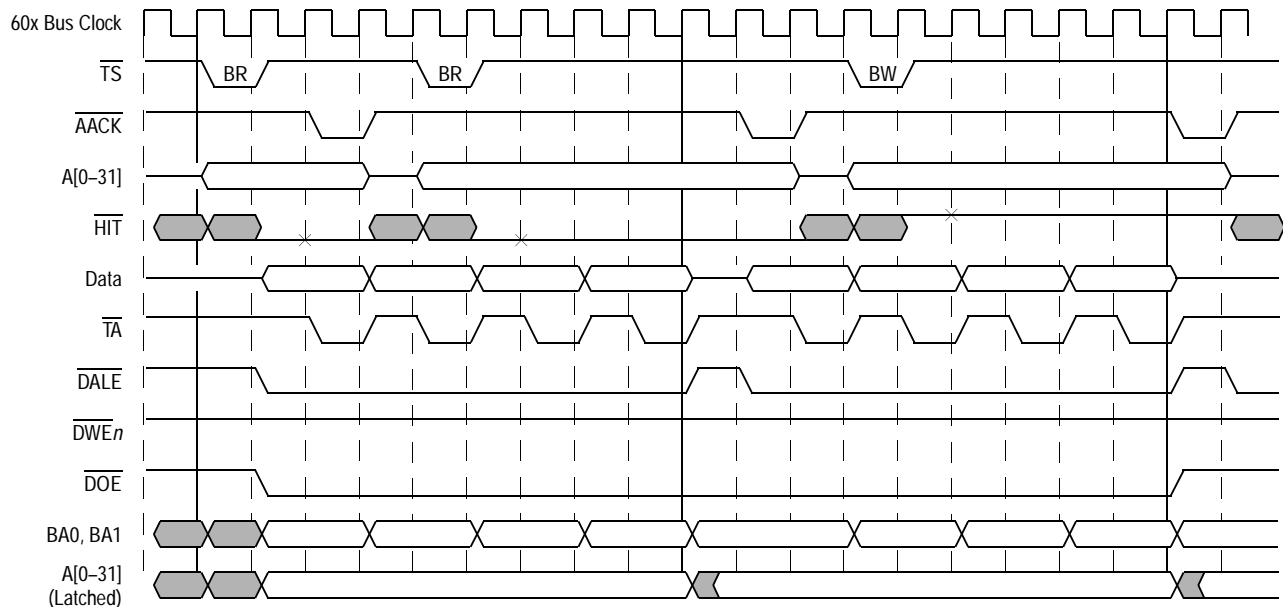


Figure 5-25. L2 Cache Burst Read Timing with CF_DOE = 1

5.5.2.2 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 parameter CF_WDATA is cleared to 0.

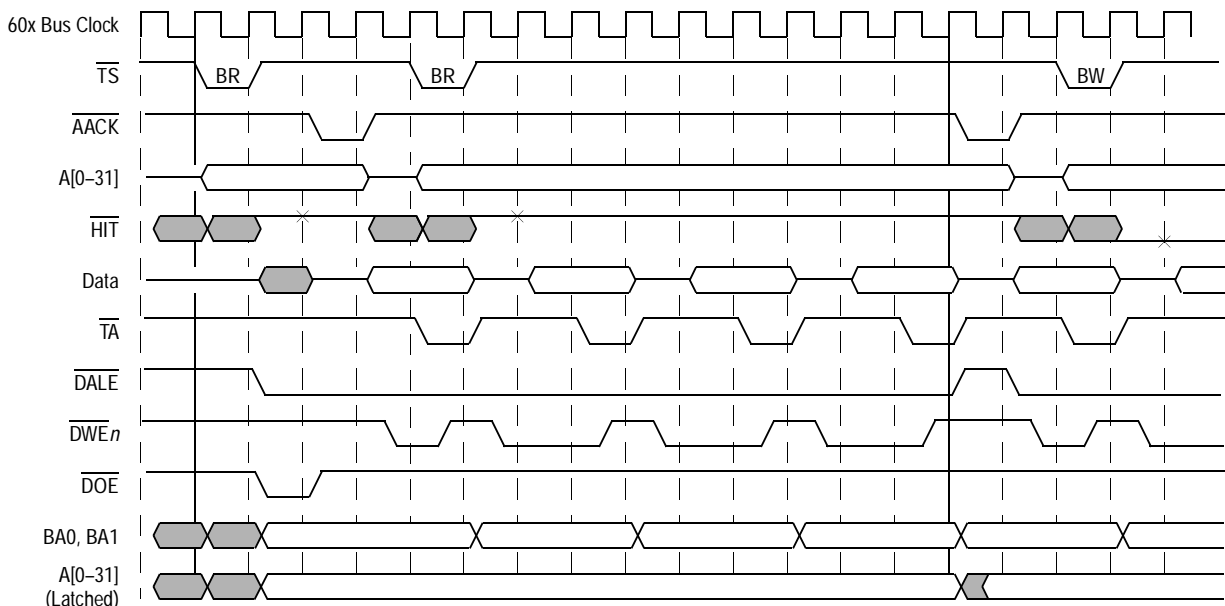


Figure 5-26. L2 Cache Burst Read Line Update Timing with CF_WDATA = 0

Figure 5-27 shows the L2 interface timing for an L2 cache line update during a burst read operation. The L2 cache configuration parameter CF_WDATA is set to 1.

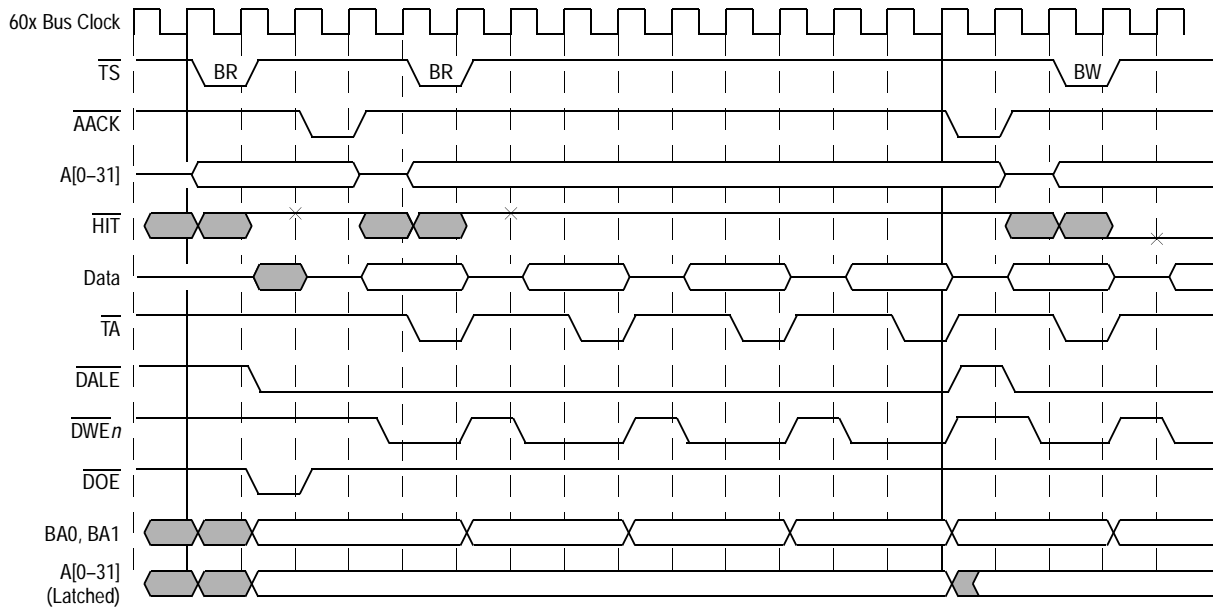


Figure 5-27. L2 Cache Burst Read Line Update Timing with CF_WDATA = 1

5.5.2.3 Burst Write Timing

Figure 5-28 shows the L2 interface timing for an L2 cache line update during a burst write operation. Note that the timing is the same for CF_WDATA = 0 or CF_WDATA = 1.

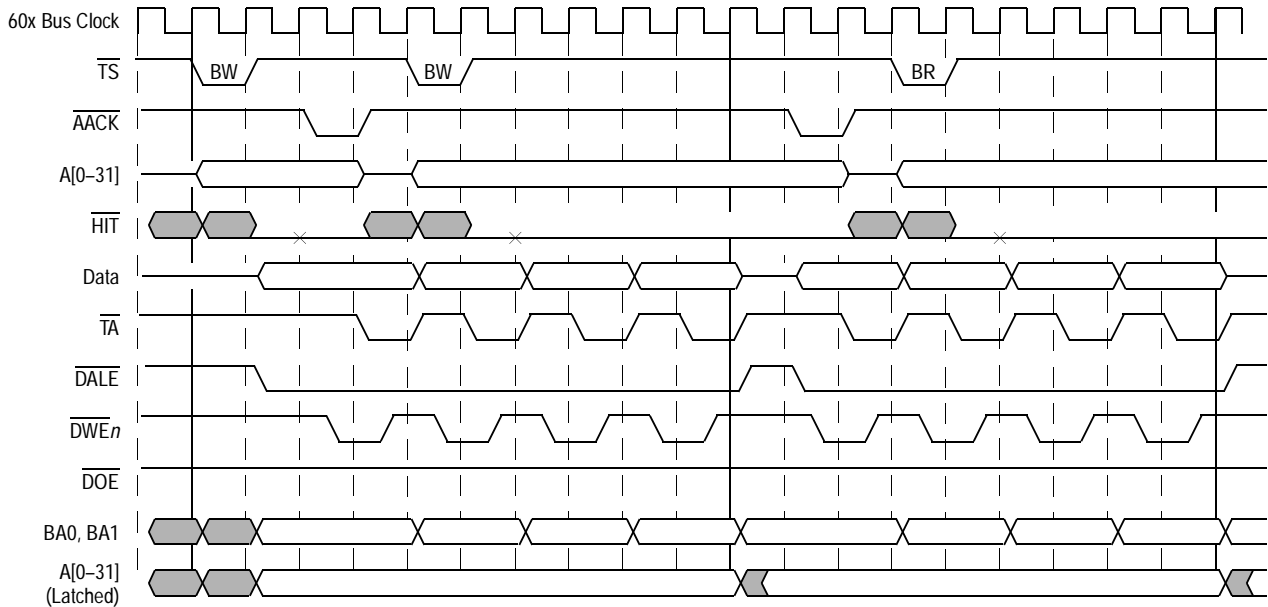
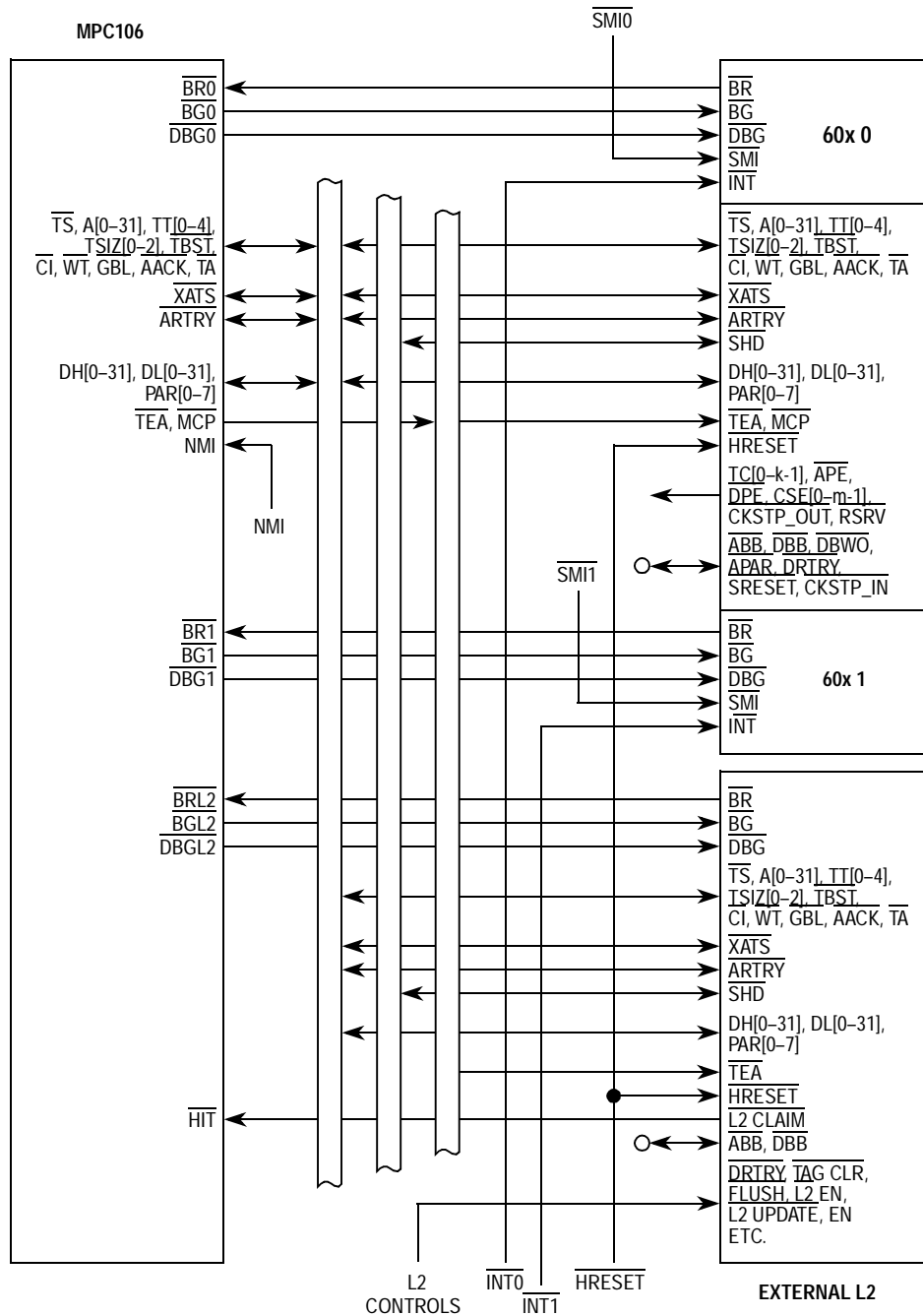


Figure 5-28. L2 Cache Burst Write Timing with CF_WDATA = 0 or 1

5.6 External L2 Cache Controller Operation

The MPC106 can support an external L2 cache controller in uniprocessor as well as multi-processor configurations. Figure 5-29 shows a typical system configuration with an external L2 cache. Note that the signal connections to the external L2 cache controller may vary depending on the external L2 cache implementation.



Tied to V_{DD} through a resistor individually.
All bidirectional signals should have a pull-up resistor.

Figure 5-29. Typical External L2 Cache Configuration

5.6.1 External L2 Cache Operation

When an externally-controlled L2 cache is used, the MPC106 samples the $\overline{\text{HIT}}$ input signal when CF_L2_HIT_DELAY expires. For 60x cycles, if $\overline{\text{HIT}}$ is asserted, the external L2 cache drives $\overline{\text{AACK}}$ and $\overline{\text{TA}}$ to complete the transaction without the MPC106 initiating a system memory transfer. The external L2 cache can assert $\overline{\text{ARTRY}}$ to retry 60x cycles, and requests the bus through $\overline{\text{BRL2}}$ to perform L2 castout operations. The MPC106 grants the address and data bus to the external L2 cache by asserting $\overline{\text{BGL2}}$ and $\overline{\text{DBGL2}}$, respectively. The external L2 cache must qualify the $\overline{\text{BGL2}}$ by $\overline{\text{ARTRY}}$ and take care not to take the bus in the $\overline{\text{ARTRY}}$ window or the window of opportunity. If the external L2 cache asserts $\overline{\text{ARTRY}}$, it should not assert $\overline{\text{HIT}}$. If the external L2 cache asserts $\overline{\text{ARTRY}}$, it should assert $\overline{\text{ARTRY}}$ on or before the clock cycle when $\overline{\text{HIT}}$ is valid and hold $\overline{\text{ARTRY}}$ asserted until one clock after $\overline{\text{AACK}}$.

If the external L2 cache asserts $\overline{\text{ARTRY}}$ for a snoop push, it should assert $\overline{\text{BRL2}}$ in the $\overline{\text{ARTRY}}$ window as well as in the window of opportunity. If a processor also asserts bus request in the window of opportunity, the MPC106 will grant the bus to the processor (an L1 snoop push has the highest priority and always takes precedence over an L2 snoop push). If a processor does not request the bus in the window of opportunity, the bus will be granted to the external L2 cache. Since a processor can assert $\overline{\text{ARTRY}}$ without asserting bus request (even though it has dirty data), the snoop has to be repeated after the external L2 cache snoop push until $\overline{\text{ARTRY}}$ is not asserted. A system using an external L2 cache controller must operate with snoop looping enabled ($\text{PICR1}[\text{CF_LOOP_SNOOP}] = 1$).

For a normal PCI write (not write-with-invalidate) snoop, the external L2 cache push data is merged with the PCI data. Then, the result is written to memory. For a PCI write-with-invalidate snoop, the external L2 cache should invalidate the cache line without a snoop push.

The system design must prevent PCI devices from reading from locations that are, or might be, modified in the external L2 cache. This can be accomplished by operating the external L2 in write-through mode, or by using caching-inhibited semaphores between the processor and PCI devices.

5.6.2 External L2 Cache Controller Interface Parameters

The CF_EXTERNAL_L2 and CF_L2_MP parameters are used to enable and disable the MPC106's external L2 cache controller interface. The following L2 cache interface parameters, described in Section 5.4, "L2 Cache Interface Parameters," should be set properly before the external L2 cache controller interface is enabled:

- CF_L2_HIT_DELAY
- CF_FAST_L2_MODE
- CF_CACHE_1G

Note that the external L2 cache modes are selected by the CF_EXTERNAL_L2 and CF_L2_MP parameters only. The external L2 cache controller interface is enabled whenever the external L2 cache modes are selected and disabled whenever the external L2 cache modes are not selected—the L2_EN parameter has no effect on the external L2 cache controller interface. Care should be exercised when enabling or disabling the external L2 cache controller interface on the MPC106. The external L2 cache must not claim any transaction (by asserting $\overline{\text{HIT}}$) when the MPC106's external L2 cache controller interface is disabled. Typically, the external L2 cache controller interface on the MPC106 is enabled before the external L2 cache is enabled and the external L2 cache is disabled before the external L2 cache controller interface on the MPC106 is disabled.



Chapter 6

Memory Interface

The memory interface of the MPC106 controls processor and PCI interactions with system memory. It is capable of supporting a variety of DRAM, EDO, or SDRAM, and ROM or Flash configurations as main memory. Note that only one of the RAM interfaces (DRAM, EDO, or SDRAM) can be used in a system; that is, a system cannot mix DRAM and EDO devices.

This chapter describes the memory interface on the MPC106—its features and limitations, buffering requirements, supported device organizations, initialization, and timing. Design examples are provided for the DRAM, EDO, SDRAM, ROM, and Flash interfaces.

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 MPC106 to coordinate memory accesses between the L2 cache, the 60x processor(s), and devices on the PCI bus.

6.1 Overview

The DRAM/EDO/SDRAM interface supports up to eight banks of 64-bit memory. Bank sizes up to 128 Mbytes provide for a maximum memory size of 1 Gbyte. Programmable parameters allow for a variety of DRAM/EDO/SDRAM organizations and timings. Note that if SDRAM is used, it must comply with the JEDEC specification for SDRAM. Two types of parity as well as ECC protection are provided for the DRAM/EDO.

The MPC106 handles parity checking and generation, with eight parity bits checked or generated for the 64-bit datapath. Read-modify-write (RMW) parity is also provided for write transactions less than the full 8-byte datapath. As an alternative to parity, the MPC106 supports ECC for the datapath to DRAM/EDO system memory. Using ECC, the MPC106 detects and corrects all single-bit errors, and detects all double-bit errors and all errors within a nibble. Note that the MPC106 does not directly support ECC for SDRAM memory configurations. However, the MPC106 can support an external, in-line, error-checking module (ECM)/buffer for use with SDRAM configurations.

The DRAM/EDO/SDRAM interface provides for doze, nap, sleep, and suspend power saving modes, defined in Appendix A, “Power Management.”

The ROM/Flash interface supports one or two banks of ROM/Flash memory on the 60x/memory bus. Bank sizes up to 8 Mbytes provide for a maximum ROM/Flash memory size of 16 Mbytes. The ROM space may also be mapped to the PCI bus or split between the 60x/memory bus and the PCI bus.

6.2 Memory Interface Signal Buffering

To reduce loading on the data bus, most system designs will require buffering between the 60x data bus and the memory data bus. The MPC106 features configurable data buffer control logic to accommodate flow-through, transparent latch, or registered data buffers. This section describes the different buffer control configurations of the MPC106. Note that in addition to the data and parity signals, certain other memory interface signals may also require buffering. The AC characteristics of the MPC106, memory operating frequency, capacitive loading, and transmission line effects of the board layout dictate which signals require buffering, and which buffer devices are appropriate. The example design in Figure 6-6 uses bidirectional/tri-state drivers on the data and parity signals.

The $\overline{\text{BCTL0}}$ and $\overline{\text{BCTL1}}$ signals control the data bus buffers (directional control and high-impedance state). The memory buffer type parameters (MCCR4[WCBUF] and MCCR4[RCBUF]) determine the type of buffer used and the data synchronization for that buffer type. The buffer mode (MCCR2[BUF_MODE]) parameter controls how the buffer control signals, $\overline{\text{BCTL0}}$ and $\overline{\text{BCTL1}}$, operate. The 501-mode (MCCR1[501_MODE]) parameter controls the polarity of the buffer control signal $\overline{\text{BCTL0}}$. The $\overline{\text{BCTL0}}$ signal is sampled at reset to determine the setting of 501_MODE. See Section 2.2.9, “Configuration Signals,” for more information.

Figure 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_MODE	501_MODE	Buffer Type	$\overline{\text{BCTL0}}$	$\overline{\text{BCTL1}}$	Typical Buffer Device
0	0	0	1	Flow-through	$\overline{\text{WE}}$	$\overline{\text{RE}}$	54/7416863
0	0	1	1	Flow-through	DIR (R/W)	$\overline{\text{OE}}$	54/7416245 54/74162245 54/74163245
0	1	0	1	Transparent latch (DRAM/EDO)	$\overline{\text{WE}}$	$\overline{\text{RE}}$	54/7416543 54/74162543
				Registered (DRAM/EDO)	$\overline{\text{WE}}$	$\overline{\text{RE}}$	54/7416952 54/74162952 54/7416601
0	1	1	1	—	DIR (R/W)	$\overline{\text{OE}}$	—
1	0	0	1	—	$\overline{\text{WE}}$	$\overline{\text{RE}}$	—

Table 6-1. Buffer Configurations (continued)

WCBUF	RCBUF	BUF_MODE	501_MODE	Buffer Type	$\overline{\text{BCTL0}}$	$\overline{\text{BCTL1}}$	Typical Buffer Device
1	0	1	1	—	DIR (R/W)	$\overline{\text{OE}}$	—
1	1	0	1	Registered (SDRAM)	$\overline{\text{WE}}$	$\overline{\text{RE}}$	54/7416952 54/74162952 54/7416601
1	1	1	1	—	DIR (R/W)	$\overline{\text{OE}}$	—
0	0	0	0	Flow-through	$\overline{\text{WE}}$	$\overline{\text{RE}}$	—
0	0	1	0	Flow-through	DIR (W/R)	$\overline{\text{OE}}$	—
0	1	0	0	Transparent latch/ Registered (DRAM/EDO)	$\overline{\text{WE}}$	$\overline{\text{RE}}$	—
0	1	1	0	—	DIR (W/R)	$\overline{\text{OE}}$	—
1	0	0	0	—	$\overline{\text{WE}}$	$\overline{\text{RE}}$	—
1	0	1	0	—	DIR (W/R)	$\overline{\text{OE}}$	—
1	1	0	0	Registered (SDRAM)	$\overline{\text{WE}}$	$\overline{\text{RE}}$	54/7416501
1	1	1	0	—	DIR (W/R)	$\overline{\text{OE}}$	—

6.2.1 Flow-Through Buffers

Both the WCBUF and RCBUF parameters should be cleared to indicate a flow-through buffer configuration. The MPC106 supports two protocols for the buffer control signals, $\overline{\text{BCTL0}}$ and $\overline{\text{BCTL1}}$. The buffer mode parameter, BUF_MODE, determines which protocol the buffer control signals use.

The default protocol (BUF_MODE = 1) uses $\overline{\text{BCTL0}}$ as a direction control signal (reads active high/writes active low), and $\overline{\text{BCTL1}}$ as a buffer output enable signal (active low). Buffers that are compatible with this protocol include the 54/7416245, 54/74162245, and 54/74163245. These buffers are available from several manufacturers. Connections to this buffer device should be made as shown in Figure 6-1.

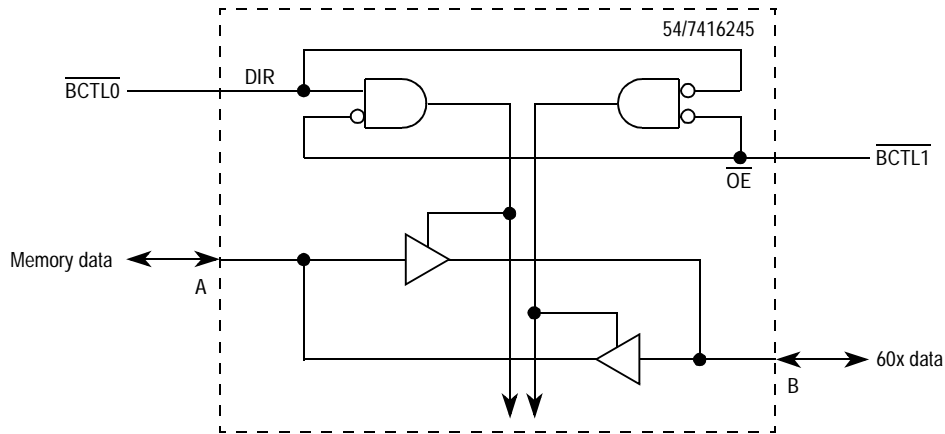


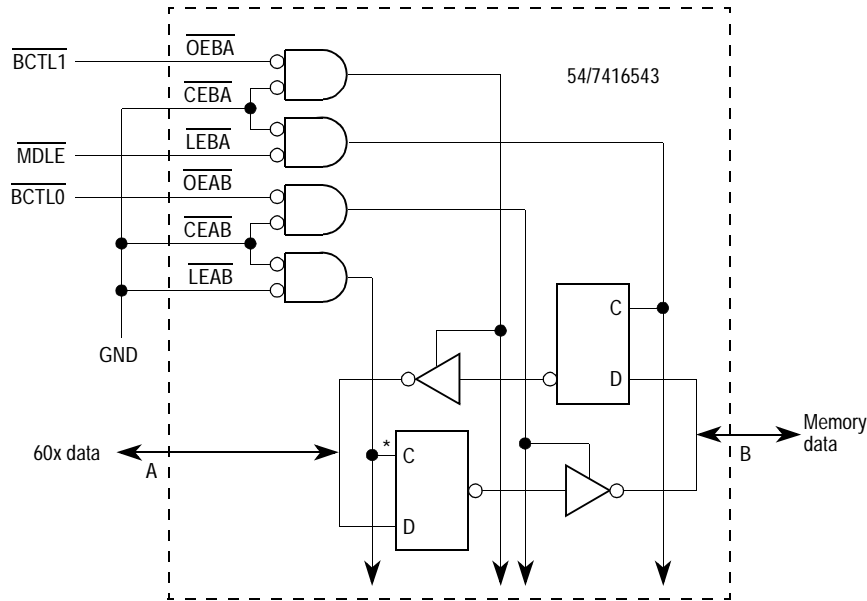
Figure 6-1. Flow-Through Buffer

The alternate protocol ($BUF_MODE = 0$) uses $\overline{BCTL0}$ as a buffer write enable signal (active low), and $\overline{BCTL1}$ as a read enable signal (active low). The 54/7416863, available from several manufacturers, is compatible with this protocol.

6.2.2 Transparent Latch Buffers

The $WCBUF$ parameter should be cleared and the $RCBUF$ parameter should be set to indicate the transparent latch configuration. The $\overline{BCTL0}$, $\overline{BCTL1}$ and \overline{MDLE} signals control the buffer. The $\overline{BCTL0}$ signal acts as a buffer write enable signal, and $\overline{BCTL1}$ acts as a read enable signal. \overline{MDLE} is used to latch the buffer to provide additional data hold time, as might be required by an asynchronous L2 cache (refer to the \overline{MDLE} waveform shown in Figure 6-9). Note that SDRAM memory systems cannot use latched buffers because the \overline{SDCAS} signal is multiplexed with the \overline{MDLE} signal.

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 6-2.



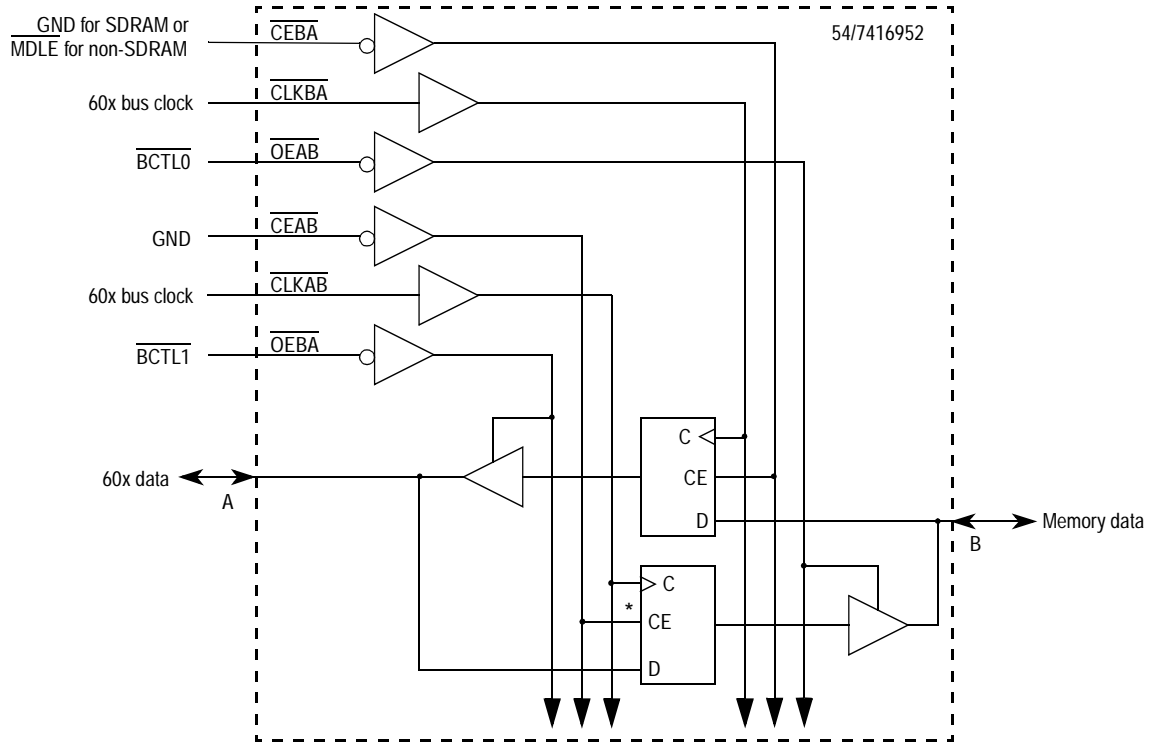
* Note that this configuration intentionally allows the write latch to remain in a flow-through state.

Figure 6-2. Transparent Latch-Type Buffer

6.2.3 Registered Buffers

Both the `WCBUF` and `RCBUF` parameters should be set to indicate the registered buffer configuration. The `BCTL0` and `BCTL1` signals control the buffer. `BCTL0` acts as a buffer write enable signal, and `BCTL1` acts as a read enable signal. The polarity of the `BCTL0` signal is programmable using the `501_MODE` parameter in `MCCR1`. In non-501-mode (`MCCR1[501_MODE] = 1`), `BCTL0` acts as an active-low, write-enable (\overline{WE}) signal. In 501-mode (`MCCR1[501_MODE] = 0`), `BCTL0` act as an active-high, write-enable (`WE`) signal.

Buffers that are compatible with the non-501-mode, registered configuration include the 54/7416952, 54/74162952, and 16601. These buffers are available from several manufacturers. Connections to the 16952-type buffer device should be made as shown in Figure 6-3.



* Note that the write-clock is always enabled.

Figure 6-3. 16952-Type Registered Buffer

Figure 6-4 shows the connections to a 16601-type registered buffer.

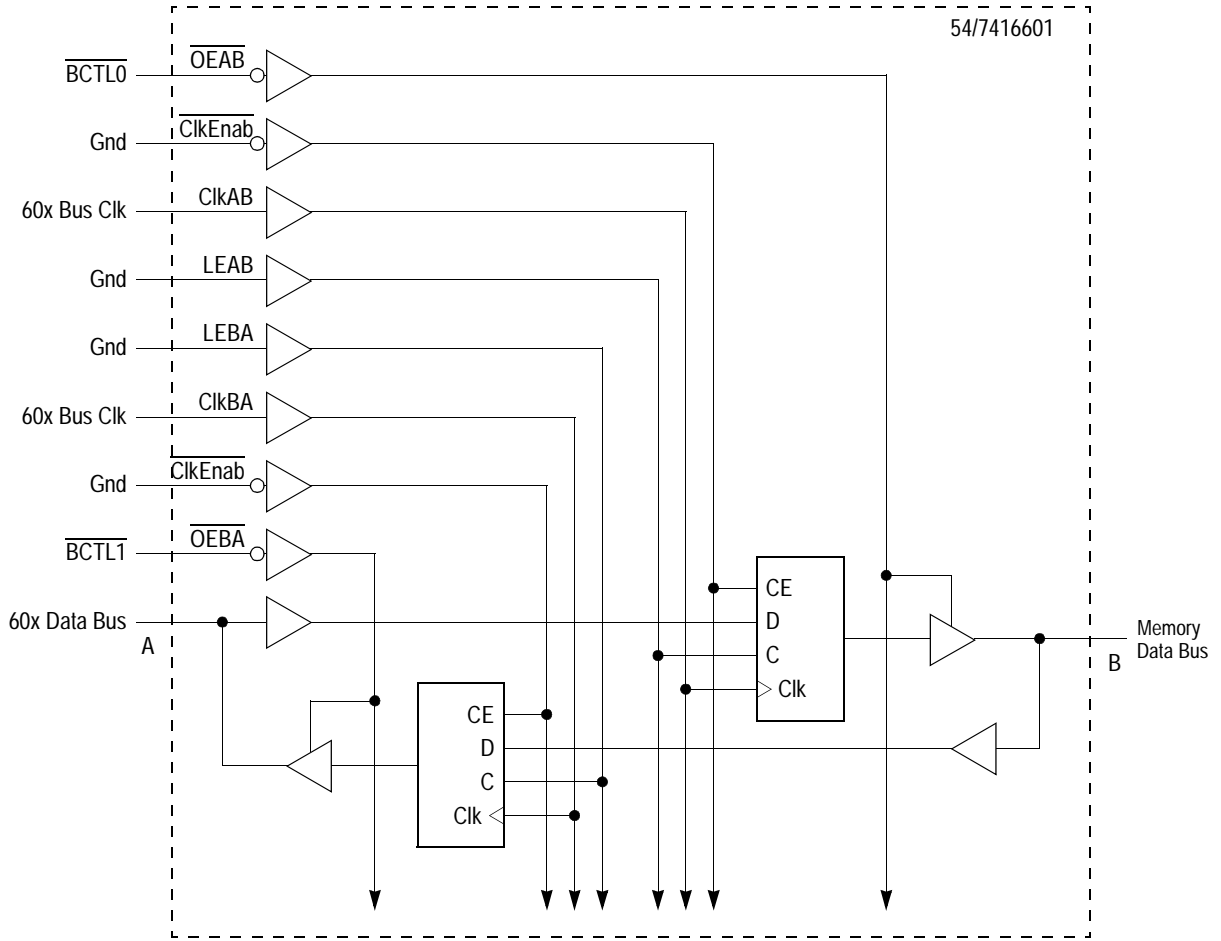


Figure 6-4. 16601-Type Registered Buffer

Figure 6-5 shows the connections to a 16501-type universal transceiver/buffer.

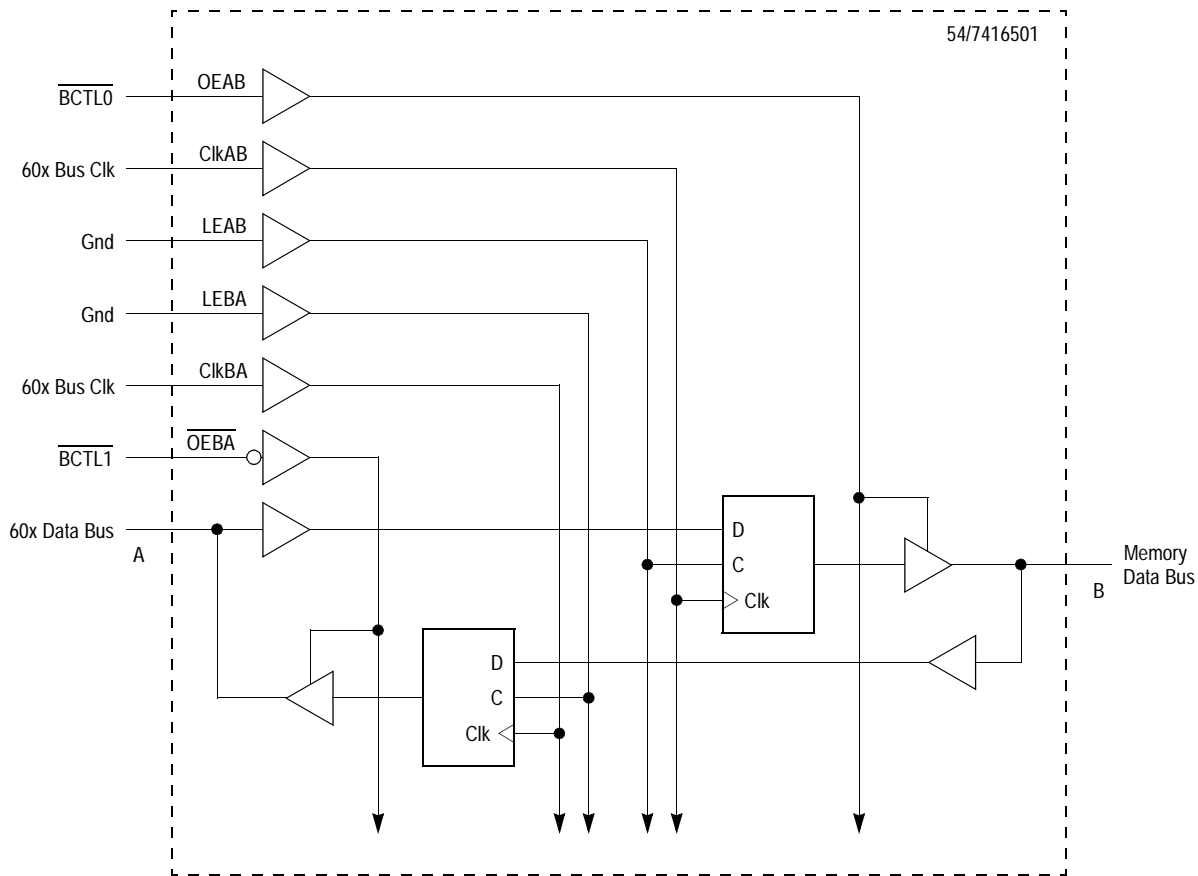


Figure 6-5. 16501-Type Universal Transceiver/Buffer

6.2.4 Parity/ECC Path Read Control

Since the MPC106 uses the parity/ECC signals for ROM/Flash address bits during ROM/Flash accesses, a transceiver must be placed between the memory parity/ECC bus and the 106 parity/ECC/ROM address bus to prevent contention. For DRAM/EDO/SDRAM accesses, the \overline{PPEN} signal functions similar to the $\overline{BCTL1}$ signal. That is, when $MCCR2[BUF_MODE] = 0$, \overline{PPEN} acts as a \overline{RE} signal; when $MCCR2[BUF_MODE] = 1$, \overline{PPEN} acts as an \overline{OE} signal. However, for ROM/Flash accesses, \overline{PPEN} remains negated, preventing the memory parity/ECC signals from conflicting with the ROM/Flash address being driven by the MPC106.

6.3 Memory Interface Signal Connections

The MPC106 memory interface uses an address bit ordering convention where bit 0 is the most-significant memory address bit. Most memory devices use a convention where bit 0 is the least-significant address bit. Figure 6-2 shows the memory address signal mappings and their corresponding address inputs on typical DRAM/EDO, SDRAM, and ROM

devices. (Note that the $\overline{\text{SDMA1/XATS}}$ signal functions as the $\overline{\text{XATS}}$ input signal until $\text{MCCR1}[\text{RAM_TYPE}]$ is cleared. This signal should be pulled-up if used as the SDMA1 output signal. The $\overline{\text{SDMA1/XATS}}$ signal must not be allowed to float or an unsupported 60x bus transaction error may be detected.)

Table 6-2. Memory Address Signal Mappings

MPC106 Signal Name (Output)		Logical Name					Typical Memory Device Address Signal Name (Input)		
		DRAM/EDO	SDRAM Address (16 Mbit, 2-Bank)	SDRAM Address (64 Mbit or 128 Mbit, 2-Bank)	SDRAM Address (64 Mbit or 128 Mbit, 4-Bank)	ROM/Flash	DRAM/EDO	SDRAM	ROM/Flash
msb	SDMA0/ SDBA1/ MA0/ AR0	MA0	—	SDMA0	SDBA1	AR0	A12	A12/BA1	A20
	PAR0/ AR1	—	—	—	—	AR1	—	—	A19
	PAR1/ AR2	—	—	—	—	AR2	—	—	A18
	PAR2/ AR3	—	—	—	—	AR3	—	—	A17
	PAR3/ AR4	—	—	—	—	AR4	—	—	A16
	PAR4/ AR5	—	—	—	—	AR5	—	—	A15
	PAR5/ AR6	—	—	—	—	AR6	—	—	A14
	PAR6/ AR7	—	—	—	—	AR7	—	—	A13
	PAR7/ AR8	—	—	—	—	AR8	—	—	A12
	SDBA0/ MA1/ AR9	MA1	SDBA0	SDBA0	SDBA0	AR9	A11	BA0/BS	A11
	SDMA1/ $\overline{\text{XATS}}$	—	—	SDMA1	SDMA1	—	—	A11	—
	SDMA2/ MA2/ AR10	MA2	SDMA2	SDMA2	SDMA2	AR10	A10	A10/AP	A10
	SDMA3/ MA3/ AR11	MA3	SDMA3	SDMA3	SDMA3	AR11	A9	A9	A9

Table 6-2. Memory Address Signal Mappings (continued)

MPC106 Signal Name (Output)		Logical Name					Typical Memory Device Address Signal Name (Input)		
		DRAM/EDO	SDRAM Address (16 Mbit, 2-Bank)	SDRAM Address (64 Mbit or 128 Mbit, 2-Bank)	SDRAM Address (64 Mbit or 128 Mbit, 4-Bank)	ROM/Flash	DRAM/EDO	SDRAM	ROM/Flash
	SDMA4/ MA4/ AR12	MA4	SDMA4	SDMA4	SDMA4	AR12	A8	A8	A8
	SDMA5/ MA5/ AR13	MA5	SDMA5	SDMA5	SDMA5	AR13	A7	A7	A7
	SDMA6/ MA6/ AR14	MA6	SDMA6	SDMA6	SDMA6	AR14	A6	A6	A6
	SDMA7/ MA7/ AR15	MA7	SDMA7	SDMA7	SDMA7	AR15	A5	A5	A5
	SDMA8/ MA8/ AR16	MA8	SDMA8	SDMA8	SDMA8	AR16	A4	A4	A4
	SDMA9/ MA9/ AR17	MA9	SDMA9	SDMA9	SDMA9	AR17	A3	A3	A3
	SDMA10/ MA10/ AR18	MA10	SDMA10	SDMA10	SDMA10	AR18	A2	A2	A2
	SDMA11/ MA11/ AR19	MA11	SDMA11	SDMA11	SDMA11	AR19	A1	A1	A1
Isb	SDMA12/ MA12/ AR20	MA12	SDMA12	SDMA12	SDMA12	AR20	A0	A0	A0

6.4 DRAM/EDO Interface Operation

The MPC106 is capable of supporting a variety of DRAM/EDO configurations. Thirteen multiplexed address signals provide for device densities from 256 K up to 16 M. Eight row address strobe ($\overline{\text{RAS}}$) signals support up to eight banks of memory. The DRAM/EDO banks can be built from standard memory modules or directly-attached memory chips. The datapath to the memory banks must be 64 bits wide (72 bits with parity or ECC). The MPC106 supports bank sizes from 2 Mbytes to 128 Mbytes. Eight column address strobe ($\overline{\text{CAS}}$) signals are used to provide byte selection for memory accesses. (Note that all $\overline{\text{CAS}}$ signals are driven in ECC mode.)

In addition to the 13 row/column multiplexed address signals (MA[0–12]), the 8 $\overline{\text{RAS}}$ signals, and the 8 $\overline{\text{CAS}}$ signals, there are 64 data signals (DH[0–31] and DL[0–31]), a write enable ($\overline{\text{WE}}$) signal, and 8 parity/ECC bits (PAR[0–7]).

Figure 6-6 shows an example of a two-bank, 16-Mbyte DRAM system.

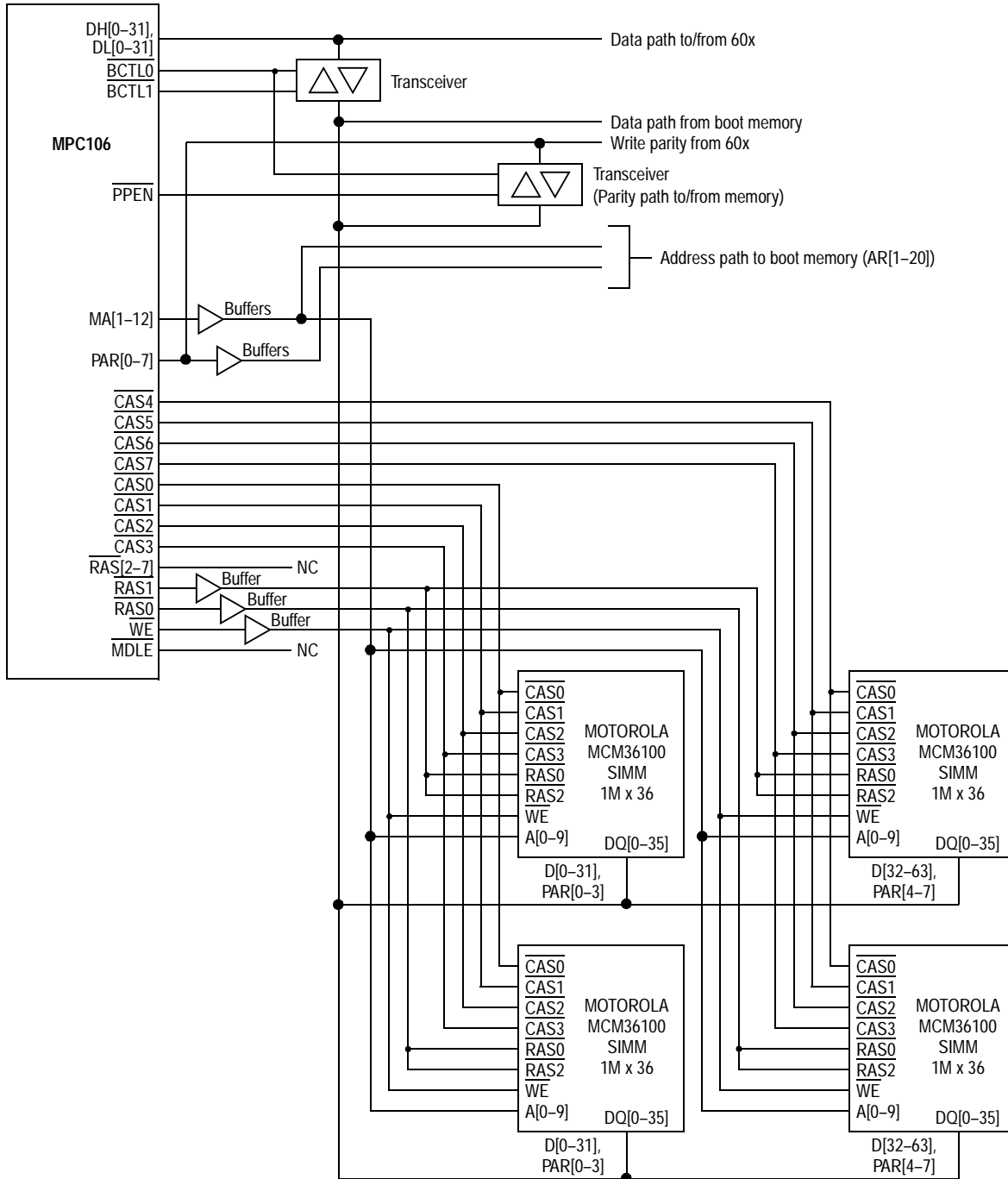


Figure 6-6. 16-Mbyte DRAM System with Parity

6.4.1 Supported DRAM/EDO Organizations

It is not necessary to use identical DRAM/EDO devices in each memory bank; individual memory banks may be of differing size. Although the MPC106 multiplexes the row and column address bits onto 13 memory address signals, individual banks may be implemented with memory devices requiring fewer than 24 address bits. The MPC106 can be configured to provide 13, 12, 11, 10, or 9 row address bits to a particular bank, and 12, 11, 10, or 9 column address bits. Note that for devices with 13 row address bits, only 11 column address bits can be provided for a 1-Gbyte address space.

The datapath to the memory banks must be 64 bits wide (72 with parity or ECC). When ECC or read-modify-write parity is enabled, all memory accesses are performed a double word at a time (that is, all $\overline{\text{CAS}}$ signals are asserted simultaneously). However, when ECC or read-modify-write parity is disabled, the memory system design must use the $\overline{\text{CAS}}$ signals for byte lane selection.

Table 6-3 summarizes some of the memory configurations supported by the MPC106. Note that Table 6-3 is not an exhaustive list of all configurations supported by the MPC106. The MPC106 can support any device that can accept the address multiplexing described in Section 6.4.2, “DRAM/EDO Address Multiplexing,” without exceeding the 128-Mbyte bank limit.

Table 6-3. Supported DRAM/EDO Device Configurations

DRAM Devices	Devices (64-Bit Bank)	Device Configuration	Row x Column Bits	64-Bit Bank Size (Mbytes)	8 Banks of Memory (Mbytes)
4 Mbits	4	256 Kbits x 16	9 x 9	2	16
	8	512 Kbits x 8	10 x 9	4	32
	16	1 Mbits x 4	10 x 10	8	64
	16	1 Mbits x 4	11 x 9	8	64
	64	4 Mbits x 1	12 x 10	32	256
	64	4 Mbits x 1	11 x 11	32	256
16 Mbits	2	512 Kbits x 32	10 x 9	4	32
	4	1 Mbits x 16	11 x 9	8	64
	4	1 Mbits x 16	10 x 10	8	64
	8	2 Mbits x 8	12 x 9	16	128
	8	2 Mbits x 8	11 x 10	16	128
	16	4 Mbits x 4	12 x 10	32	256
	16	4 Mbits x 4	11 x 11	32	256
	64	16 Mbits x 1	13 x 11	128	1024
	64	16 Mbits x 1	12 x 12	128	1024

Table 6-3. Supported DRAM/EDO Device Configurations (continued)

DRAM Devices	Devices (64-Bit Bank)	Device Configuration	Row x Column Bits	64-Bit Bank Size (Mbytes)	8 Banks of Memory (Mbytes)
64 Mbits	2	2 Mbits x 32	12 x 9	16	128
	2	2 Mbits x 32	11 x 10	16	128
	4	4 Mbits x 16	12 x 10	32	256
	4	4 Mbits x 16	11 x 11	32	256
	8	8 Mbits x 8	12 x 11	64	512
	16	16 Mbits x 4	13 x 11	128	1024
	16	16 Mbits x 4	12 x 12	128	1024

By using a memory polling algorithm at power-on reset, system firmware configures the MPC106 to correctly map the size of each bank in memory (using the memory boundary registers). The MPC106 uses its bank map to assert the appropriate $\overline{\text{RAS}}$ signal for memory accesses according to the provided bank depths.

6.4.2 DRAM/EDO Address Multiplexing

System software must configure the MPC106 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 memory control configuration register 1 (MCCR1) and the bank size into the memory boundary registers. Table 6-4 shows the settings for MCCR1[Bank n row] and the corresponding DRAM/EDO address multiplexing.

Table 6-4. DRAM/EDO Address Multiplexing

MCCR1[Bank n row]	DRAM/EDO Address Phase	Processor/Memory Address
0b00 (9 row bits)	Row ($\overline{\text{RAS}}_n$ asserted)	A[12–20] = MA[4–12]
	Column ($\overline{\text{CAS}}_n$ asserted)	A11 = MA4 A[21–28] = MA[5–12]
0b01 (10 row bits)	Row	A[11–20] = MA[3–12]
	Column	A[9–10] = MA[3–4] A[21–28] = MA[5–12]
0b10 (11 row bits)	Row	A[10–20] = MA[2–12]
	Column	A[7–9] = MA[2–4] A[21–28] = MA[5–12]

Table 6-4. DRAM/EDO Address Multiplexing

MCCR1[Bank n row]	DRAM/EDO Address Phase	Processor/Memory Address
0b11 (12 or 13 row bits)	Row	A5 = MA0* A[9–20] = MA[1–12]
	Column	A[5–8] = MA[1–4] A[21–28] = MA[5–12]
*Note that MA[0] is only used as the most-significant row address bit for 13 x 11 DRAM/EDO arrays.		

Figure 6-7 shows an alternate view of the DRAM/EDO address multiplexing.

Row x Col	Memory Address Phase	Processor (Physical) Address																													
		msb																													lsb
		0–4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
13x11	RAS						0			1	2	3	4	5	6	7	8	9	10	11	12										
	CAS																														
12x12	RAS									1	2	3	4	5	6	7	8	9	10	11	12										
	CAS																														
12x11	RAS									1	2	3	4	5	6	7	8	9	10	11	12										
	CAS																														
12x10	RAS									1	2	3	4	5	6	7	8	9	10	11	12										
	CAS																														
12x9	RAS									1	2	3	4	5	6	7	8	9	10	11	12										
	CAS																														
11x11	RAS																														
	CAS																														
11x10	RAS																														
	CAS																														
11x9	RAS																														
	CAS																														
10x10	RAS																														
	CAS																														
10x9	RAS																														
	CAS																														
9x9	RAS																														
	CAS																														

Figure 6-7. DRAM/EDO Address Multiplexing

The MPC106 uses the bank sizes programmed into the memory boundary registers and the higher-order address signals to determine which $\overline{\text{RAS}}[0-7]$ signal to assert for a given memory access. Also, the MPC106 decodes the processor transfer size (TSIZ[0-2] and TBST), and the three low-order address signals (A[29-31]) to determine which $\overline{\text{CAS}}[0-7]$ signals to assert for a given memory access.

6.4.3 DRAM/EDO 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, the memory page mode register, and the memory control configuration registers (MCCRs). See Section 3.2.8, “Memory Interface Configuration Registers,” for more detailed descriptions of the MICRs.

The programmable parameters relevant to the DRAM/EDO interface are:

- Memory bank starting and ending addresses (memory boundary registers)
- Memory bank enables (memory bank enable register)
- PGMAX—maximum $\overline{\text{RAS}}$ assertion interval for page mode retention
- SREN—self-refresh enable
- PCKEN—memory interface parity checking/generation enable
- Row address bit count for each bank
- ECC_EN—ECC enable
- EDO—EDO enable
- REFINT—refresh interval
- BUF_MODE—buffer mode
- RMW_PAR—read-modify-write parity enable
- CPX— $\overline{\text{CAS}}$ write timing modifier
- RAS_{6P}— $\overline{\text{RAS}}$ assertion interval for CBR refresh
- CAS₅— $\overline{\text{CAS}}$ assertion interval for page mode access
- CP₄— $\overline{\text{CAS}}$ precharge interval
- CAS₃— $\overline{\text{CAS}}$ assertion interval for a single beat, or for the first access in a burst
- RCD₂— $\overline{\text{RAS}}$ to $\overline{\text{CAS}}$ delay interval
- RP₁— $\overline{\text{RAS}}$ precharge interval
- RCBUF—memory buffer type

Note that any unused banks should have their starting and ending addresses programmed out of the range of memory banks in use. If a disabled bank has its starting and ending address defined as overlapping an enabled bank’s address space, there may be system

memory corruption in the overlapping address range. Always map unused memory banks' starting and ending addresses to memory space that is not used by the system.

Once all the memory parameters are configured, system software should set the MEMGO bit (MCCR1, bit 19) to enable the memory interface. Note that 100 μ s 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 MPC106 then performs eight refreshes on the DRAM/EDO array (in accordance with REFINT) to prepare it for system access. When these refreshes are complete, the memory array is ready for access.

6.4.3.1 Supported Memory Interface Configurations

The MPC106 can only support certain configurations of parity/RMW/ECC, DRAM/EDO/SDRAM, and buffer types. Table 6-5 shows the supported memory interface configurations.

Table 6-5. Supported Memory Interface Configurations

Memory Interface Parameter						
PCKEN	ECC_EN	EDO	RAM_TYPE	RMW_PAR	WCBUF	RCBUF
x	0 (ECC disabled)	x (DRAM or EDO)	1 (DRAM or EDO)	x ¹	x (flow-through, latched, or registered type)	x (flow-through, latched, or registered type)
x	0 (ECC disabled)	x (DRAM or EDO)	x (DRAM, EDO, or SDRAM)	x ¹	x (flow-through or registered type) ²	x (flow-through or registered type) ²
x ³	1 (ECC enabled) ⁴	0 (DRAM)	1 (DRAM)	0 (RMW parity disabled)	x (flow-through, latched, or registered type)	x (flow-through, latched, or registered type)
x ³	1 (ECC enabled) ⁴	1 (EDO)	1 (DRAM)	0 (RMW parity disabled)	0 (flow-through) ⁵	0 (flow-through) ⁵

Note:

1. Parity checking must be enabled (PCKEN = 1) when RMW parity is enabled (RMW_PAR = 1).
2. SDRAM systems cannot use latched buffers. Therefore, [WCBUF, RCBUF] can be 0b00 or 0b11, but not 0b01.
3. To enable L2 parity checking when ECC is enabled, parity checking must be enabled (PCKEN = 1).
4. SDRAM systems cannot use ECC.
5. Systems with EDO and latched buffers can use ECC, but the EDO parameter must be set for DRAM (EDO = 0).

6.4.4 DRAM/EDO Interface Timing

System software is responsible for optimal configuration of the DRAM/EDO programmable timing parameters (RP₁, RCD₂, CAS₃, CP₄, CAS₅, and CPX) at reset. The programmable timing parameters are applicable to both read and write timing

configuration. The $\overline{\text{CAS}}$ write timing modifier (CPX) compensates for data propagation delays by adjusting the CAS timing parameters (CP_4 and CAS_5) during burst write operations. The configuration process must be completed, before any accesses to DRAM/EDO are attempted.

Note that EDO systems must use a CP_4 value of one clock; larger values are not supported. Also note that if the CF_L2_HIT_DELAY parameter is programmed to three clock cycles, then the $\overline{\text{RAS}}$ to $\overline{\text{CAS}}$ delay (RCD_2) must be programmed to three or more clock cycles. Finally, the MPC106 should never be programmed such that successive assertions of $\overline{\text{CAS}}$ within a burst will have more than eight PCI clocks between the start of assertions. This should be easy to achieve with existing DRAM/EDO technology.

Table 6-6 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 6-6. 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/EDO timing for various types of accesses with ECC disabled; see Figure 6-8 for a single-beat read operation, Figure 6-9 for a DRAM burst read operation, Figure 6-10 for an EDO burst read operation, Figure 6-11 for a single-beat write operation, and Figure 6-12 for a burst write operation. Note that all signal transitions occur on the rising edge of the 60x bus clock.

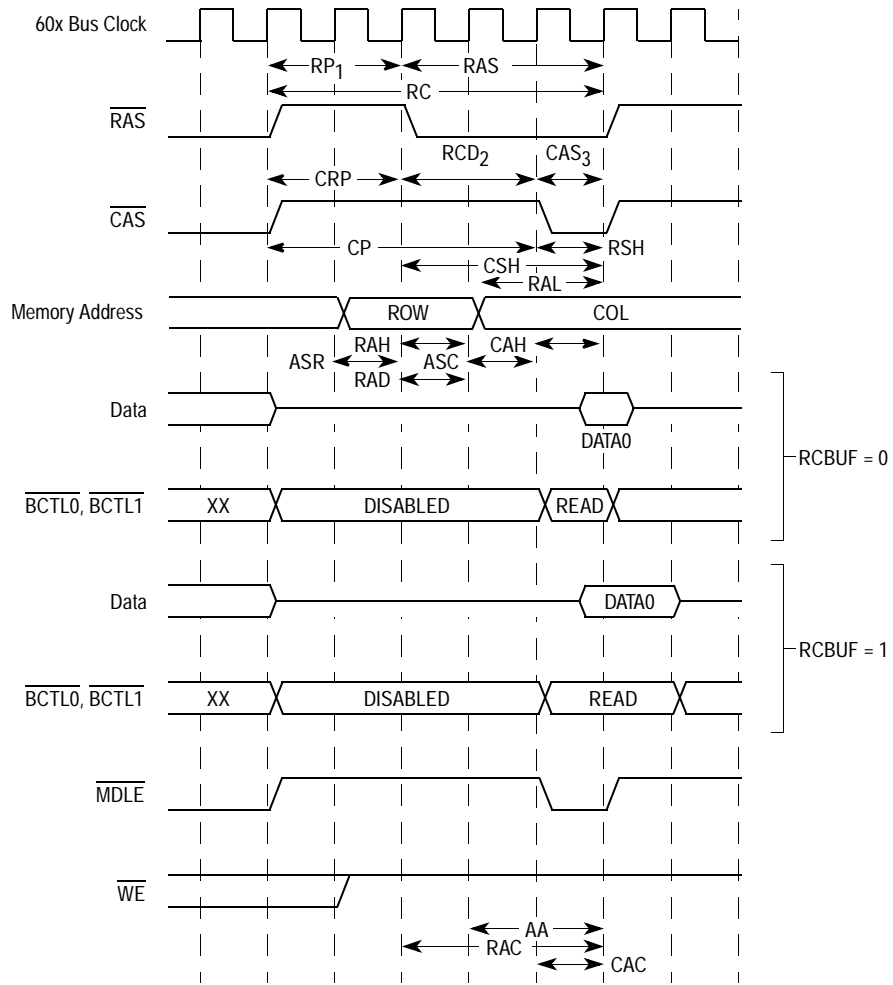
Table 6-7 describes the acronyms used in the DRAM/EDO timing diagrams. The acronyms with subscripted numbers represent the programmable timing parameters of the MPC106.

Table 6-7. DRAM/EDO 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 $\overline{\text{CAS}}$ assertion
CAH	Column address hold time
CAS_3	$\overline{\text{CAS}}$ assertion interval for a single beat, or for the first access in a burst
CAS_5	$\overline{\text{CAS}}$ assertion interval for page mode access

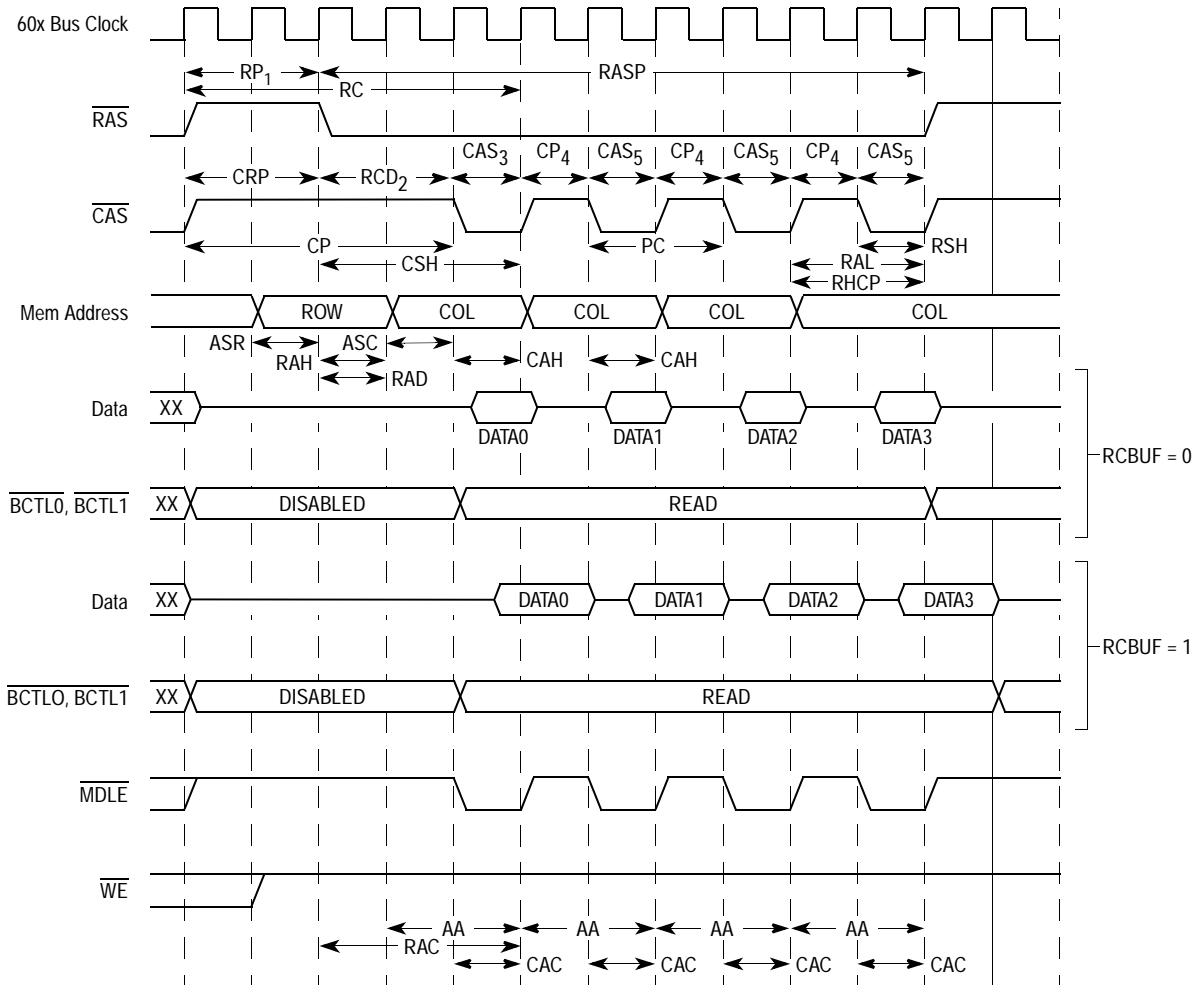
Table 6-7. DRAM/EDO Timing Parameters (continued)

Symbol	Timing Parameter
CHR	$\overline{\text{CAS}}$ hold time for CBR refresh
CP	$\overline{\text{CAS}}$ precharge time
CP ₄	$\overline{\text{CAS}}$ precharge interval during page-mode access
CRP	$\overline{\text{CAS}}$ to $\overline{\text{RAS}}$ precharge time
CSH	$\overline{\text{CAS}}$ hold time
CSR	$\overline{\text{CAS}}$ 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 $\overline{\text{RAS}}$ assertion
RAD	$\overline{\text{RAS}}$ to column address delay time
RAH	Row address hold time
RAL	Column address to $\overline{\text{RAS}}$ lead time
RAS _{6P}	$\overline{\text{RAS}}$ assertion interval for CBR refresh
RAS	$\overline{\text{RAS}}$ assertion interval
RASP	$\overline{\text{RAS}}$ pulse width (page-mode)
RASS	Self-refresh interval (power saving modes only)
RC	Random access (read, write, or refresh) cycle time
RCD ₂	$\overline{\text{RAS}}$ to $\overline{\text{CAS}}$ delay interval
RHCP	$\overline{\text{RAS}}$ hold time from $\overline{\text{CAS}}$ precharge (page mode only)
RP ₁	$\overline{\text{RAS}}$ precharge interval
RPC	$\overline{\text{RAS}}$ precharge to $\overline{\text{CAS}}$ active time
RSH	$\overline{\text{RAS}}$ hold time
WCH	Write command hold time (referenced to $\overline{\text{CAS}}$)
WCS	Write command setup time
WP	Write command pulse width
WRH	Write to $\overline{\text{RAS}}$ hold time (CBR refresh)
WRP	Write to $\overline{\text{RAS}}$ precharge time (CBR refresh)



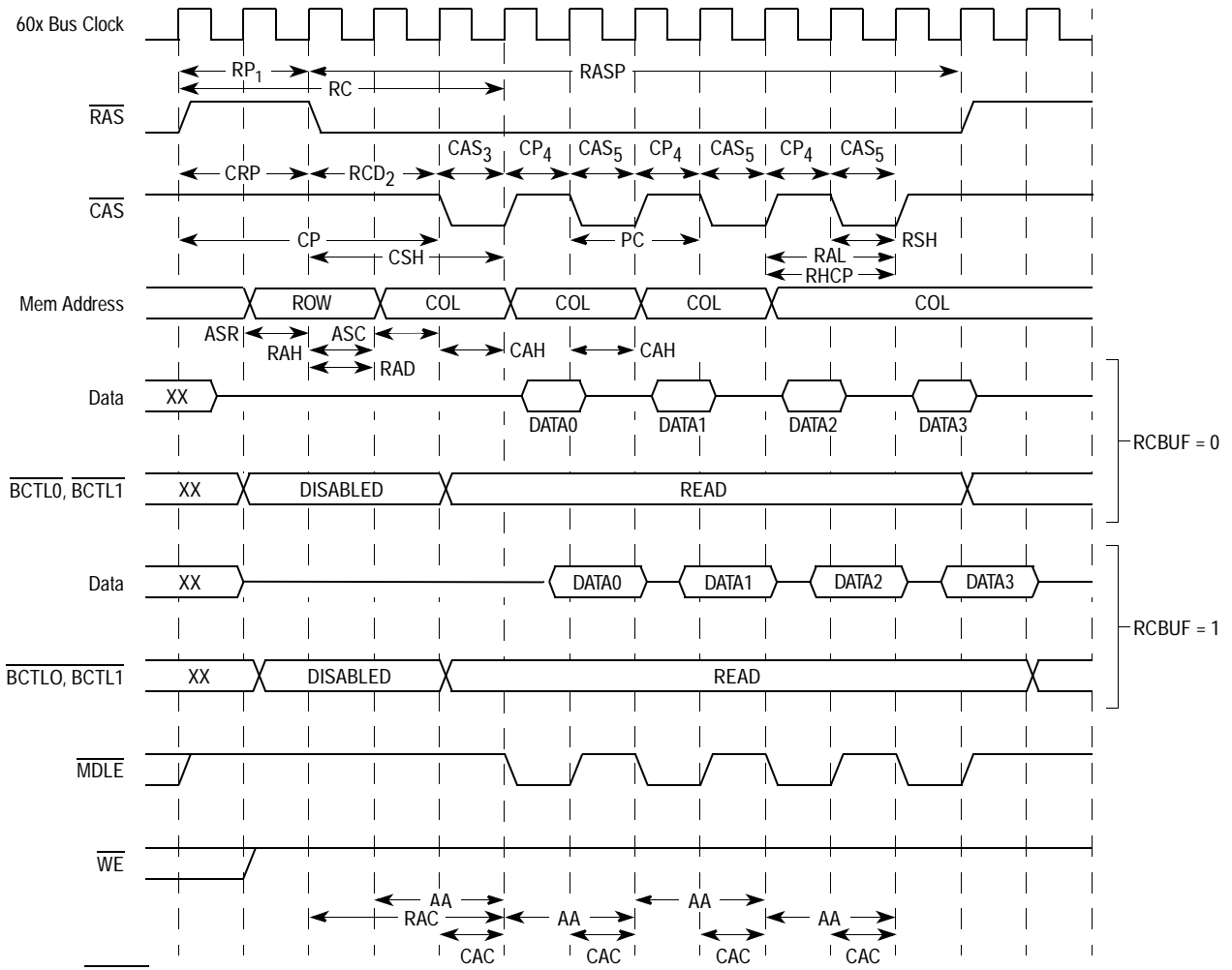
Note: MDLE is driven only if RCBUF = 1.

Figure 6-8. DRAM Single-Beat Read Timing—No ECC



Note: \overline{MDLE} is driven only if $RCBUF = 1$.

Figure 6-9. DRAM Burst Read Timing—No ECC



Note: MDLE is driven only if RCBUF = 1.

Figure 6-10. EDO Burst Read Timing—No ECC

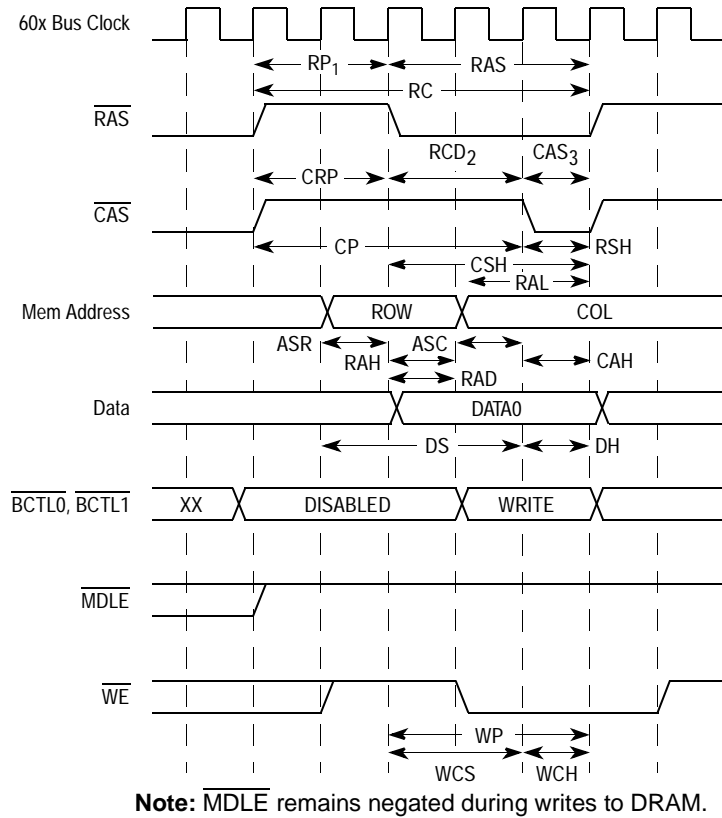
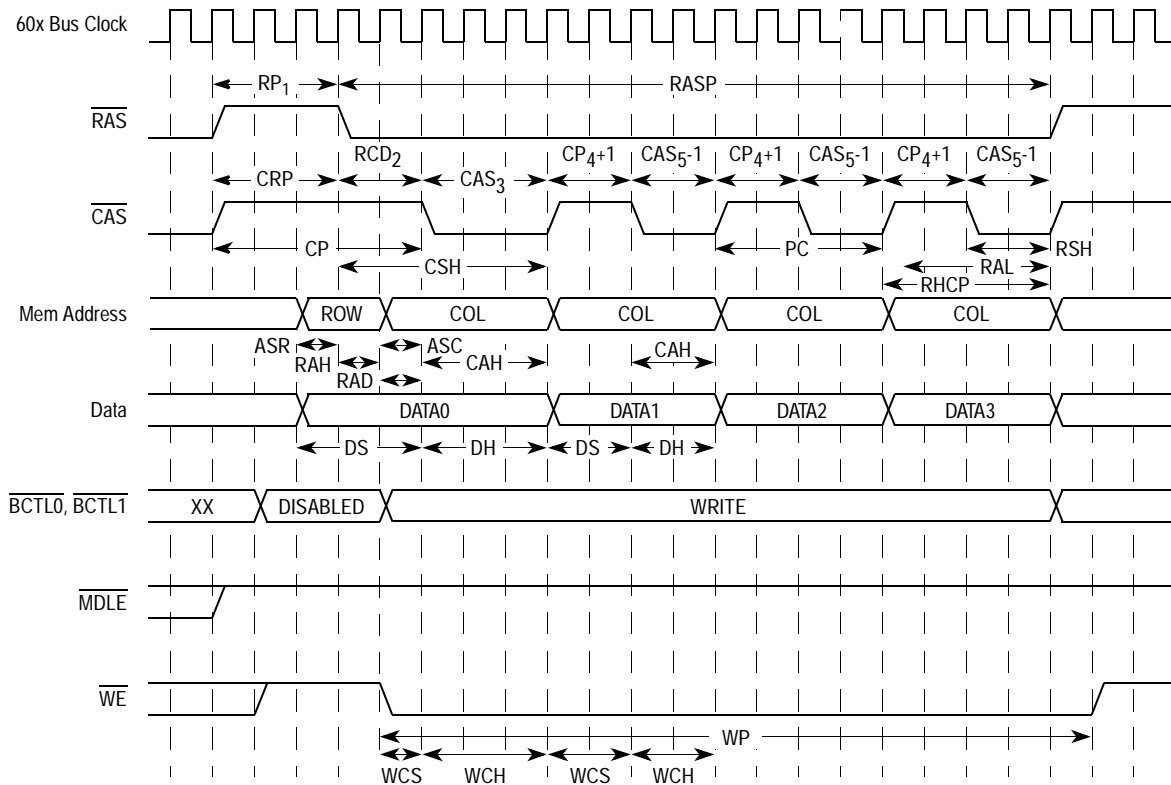


Figure 6-11. DRAM Single-Beat Write Timing—No ECC



Note that if $CPX = 1$, CP_4 and CAS_5 are automatically adjusted by one cycle during write bursts to allow greater data propagation time. Also, note that $MDLE$ remains negated during writes to DRAM/EDO.

Figure 6-12. DRAM/EDO Burst Write Timing—No ECC, $CPX = 1$

6.4.5 DRAM/EDO Burst Wrap

The MPC106 supports burst-of-four, double-word data beats for accesses to system memory. The burst is always sequential, and the critical double word is always supplied first. For example, if the 60x requests the third double word of a cache line, the MPC106 reads double words from memory in the order 2-3-0-1.

6.4.6 DRAM/EDO Latency

Table 6-8 summarizes the estimated DRAM latency (in processor cycles) for a burst read operation. The first number is the number of processor clock cycles from the assertion of \overline{TS} by the 60x processor to the time the first data is returned; the next three numbers are the number of clock cycles for the subsequent three data beats. The latency calculations assume ECC has been disabled, \overline{RAS} has been precharged, the MPC106 is idle, and the 60x processor and the MPC106 are operating at the same frequency.

Table 6-8. Estimated DRAM Latency

DRAM Access Time	System Timing	Processor Bus Frequency			
		25 MHz	33 MHz	50 MHz	66 MHz
60 ns	Aggressive (lightly loaded)	5-2-2-2	5-2-2-2	6-3-3-3	7-3-3-3
	Conservative (heavily loaded)	6-2-2-2	6-2-2-2	7-3-3-3	8-4-4-4
70 ns	Aggressive (lightly loaded)	5-2-2-2	5-3-3-3	7-3-3-3	8-4-4-4
	Conservative (heavily loaded)	6-2-2-2	6-3-3-3	8-3-3-3	9-4-4-4

Table 6-9 summarizes the estimated EDO latency (in processor cycles) for a burst read operation. The first number is the number of processor clock cycles from the assertion of \overline{TS} by the 60x processor to the time the first data is returned; the next three numbers are the number of clock cycles for the subsequent three data beats. The latency calculations assume ECC has been disabled, \overline{RAS} has been precharged, the MPC106 is idle, and the 60x processor and the MPC106 are operating at the same frequency.

Table 6-9. Estimated EDO Latency

EDO Access Time	System Timing	Processor Bus Frequency			
		25 MHz	33 MHz	50 MHz	66 MHz
60 ns	Aggressive (lightly loaded)	5-2-2-2	5-2-2-2	6-2-2-2	7-2-2-2
	Conservative (heavily loaded)	6-2-2-2	6-2-2-2	7-2-2-2	8-3-3-3

6.4.7 DRAM/EDO Page Mode Retention

Under certain conditions, the MPC106 retains the currently active DRAM/EDO page by holding \overline{RAS} asserted for pipelined burst accesses. These conditions are:

- A pending transaction (read or write) hits the currently active DRAM/EDO page.
- There are no pending refreshes.
- The maximum \overline{RAS} assertion interval (controlled by PGMAX) has not been exceeded.

Page mode can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save three to four clock cycles from subsequent burst accesses that hit in an active page. Page mode is disabled by clearing the PGMAX parameter (that is, PGMAX = 0x00).

6.4.8 DRAM/EDO Parity and RMW Parity

When configured for DRAM or EDO, the MPC106 supports two forms of parity checking and generation—normal parity and read-modify-write (RMW) parity. Normal parity assumes that each of the eight parity bits is controlled by a separate $\overline{\text{CAS}}$ signal. Thus, for a single-beat write from PCI to system memory, the MPC106 generates a parity bit for each byte written to memory.

RMW parity assumes that all eight parity bits are controlled by a single $\overline{\text{CAS}}$ signal and therefore must be written as a single 8-bit quantity (that is, a byte). Therefore, for any write operation to system memory that is less than a double word, the MPC106 must latch the write data, read an entire 64-bit double word from memory, check the parity of the double word read from memory, merge the write data with the double word read from memory, regenerate parity for the new double word, and finally write the new double word back to memory.

The MPC106 checks parity on all memory reads, provided parity checking is enabled ($\text{PCKEN} = 1$). The MPC106 generates parity for the following operations:

- PCI to memory write operations
- L1 and L2 copyback operations
- L2 castout operations
- 60x single-beat write operations with RMW parity enabled ($\text{RMW_PAR} = 1$)

The 60x processor is expected to generate parity for all other 60x to memory write operations as the data goes directly to memory and does not pass through the MPC106.

6.4.8.1 RMW Parity Latency Considerations

When RMW parity is enabled, the time required to read, modify, and write increases latency for some transactions.

For 60x processor single-beat writes to system memory, the MPC106 latches the data, performs a double-word read from system memory (checking parity), and then merges the write data from the processor with the data read from memory. The MPC106 then generates new parity bits for the merged double word and writes the data and parity to memory. The read-modify-write process adds six clock cycles to a single-beat write operation. If page mode retention is enabled ($\text{PGMAX} \neq 0$), then the MPC106 will keep the memory in page mode for the read-modify-write sequence. Figure 6-15 shows DRAM/EDO timing for a 60x single-beat write operation with RMW parity enabled.

For PCI writes to system memory with RMW parity enabled, the MPC106 latches the data in the internal PCI-to-system-memory-write buffer (PCMWB). If the PCI master writes complete double words to system memory, the MPC106 generates the parity bits when the PCMWB is flushed to memory. However, if the PCI master writes 32-, 16-, or 8-bit data that cannot be gathered into a complete double word in the PCMWB, a read-modify-write

operation is required. The MPC106 performs a double-word read from system memory (checking parity), and then merges the write data from the PCI master with the data read from memory. The MPC106 then generates new parity for the merged double word and writes the data and parity to memory. If page mode retention is enabled ($PGMAX \neq 0$), the MPC106 keeps the memory in page mode for the read-modify-write sequence.

Since the processor drives all eight parity bits during 60x burst writes to system memory, these transactions go directly to the DRAMs with no performance penalty. All other transactions are unaffected and operate as in normal parity mode.

6.4.9 Error Checking and Correction—ECC

As an alternative to simple parity, the MPC106 supports ECC for the datapath between the MPC106 and system memory. ECC not only allows the MPC106 to detect errors in the memory datapath, but also allows the MPC106 to correct single-bit errors in the 64-bit datapath. The ECC logic in the MPC106 detects and corrects all single-bit errors and detects all double-bit errors and all errors within a nibble. Other errors may be detected, but are not guaranteed to be corrected. Multibit errors are always reported. However, when a single-bit error occurs, the single-bit error counter register is incremented and its value compared to the single-bit error trigger register. If the values are not equal, no error is reported; if the values are equal, then an error is reported. Thus, the single-bit error registers may be programmed such that minor faults with memory are corrected and ignored, but a catastrophic memory failure generates an interrupt.

The MPC106 supports concurrent ECC for the DRAM/EDO datapath and parity for the 60x processor/L2 cache datapath. ECC and parity may be independently enabled or disabled. The eight signals used for ECC (PAR[0–7]) are also used for 60x processor/L2 cache parity. The parity path and datapath buffers effectively isolate the ECC (106 to/from memory) transactions from the parity (106 to/from 60x/L2) transactions.

The MPC106 checks ECC on all memory reads, provided $ECC_EN = 1$, and generates parity on all 60x processor reads that are latched by the L2 cache (provided $PCKEN = 1$). Note that when the MPC106 is generating ECC for 60x processor writes to system memory, it does not check data parity generated by the processor—even when $PCKEN = 1$. If data gets corrupted on the 60x bus, the MPC106 does not report a parity error and generates ECC bits based on the corrupt data.

6.4.9.1 DRAM/EDO Interface Timing with ECC

When ECC is enabled, the time required to check/generate the ECC codes increases access latency. Figure 6-13 through Figure 6-15 illustrate DRAM/EDO timing for various types of accesses with ECC enabled; see Figure 6-13 for a DRAM burst read operation, Figure 6-14 for an EDO burst read operation, and Figure 6-15 for a single-beat write operation.

For processor burst reads from system memory, checking the ECC codes for errors requires an additional two clock cycles for the first data returned, and at least four clock cycles for subsequent beats. If an asynchronous L2 cache is used in the system, then the subsequent beats require a minimum of five clock cycles. These requirements do not depend on the buffer type or whether DRAM or EDO is used for system memory.

For 60x processor single beat writes to system memory, the MPC106 latches the data, performs a double-word read from system memory (checking and correcting any ECC errors), and merges the write data from the processor with the data read from memory. The MPC106 then generates a new ECC code for the merged double word and writes the data and ECC code to memory. This read-modify-write process adds six clock cycles to a single-beat write operation. If page mode retention is enabled ($PGMAX \neq 0$), the MPC106 keeps the memory in page mode for the read-modify-write sequence.

For 60x processor burst writes to system memory, the MPC106 latches the data in the internal copyback buffer and flushes the buffer to memory at the earliest opportunity. The MPC106 generates the ECC codes when the flush occurs. Note that the MPC106 does not check the data being overwritten in memory.

For PCI writes to system memory with ECC enabled, the MPC106 latches the data in the internal PCI to memory write buffer (PCMWB). If the PCI master writes complete double words to system memory, the MPC106 generates the ECC codes when the PCMWB is flushed to memory. However, if the PCI master writes 32-, 16-, or 8-bit data that cannot be gathered into a complete double word in the PCMWB, a read-modify-write operation is required. The MPC106 performs a double word read from system memory (checking and correcting any ECC errors), and then merges the write data from the PCI master with the data read from memory. The MPC106 then generates a new ECC code for the merged double word and writes the data and ECC code to memory. If page mode retention is enabled ($PGMAX \neq 0$), the MPC106 keeps the memory in page mode for the read-modify-write sequence.

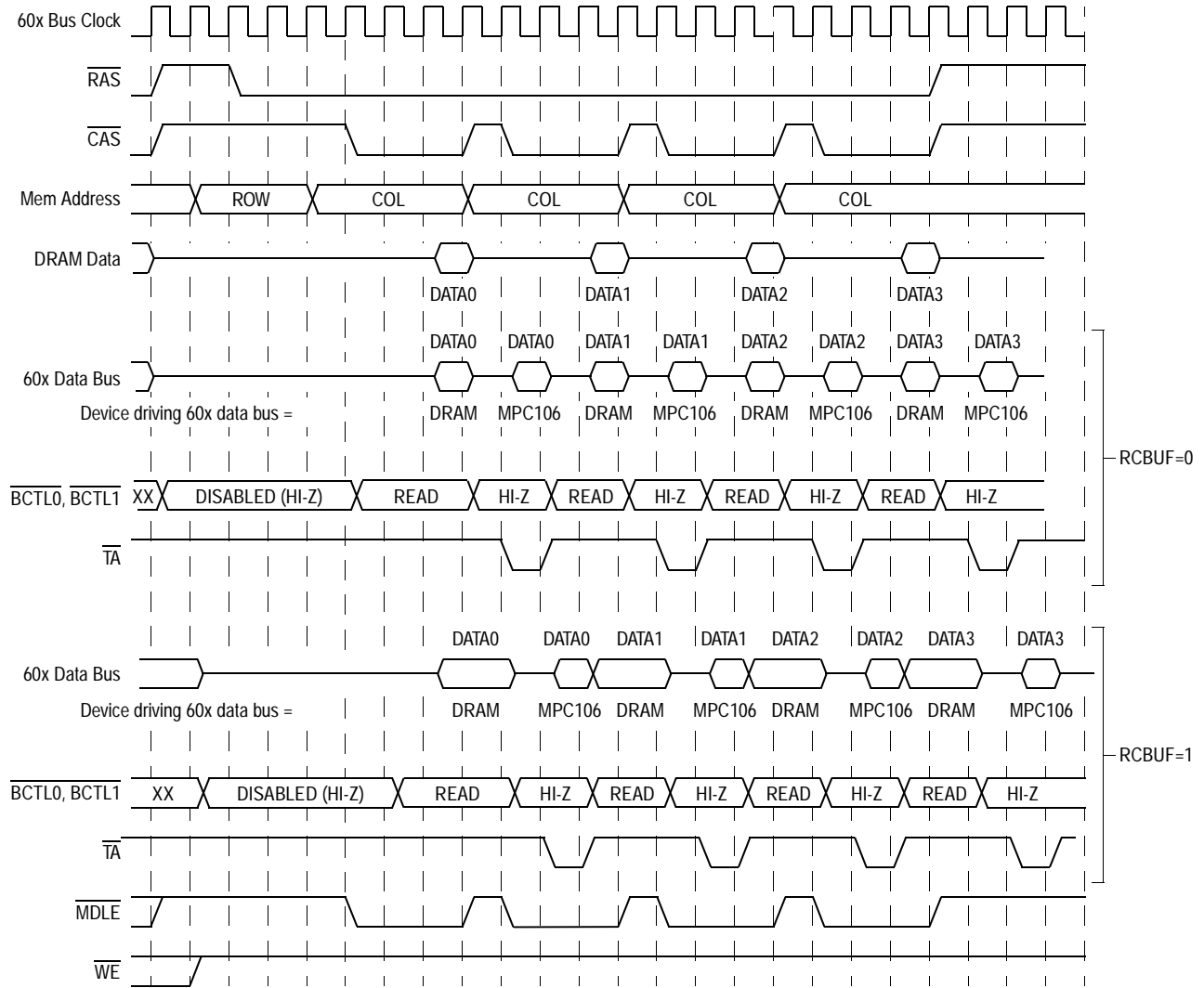
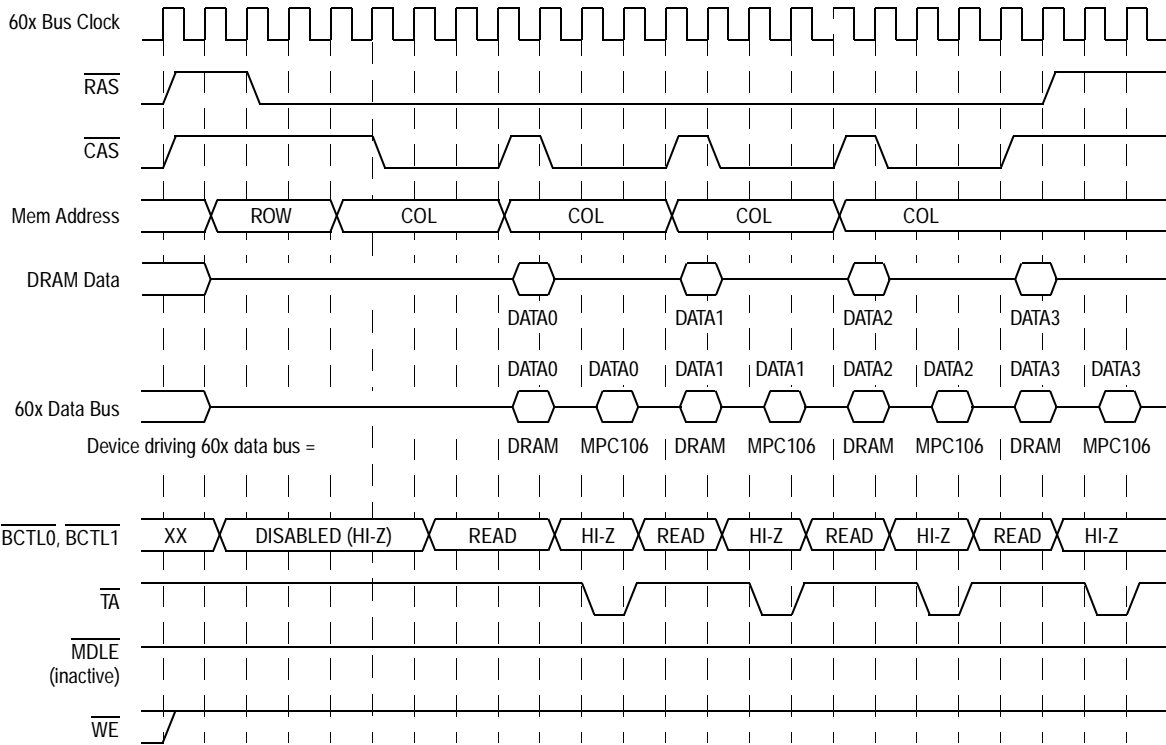


Figure 6-13. DRAM Burst Read with ECC



Notes:

- RCBUF must be programmed to zero for systems with EDO and ECC.
- MDLE will be inactive for this configuration.

Figure 6-14. EDO Burst Read Timing with ECC

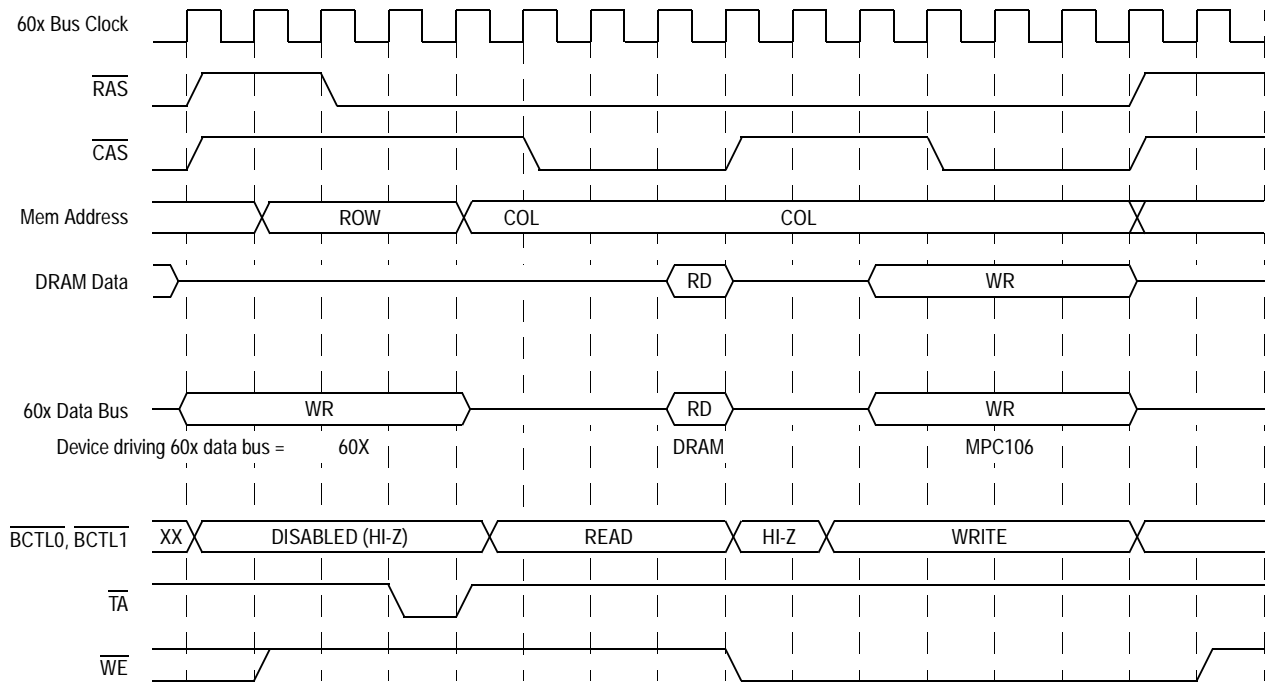


Figure 6-15. DRAM Single-Beat Write Timing with RMW or ECC Enabled

6.4.10 DRAM/EDO Refresh

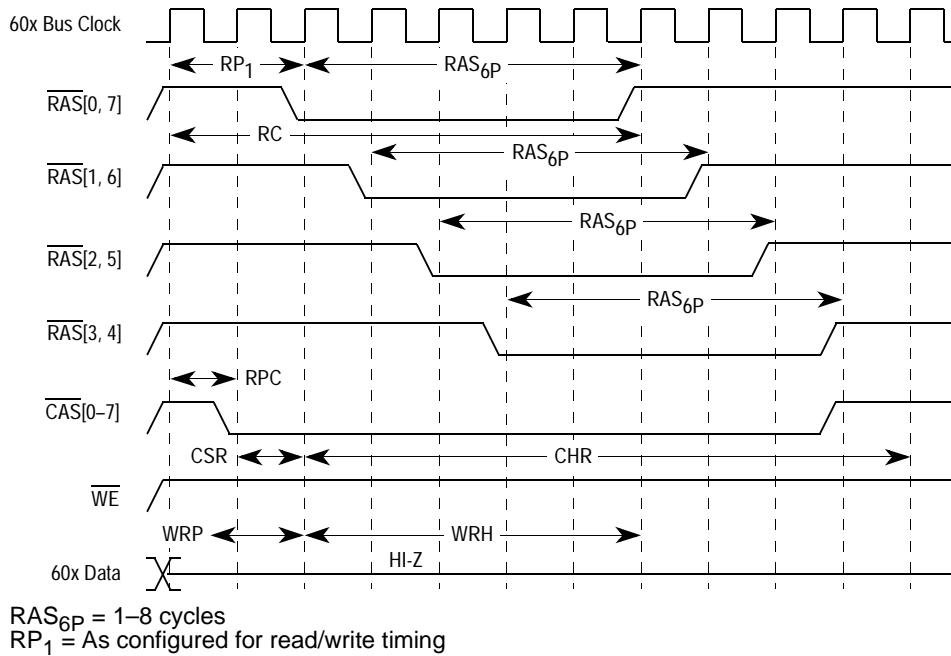
The memory interface supplies $\overline{\text{CAS}}$ before $\overline{\text{RAS}}$ (CBR) refreshes to DRAM/EDO according to the refresh interval, MCCR2[REFINT]. The value stored in REFINT represents the number of 60x bus clock cycles required between CBR refreshes. The value for REFINT depends on the specific DRAM/EDOs used and the operating frequency of the MPC106. The value should allow for any potential collisions between DRAM/EDO 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 $64 \text{ ms} \div 4096 \text{ rows} = 15.6 \mu\text{s}$. If the 60x bus clock is running at 66 MHz, the number of clock cycles per row refresh is $15.6 \mu\text{s} \times 66 \text{ MHz} = 1030$ 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, $1030 - 24 = 1006$).

6.4.10.1 DRAM/EDO Refresh Timing

The refresh timing for DRAM/EDO is controlled by the programmable timing parameter MCCR3[RAS_{6P}]. RAS_{6P} determines the number of 60x bus clock cycles that $\overline{\text{RAS}}$ is held asserted during a CBR refresh. The MPC106 implements bank staggering for CBR refreshes, as shown in Figure 6-16; see Table 6-7 for the acronyms used in Figure 6-16. The acronyms with subscripted numbers represent the programmable timing parameters of the MPC106.


Figure 6-16. DRAM/EDO 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/EDO are attempted. Table 6-10 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 6-10. 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

6.4.10.2 DRAM/EDO Refresh and Power Saving Modes

The MPC106 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 MPC106 supplies the normal CBR refresh to DRAM/EDO. In sleep mode, the MPC106 can be configured to take advantage of self-refreshing DRAM/EDOs, provide normal refresh to DRAM/EDO, or provide no refresh support. In suspend mode, the MPC106 can be configured to take

advantage of self-refreshing DRAM/EDOs, provide CBR refreshes based on the RTC input signal, or provide no refresh support. Normal refresh is not available for DRAM/EDO in suspend mode. Table 6-11 summarizes the refresh types available in each of the power saving modes and the relevant configuration parameters.

Table 6-11. DRAM/EDO Power Saving Modes Refresh Configuration

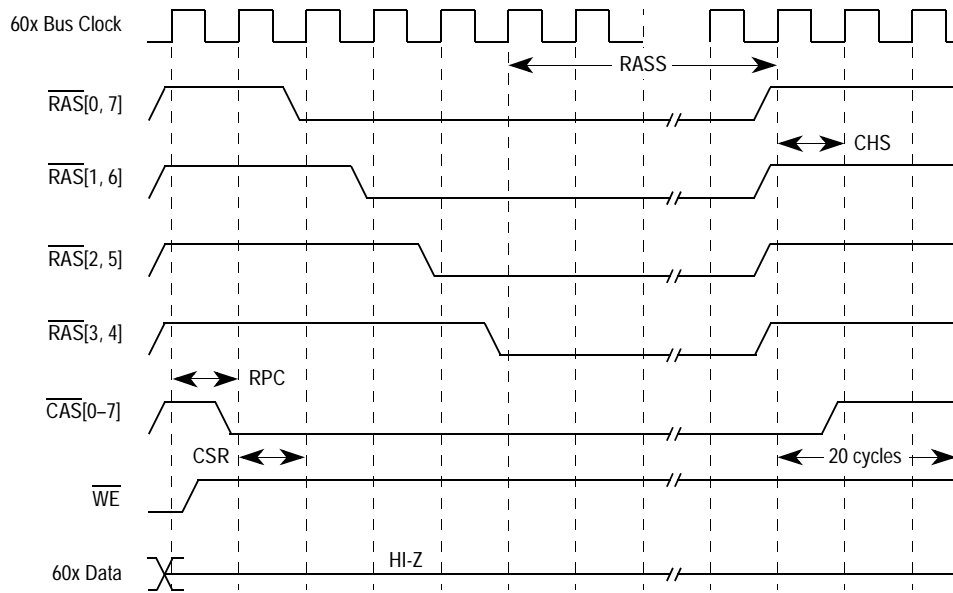
Power Saving Mode	Refresh Type	SUSPEND Signal	Power Management Control Register (PMCR)					MCCR1 [SREN]
			[PM]	[DOZE]	[NAP]	[SLEEP]	[LP_REF_EN]	
Doze	Normal	Negated (High)	1	1	0	0	—	—
Nap	Normal	Negated (High)	1	—	1	0	—	—
Sleep	Self	Negated (High)	1	—	—	1	1	1
	Normal	Negated (High)	1	—	—	1	1	0
	None	Negated (High)	1	—	—	1	0	—
Suspend	Self	Asserted (Low)	1	—	—	0	1	1
	RTC	Asserted (Low)	1	—	—	0	1	0
	None	Asserted (Low)	1	—	—	0	0	—

Note: Normal refresh is not available in suspend mode.

Note that in the absence of refresh support, system software must preserve DRAM/EDO data (such as by copying the data to disk) before entering the power saving state.

6.4.10.2.1 Self-Refresh in Sleep and Suspend Modes

The setup timing for self-refreshing DRAM/EDOs is shown in Figure 6-17; see Table 6-7 for acronyms used in Figure 6-17.



RASS = Self-refresh interval (must be greater than 100 μ s).

Figure 6-17. DRAM/EDO Self-Refresh Timing in Sleep and Suspend Modes

6.4.10.2.2 RTC Refresh in Suspend Mode

If the MPC106 is configured to provide refresh cycles based on the RTC signal in suspend mode, $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ are driven as shown in Figure 6-18. The DRAM/EDOs 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 DRAM/EDOs must have a low-power refresh interval greater than 122 μ s to allow use of the RTC refresh feature.

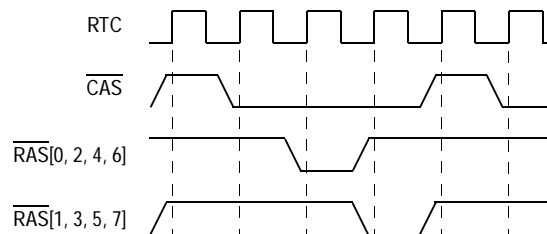


Figure 6-18. Suspend Mode—Real Time Clock Refresh

6.4.11 Processor-to-DRAM Transaction Examples

The figures in this section provide signal timing examples of 60x processor-to-system-memory transactions for a DRAM configuration. Figure 6-19 shows a series of processor burst reads from system memory. Figure 6-20 shows a processor burst write to system memory.

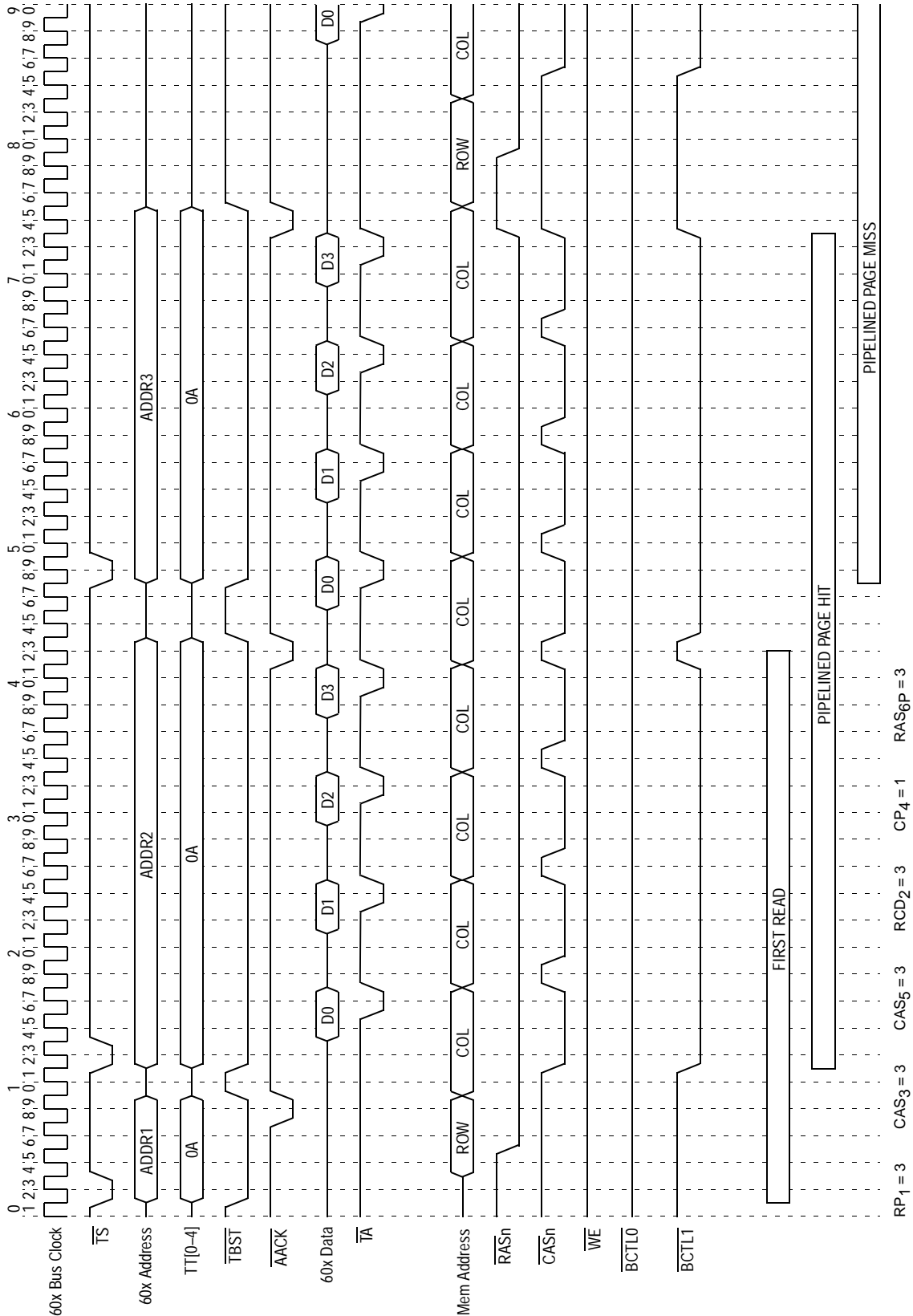


Figure 6-19. Processor Burst Reads from Memory—60-ns DRAM with Flow-Through Buffers

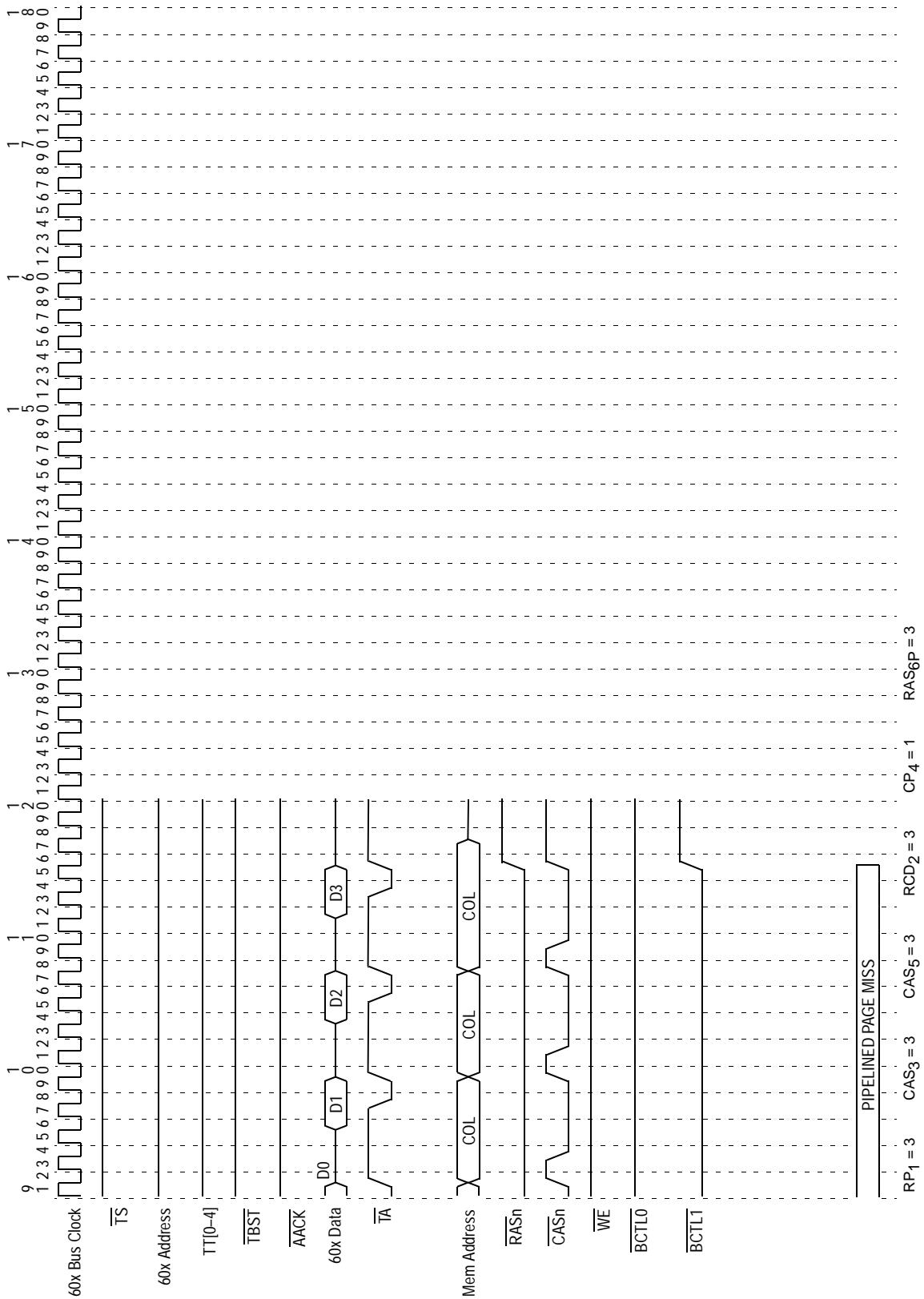


Figure 6-19. (Continued) Processor Burst Reads from Memory—60-ns DRAM with Flow-Through Buffers

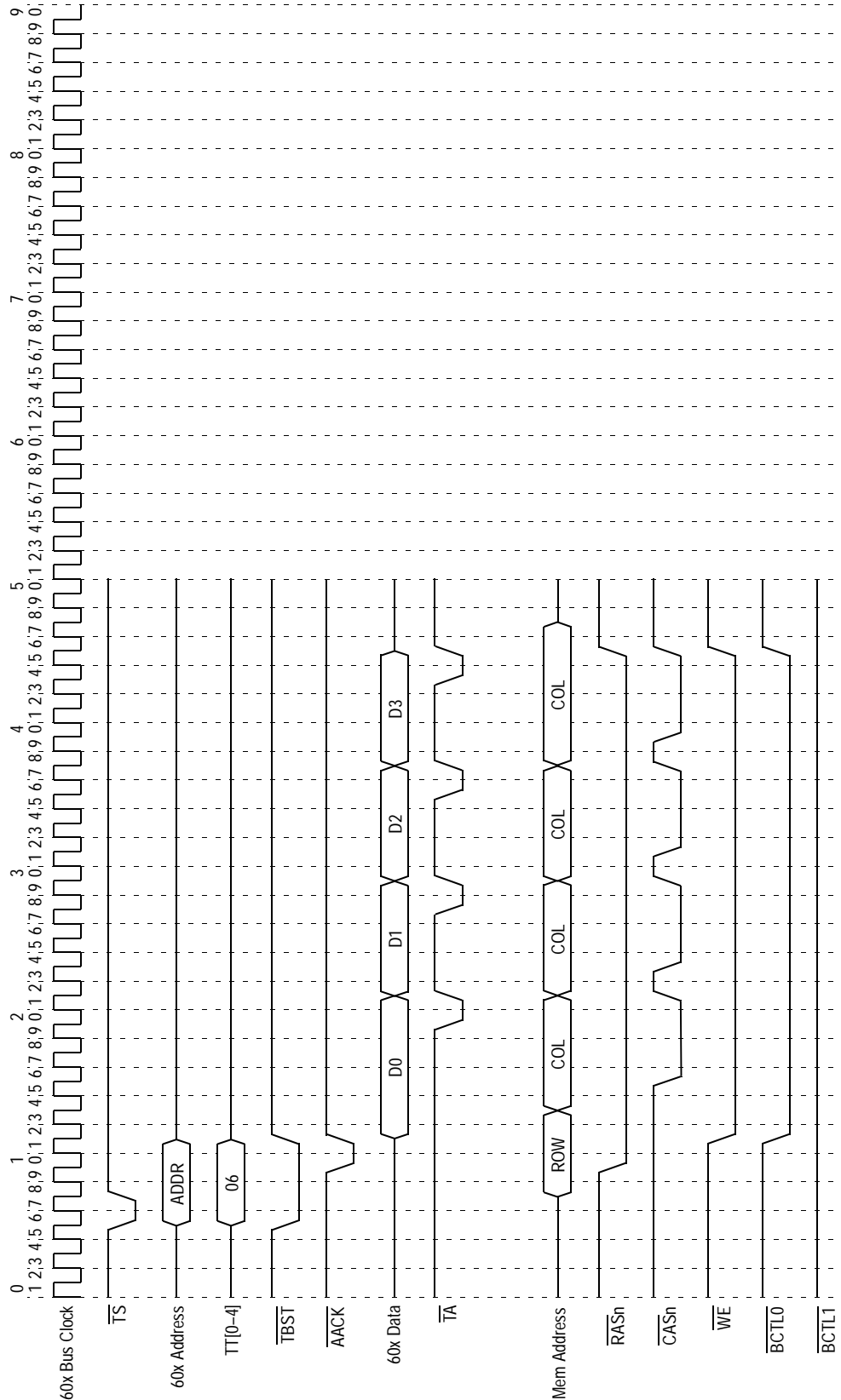


Figure 6-20. Processor Burst Write to Memory—60-ns DRAM with Flow-Through Buffers

6.4.12 PCI-to-DRAM Transaction Examples

The figures in this section provide signal timing examples of PCI-to-system-memory transactions for a DRAM configuration. Figure 6-21 shows a series of PCI burst reads from system memory with speculative reading enabled (PICR1[Speculative PCI Reads] = 1). Figure 6-22 shows a series of PCI burst reads from system memory with speculative reading disabled (PICR1[Speculative PCI Reads] = 0). Figure 6-23 shows a series of PCI burst writes to system memory.

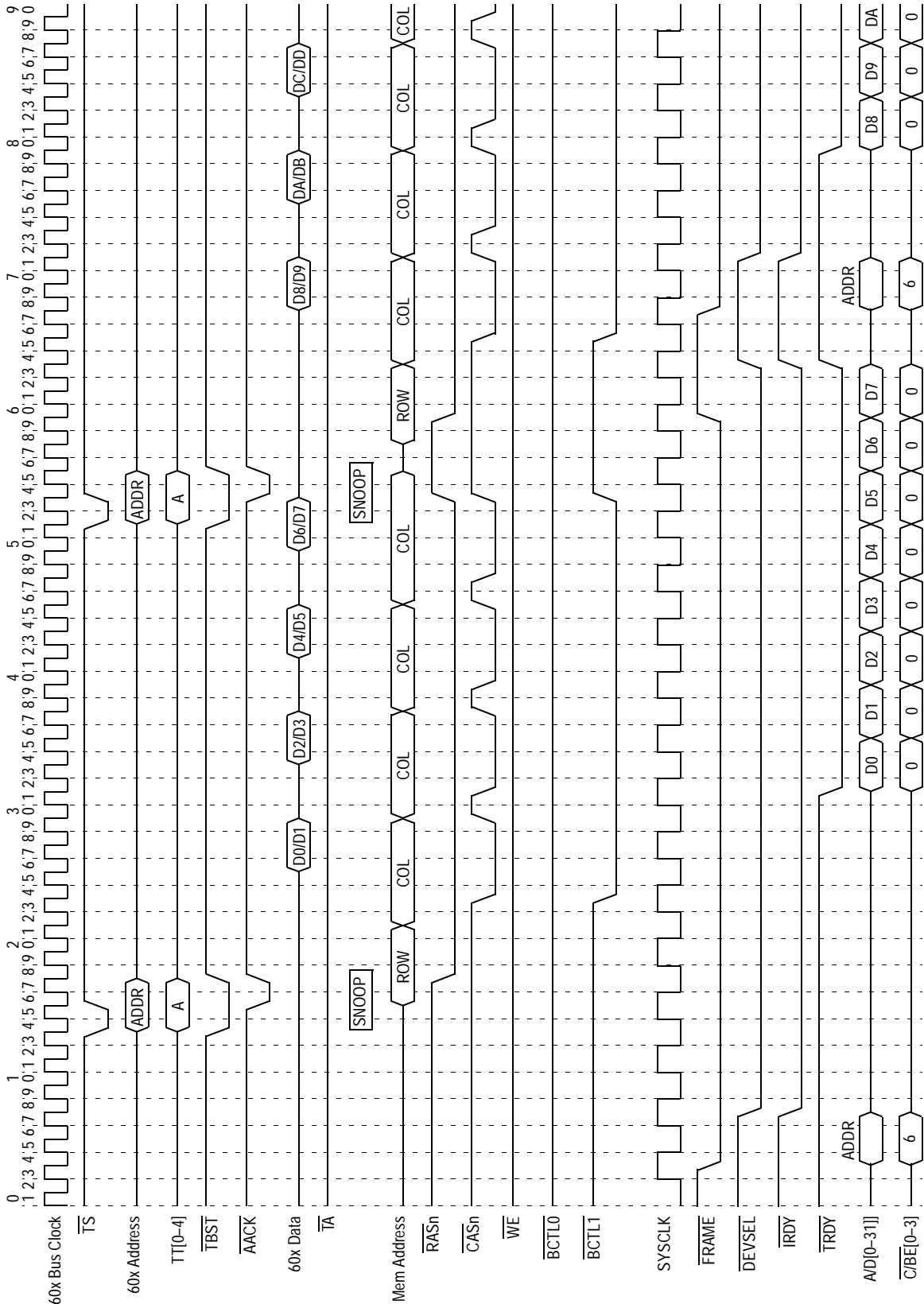


Figure 6-21. PCI Reads from Memory—Speculative Reads Enabled—60-ns DRAM with Flow-Through Buffers

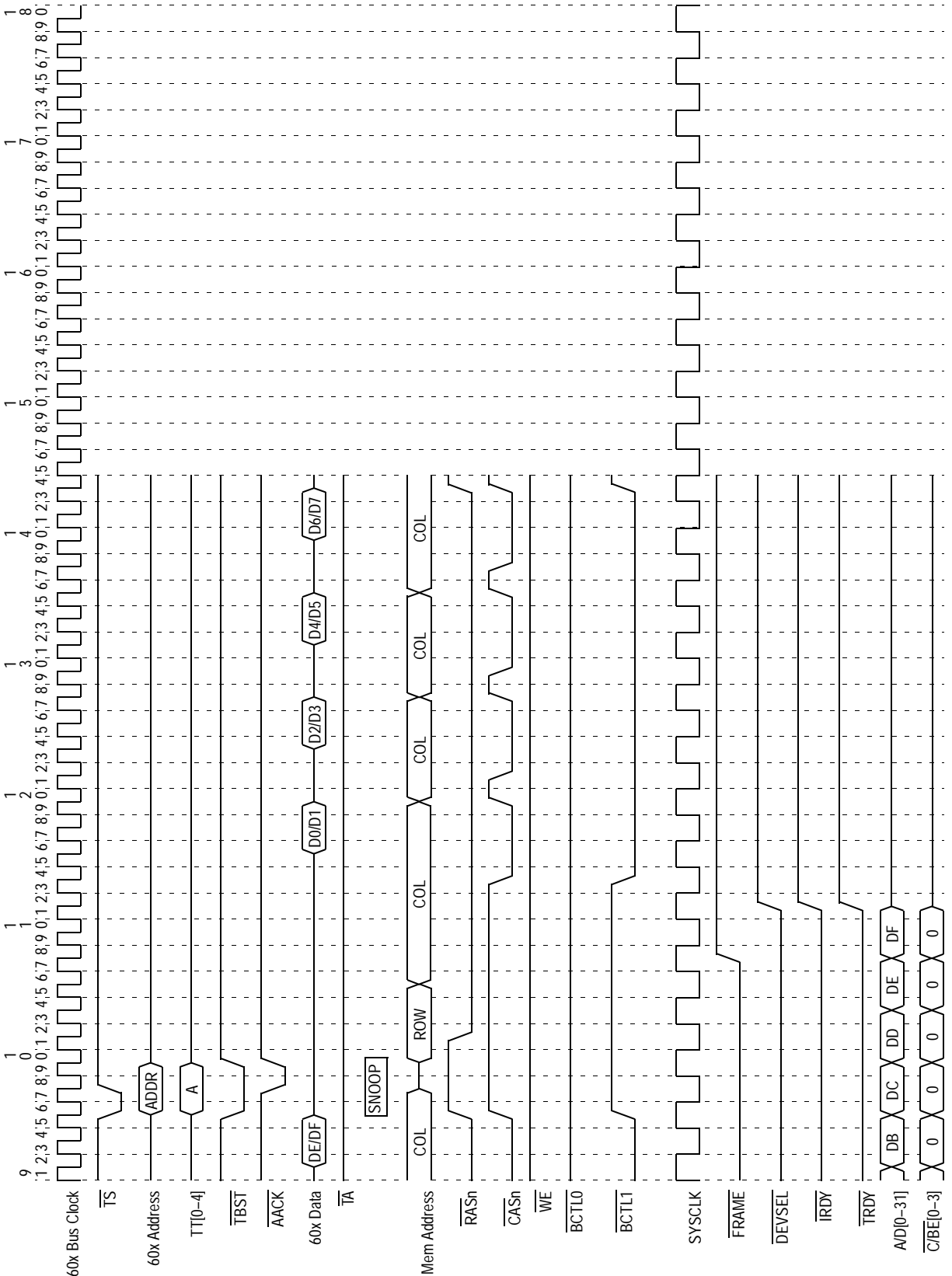


Figure 6-21. (Continued) PCI Reads from Memory—Speculative Reads Enabled—60-ns DRAM with Flow-Through Buffers

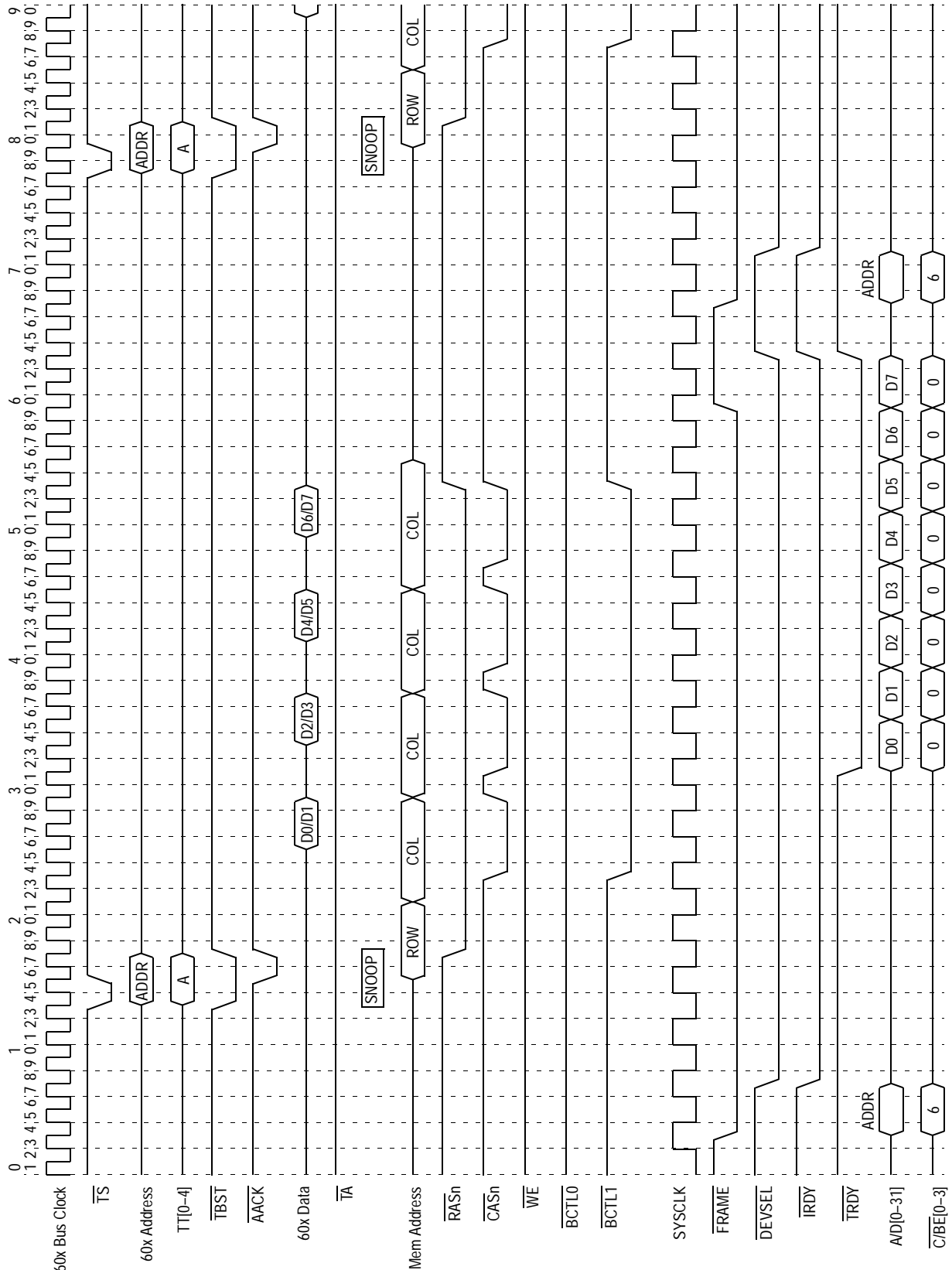


Figure 6-22. PCI Reads from Memory—Speculative Reads Disabled—60-ns DRAM with Flow-Through Buffers

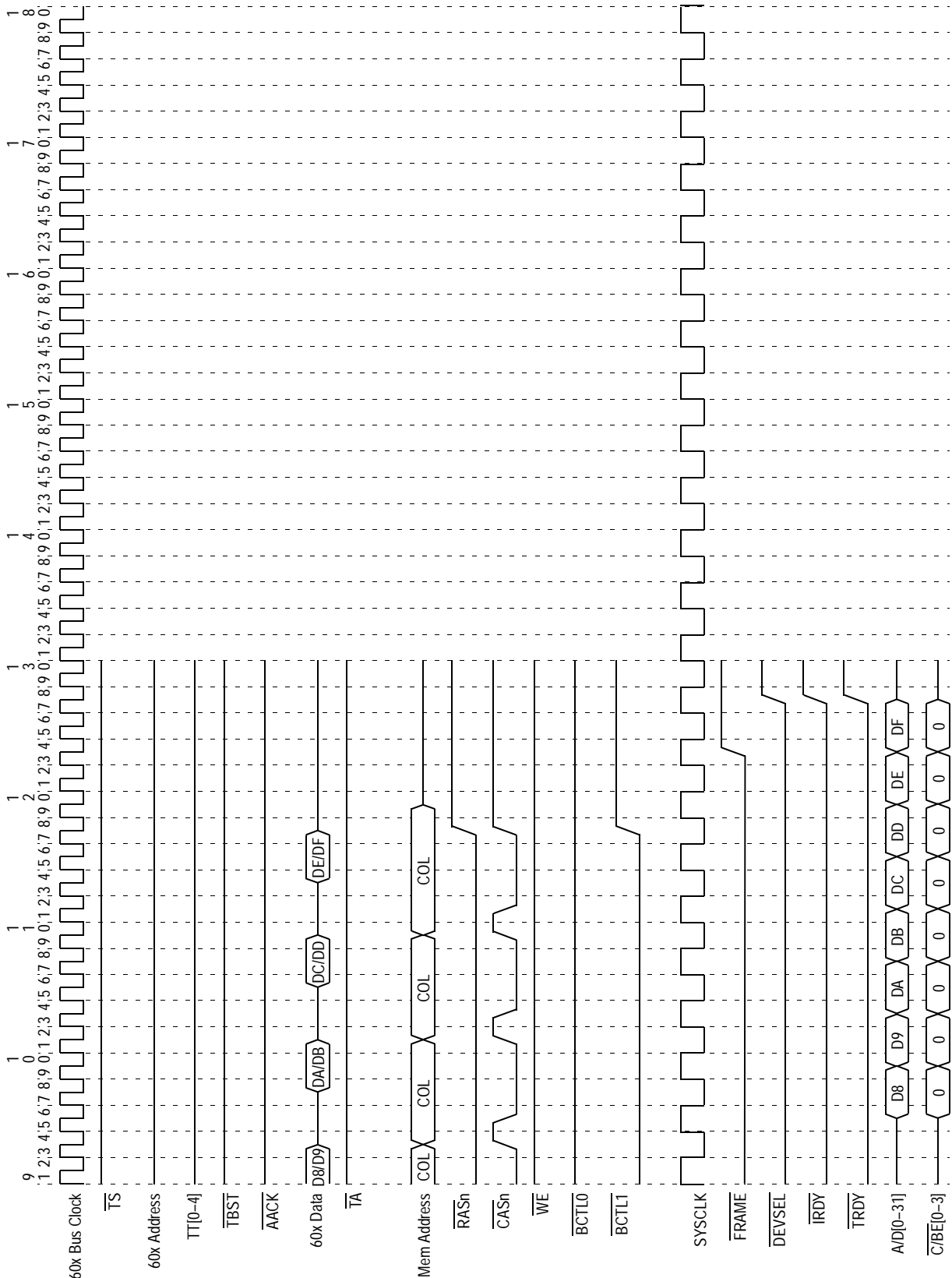


Figure 6-22. (Continued) PCI Reads from Memory—Speculative Reads Disabled—60-ns DRAM with Flow-Through Buffers

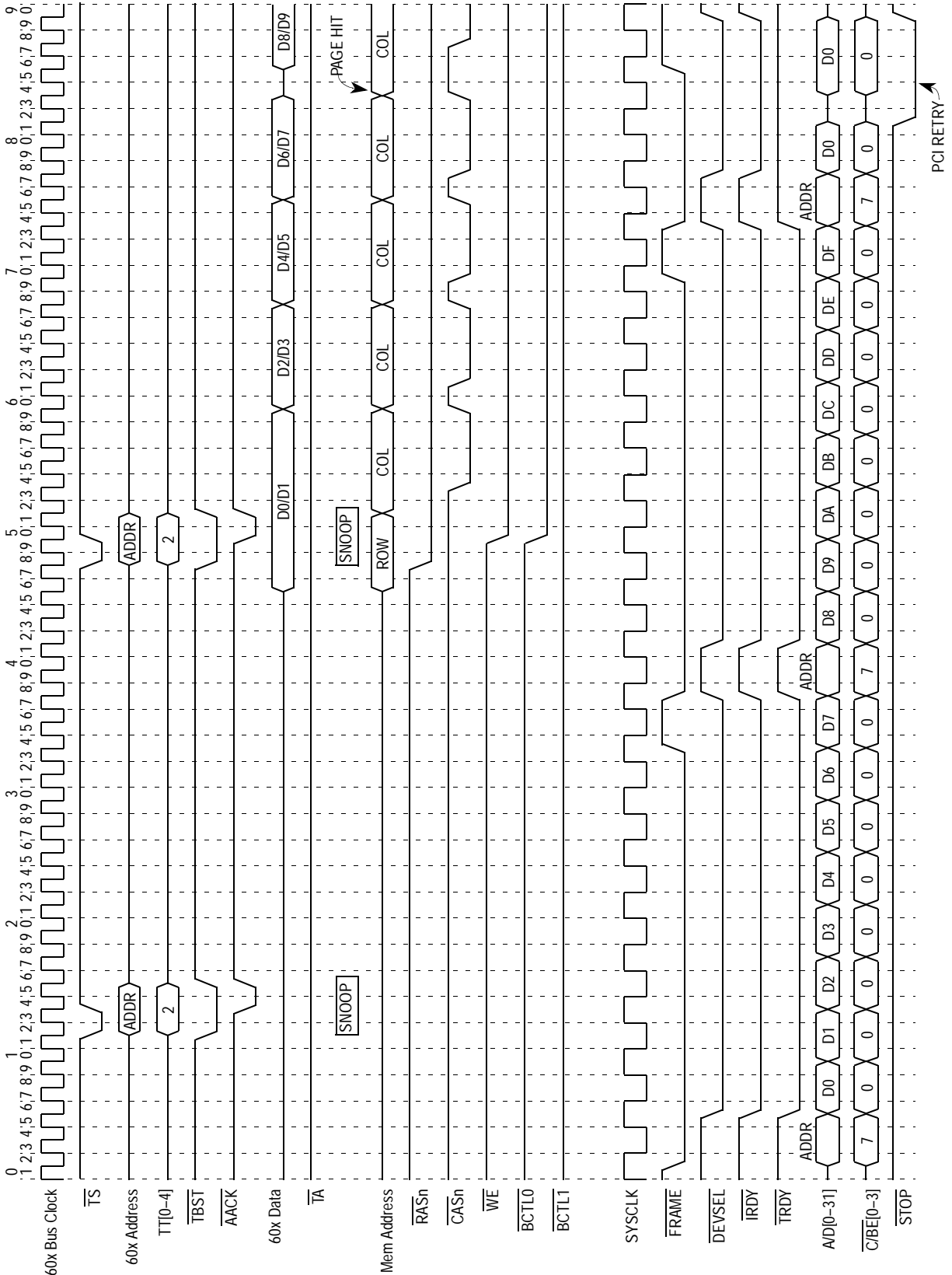


Figure 6-23. PCI Writes to Memory—60-ns DRAM with Flow-Through Buffers

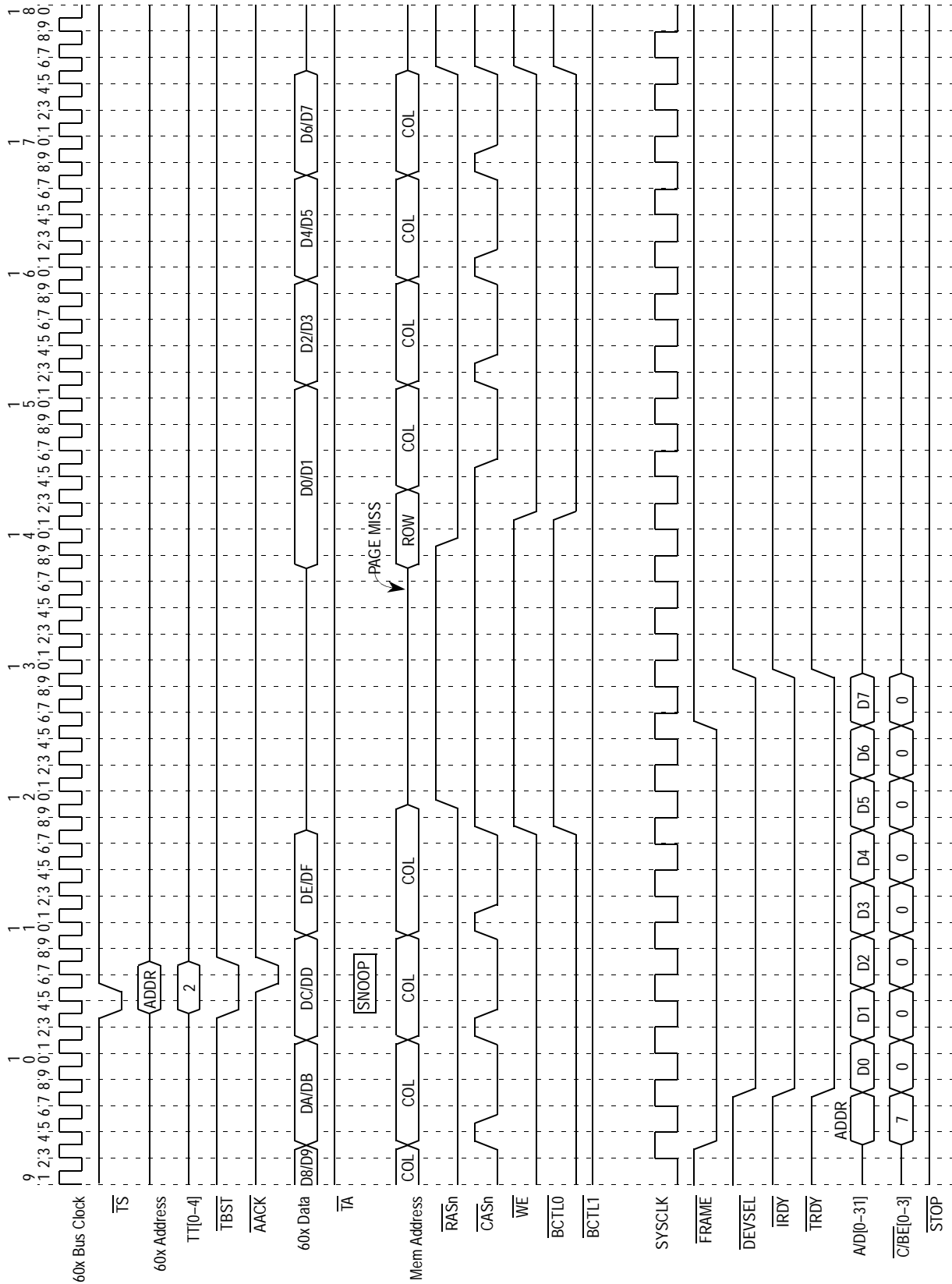


Figure 6-23. (Continued) PCI Writes to Memory—60-ns DRAM with Flow-Through

6.5 SDRAM Interface Operation

The MPC106 provides the necessary control functions and signals for JEDEC-compliant SDRAM devices. The MPC106 provides 1 Gbyte of addressable memory.

The 12 row/column multiplexed address signals (SDMA[1–12]) and two internal bank selects (SDBA[0–1]) provide for SDRAM densities from 1M to 128M in depth. (Note that SDBA1 can be used as SDMA0 for 64-/128-Mbit devices with only two internal banks.) Eight SDRAM chip select ($\overline{\text{CS}}$) signals support up to eight banks of memory. Eight SDRAM data input/output mask (DQM) signals are used to provide byte selection for 64-bit accesses.

In addition to DQM[0–7], $\overline{\text{CS}}$ [0–7], SDBA[0–1], and SDMA[1–11], there are 64 data signals (DH[0–31] and DL[0–31]), a write enable ($\overline{\text{WE}}$) signal, a column address strobe ($\overline{\text{SDCAS}}$) signal, a row address strobe ($\overline{\text{SDRAS}}$), a memory clock enable (CKE) signal, and eight parity bits (PAR[0–7]).

Note that when configured for SDRAM, the following MPC106 features are not supported:

- ECC
- Asynchronous L2 cache
- Partial L2 update with external byte write decode (that is, CF_WMODE must be cleared to 0b00)
- Modified memory tracking in PC emulation mode ($\overline{\text{SDRAS}}$ is multiplexed with $\overline{\text{PIRQ}}$)
- Latched data buffers ($\overline{\text{SDCAS}}$ is multiplexed with $\overline{\text{MDLE}}$)

Figure 6-24 shows an example of an eight-bank, 128-Mbyte system, comprised of 2 Mbit 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.

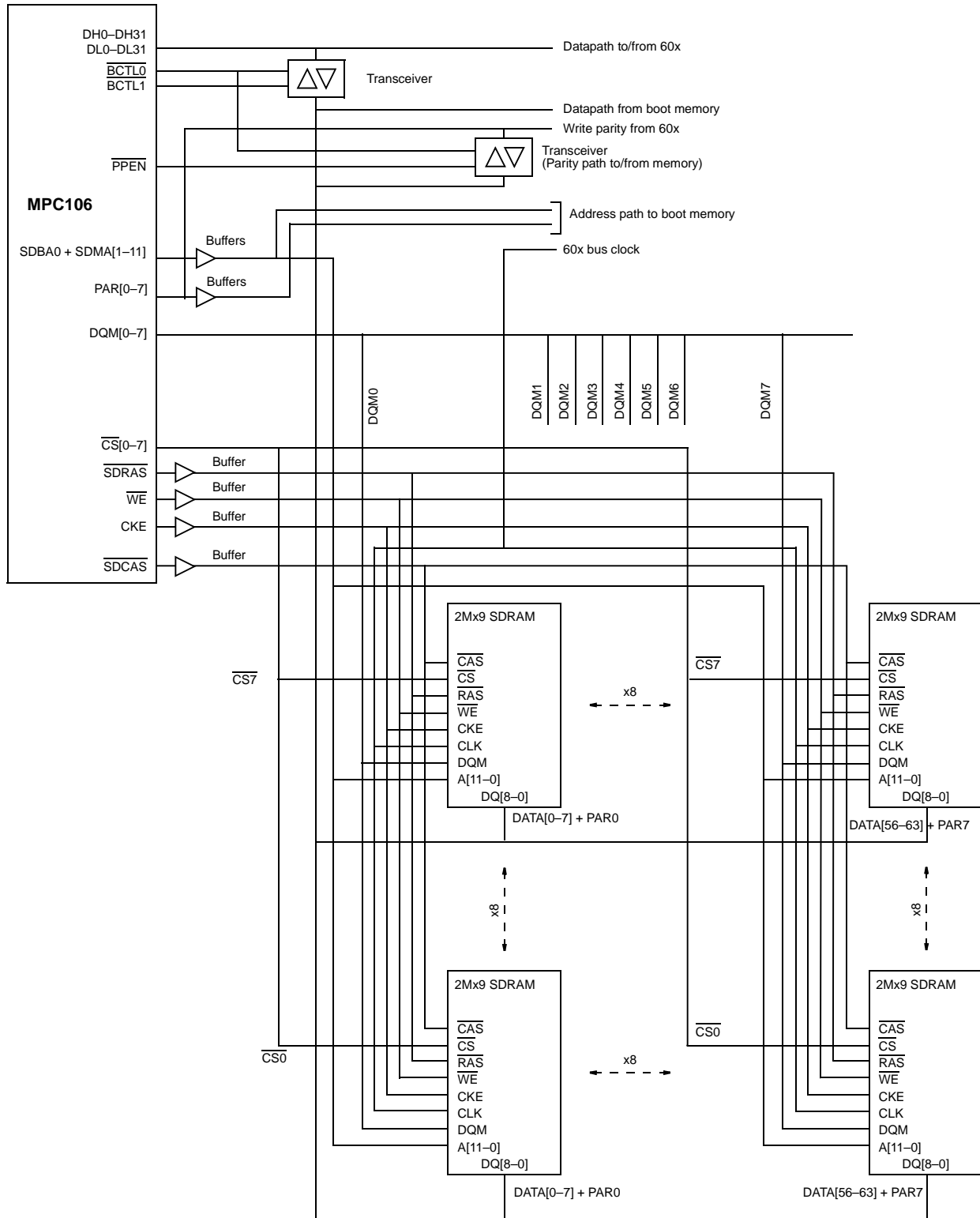


Figure 6-24. 128-Mbyte SDRAM System with Parity

6.5.1 Supported SDRAM Organizations

It is not necessary to use identical memory devices in each memory bank; individual memory banks may be of differing size. The MPC106 multiplexes the row and column address bits on either 12 memory address signals and 2 bank select signals or 13 memory address signals and 1 bank select signals. Individual SDRAM banks may be implemented with memory devices requiring fewer than 24 address bits.

Table 6-12 summarizes some of the memory configurations supported by the MPC106. Note that Table 6-12 is not an exhaustive list of all configurations that the MPC106 can support. The MPC106 can support any device that can accept the address multiplexing described in 6.5.2, “SDRAM Address Multiplexing,” without exceeding the 128-Mbyte physical bank limit.

Table 6-12. Supported SDRAM Device Configurations

SDRAM Devices	Number of Devices in a Physical Bank ¹	Device Configuration	Row Bits x Column Bits x Logical Banks ²	Physical Bank Size (Mbytes)	Eight Physical Banks of Memory (Mbytes)
16 Mbits 2 bank	16	2M x 2 banks x 4 bits	11 x 10 x 2	32	256
	8	1M x 2 banks x 8 bits	11 x 9 x 2	16	128
	4	512K x 2 banks x 16 bits	11 x 8 x 2	8	64
64 Mbits 2 bank	16	8M x 2 banks x 4 bits	13 x 10 x 2	128	1024
	8	4M x 2 banks x 8 bits	13 x 9 x 2	64	512
	4	2M x 2 banks x 16 bits	13 x 8 x 2	32	256
	2	1M x 2 banks x 32 bits	12 x 8 x 2	16	128
64 Mbits 4 bank	16	4M x 4 banks x 4 bits	12 x 10 x 4	128	1024
	8	2M x 4 banks x 8 bits	12 x 9 x 4	64	512
	4	1M x 4 banks x 16 bits	12 x 8 x 4	32	256
	2	512K x 4 banks x 32 bits	11 x 8 x 4	16	128
128 Mbits 2 banks	8	8M x 2 banks x 8 bits	13 x 10 x 2	128	1024
	4	4M x 2 banks x 16 bits	13 x 9 x 2	64	512
	2	2M x 2 banks x 32 bits	13 x 8 x 2	32	256
128 Mbits 4 banks	8	4M x 4 banks x 8 bits	12 x 10 x 4	128	1024
	4	2M x 4 banks x 16 bits	12 x 9 x 4	64	512
	2	1M x 4 banks x 32 bits	12 x 8 x 4	32	256

¹ A physical bank is defined for the MPC106 as a portion of memory addressed through a single SDRAM chip select signal (CS[0–7]). Certain SDRAM modules may have two physical banks and require two chip selects to be programmed to support the single module.

² A logical bank is defined for the MPC106 as a portion of memory addressed through a SDRAM bank select (SDBA[0–1]).

By using a memory polling algorithm at power-on reset, system firmware configures the MPC106 to correctly map the size of each bank in memory (the memory boundary registers). The MPC106 uses its bank map to assert the appropriate \overline{CS}_n signal for memory accesses according to the provided bank depths.

6.5.2 SDRAM Address Multiplexing

This section describes how the MPC106 translates processor addresses into SDRAM memory addresses. Note that the MPC106 SDRAM memory address signals SDMA[0–12] are labeled with SDMA0 as the most-significant address bit and SDMA12 as the least-significant address bit. Most SDRAM devices are labelled with A0 as the least significant address input. Therefore, the MPC106 SDMA[0–12] signals should be connected to SDRAM devices according to Figure 6-2.

System software must configure the MPC106 at power-on reset to appropriately multiplex the row and column address bits for each bank. This is accomplished using the MCCR1[Bank *n* row] parameters. Figure 6-13 shows the settings for MCCR1[Bank *n* row] and the corresponding address multiplexing.

Table 6-13. SDRAM Address Multiplexing

MCCR1[Bank <i>n</i> row]	SDRAM Address Phase	Processor/Memory Address
0b00 (64-/128-Mbit, 4 internal bank devices)	Row (\overline{SDRAS} asserted)	A7 = SDMA1 A8 = SDBA1 A9 = SDBA0 A[10–20] = SDMA[2–12]
	Column (\overline{SDCAS} asserted)	A[5–6] = SDMA[3–4] A8 = SDBA1 A9 = SDBA0 A[21–28] = SDMA[5–12]
0b01 (64-/128-Mbit, 2 internal bank devices)	Row	A7 = SDMA1 A8 = SDMA0 A9 = SDBA0 A[10–20] = SDMA[2–12]
	Column	A[5–6] = SDMA[3–4] A9 = SDBA0 A[21–28] = SDMA[5–12]
0b10 (Reserved)	—	—
0b11 (16-Mbit, 2 internal bank devices)	Row	A9 = SDBA0 A[10–20] = SDMA[2–12]
	Column	A[7–8] = SDMA[3–4] A9 = SDBA0 A[21–28] = SDMA[5–12]

Figure 6-25 shows an alternate view of the SDRAM address multiplexing.

Row x Col x Banks		msb Processor (Physical) Address lsb																																		
		0-4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31							
13x10x2	SDRAS						1	0	B	2	3	4	5	6	7	8	9	10	11	12																
	SDCAS				3	4			B													5	6	7	8	9	10	11	12							
13x9x2	SDRAS						1	0	B	2	3	4	5	6	7	8	9	10	11	12																
	SDCAS					4			B													5	6	7	8	9	10	11	12							
13x8x2 or 12x8x2	SDRAS						1	0	B	2	3	4	5	6	7	8	9	10	11	12																
	SDCAS								B													5	6	7	8	9	10	11	12							
12x10x4	SDRAS						1	B	B	2	3	4	5	6	7	8	9	10	11	12																
	SDCAS				3	4		B	B													5	6	7	8	9	10	11	12							
12x9x4	SDRAS						1	B	B	2	3	4	5	6	7	8	9	10	11	12																
	SDCAS					4		B	B													5	6	7	8	9	10	11	12							
12x8x4 or 11x8x4	SDRAS						1	B	B	2	3	4	5	6	7	8	9	10	11	12																
	SDCAS							B	B													5	6	7	8	9	10	11	12							
11x10x2	SDRAS								B	2	3	4	5	6	7	8	9	10	11	12																
	SDCAS						3	4	B													5	6	7	8	9	10	11	12							

- The maximum activate-to-precharge interval (controlled by PGMAX) has not expired.

Page mode can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save clock cycles from subsequent burst accesses that hit in an active page. SDRAM page mode is controlled by the BSTOPRE[0–9], and PGMAX parameters. Page mode is disabled by clearing either BSTOPRE[0–9], or the PGMAX parameter.

The page open duration counter is loaded with BSTOPRE[0–9] every time the page is accessed (including page hits). When the counter expires (or when PGMAX expires) the open page is closed with a precharge bank command. Page hits can occur at any time in the interval specified by BSTOPRE.

The 1-byte memory page mode register at address offset 0xA3 contains the PGMAX parameter that controls how long the MPC106 retains the currently accessed page (row) in memory. The PGMAX parameter specifies the activate-to-precharge interval (sometimes called row active time or t_{RAS}). The PGMAX value is multiplied by 64 to generate the actual number of clock cycles for the interval. When PGMAX is programmed to 0x00, page mode is disabled.

The value for PGMAX depends on the specific SDRAM devices used, the ROM system, and the operating frequency of the MPC106. When the interval specified by PGMAX expires, the MPC106 must close the active page by issuing a precharge bank command. PGMAX must be sufficiently less than the maximum row active time for the SDRAM device to ensure that the issuing of a precharge command is not stalled by a memory access. If a memory access is in progress at the time PGMAX expires, the MPC106 must wait for the access to complete before issuing the precharge command to the SDRAM. In the worst case, the MPC106 initiates a memory access one clock cycle before PGMAX expires. If ROM is located on the 60x/memory bus, the longest access that could potentially stall a precharge is a burst read from ROM. If ROM is located on the PCI bus, the longest memory access is a burst read from the SDRAM.

The MPC106 also requires two clock cycles to issue a precharge bank command to the SDRAM device. So, the PGMAX interval must be further reduced by two clock cycles.

Therefore, PGMAX should be programmed according to the following equation, and shown in Figure 6-26:

$$PGMAX < [t_{RAS(MAX)} - (\text{worst case memory access}) - 2] \div 64$$

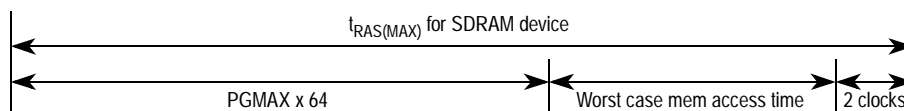


Figure 6-26. PGMAX Parameter Setting for SDRAM Interface

For example, consider a system with a 60x bus clock frequency of 66 MHz using SDRAMs with a maximum row active time ($t_{RAS(MAX)}$) of 100 μ s. The maximum number of clock cycles between activate bank and precharge bank commands is 66 MHz \times 100 μ s = 6600 clock cycles.

If the system uses 8-bit ROMs on the 60x/memory bus, a burst read from ROM follows the timing shown in Figure 6-50. The minimum time allowed for ROM devices to enter high impedance is two clock cycles. This delay is enforced after all ROM accesses preventing any other memory access from starting. Therefore a burst read from an 8-bit ROM takes:

$$\{[(ROMFAL + 2) \times 8 + 3] \times 4 + 5\} + 2 \text{ clock cycles to enter high-impedance}$$

So, if $MCCR1[ROMFAL] = 4$, the interval for a 60x burst read from an 8-bit ROM takes:

$$\{[(4 + 2) \times 8 + 3] \times 4 + 5\} + 2 = 209 + 2 = 211 \text{ clock cycles}$$

Plugging the values into the PGMAX equation above:

$$PGMAX < (6600 - 211 - 2) \div 64 = 99.8 \text{ clock cycles}$$

The value stored in PGMAX would be 0b0110_0011 (or 99 clock cycles).

6.5.5 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, the memory page mode register, and the memory control configuration registers (MCCRs). See Section 3.2.8, “Memory Interface Configuration Registers,” for more detailed descriptions of the MICRs and MCCRs.

NOTE

The MPC106 can only support certain configurations of parity/RMW/ECC, DRAM/EDO/SDRAM, and buffer types. Section 6.4.3.1, “Supported Memory Interface Configurations,” describes the supported memory interface configurations.

The programmable parameters relevant to the SDRAM interface are:

- Memory bank starting and ending addresses (memory boundary registers)
- Memory bank enables (memory bank enable register)
- PGMAX—maximum activate to precharge interval (also called row active time or t_{RAS})
- 501_MODE—501 buffer mode
- SREN—self-refresh enable
- RAMTYP—RAM type
- PCKEN—memory interface parity checking/generation enable

- Row address configuration for each bank
- BSTOPRE[0–9]—burst to precharge interval
- EXT_ECM_PAR_EN—External ECM parity enable
- EXT_ECM_ECC_EN—External ECM ECC enable
- ECC_EN—ECC enable
- REFINT—refresh interval
- BUF_MODE—buffer mode
- RMW_PAR—read-modify-write parity enable
- REFREC—refresh recovery interval
- RDLAT—data latency from read command
- PRETOACT—precharge-to-activate interval
- ACTOPRE—activate-to-precharge interval
- WCBUF—memory write buffer type
- RCBUF—memory read buffer type
- SDMODE—SDRAM mode register
- ACTORW—activate-to-read/write interval

Note that any unused banks should have their starting and ending addresses programmed out of the range of memory banks in use. If a disabled bank has its starting and ending address defined as overlapping an enabled bank's address space, there may be system memory corruption in the overlapping address range. Always map unused memory banks' starting and ending addresses to memory space that is not used by the system.

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 μ s 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 MPC106 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 6.5.6, “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.

6.5.6 JEDEC Standard SDRAM Interface Commands

The MPC106 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 MPC106 provides the following SDRAM interface commands:

- **Activate-bank**—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-bank**—Restores data from the sense amplifiers to the appropriate row. Also initializes the sense amplifiers in preparation for reading another row in the SDRAM internal bank. A precharge command must be issued after a read or write, if the row address changes on the next access. Note that the MPC106 uses SDMA2 to distinguish between the precharge-bank and precharge-all-banks commands. This corresponds to the A10/AP input on typical SDRAM devices. The SDRAMs must be compatible with this format.
- **Precharge-all-banks**—Initializes the sense amplifiers of all SDRAM internal banks. Note that the MPC106 uses SDMA2 to distinguish between the precharge-all-banks and precharge-bank commands. The SDRAMs must be compatible with this format.
- **Read**—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.
- **Write**—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.
- **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 MPC106 only supports a burst of four. Burst lengths of 1, 2, 8, and a page for SDRAMs are not supported by the MPC106.

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 MPC106 then transfers the information in MCCR4[SDMODE] to the SDRAM array by issuing the mode-set command. See Section 6.5.7.2, “SDRAM Mode-Set Command Timing,” for timing information.

Note that for 64- and 128-Mbit SDRAMs, the mode-set command uses a 14-bit mode register data field. On the MPC106, the 2 most significant bits are forced to 0 and concatenated to the 12-bit MCCR4[SDMODE] parameter during mode-set command execution.

- Self-refresh—Used when the SDRAM device is in standby for very long periods of time (corresponding with sleep or suspend mode on the MPC106). 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.

The MPC106 automatically issues a precharge command to the SDRAM when the BSTOPRE or PGMAX intervals have expired, regardless of pending memory transactions from the PCI bus or 60x. The MPC106 can perform precharge cycles concurrent with snoop broadcasts for PCI transactions.

The SDRAM interface command encodings are summarized in Table 6-14.

Table 6-14. SDRAM Command Encodings

	Command							
	Activate Bank	Precharge All Banks ¹	Precharge Bank	Read	Write	Refresh (CBR)	Mode Set	Self-Refresh
Previous CKE	High	High	High	High	High	High	High	High
Current CKE	x	x	x	x	x	High	x	Low
$\overline{\text{CS}}_n$	Low	Low	Low	Low	Low	Low	Low	Low
$\overline{\text{SDRAS}}$	Low	Low	Low	High	High	Low	Low	Low
$\overline{\text{SDCAS}}$	High	High	High	Low	Low	Low	Low	Low
$\overline{\text{WE}}$	High	Low	Low	High	Low	High	Low	High
DQM_n	x	x	x	Low	Low	x	x	x
SDBA[0–1]	Bank select	x	Bank select	Bank select	Bank select	x	Opcode	x
SDMA1	Row	x	x	Column	Column	x	Opcode	x

Table 6-14. SDRAM Command Encodings (continued)

	Command							
	Activate Bank	Precharge All Banks ¹	Precharge Bank	Read	Write	Refresh (CBR)	Mode Set	Self-Refresh
SDMA2	Row	High	Low	Column	Column	x	Opcode	x
SDMA[3–12]	Row	x	x	Column	Column	x	Opcode and mode	x

¹ Note that the MPC106 only issues the precharge all banks command at initialization when the MCCR1[MEMGO] parameter is set.

6.5.7 SDRAM Interface Timing

System software is responsible for optimal configuration of the SDRAM programmable timing parameters (RDLAT, BSTOPRE, PRETOACT, ACTOPRE, 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.

6.5.7.1 Processor-to-SDRAM Transaction Examples

The figures in this section provide signal timing examples of 60x processor-to-system-memory transactions for an SDRAM configuration. Figure 6-27 shows several burst read operations, Figure 6-28 shows a burst write operation, Figure 6-29 shows burst read followed by burst write operations, Figure 6-30 shows a single-beat read operation, and Figure 6-31 shows a single-beat write operation.

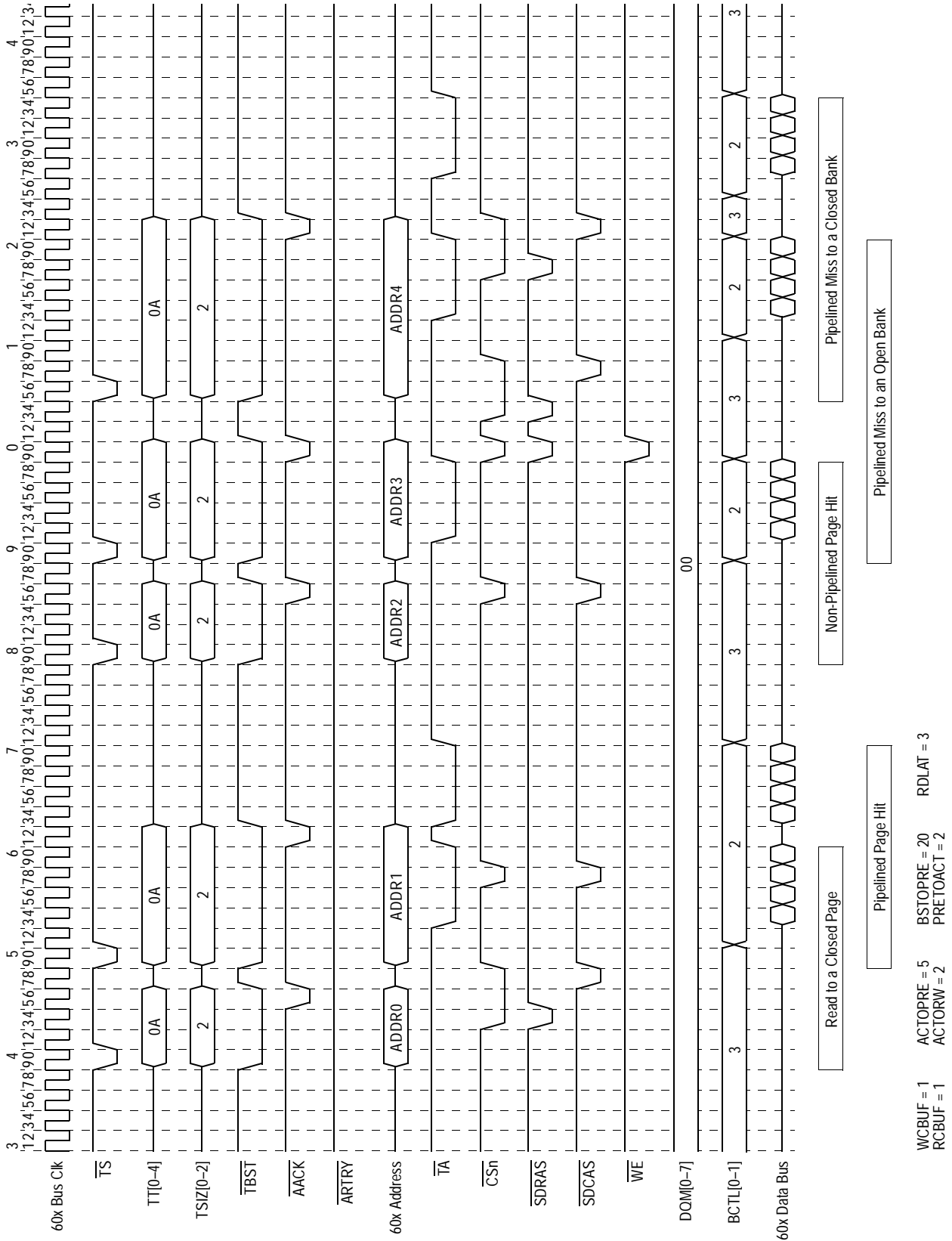


Figure 6-27. SDRAM Burst Read Timing

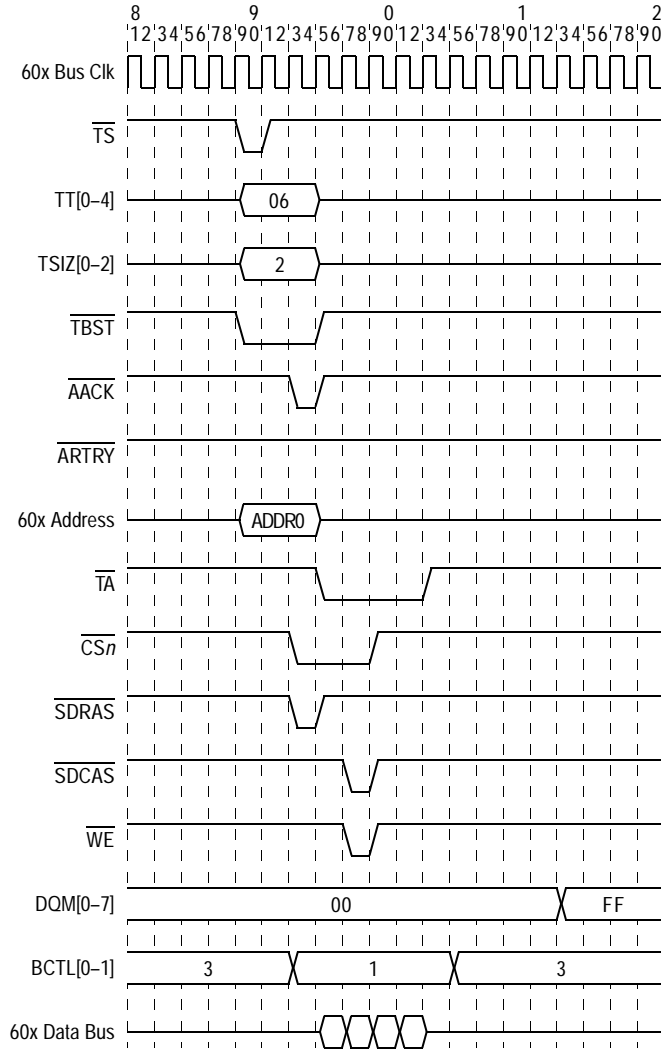


Figure 6-28. SDRAM Burst Write Timing

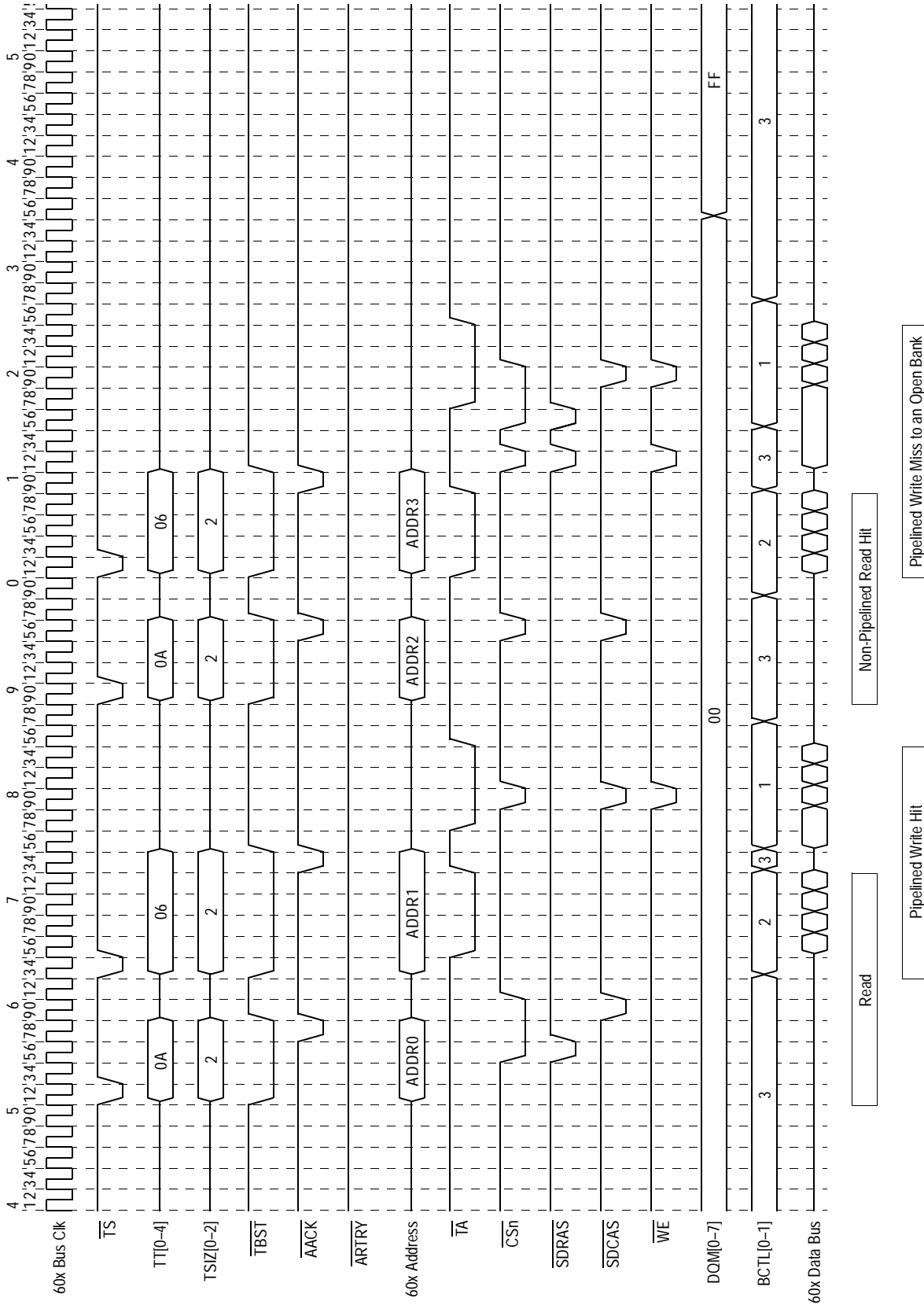


Figure 6-29. SDRAM Burst Read Followed By Burst Write Timing

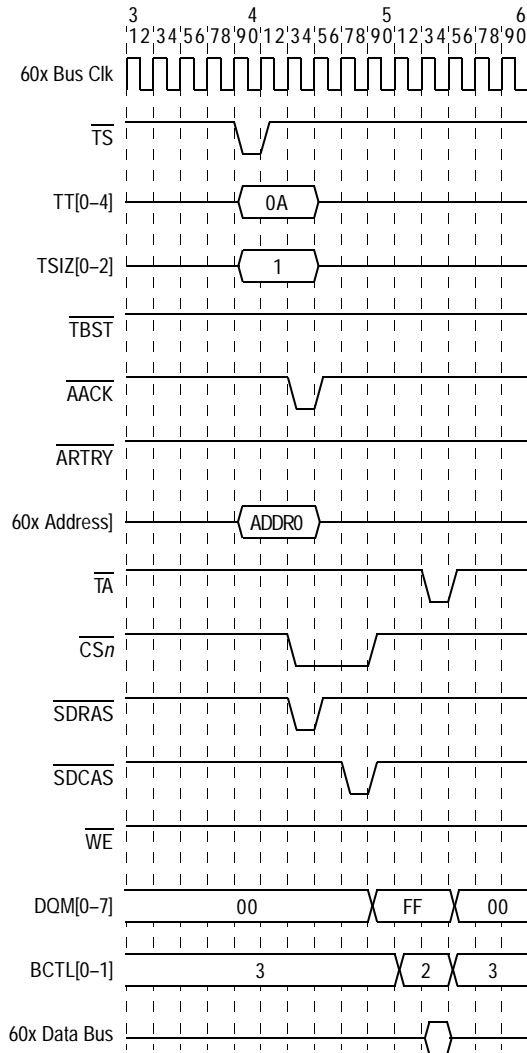
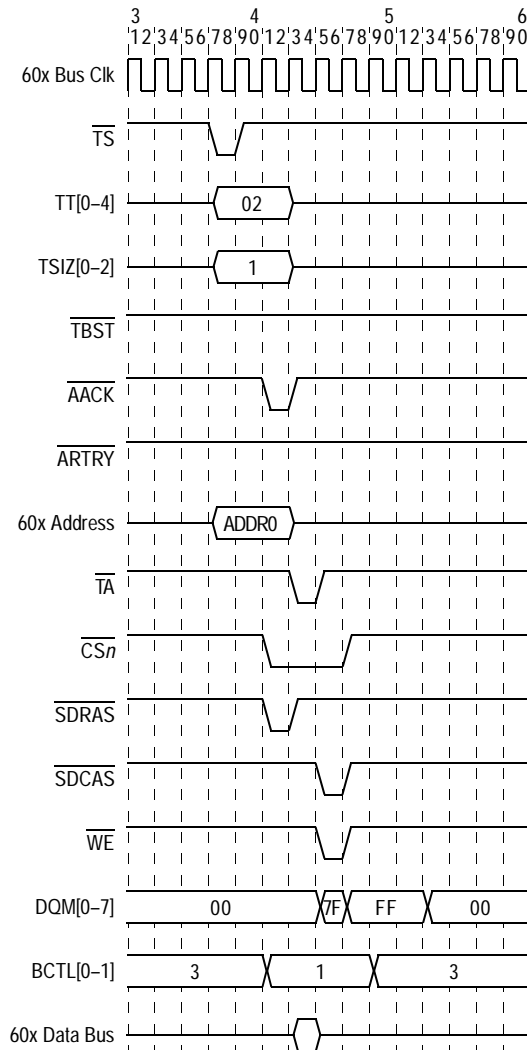
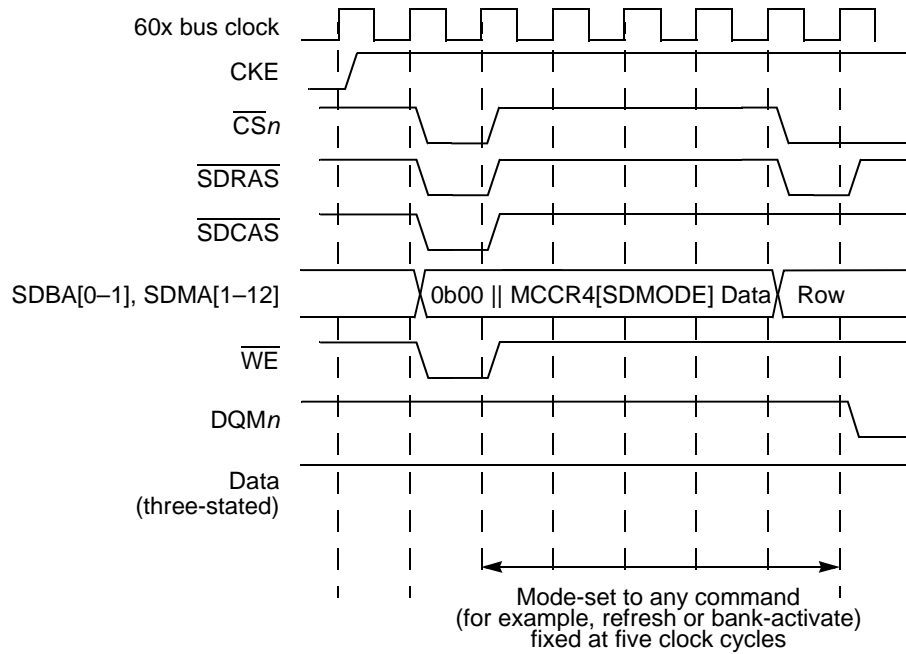


Figure 6-30. SDRAM Single-Beat Read Timing


Figure 6-31. SDRAM Single-Beat Write Timing

6.5.7.2 SDRAM Mode-Set Command Timing

The MPC106 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. Note that for 64- and 128-Mbit SDRAMs, the mode-set command uses a 14-bit mode register data field. On the MPC106, the 2 most significant bits are forced to 0 and concatenated to the 12-bit MCCR4[SDMODE] parameter during mode-set command execution. The timing of the mode-set command is shown in Figure 6-32.


Figure 6-32. SDRAM Mode-Set Command Timing

6.5.8 SDRAM Parity and RMW Parity

When configured for SDRAM, the MPC106 supports two forms of parity checking and generation—normal parity and read-modify-write (RMW) parity. Normal parity assumes that each of the eight parity bits is controlled by a separate DQM signal. Thus, for a single-beat write to system memory, the MPC106 generates a parity bit for each byte written to memory.

RMW parity assumes that all eight parity bits are controlled by a single DQM signal and therefore must be written as a single 8-bit quantity (that is, a byte). Therefore, for any write operation to system memory that is less than a double word, the MPC106 must latch the write data, read an entire 64-bit double word from memory, check the parity of the double word read from memory, merge the write data with the double-word read from memory, regenerate parity for the new double word, and finally write the new double word back to memory.

The MPC106 checks parity on all memory reads, provided parity checking is enabled (PCKEN = 1). The MPC106 generates parity for the following operations:

- PCI to memory write operations
- L1 and L2 copyback operations
- L2 castout operations
- 60x single-beat write operations with RMW parity enabled (RMW_PAR = 1)

The 60x processor is expected to generate parity for all other 60x to memory write operations as the data goes directly to memory and does not pass through the MPC106.

6.5.8.1 RMW Parity Latency Considerations

When RMW parity is enabled, the time required to read, modify, and write increases latency for processor single-beat writes to system memory and PCI writes to system memory. All other transactions are unaffected and operate as in normal parity mode.

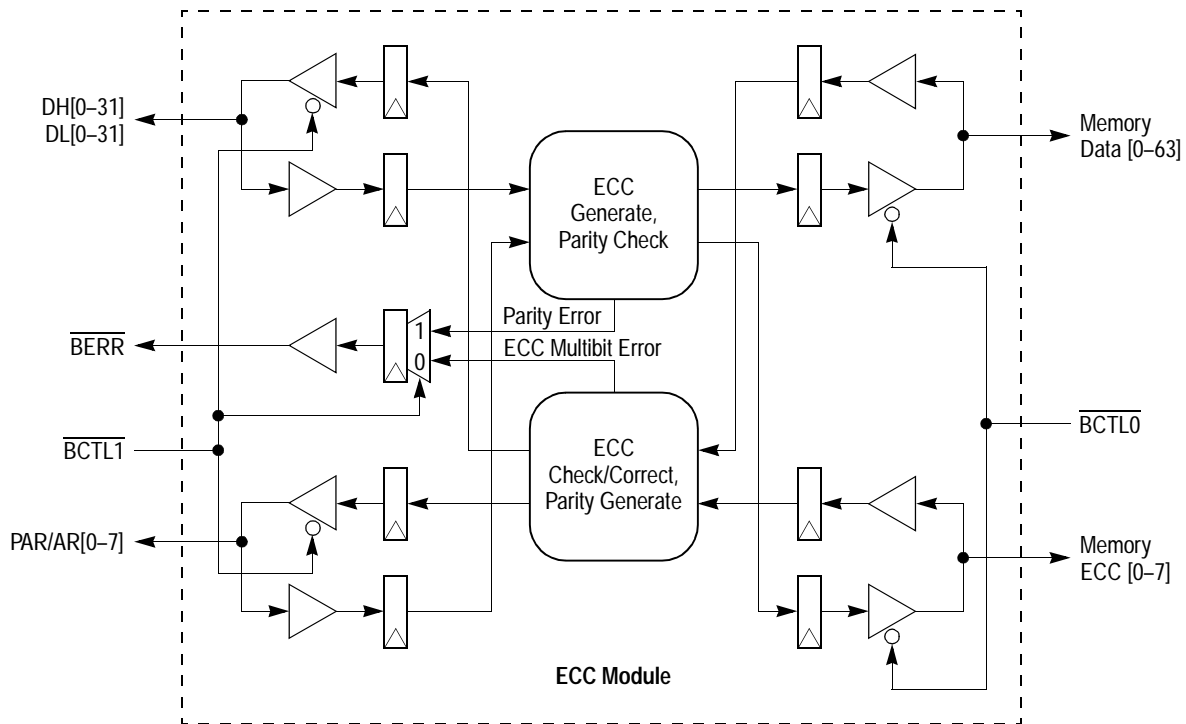
For 60x processor single-beat writes to system memory, the MPC106 latches the data, performs a double-word read from system memory (checking parity), and then merges the write data from the processor with the data read from memory. The MPC106 then generates new parity bits for the merged double word and writes the data and parity to memory. The read-modify-write process adds six clock cycles to a single-beat write operation. If page mode retention is enabled ($BSTOPRE \neq 0$ and $PGMAX \neq 0$), then the MPC106 will keep the memory in page mode for the read-modify-write sequence. Since the processor drives all eight parity bits during 60x burst writes to system memory, these transactions go directly to the SDRAMs with no performance penalty.

For PCI writes to system memory with RMW parity enabled, the MPC106 latches the data in the internal PCI-to-system-memory-write buffer (PCMWB). If the PCI master writes complete double words to system memory, the MPC106 generates the parity bits when the PCMWB is flushed to memory. However, if the PCI master writes 32-, 16-, or 8-bit data that cannot be gathered into a complete double word in the PCMWB, a read-modify-write operation is required. The MPC106 performs a double-word read from system memory (checking parity), and then merges the write data from the PCI master with the data read from memory. The MPC106 then generates new parity for the merged double word and writes the data and parity to memory. If page mode retention is enabled ($BSTOPRE \neq 0$ and $PGMAX \neq 0$), the MPC106 keeps the memory in page mode for the read-modify-write sequence.

6.5.9 External Error Checking Module (ECM) Support

With Rev. 4.0, the MPC106 supports an external, in-line, error checking module (ECM)/buffer for use with SDRAM, including enhancements to the SDRAM error logging hardware for this mode.

Figure 6-33 shows a block diagram of the recommended external ECM implementation.


Figure 6-33. External ECM Block Diagram

From Figure 6-33, note the following:

- The MPC106 controls $\overline{\text{BCTL0}}$ (write enable) and $\overline{\text{BCTL1}}$ (read enable).
- The MPC106 assumes that the $\overline{\text{BERR}}$ signal is valid on clock cycles when data is valid on the destination bus (that is, the memory bus for write operations and the 60x bus for read operations). The MPC106 samples the $\overline{\text{BERR}}$ signal at the appropriate time to mask invalid error indications.
- The 60x bus has optional parity. The memory bus can have either parity or ECC.

An error indicator signal from the ECM is connected to the $\overline{\text{BERR}}$ input signal. Since the $\overline{\text{BERR}}$ signal is multiplexed with the $\overline{\text{ISA_MASTER}}$ signal, systems that use the $\overline{\text{ISA_MASTER}}$ signal cannot use an external ECM. Also, the signal functions as $\overline{\text{BERR}}$ only if external ECM mode is enabled ($\text{MCCR2}[\text{EXT_ECM_ECC_EN}] = 1$ and $\text{MCCR2}[\text{EXT_ECM_PAR_EN}] = 1$). With this implementation, $\overline{\text{BERR}}$ is used to report both read parity/ECC multibit errors and write parity errors. See Section 9.3.2.3.1, “External ECM Errors,” for more information about external ECM error reporting.

Figure 6-15 describes the parameter settings necessary to configure the MPC106 to use an external ECM.

Table 6-15. MPC106 Parameter Values for External ECM

Address Offset	Register	Bit(s)	Field Name	Setting for External ECM	Description
0xC0	ErrEnR1	2	Memory parity/ECC enabled	1	Memory parity/ECC enable is enabled
0xC4	ErrEnR2	3	ECC multibit error enable	0	On-board ECC multibit error is disabled
0xF0	MCCR1	16	PCKEN	0	PCKEN is disabled
0xF4	MCCR2	19	EXT_ECM_ECC_EN	1	EXT_ECM_ECC_EN is enabled
0xF4	MCCR2	18	EXT_ECM_PAR_EN	1	EXT_ECM_PAR_EN is enabled
0xF4	MCCR2	17	ECC_EN	0	On-board ECC is disabled
0xF4	MCCR2	0	RMW_PAR	1	RMW parity is enabled. This is necessary to prevent corruption on byte, half-word, and word writes to memory.
0xF8	MCCR3	23–20	RDLAT	2+ CAS latency	Set MCCR3[RDLAT] to 2 + CAS latency value programmed in MCCR4[SDMODE]
0xFC	MCCR4	21	WCBUF	1	WCBUF is enabled
0xFC	MCCR4	20	RCBUF	1	RCBUF is enabled

6.5.9.1 External ECM Timing Examples

Figure 6-34 shows burst read timing when configured for an external ECM.

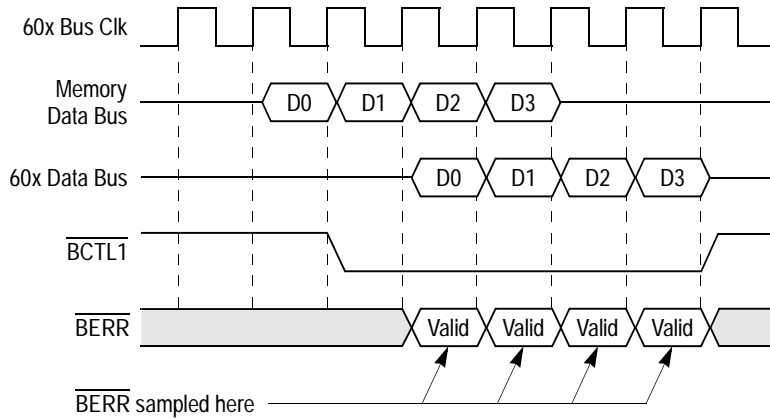


Figure 6-34. Burst Read Timing with External ECM

Figure 6-35 shows single-beat read timing when configured for an external ECM.

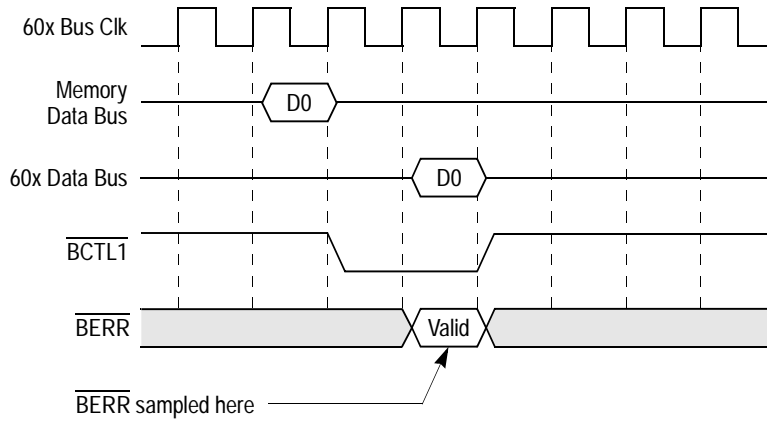


Figure 6-35. Single-Beat Read Timing with External ECM

Figure 6-36 shows burst write timing when configured for an external ECM.

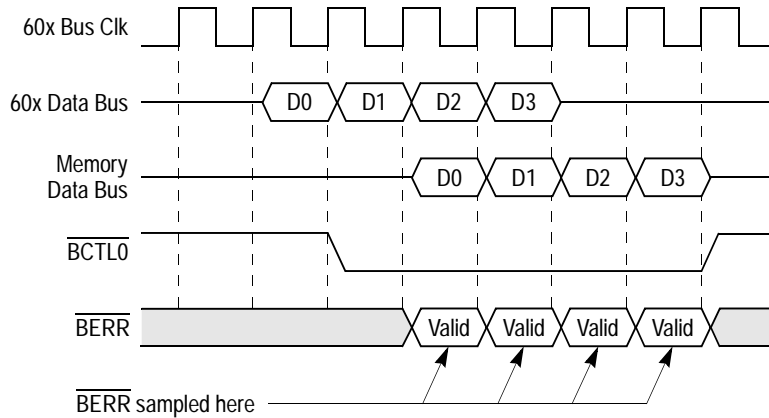


Figure 6-36. Burst Write Timing with External ECM

Figure 6-37 shows single-beat write timing when configured for an external ECM.

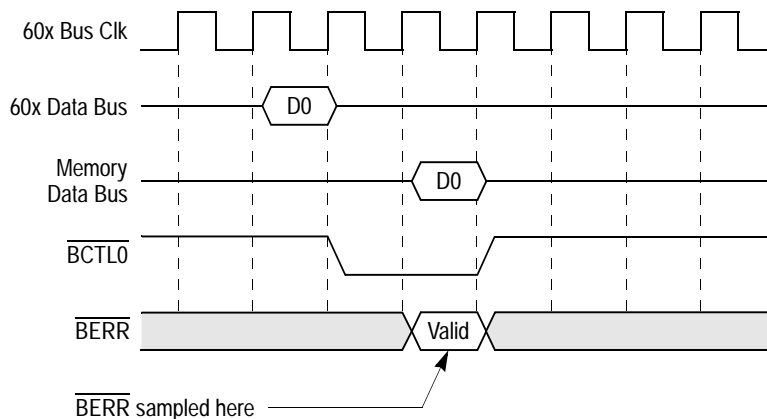


Figure 6-37. Single-Beat Write Timing with External ECM

6.5.10 SDRAM Refresh

The memory interface supplies CBR refreshes to SDRAM according to the interval specified in MCCR2[REFINT]. REFINT is the refresh interval. When REFINT expires and the memory bus is idle, the MPC106 issues a precharge and then a refresh command to the SDRAM devices. However, if the memory bus is busy with a transaction, the refresh request is not performed and an internal, 4-bit, missed-refresh counter is incremented. The refresh interval timer is reset to the value in REFINT and the process begins again. When the refresh interval counter expires and the bus is idle, the MPC106 performs all the missed refreshes back-to-back and the missed refresh counter is cleared. If the number of missed refreshes exceeds 16, the counter overflows and causes a refresh overflow error. See Section 9.3.2.5, “System Memory Refresh Overflow Error,” for more information about the reporting of these errors. In the worst case, the MPC106 misses 16 refreshes and must perform all 16 refreshes. Figure 6-38 shows this worst case situation repeated over the devices refresh period.

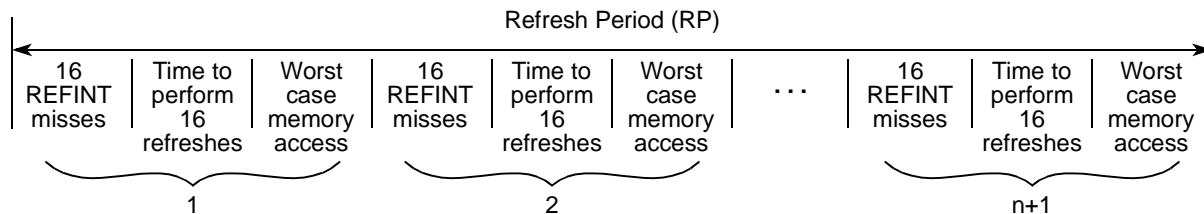


Figure 6-38. SDRAM Refresh Period

The value stored in REFINT must permit the MPC106 to supply refreshes within the refresh period specified by the SDRAM device. Another factor in calculating the value for REFINT is the overhead for the MPC106 to actually issue a refresh command to the SDRAM device. The MPC106 has to precharge any open banks before it can issue the refresh command. The MPC106 requires two clock cycles to issue a precharge to an internal bank; with the possibility of two banks open simultaneously, this equates to four clock cycles. The MPC106 must also wait for the PRETOACT interval to pass before issuing the refresh command. The refresh command itself takes four clock cycles (see Figure 6-39) with a dead cycle needed between subsequent refresh commands.

REFINT must also allow for a potential collision between memory accesses and refresh cycles. In the worst case, the refresh may have to wait the number of clock cycles required by the longest access. For example, if ROM is located on the 60x/memory bus and a ROM access is in progress at the time a refresh operation needs to be performed, the refresh must wait until the ROM access has completed. If ROM is located on the 60x/memory bus, the longest access that could potentially stall a refresh is a burst read from ROM. If ROM is located on the PCI bus, the longest memory access is a burst read from the SDRAM.

Therefore, REFINT should be programmed according to the following equation:

$$\text{REFINT} < \frac{\text{RP}}{(n + 1)16} - \frac{16\text{ROH}}{16} - \frac{\text{TWACC}}{16}$$

Where:

RP is the refresh period of the device = refresh period per bank x the number of banks x memory frequency

$n = (\text{the number of rows per bank} \times \text{the number of banks per device}) \div 16$

ROH is the refresh overhead imposed by the MPC106 and is composed of the precharge, the PRETOACT interval, the 4 clock cycles to issue the refresh command, and one dead cycle between refreshes.

TWACC is the worst case access time for the slowest device on the memory bus.

Consider a typical SDRAM device having two internal banks, 2K rows in each bank (4K rows total) with a refresh period of 32 ms for 2K rows. This means that the MPC106 must refresh each internal bank (2K rows) every 32 ms. In this example there are two banks, so to refresh the whole SDRAM it takes 64 ms. If the memory bus operates at 66 MHz, $\text{RP} = 64 \text{ ms} \times 66 \text{ MHz} = 4224000$ clock cycles to refresh all 4K rows. In this example $n = 2048 \times 2 \div 16 = 256$. So, the value of the first term in the REFINT equation above is $4224000 \div [(256 + 1) \times 16] = 1027.237$

For this example, suppose PRETOACT is set to 2 clock cycles. In this case,

$$\text{ROH} = (2 \times 2) + 2 + 4 + 1 = 11$$

If the system uses 8-bit ROMs on the 60x/memory bus, a burst read from ROM will follow the timing shown in Figure 6-50. In addition, the minimum time allowed for ROM devices to enter high impedance is two clock cycles. This delay is enforced after all ROM accesses preventing any other memory access from starting. Therefore a burst read from an 8-bit ROM will take:

$$\{[(\text{ROMFAL} + 2) \times 8 + 3] \times 4 + 5\} + 2 \text{ clock cycles}$$

So, if $\text{MCCR1}[\text{ROMFAL}] = 4$, the interval for a 60x burst read from an 8-bit ROM will take:

$$\text{TWACC} = \{[(4 + 2) \times 8 + 3] \times 4 + 5\} + 2 = 211 \text{ clock cycles}$$

Plugging the values into the REFINT equation above:

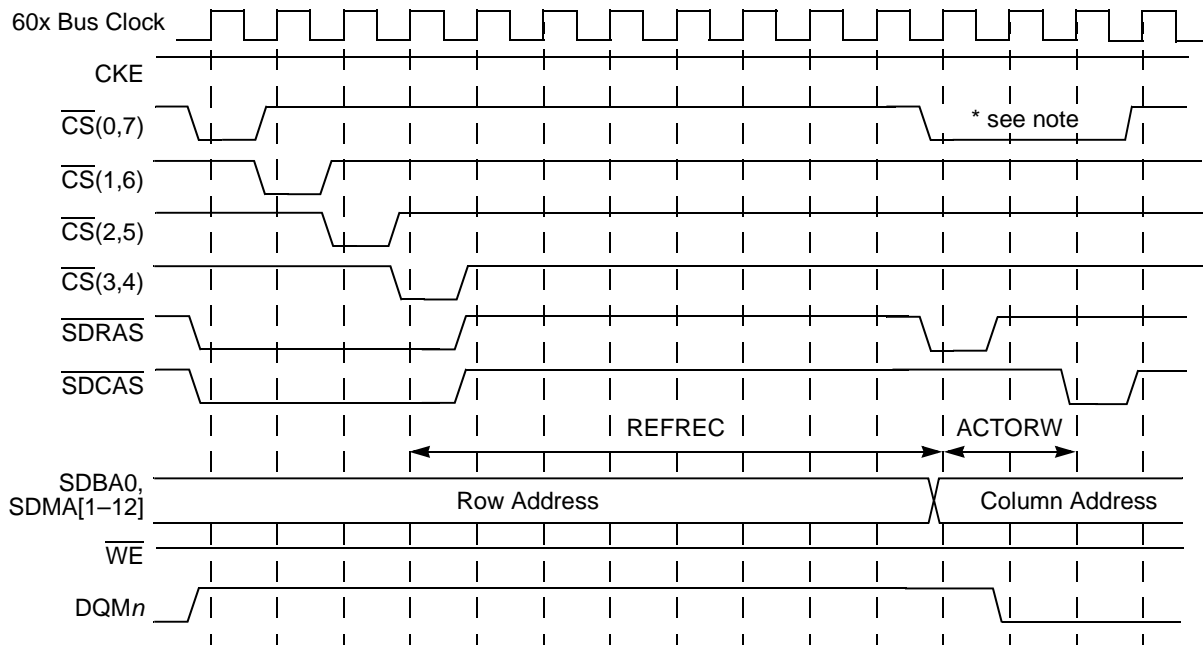
$$\text{REFINT} < 1027.237 - 11 - (211 \div 16) = 1003 \text{ clock cycles (rounded down)}$$

The value stored in REFINT would be 0b00_0011_1110_1011 (or 1003 clock cycles).

6.5.10.1 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 MPC106 implements bank staggering for CBR refreshes, as shown in Figure 6-39. This reduces instantaneous current consumption for memory refresh operations.



* Note that only one \overline{CS} signal is asserted for the bank-activate and read commands.

Figure 6-39. SDRAM Bank-Staggered CBR Refresh Timing

6.5.10.2 SDRAM Refresh and Power Saving Modes

The MPC106 memory interface provides for doze, nap, sleep, and suspend power saving modes defined in Appendix A, “Power Management.”

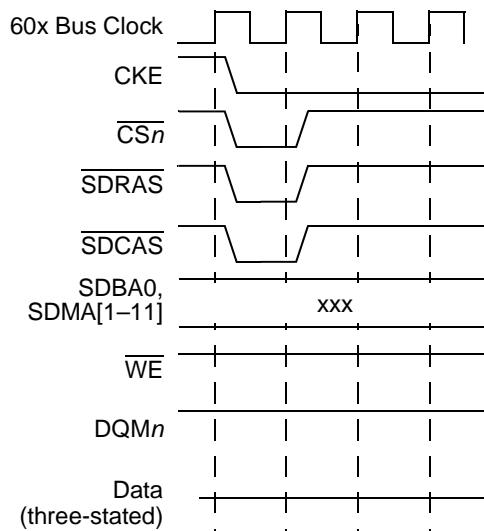
In doze and nap power saving modes and in full-on mode, the MPC106 supplies the normal CBR refresh to SDRAM. In sleep mode, the MPC106 can be configured to take advantage of self-refreshing SDRAMs or provide normal refresh to the SDRAMs. In suspend mode, the MPC106 can be configured to take advantage of self-refreshing SDRAMs. 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 6-16 summarizes the refresh types available in each of the power saving modes and the relevant configuration parameters.

Table 6-16. 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
Suspend ¹	Self	Asserted (Low)	1	—	—	0	1	1

Note: ¹Normal and RTC refresh are not available in suspend mode.

The entry timing for self-refreshing SDRAMs is shown in Figure 6-40. The exit timing for self-refreshing SDRAMs is shown in Figure 6-41.


Figure 6-40. SDRAM Self-Refresh Entry Timing

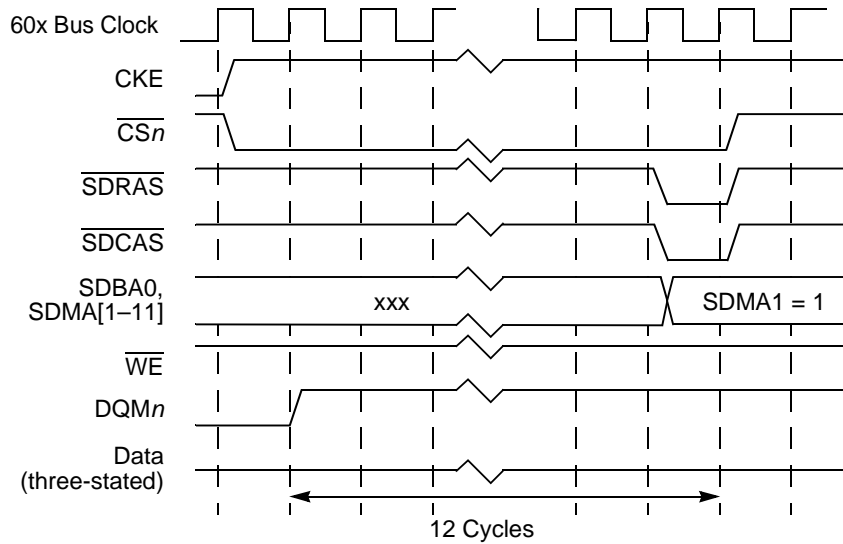


Figure 6-41. SDRAM Self-Refresh Exit Timing

6.6 ROM/Flash Interface Operation

For the ROM/Flash interface, the MPC106 provides 21 address bits, two bank selects, one Flash output enable, and one Flash write enable. Figure 6-42 shows an example of a 16-Mbyte ROM system.

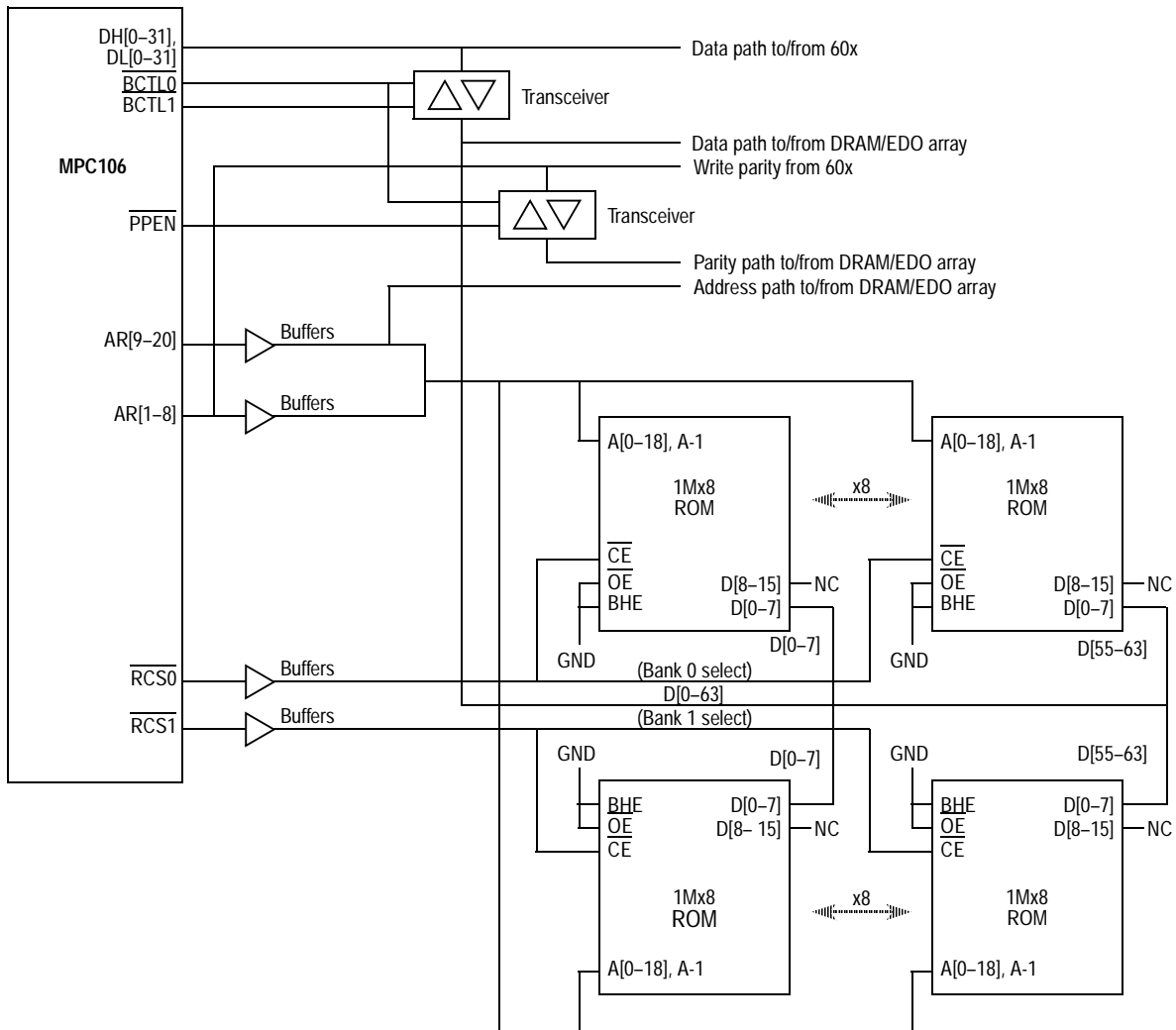
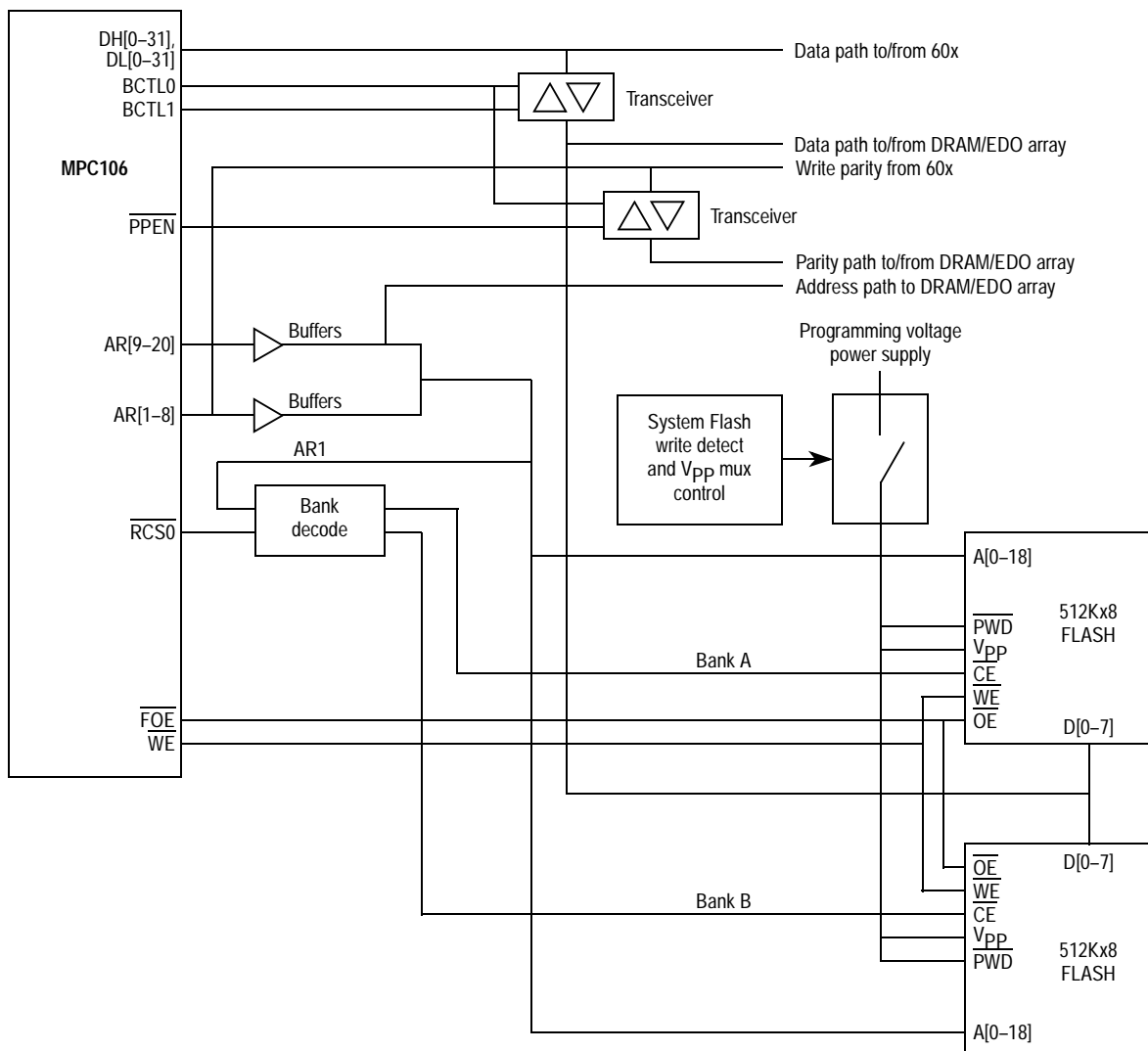


Figure 6-42. 16-Mbyte ROM System

Figure 6-43 shows an example of a 1-Mbyte Flash system. Note that because of the 8-bit datapath, external decode logic is necessary to select between the two devices.


Figure 6-43. 1-Mbyte Flash System

The MPC106 supports an 8- or 64-bit datapath to bank 0. A configuration signal ($\overline{\text{FOE}}$) sampled at reset, determines the bus width of the ROM or Flash device (8-bit or 64-bit) in bank 0. The data bus width for ROM bank 1 is always 64 bits.

The most-significant ROM/Flash address bit (AR0) is only supported for bank 0 when using the 8-bit datapath. The extra address bit allows for up to 2 Mbytes of ROM/Flash space in bank 0 for the 8-bit datapath configuration. For the 64-bit datapath, 20 address bits (AR[1–20]) allows for up to 8 Mbytes per bank.

For systems using the 8-bit interface to bank 0, the ROM/Flash device must be connected to the most-significant byte lane of the data bus (DH[0–7]). The MPC106 performs byte lane alignment for single-byte reads from ROM/Flash memory. The MPC106 can also perform byte gathering (up to eight bytes) for ROM/Flash read operations. The data bytes are gathered and aligned within the MPC106, then forwarded to the 60x processor.

The 16-Mbyte ROM/Flash space is subdivided into two 8-Mbyte banks. Bank 0 (selected by $\overline{\text{RCS0}}$) is addressed from 0xFF80_0000 to 0xFFFF_FFFF. Bank 1 (selected by $\overline{\text{RCS1}}$) is addressed from 0xFF00_0000 to 0xFF7F_FFFF. Implementations that require less than 16 Mbytes may allocate the required ROM/Flash to one or both banks.

For example, an implementation that requires only 4 Mbytes of ROM/Flash could locate the ROM/Flash entirely within bank 0 at addresses 0xFFC0_0000–0xFFFF_FFFF. Alternately, the ROM/Flash could be split across both banks with 2 Mbytes in bank 0 at 0xFFE0_0000–0xFFFF_FFFF, and 2 Mbytes in bank 1 at 0xFF60_0000–0xFF7F_FFFF. Any system ROM space that is not physically implemented within a bank will be aliased to the physical device(s) within that bank.

The MPC106 can be configured to support ROM/Flash devices located on the 60x/memory bus or on the PCI bus. The $\overline{\text{RCS0}}$ signal is sampled at reset to determine the location of ROM/Flash. See Section 2.2.9, “Configuration Signals,” for more information. If the system ROM space is mapped to the PCI bus, the MPC106 directs all system ROM accesses to the PCI bus.

The MPC106 also supports splitting the system ROM space between PCI and the 60x/memory bus. The entire ROM space is mapped to the PCI space, and then, by setting the configuration parameter PICR2[CF_FF0_LOCAL], the lower half of the ROM space (FF00_0000–FF7F_FFFF) is remapped onto the 60x/memory bus. This allows the system to have the upper half of ROM space on the PCI bus for boot firmware and the lower half of the ROM space on the 60x/memory bus for performance critical firmware. The ROM/Flash on the 60x/memory bus is selected by $\overline{\text{RCS1}}$ and the datapath must be 64 bits wide.

6.6.1 ROM/Flash Cacheability

Data in ROM/Flash memory is cacheable with certain restrictions—the L2 cache must use synchronous burst SRAMs, and the data in the cache must not be modified. Writes to Flash must be single-beat datapath sized writes; burst writes to Flash cause a Flash write error (see Section 6.6.6, “ROM/Flash Interface Write Operations,” for more information). Therefore, if the system performs writes to Flash locations that may be contained in the L2 cache, the L2 cache must operate in write-through mode.

The MPC106 does not generate parity for transactions in the system ROM space, so incorrect parity is stored with ROM/Flash data in the L2 cache. However, this does not cause a parity error because the MPC106 does not check parity/ECC for any transaction in the system ROM address space.

6.6.2 64-Bit ROM/Flash Interface Timing

The ROM/Flash interface of the MPC106 supports burst and nonburst devices. The MPC106 provides programmable access timing for the ROM/Flash interface. The

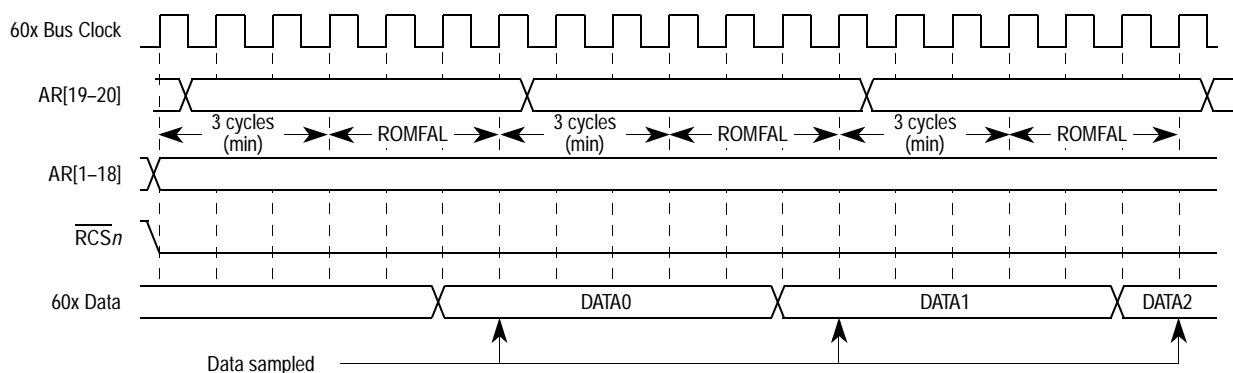
programmable timing parameters for the ROM/Flash interface are MCCR1[ROMNAL], MCCR1[ROMFAL], and MCCR1[BURST]. See Section 3.2.8.4, “Memory Control Configuration Registers,” for more information.

When configured for a 64-bit datapath, MCCR1[ROMFAL] controls the latency for nonburst devices. For burst devices, ROMFAL controls the latency for the first data beat only. The actual latency cycle count for the 64-bit interface is three clock 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 devices for the burst data beats following the first data beat. The actual latency cycle count is three cycles more than the value specified in ROMNAL. For example, when ROMNAL = 0b0000, the latency is three clock cycles; when ROMNAL = 0b0001, the latency is four clock cycles; when ROMNAL = 0b0010, the latency is five clock cycles; and so on. MCCR1[BURST] controls whether the MPC106 uses burst (BURST = 1) or nonburst (BURST = 0) access timing.

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 device timing.

Figure 6-44 shows the 64-bit ROM/Flash interface timing for the nonburst ROM device configuration.



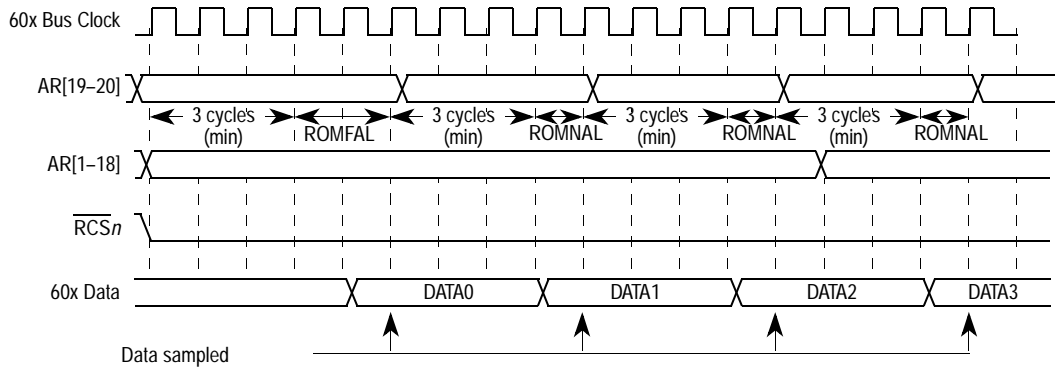
Notes:

ROMFAL (ROM first access latency) = 0–31 clock cycles

MCCR1[BURST] = 0 (default value at reset)

Figure 6-44. 64-Bit ROM/Flash Interface—Nonburst ROM Timing

Figure 6-45 shows the 64-bit ROM/Flash interface timing for the burst ROM device configuration.



Notes:

- ROMFAL (ROM first access latency) = 0–31 clock cycles
- ROMNAL (ROM next access latency) = 0–15 clock cycles
- MCCR1[BURST] = 1

Figure 6-45. 64-Bit ROM/Flash Interface—Burst ROM Timing

6.6.3 Processor Read from 64-Bit ROM Transaction Examples

The figures in this section provide signal timing examples of 60x processor reads from 64-bit ROM. Figure 6-46 shows a processor cache-line (32-bytes) burst read from a nonbursting 64-bit ROM. Figure 6-47 shows a processor single-beat (8-bytes) read from a nonbursting 64-bit ROM.

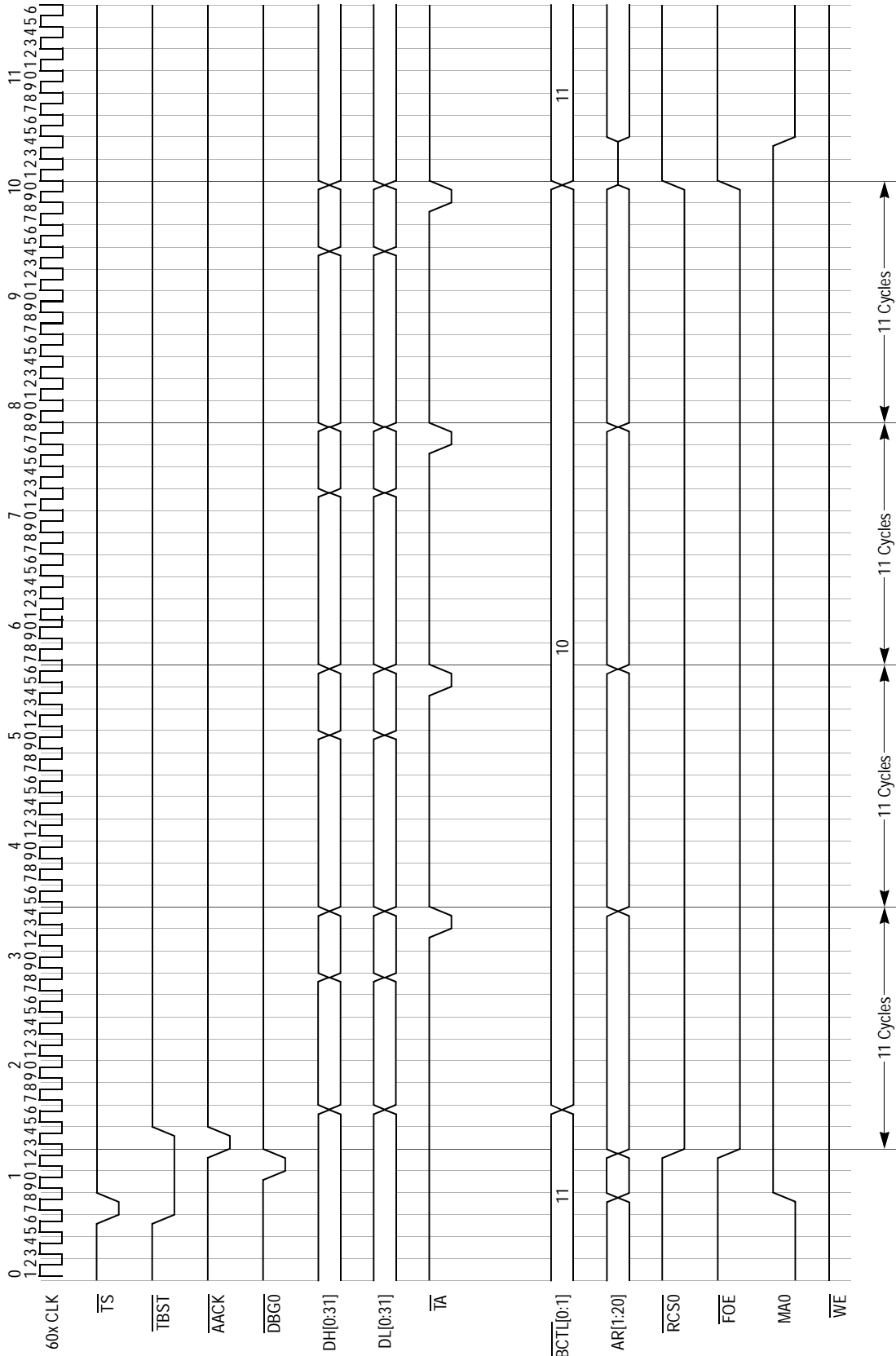


Figure 6-46. 64-Bit ROM, 32-Byte Read, Flow-Through Buffers

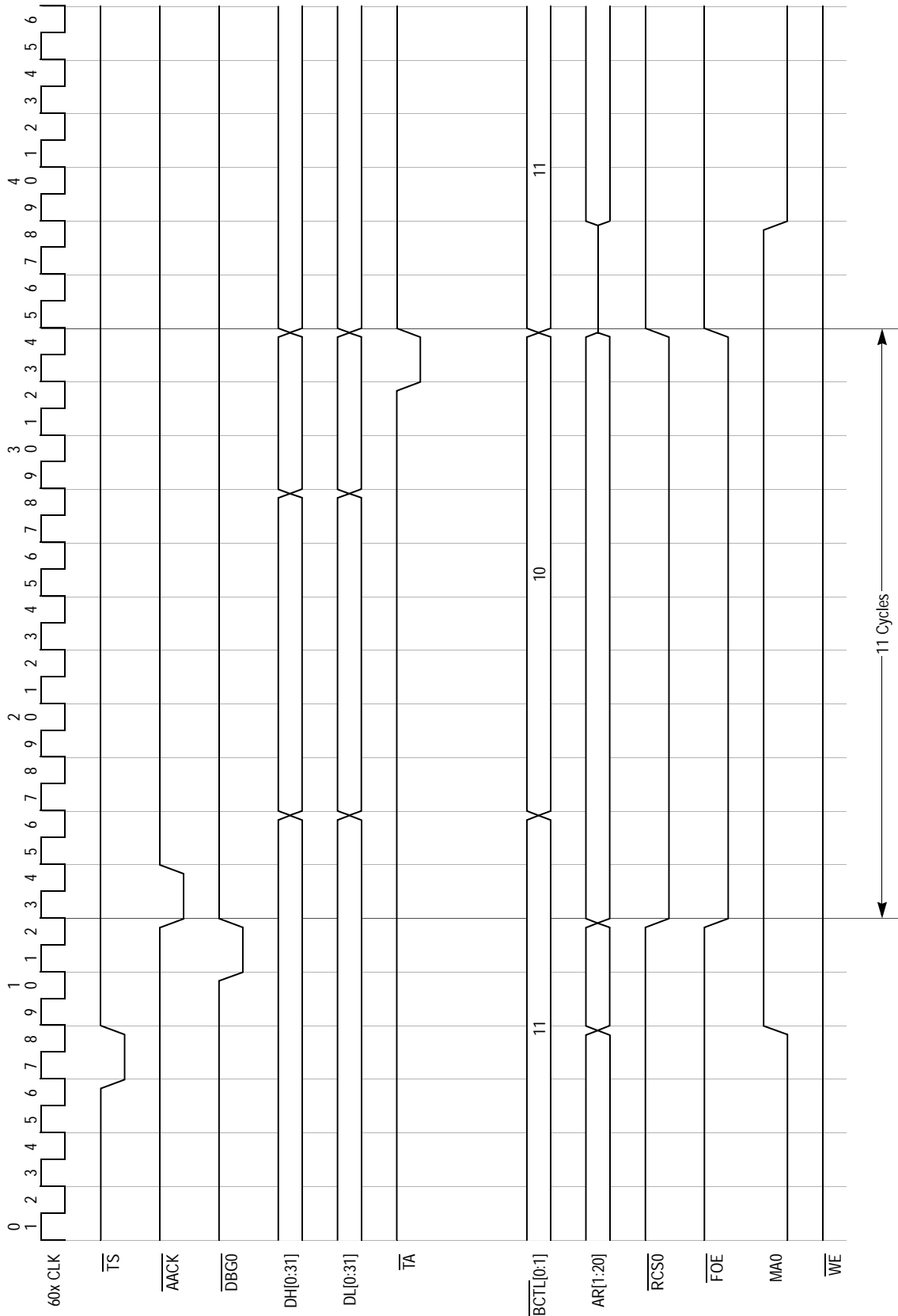


Figure 6-47. 64-Bit ROM, 8-Byte Read, Flow-Through Buffers

6.6.4 Eight-Bit ROM/Flash Interface Timing

The MPC106 provides programmable latency for accessing ROM/Flash memory so that systems of various clock frequencies may properly interface to the ROM/Flash devices. The programmable timing parameter MCCR1[ROMFAL] controls the access latency to ROM/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. ROMFAL is set to its maximum value at reset in order to accommodate initial boot code fetches from a slow device. To improve access latency, initialization software should program a more appropriate ROMFAL value for the actual device being used. The 8-bit interface only provides nonburst device access timing, so the parameter MCCR1[BURST] is ignored for the 8-bit interface.

The following figures illustrate the 8-bit ROM/Flash interface timing for various read accesses. Figure 6-48 shows a single-byte read access. Figure 6-49 shows a half-word read access. Word and double-word accesses require using the burst access timing shown in Figure 6-50.

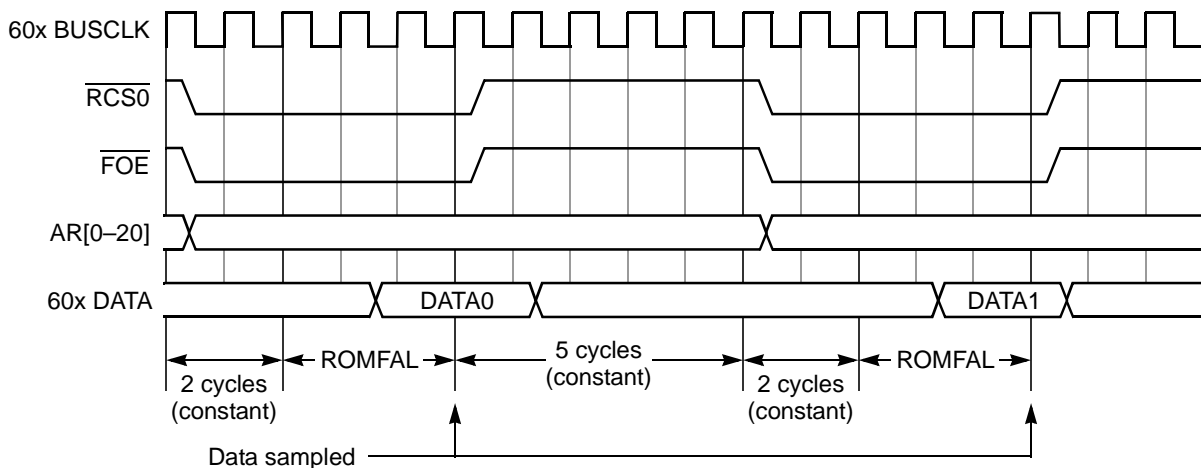
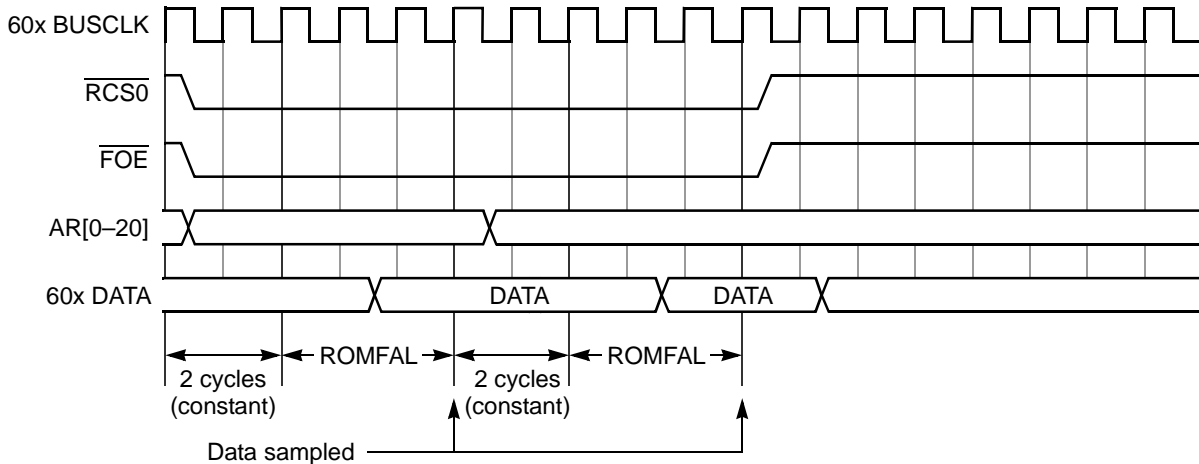
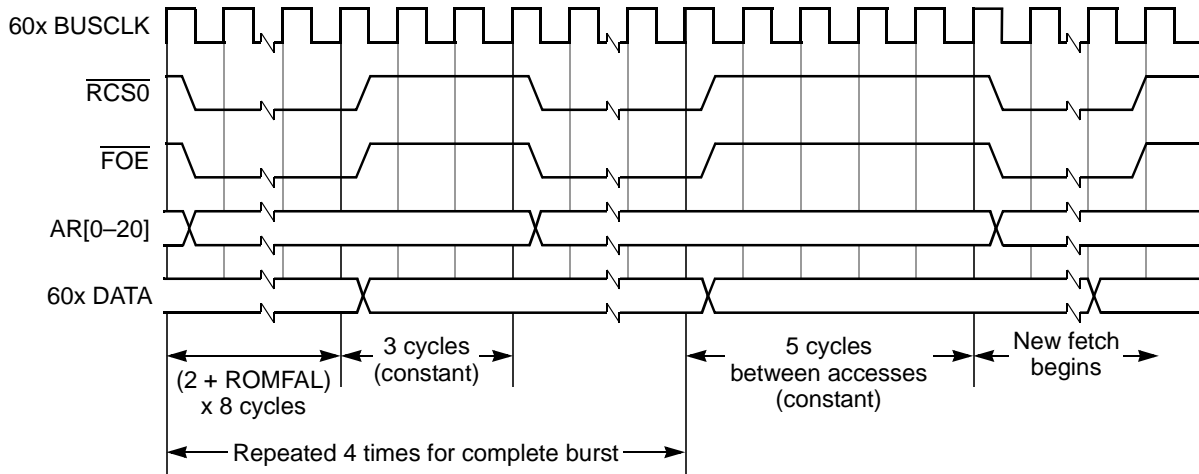


Figure 6-48. 8-Bit ROM/Flash Interface—Single-Byte Read Timing


Figure 6-49. 8-Bit ROM/Flash Interface—Half-Word Read Timing

Figure 6-50. 8-Bit ROM/Flash Interface—Burst Read Timing

6.6.5 Processor Read from 8-Bit ROM Transaction Examples

The figures in this section provide signal timing examples of 60x processor reads from 8-bit ROM. Figure 6-51 shows a processor single-byte read from an 8-bit ROM. Figure 6-52 shows a processor double-word (8-bytes) read from an 8-bit ROM.

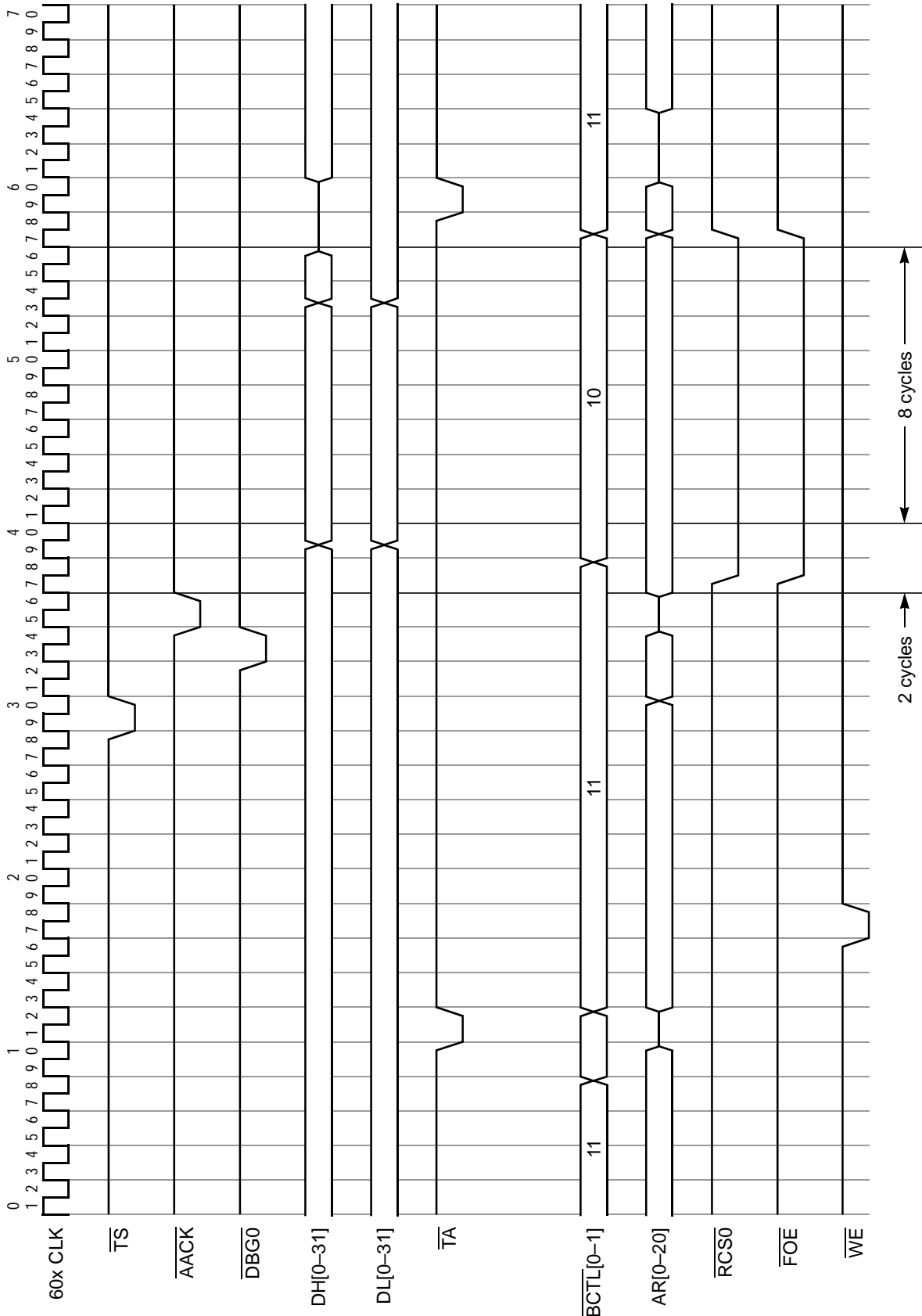


Figure 6-51. 8-Bit ROM, 1-Byte Read, Flow-Through Buffers

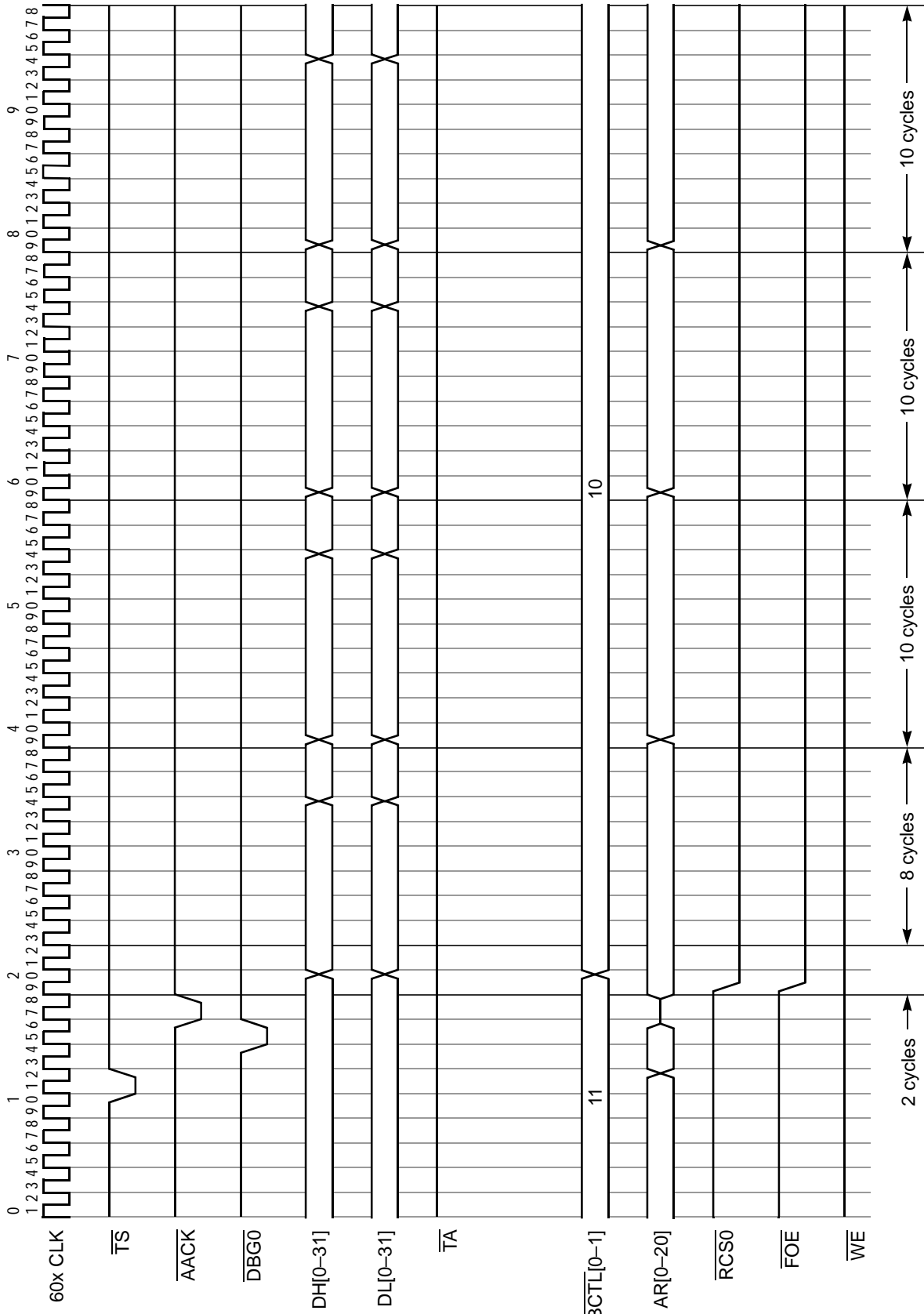


Figure 6-52. 8-Bit ROM, 8-Byte Read, Flow-Through Buffers

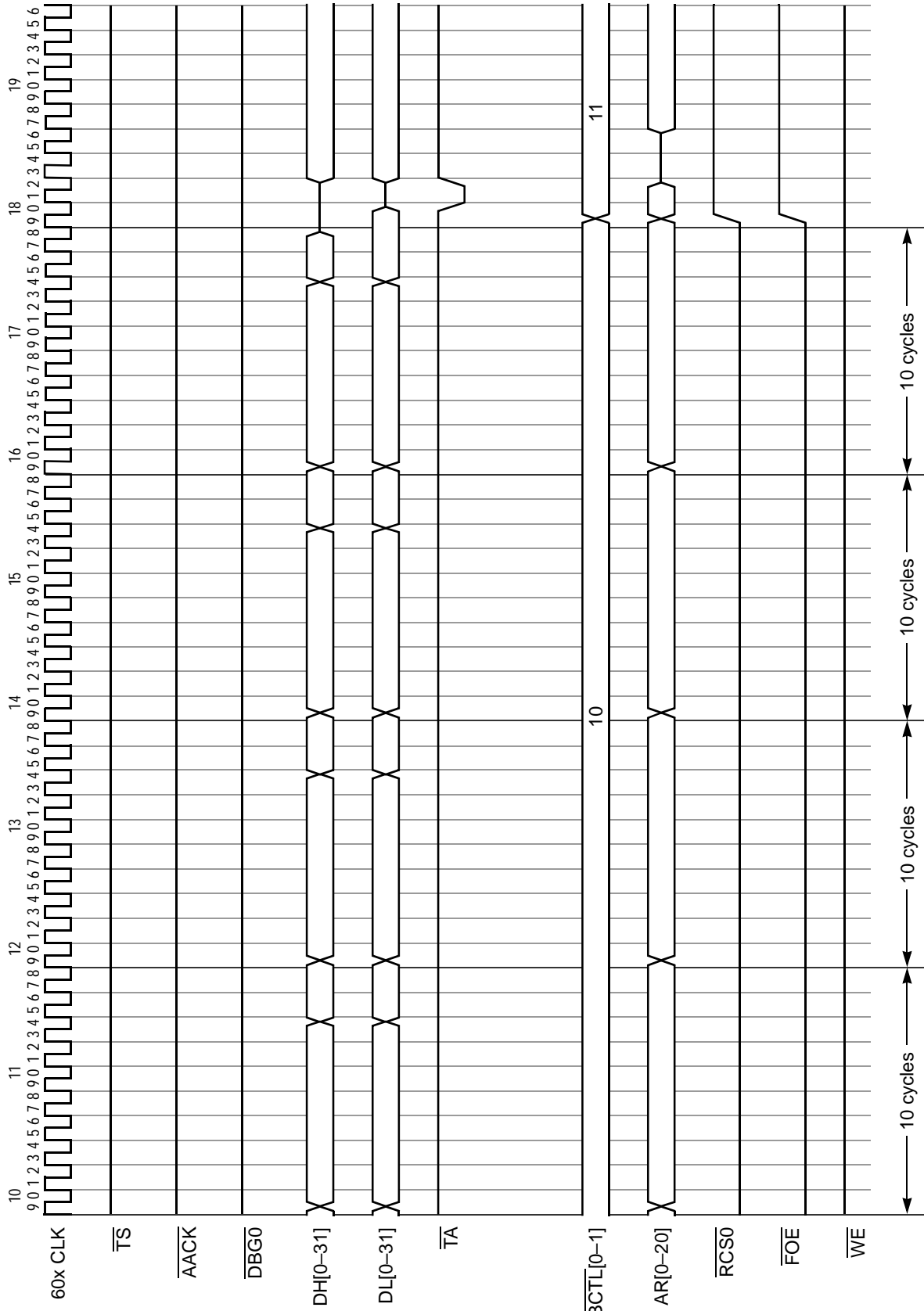


Figure 6-52. (continued) 8-Bit ROM, 8-Byte Read, Flow-Through Buffers

6.6.6 ROM/Flash Interface Write Operations

The MPC106 accommodates only single-beat, datapath sized (8- or 64-bit), writes from the 60x processor to Flash memory. This requires that all writes to system ROM space must be either caching-inhibited (\overline{CI} asserted), or caching-allowed/write-through (\overline{CI} negated and \overline{WT} asserted). Software must partition larger data into individual datapath sized (8- or 64-bit) write operations. If an attempt is made to write to Flash with a data size other than the full datapath size, the MPC106 will assert \overline{TEA} (transfer error acknowledge) and \overline{MCP} , provided \overline{TEA} is enabled in PICR1.

Note that for address map A, the system ROM space is not accessible by PCI agents when the system ROM space is located on the 60x/memory bus. Alternately, for address map B, the MPC106 responds to PCI writes to Flash with a target-abort when the system ROM space is located on the 60x/memory bus. See Section 7.4.3.2, “Target-Initiated Termination,” for more information.

The MPC106 latches processor writes to system ROM space before initiating the actual write operation to Flash. This is necessary to avoid contention on the multiplexed parity/ROM address signals. The 60x initiates the write to system ROM space and drives parity during the data tenure, the MPC106 latches the data, and then MPC106 drives the ROM address and data for the write to Flash. System logic (external to the MPC106) is responsible for multiplexing high voltage to the Flash memory as required for write operations.

MCCR1[MEMGO] must be set before any writes to Flash are attempted. Also, 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. FLASH_WR_EN is cleared at reset to disable write operations to Flash memory.

Writing to Flash can be locked out by setting PICR2[FLASH_WR_LOCKOUT]. When this bit is set, the MPC106 disables writing to Flash memory, even if FLASH_WR_EN is set. Once set, the FLASH_WR_LOCKOUT parameter can only be cleared by a hard reset.

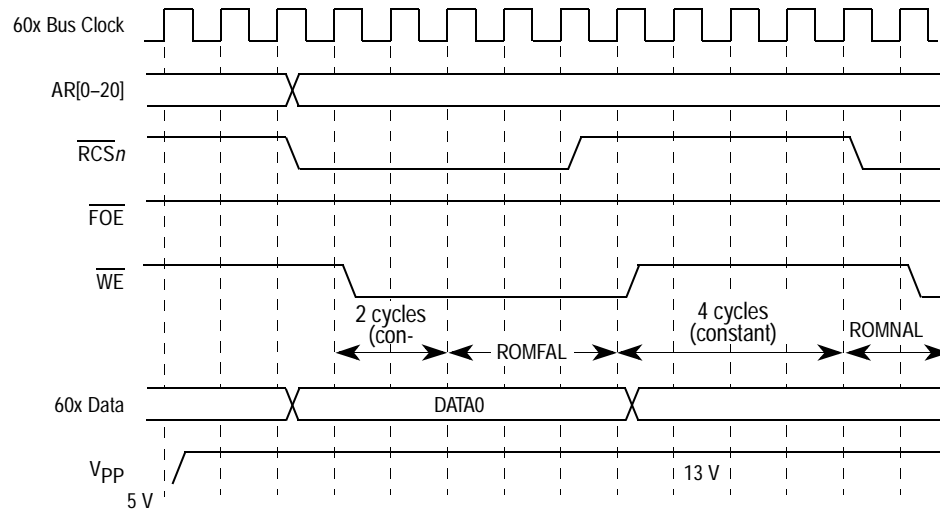
If the system attempts to write to read-only devices in a bank, then bus contention may occur. This is because the write data is driven onto the data bus when the read-only device is also trying to drive its data onto the data bus. This situation can be avoided by disabling writes to the system ROM space using FLASH_WR_EN or FLASH_WR_LOCKOUT or by connecting the Flash output enable (\overline{FOE}) signal to the output enable on the read-only device.

6.6.6.1 ROM/Flash Interface Write Timing

The parameter 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 four cycles more than the value specified in ROMNAL. For example, when ROMNAL = 0b0000, the write recovery time is four clock cycles; when ROMNAL = 0b0001, the write

recovery time is five clock cycles; when ROMNAL = 0b0010, the write recovery time is six clock cycles; and so on. ROMNAL is set to the maximum value at reset. To improve performance, it is recommended that initialization software program a more appropriate value for the actual device being used.

Figure 6-53 illustrates the write access timing of the Flash interface.



Note: V_{pp} multiplexed by system logic with appropriate setup time to write cycle.

Figure 6-53. Flash Memory Write Timing

6.6.6.2 Processor Write to 8-Bit ROM Transaction Example

Figure 6-54 shows a processor single-byte write to an 8-bit Flash device followed by a single-byte read.

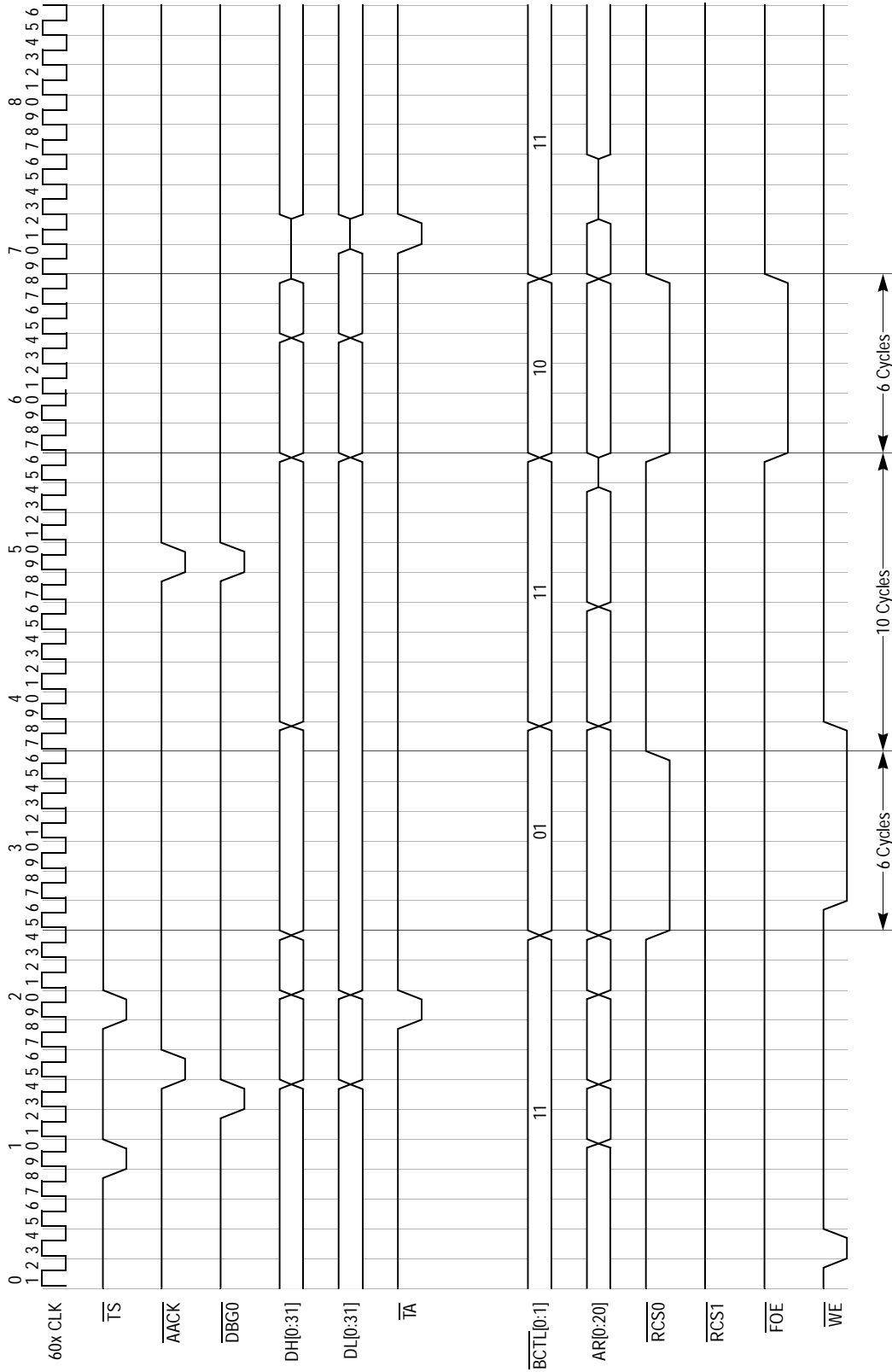


Figure 6-54. 8-Bit Flash, 1-Byte Write Followed by 1-Byte Read, Flow-Through Buffers



Chapter 7

PCI Bus Interface

One of the primary functions of the MPC106 is to serve as a bridge between the 60x processor bus (the host bus) and the PCI bus. The MPC106's PCI interface complies with the *PCI Local Bus Specification*, Revision 2.1. It is 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 MPC106 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 2.1 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. Since this is inconsistent with the terminology in this manual, the terms 'word' and 'double word' are not used in this chapter. Instead, the number of bits or bytes indicates 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 PCI interface supports address and data parity with error checking and reporting.

The PCI interface of the MPC106 functions as both a master (initiator) and target device. Internally, the PCI interface of the MPC106 is controlled by two state machines (one for master and one for target) running independently of each other. This allows the MPC106 to run two separate transactions simultaneously. For example, if the MPC106, as a 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 MPC106, as a target, can accept the burst-read transfer. When the MPC106 is granted mastership of the PCI bus, the burst-write transaction continues.

As a master, the MPC106 supports read and write operations to the PCI memory space, the PCI I/O space, and the PCI 256-byte configuration space. As a master, the MPC106 also

supports generating PCI special-cycle and interrupt-acknowledge transactions. As a target, the MPC106 supports read and write operations to system memory.

Internal 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 Buffers,” for more information.

The interface can be programmed for either little-endian or big-endian formatted data, and provides 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 MPC106.

7.1.1 MPC106 as a PCI Master

Upon detecting a 60x-to-PCI transaction, the MPC106 requests the use of the PCI bus. For 60x-to-PCI bus write operations, the MPC106 requests mastership of the PCI bus when the 60x completes the write operation on the 60x processor bus. For 60x-to-PCI read operations, the MPC106 requests mastership of the PCI bus when it decodes that the access is for PCI address space.

Once granted, the MPC106 drives the 32-bit PCI address ($AD[31-0]$) and the bus command ($\overline{C/BE}[3-0]$) 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 MPC106 as a PCI Target

As a target, upon detection of a PCI address phase the MPC106 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, decodes the PCI bus 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 meaningful data.

The target interface of the MPC106 can issue target-abort, target-retry, and target-disconnect cycles. 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 timing of system memory

devices. The target interface disconnects when a transaction reaches the end of a cache line (32 bytes).

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 ($\overline{\text{REQ}}$) output and grant ($\overline{\text{GNT}}$) input 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 MPC106 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, interrupt controller, and PCI-to-ISA bridge functions into a single device.

7.3 PCI Bus Protocol

This section provides a general description of the PCI bus protocol. Specific PCI bus transactions are described in Section 7.4, “PCI Bus Transactions.” Refer to Figure 7-1, Figure 7-2, Figure 7-3, and Figure 7-4 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— $\overline{\text{FRAME}}$ (frame), $\overline{\text{IRDY}}$ (initiator ready), and $\overline{\text{TRDY}}$ (target ready). 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 $\overline{\text{IRDY}}$ to force wait cycles. A target negates $\overline{\text{TRDY}}$ 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 $\overline{\text{C/BE}}[3-0]$ 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 MPC106.

Table 7-1. PCI Bus Commands

$\overline{\text{C/BE}}[3-0]$	PCI Bus Command	MPC106 Supports as a Master	MPC106 Supports as a Target	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 Transactions," 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 Transactions," 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	—

Table 7-1. PCI Bus Commands (continued)

$\overline{C/BE[3-0]}$	PCI Bus Command	MPC106 Supports as a Master	MPC106 Supports as a Target	Definition
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 MPC106 (the target) fetches data 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 information on PCI configuration cycles.
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 information on PCI configuration accesses.
1100	Memory-read-multiple	No	Yes	The memory-read-multiple command functions similar to the memory-read command, but it also causes a prefetch of the next cache line (32 bytes). Note that for PCI reads from system memory, prefetching for all reads may be forced by setting bit 2 (PCI speculative read enable) of PICR1. See Section 8.1.3.1.1, "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 MPC106 does not respond to this command.

Table 7-1. PCI Bus Commands (continued)

$\overline{C/BE}[3-0]$	PCI Bus Command	MPC106 Supports as a Master	MPC106 Supports as a Target	Definition
1110	Memory-read-line	Yes	Yes	The memory-read-line command indicates that a master is requesting the transfer of an entire cache line (32 bytes).
1111	Memory-write-and-invalidate	No	Yes	The memory-write-and-invalidate command indicates that a master is transferring an entire cache line (32 bytes), and, if this data is in any cacheable memory, that cache line needs to be invalidated.

¹ Reserved command encodings are reserved for future use. The MPC106 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 MPC106 address map (A, B, or emulation mode map) being used. The address 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 ($AD[1-0]$) 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.

7.3.3.1 Memory Space Addressing

For memory accesses, PCI defines two types of burst ordering controlled by the two low-order bits of the address—linear incrementing ($AD[1-0] = 0b00$) and cache wrap mode ($AD[1-0] = 0b10$). The other two $AD[1-0]$ possibilities ($0b01$ and $0b11$) are reserved. As a target, the MPC106 executes a target disconnect after the first data phase completes if $AD[1-0] = 0b01$ or $AD[1-0] = 0b11$ during the address phase of a system memory access. As a master, the MPC106 always encodes $AD[1-0] = 0b00$ for PCI memory space accesses.

For linear incrementing mode, the memory address is encoded/decoded using $AD[31-2]$. 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 on the address bus are included in all parity calculations.

For cache wrap mode ($AD[1-0] = 0b10$) reads, the critical memory address is decoded using $AD[31-2]$. The address is incremented by 4 bytes after each data phase completes until the end of the cache line is reached. Thereafter, the address wraps to the beginning of the current cache line and continues incrementing until the entire cache line (32 bytes) is read. The MPC106 does not support cache wrap mode write operations and executes a target disconnect after the first data phase completes for writes with $AD[1-0] = 0b10$. Again, note that the two low-order bits on the address bus are included in all parity calculations.

7.3.3.2 I/O Space Addressing

For PCI I/O accesses, all 32 address signals ($AD[31-0]$) are used to provide a byte address. After 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 error. See Section 7.4.3.2, “Target-Initiated Termination,” for more information.

7.3.3.3 Configuration Space Addressing

PCI supports two types of configuration accesses which use different formats for the $AD[31-0]$ signals during the address phase. The two low-order bits of the address indicate the format used for the configuration address phase—type 0 ($AD[1-0] = 0b00$) or type 1 ($AD[1-0] = 0b01$). 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{DEVSEL} signal is driven by the target of the current transaction. \overline{DEVSEL} indicates to the other devices on the PCI bus that the target has decoded the address and claimed the transaction. \overline{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{DEVSEL} within three clocks of \overline{FRAME} , the agent responsible for subtractive decoding may claim the transaction by asserting \overline{DEVSEL} .

A target must assert \overline{DEVSEL} (claim the transaction) before or coincident with any other target response (assert its \overline{TRDY} , \overline{STOP} , or data signals). In all cases except target-abort, once a target asserts \overline{DEVSEL} , it must not negate \overline{DEVSEL} until \overline{FRAME} is negated (with

$\overline{\text{IRDY}}$ asserted) and the last data phase has completed. For normal termination, negation of $\overline{\text{DEVSEL}}$ coincides with the negation of $\overline{\text{TRDY}}$ or $\overline{\text{STOP}}$.

If the first access maps into a target's address range, that target asserts $\overline{\text{DEVSEL}}$ to claim the access. However, if the master attempts to continue the burst access across the resource boundary, then the target must issue a target disconnect.

The MPC106 is hardwired for fast device select timing (PCI status register[10–9] = 0b00). Therefore, when the MPC106 is the target of a transaction (system memory access), it asserts $\overline{\text{DEVSEL}}$ one clock cycle following the address phase.

As a master, if the MPC106 does not detect the assertion of $\overline{\text{DEVSEL}}$ within four clocks after the address phase (five clocks after it asserts $\overline{\text{FRAME}}$), it terminates the transaction with a master-abort termination; see Section 7.4.3.1, “Master-Initiated Termination,” for more information.

7.3.5 Byte Alignment

The byte enable signals of the PCI bus ($\overline{\text{C/BE}}[3-0]$, during a data phase) 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 stay valid for the entire data phase. Note that parity is calculated for all bytes regardless of the state of the byte enable signals. See Section 7.6.1, “PCI Parity,” for more information.

If the MPC106, as a target, detects that no byte enables are asserted and completes the current data phase with no permanent change. This implies that on a read transaction, the MPC106 expects that the data is not changed, and on a write transaction, the data is not stored.

7.3.6 Bus Driving and Turnaround


To avoid contention, a turnaround cycle is required on all signals that may be driven by more than one agent. The turnaround cycle occurs at different times for different signals. The $\overline{\text{IRDY}}$, $\overline{\text{TRDY}}$, $\overline{\text{DEVSEL}}$, and $\overline{\text{STOP}}$ signals use the address phase as their turnaround cycle. $\overline{\text{FRAME}}$, $\overline{\text{C/BE}}[3-0]$, and $\text{AD}[31-0]$ signals use the idle cycle between transactions (when both $\overline{\text{FRAME}}$ and $\overline{\text{IRDY}}$ are negated) as their turnaround cycle. The $\overline{\text{PERR}}$ signal has a turnaround cycle on the fourth clock after the last data phase.

The PCI address/data signals, $\text{AD}[31-0]$, 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 7.6.1, “PCI 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 described as generic read transactions and generic write transactions.

The timing diagrams in this section 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. High-impedance 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 PCI Read Transactions

Figure 7-1 shows a PCI single-beat read transaction. Figure 7-2 shows a PCI burst read transaction. A PCI read transaction starts with the address phase, occurring when a master asserts $\overline{\text{FRAME}}$. During the address phase, AD[31–0] contain a valid address and $\overline{\text{C/BE}}[3–0]$ 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 AD[31–0] as address signals to the target driving AD[31–0] as data signals. The turnaround cycle is enforced by the target with the $\overline{\text{TRDY}}$ signal. The target provides valid data no sooner than one cycle after the turnaround cycle. The target must drive the AD[31–0] signals when $\overline{\text{DEVSEL}}$ is asserted.

During the data phase, the $\overline{\text{C/BE}}[3–0]$ signals indicate which byte lanes are involved in the current data phase. A data phase may consist of a data transfer and wait cycles. The $\overline{\text{C/BE}}[3–0]$ 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.

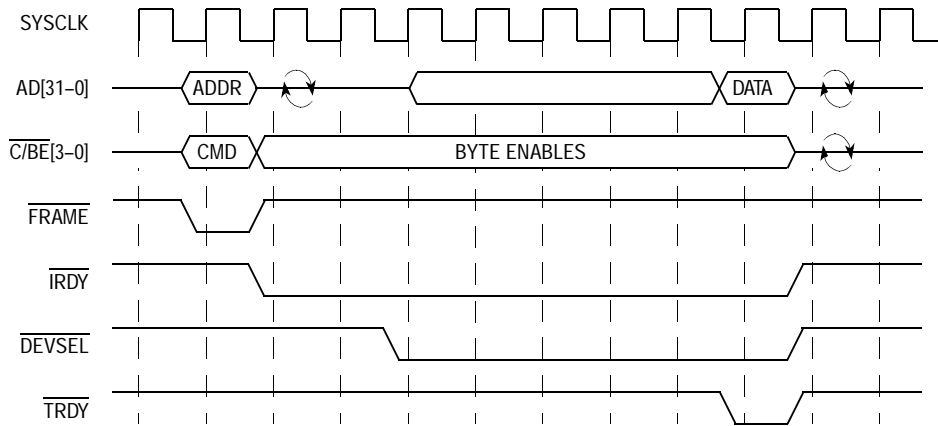


Figure 7-1. PCI Single-Beat Read Transaction

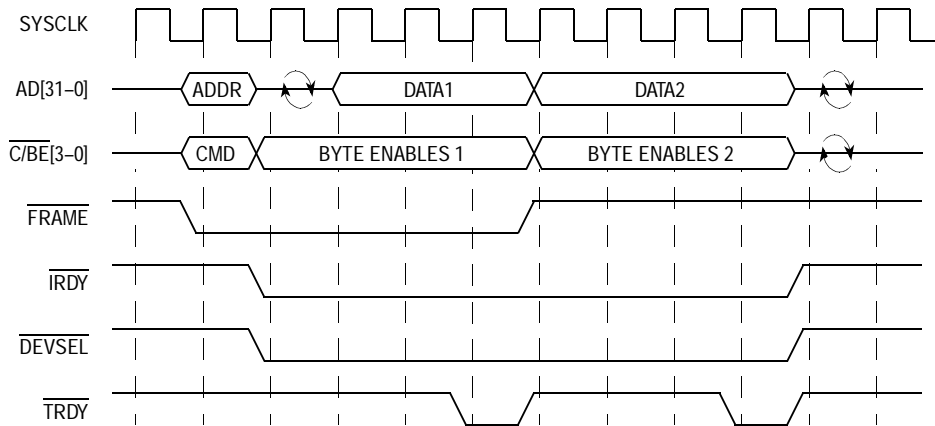


Figure 7-2. PCI Burst Read Transaction

7.4.2 PCI Write Transactions

Figure 7-3 shows a PCI single-beat write transaction. Figure 7-4 shows a PCI burst write transaction. A PCI write transaction starts with the address phase, occurring when a master asserts $\overline{\text{FRAME}}$. 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. Although not shown in the figures, the master must drive the $\overline{\text{C/BE}}[3-0]$ signals, even if the master is not ready to provide valid data ($\overline{\text{IRDY}}$ negated).

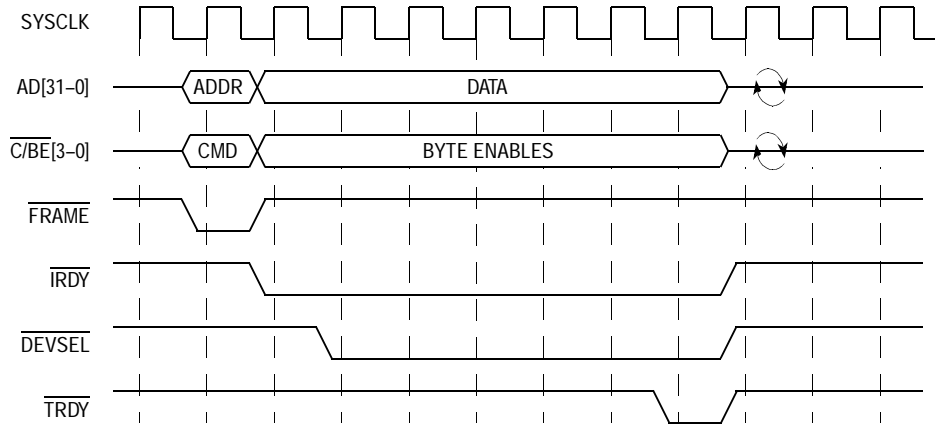


Figure 7-3. PCI Single-Beat Write Transaction

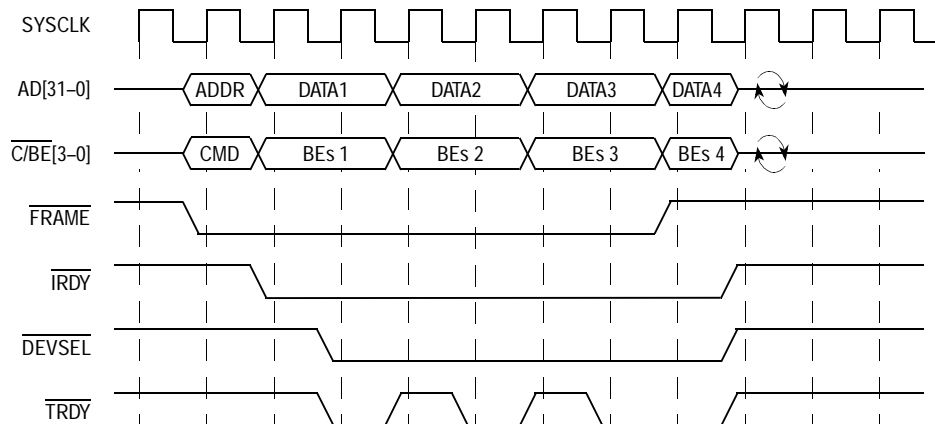


Figure 7-4. PCI Burst Write Transaction

7.4.3 Transaction Termination

A PCI transaction may be terminated by either the master or the target. The master is ultimately responsible for concluding all transactions, 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).

There are three types of master-initiated termination:

- **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 ($\overline{\text{GNT}}$ is negated) and its internal latency timer has expired. The intended transaction is not necessarily concluded. Note that the MPC106 does not have a latency timer. Latency for the MPC106 acting as a master is determined by the target.
- **Master-abort** 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, 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 command), the received master-abort bit (bit 13) of the PCI status register is set.

As a master, if the MPC106 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. On reads that are aborted, the MPC106 returns all 1s (0xFFFF_FFFF). On writes that are aborted, the data is lost.

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.

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 corresponding to 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, it can assert $\overline{\text{REQ}}$ whenever it needs to use the PCI bus again.

There are three types of target-initiated termination:

- **Disconnect** Disconnect refers to termination requested because the target is temporarily unable to continue bursting. Disconnect implies that some data has been transferred. The initiator 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 the *PCI Local Bus Specification, Revision 2.1* requires that all retried transactions must be completed.
- **Target-abort** 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.

As a target, the MPC106 terminates a transaction with a target disconnect due to the following:

- It is unable to respond within eight PCI clocks (not including the first data phase).
- A cache line (32 bytes) of data was transferred. (See the discussion of cache wrap mode in Section 7.3.3.1, "Memory Space Addressing," for more information.)
- The last four bytes of a cache line were transferred in linear-incrementing address mode. (See the description of linear-incrementing mode in Section 7.3.3.1, "Memory Space Addressing," for more information.)
- If $\text{AD}[1-0] = 0b01$ or $\text{AD}[1-0] = 0b11$ during the address phase of a system memory access. (See Section 7.3.3.1, "Memory Space Addressing," for more information.)

As a target, the MPC106 responds to a transaction with a retry due to the following:

- A 60x bus copyback operation is in progress.
- A PCI write to system memory was attempted when the internal PCI-to-system-memory-write buffers (PCMWBs) are full.
- A nonexclusive access was attempted to system memory while the MPC106 is locked.
- A configuration write to a PCI device is underway and $\text{PICR2}[\text{NO_SERIAL_CFG}] = 0$.
- An access to one of the MPC106 internal configuration registers is in progress.
- The 32-clock latency timer has expired and the first data phase was not begun.

As a target, the MPC106 responds with a target-abort if a PCI master attempts to write to the ROM/Flash ROM space in address map B. For PCI writes to system memory, if an address parity error or data parity error occurs, the MPC106 aborts the transaction internally, but continues the transaction on the PCI bus.

If the MPC106 initiates a memory-read-line transaction that is retried by a PCI target, the transaction will be rerun as a memory-read command.

Figure 7-5 shows several target-initiated terminations. The three disconnect terminations are unique in the data transferred at the end of the transaction. For disconnect A, the master is negating $\overline{\text{IRDY}}$ when the target asserts $\overline{\text{STOP}}$ and data is transferred only at the end of the current data phase. For disconnect B, the target negates $\overline{\text{TRDY}}$ one clock after it asserts $\overline{\text{STOP}}$, indicating that the target can accept the current data, but no more data can be transferred. For the disconnect without data, the target asserts $\overline{\text{STOP}}$ when $\overline{\text{TRDY}}$ is negated indicating that the target cannot accept any more data.

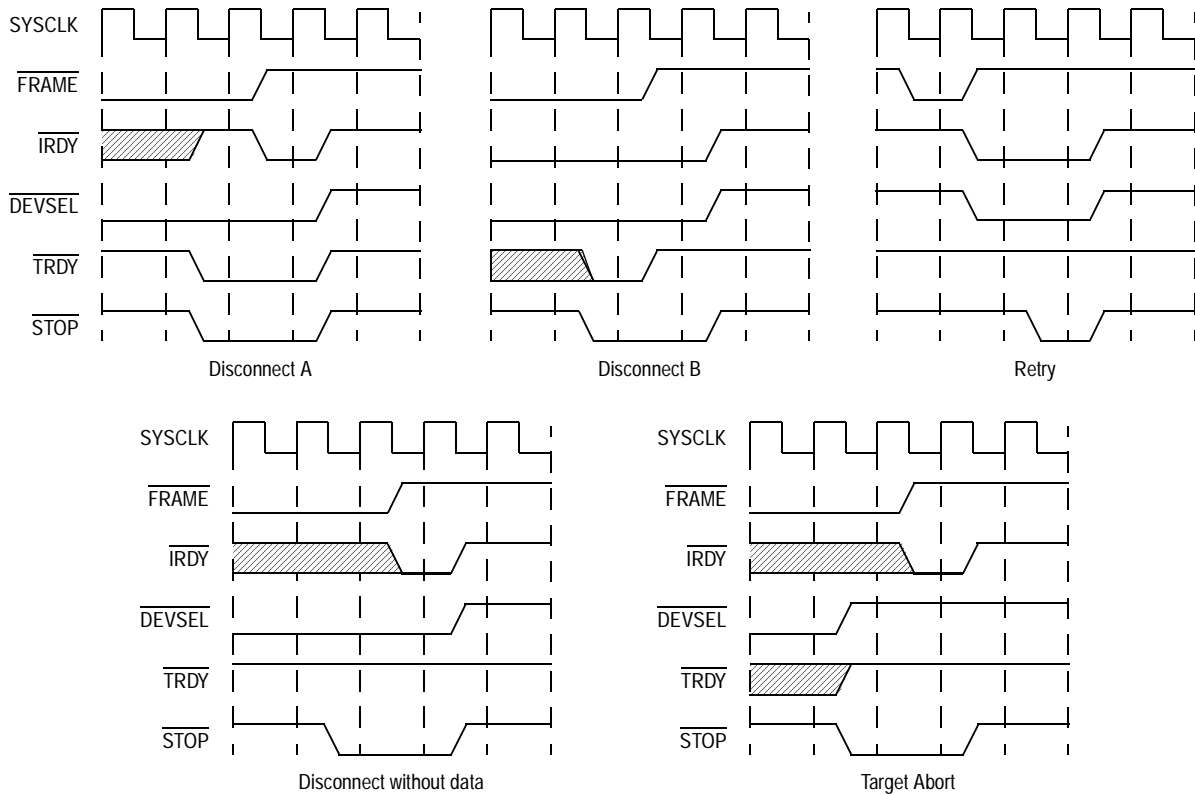


Figure 7-5. PCI Target-Initiated Terminations

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 in the clock cycle 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 MPC106 does not perform any fast back-to-back transactions. As a target, the MPC106 supports both types of fast back-to-back transactions.

During fast back-to-back transactions, the MPC106 monitors the bus states to determine if it is the target of a transaction. If the previous transaction was not directed to the MPC106 and the current transaction is directed at the MPC106, the MPC106 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 PCI configuration cycles used for configuring standard PCI devices. Configuring the internal registers of the MPC106 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 PCI configuration space should be limited to initialization and error-handling software.

7.4.5.1 The PCI 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 7-6. 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 use the configuration header layout shown in Figure 7-6.

				Address Offset
Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C
Base Address Registers				10
				14
				18
				1C
Reserved				20
Reserved				24
Expansion ROM Base Address				28
Reserved				30
Reserved				34
Reserved				38
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3C

Figure 7-6. 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 (special-interest group) 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 7-6 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 μ s units
3F	Max_Lat	Specifies how often the device needs to gain access to the bus in 0.25 μ s 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 MPC106). 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 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 MPC106 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 MPC106 at 0x8000_0CF8. For address map A in the discontinuous mode, the 60x can access the CONFIG_ADDR register through the MPC106 at 0x8006_7018. For address map B and the emulation mode address map, the 60x can access the CONFIG_ADDR register at any location in the address range from 0xFEC0_0000 to

0xFEDF_FFFF. For simplicity, the address for CONFIG_ADDR is sometimes referred to as CF8, 0xn_{nnnn}_nCF8, or (in the PCI literature as) CF8h. Although systems implementing address map B or emulation mode address map can use any address in the range from 0xFEC0_0000 to 0xFEDF_FFFF for the CONFIG_ADDR register, the address 0xFEC0_0CF8 may be the most intuitive location.

The format of CONFIG_ADDR is shown in Figure 7-7.

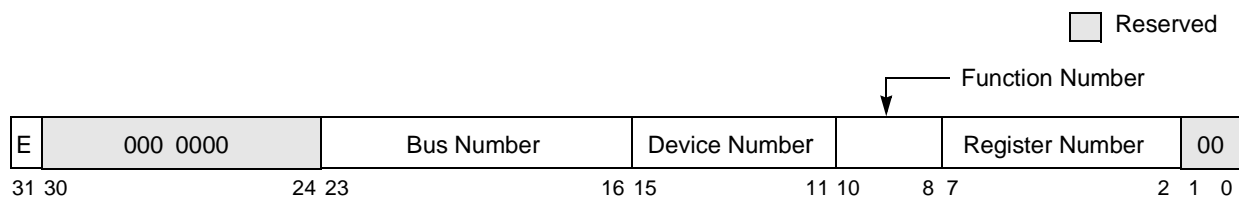


Figure 7-7. CONFIG_ADDR Register Format

Table 7-3 describes the fields within CONFIG_ADDR.

Table 7-3. CONFIG_ADDR Register Fields

Bits	Field Name	Description
31	E(nable)	The enable flag 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 MPC106, 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 MPC106 at 0x8000_0CFC–0x8000_0CFF. For address map A in the discontinuous mode, the 60x can access the CONFIG_DATA register through the MPC106 at 0x8006_701C–0x8006_701F. For address map B and the emulation mode address map, the 60x can access the CONFIG_DATA register at any location in the address range from 0xFEE0_0000 to 0xFEEF_FFFF. For simplicity, the address for CONFIG_DATA is sometimes referred to as CFC, 0xn_{nnnn}_nCFC, or (in the PCI literature as) CFCh. Although systems implementing address map B or emulation mode address map can use any address in the range from

0xFEE0_0000 to 0xFEEF_FFFF for the CONFIG_DATA register, the address 0xFEE0_0CFC–0xFEE0_0CFF may be the most intuitive location.

Note that the CONFIG_DATA register may contain 1, 2, 3, or 4 bytes depending on the size of the register being accessed.

When the MPC106 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 MPC106 performs a configuration cycle translation function 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 MPC106 performs a type 0 configuration cycle translation. If the bus number indicates a nonlocal PCI bus, the MPC106 performs a type 1 configuration cycle translation.

Figure 7-8 shows the type 0 translation function performed on the contents of the CONFIG_ADDR register to the AD[31–0] signals on the PCI bus during the address phase of the configuration cycle.

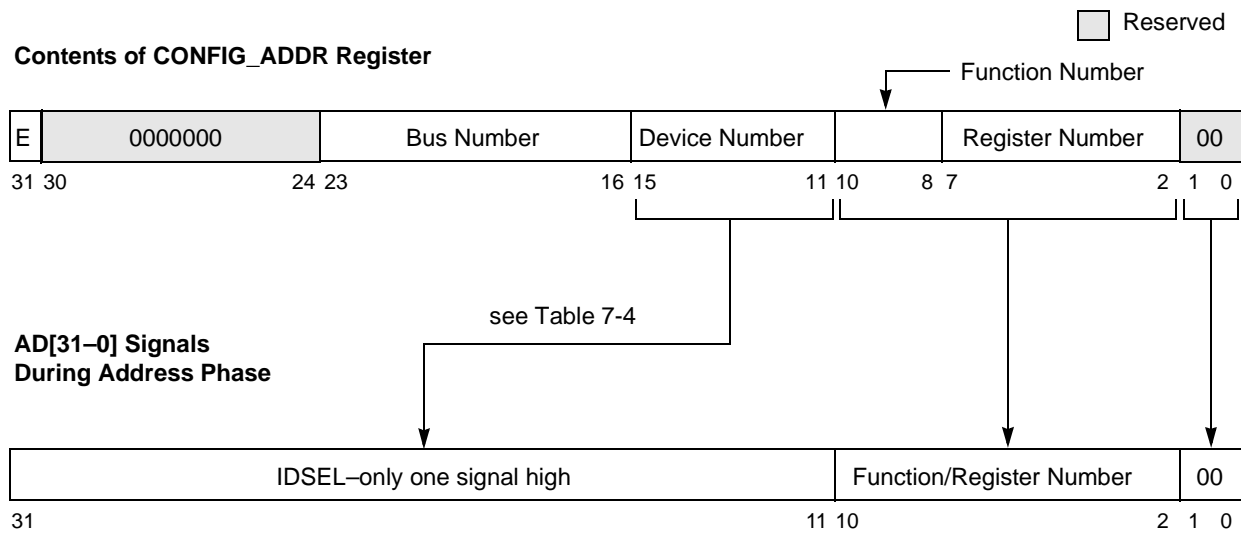


Figure 7-8. Type 0 Configuration Translation

For type 0 configuration cycles, the MPC106 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 AD[31–11] signals. For type 0 configuration cycles, the MPC106 translates the device number to IDSEL as shown in Table 7-4.

Table 7-4. Type 0 Configuration—Device Number to IDSEL Translation

Device Number		IDSEL
Binary	Decimal	
0b0_0000–0b0_1001	0–9	—
0b0_1010	10	AD31
0b0_1011	11	AD11
0b0_1100	12	AD12
0b0_1101	13	AD13
0b0_1110	14	AD14
0b0_1111	15	AD15
0b1_0000	16	AD16
0b1_0001	17	AD17
0b1_0010	18	AD18
0b1_0011	19	AD19
0b1_0100	20	AD20
0b1_0101	21	AD21
0b1_0110	22	AD22
0b1_0111	23	AD23
0b1_1000	24	AD24
0b1_1001	25	AD25
0b1_1010	26	AD26
0b1_1011	27	AD27
0b1_1100	28	AD28
0b1_1101	29	AD29
0b1_1110	30	AD30
0b1_1111 ¹	31	—

¹ A device number of all 1s indicates a PCI special-cycle or interrupt-acknowledge transaction

For type 0 translations, the function number and register number fields are copied without modification onto the AD[10–2] signals during the address phase. The AD[1–0] signals are driven to 0b00 during the address phase for type 0 configuration cycles.

For systems implementing address map A on the MPC106, there is an alternate method for generating type 0 configuration cycles, called the direct access method. The direct access configuration method bypasses the CONFIG_ADDR translation step as shown in Figure 7-9.

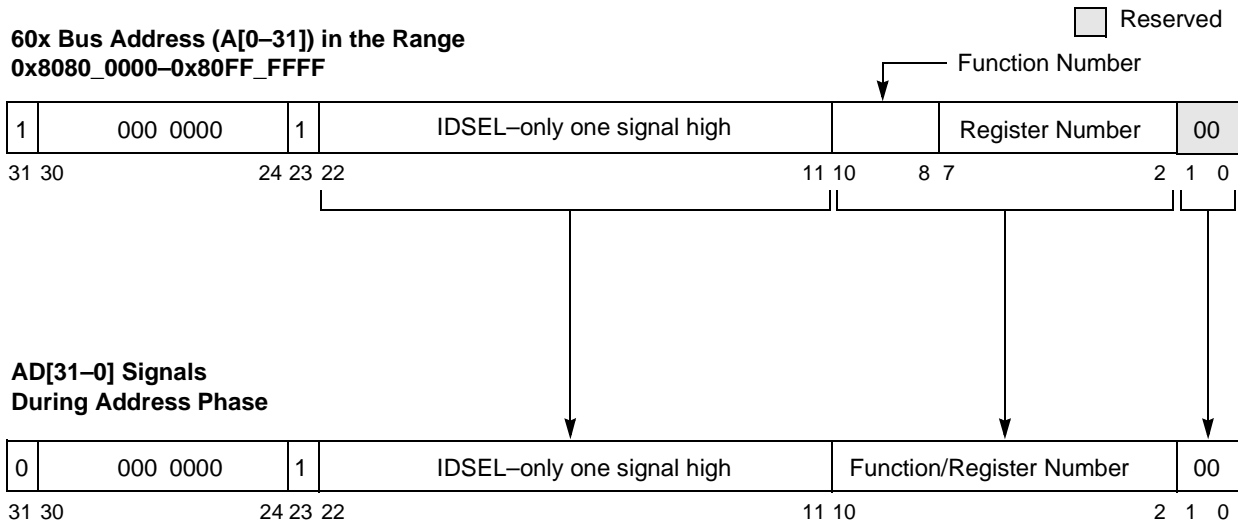


Figure 7-9. Direct-Access PCI Configuration Transaction

During the address phase of a configuration cycle, the MPC106 decodes transactions from the 60x processor in the address range from 0x8080_0000–0x80FF_FFFF, clears the most-significant address bit, and copies the 30 low-order bits of the 60x bus address (without modification) onto the AD[31–0] signals. The two least-significant bits of the 60x bus address, A[30–31], must be 0b00. Note that the direct-access method is limited to generating IDSEL on AD[11–22]. Also note that AD23 is always driven high for direct-access configuration cycles. Therefore, no PCI device should use AD23 for the IDSEL input on systems using address map A.

For type 1 translations, the MPC106 copies the 30 high-order bits of the CONFIG_ADDR register (without modification) onto the AD[31–2] signals during the address phase. The MPC106 automatically translates AD[1–0] 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 MPC106 supports—interrupt-acknowledge and special-cycles. As a master, the MPC106 may initiate both interrupt-acknowledge and special-cycle transactions; however, as a target, the MPC106 ignores interrupt-acknowledge and special-cycle transactions. Both transactions make use of the CONFIG_ADDR and CONFIG_DATA registers described in Section 7.4.5.2, “Accessing the PCI Configuration Space.”

7.4.6.1 Interrupt Acknowledge Transactions

The PCI bus supports an interrupt-acknowledge transaction. The interrupt-acknowledge command is a read operation implicitly addressed to the system interrupt controller.

When the MPC106 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 bus number corresponds to the local PCI bus (bus number = 0x00), the device number is all 1s (0b1_1111), the function number is all 1s (0b111), and the register number is zero (0b00_0000), then the MPC106 performs an interrupt-acknowledge transaction. If the bus number indicates a nonlocal PCI bus, the MPC106 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 interrupt-acknowledge command ($\overline{C/BE}[3-0] = 0b0000$). Although there is no explicit address, AD[31-0] 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{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 AD[31-0] when \overline{TRDY} is asserted. The size of the interrupt vector returned is indicated by the value driven on the $\overline{C/BE}[3-0]$ signals.

The MPC106 also provides a direct method for generating PCI interrupt-acknowledge transactions. For address map A, 60x reads to 0xBFFF_FFF0 generate PCI interrupt acknowledge transactions. For address map B and the emulation mode address map, 60x reads to any location in the address range 0xFEFF0_0000-FEFF_FFFF generate PCI interrupt-acknowledge transactions. Note that 60x writes to these addresses cause the MPC106 to assert the \overline{TEA} signal (if enabled).

7.4.6.2 Special-Cycle Transactions

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 MPC106 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 bus number corresponds to the local PCI bus (bus number = 0x00), the device number is all 1s (0b1_1111), the function number is all 1s (0b111), and the register number is zero (0b00_0000), then the MPC106 performs a special-cycle transaction on the local PCI bus. If the bus number indicates a nonlocal PCI bus, the MPC106 performs a type 1 configuration cycle translation, similar to any other configuration cycle for which the bus number does not match.

Aside from the special-cycle command ($\overline{C/BE}[3-0] = 0b0001$), the address phase contains no other valid information. Although there is no explicit address, AD[31-0] are driven to a stable state and parity is generated. During the data phase, AD[31-0] contain the

special-cycle message and an optional data field. The special-cycle message is encoded on the 16 least-significant bits (AD[15–0]); the optional data field is encoded on the most-significant 16 lines (AD[31–16]). The special-cycle message encodings are assigned by the PCI SIG steering committee. The current list of defined encodings and how the MPC106 implements them are provided in Table 7-5.

Table 7-5. Special-Cycle Message Encodings

AD[15–0]	Message	Description
0x0000	SHUTDOWN	Indicates the MPC106 is entering the sleep power saving mode
0x0001	HALT	Indicates the MPC106 is entering either the nap or sleep power saving mode
0x0002	x86 architecture-specific	This message type is not used by the MPC106.
0x0003–0xFFFF	—	Reserved

Note that the power management configuration register (PMCR) controls which special-cycle messages (if any) the MPC106 broadcasts to the PCI bus. See Section 3.2.5, “Power Management Configuration Registers (PMCRs),” for a description of the PMCR.

Each receiving agent must determine whether the special-cycle 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 transaction can insert wait states but since there is no specific target, the special-cycle message and optional data field are valid on the first clock $\overline{\text{IRDY}}$ is asserted. All special-cycle transactions 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 Exclusive Access

PCI provides an exclusive access mechanism referred to as a resource lock. The mechanism locks only the selected PCI resource (typically memory) but allows other nonexclusive accesses to unlocked targets. In this section, the term ‘locked operation’ means an exclusive access to a locked target that may span several PCI transactions. A full description of exclusive access is contained in the *PCI Local Bus Specification*.

The $\overline{\text{LOCK}}$ signal indicates that 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 the resource lock, agents performing nonexclusive accesses are free to proceed even while another master retains ownership of $\overline{\text{LOCK}}$.

7.5.1 Starting an Exclusive Access

To initiate a locked operation, a master must receive $\overline{\text{GNT}}$ when the $\overline{\text{LOCK}}$ signal is not busy. The initiator is then said to own the $\overline{\text{LOCK}}$ signal. To request a resource lock, the

master must hold $\overline{\text{LOCK}}$ negated during the address phase of a read command and assert $\overline{\text{LOCK}}$ in the clock cycle following the address phase. Note that the first transaction of a locked operation must be a read transaction.

The locked operation is not established on the PCI bus until the first data transfer ($\overline{\text{IRDY}}$ and $\overline{\text{TRDY}}$ asserted) completes. Once the lock is established, the master may retain ownership of the $\overline{\text{LOCK}}$ signal and the target may remain locked beyond the end of the current transaction. The initiator holds $\overline{\text{LOCK}}$ asserted until either the locked operation completes or until an error (master-abort or target-abort) causes an early termination. A target remains in the locked state until both $\overline{\text{FRAME}}$ and $\overline{\text{LOCK}}$ are negated. If the target retries the first transaction without a data phase completing, the master should not only terminate the transaction but should also negate $\overline{\text{LOCK}}$.

7.5.2 Continuing an Exclusive Access

When the lock owner is granted access to the bus for another exclusive access to the previously-locked target, it negates the $\overline{\text{LOCK}}$ signal during the address phase to reestablish the lock. The locked target accepts the transaction and claims the transaction. The initiator then asserts $\overline{\text{LOCK}}$ in the clock cycle following the address phase. If the initiator plans to continue the locked operation, it continues to assert $\overline{\text{LOCK}}$.

7.5.3 Completing an Exclusive Access

When a master is ready to complete an exclusive access, it should negate $\overline{\text{LOCK}}$ when $\overline{\text{IRDY}}$ is negated following the completion of the last data phase of the locked operation. This is to insure that the target is released prior to any other operation, and to insure that the resource is no longer blocked.

7.5.4 Attempting to Access a Locked Target

If $\overline{\text{LOCK}}$ is asserted during the address phase to a locked target, the locked target signals a retry, terminating the transaction without transferring any data. (The lock master always negates $\overline{\text{LOCK}}$ during the address phase of a transaction to a locked target.) Nonlocked targets ignore the $\overline{\text{LOCK}}$ signal when decoding the address. This allows other PCI agents to initiate and respond to transactions while maintaining exclusive access to the locked target.

7.5.5 Exclusive Access and the MPC106

As an initiator, the MPC106 does not generate locked operations. As a target, the MPC106 responds to locked operations by limited access exclusion to system memory from the point-of-view of the PCI bus. From the point of view of the 60x bus, only the cache line (32 bytes) of the transaction is locked.

If an initiator on the PCI bus asserts $\overline{\text{LOCK}}$ for a read transaction to system memory, the MPC106 completes the snoop transactions for any previous PCI-to-system-memory write operations and performs a snoop transaction for the locked read operation on the 60x bus. Subsequent 60x processor accesses to system memory, when $\overline{\text{LOCK}}$ is asserted, are permitted with the exception that if a 60x processor attempts to access addresses within the locked cache line, the MPC106 will retry the processor until the locked operation is completed. If a locked operation covers more than one cache line (32 bytes), only the most recently accessed cache line is locked from the processor. Since a snoop transaction is required to establish a lock, the MPC106 does not honor the assertion of $\overline{\text{LOCK}}$ when PICR1[NO_SNOOP_EN] is set.

The MPC106 releases the lock with the first write attempt to the locked line from a PCI master. If a PCI master is expecting to make several locked writes, or if it receives a retry on the first locked write, then the lock is lost and other PCI masters are able to perform non-locked reads or writes to that location. Conventional semaphore operations are not affected by this behavior. For example, a PCI master might perform a locked read followed by a locked write, and then release the lock. However, multiple locked write cycles must be avoided. If one master is expecting to access a locked location, then all masters should access that location with locked cycles.

Speculative read operations and memory-read-multiple commands ($\overline{\text{C/BE}}[3-0] = 0xC$) are incompatible with the MPC106's implementation of the lock protocol. If locked reads are necessary, software must disable speculative reads (PICR1[2] = 0). Alternately, software may use the PCI Command Register[Read_Lock] bit to force a locked access without requiring the assertion of $\overline{\text{LOCK}}$ signal. Speculative reads are allowed when Read_Lock = 1.

7.6 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 registers 1 and 2 provide for selective enabling of specific PCI error detection. The PCI status register, error detection registers 1 and 2, 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 3.2.7, "Error Handling Registers."

7.6.1 PCI Parity

Generating parity is not optional; it must be performed by all PCI-compliant devices. All PCI transactions, regardless of type, calculate even parity; that is, the number of 1s on the AD[31-0], $\overline{\text{C/BE}}[3-0]$, and PAR signals 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 $\overline{C/BE}[3-0]$ 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 responsible for driving the bus must also drive even parity on PAR one clock cycle after a valid address phase or valid data transfer, as shown in Figure 7-10.

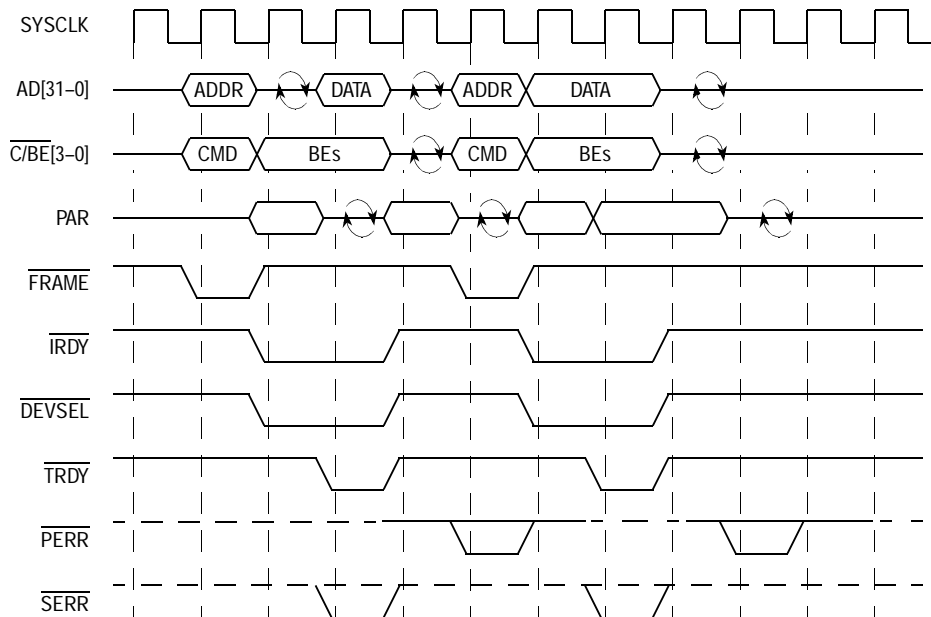


Figure 7-10. PCI Parity Operation

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.

7.6.2 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{PERR} and \overline{SERR} . The \overline{PERR} signal is used exclusively to report data parity errors on all transactions except special-cycles. The \overline{SERR} signal is used for other error signaling including data parity errors on special-cycle transactions, 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 MPC106 actions due to parity and other errors.

7.7 MPC106-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 MPC106 implements four PCI sideband signals—ISA_MASTER, FLSHREQ, MEMACK, and PIRQ. This section describes the use of the ISA_MASTER, FLSHREQ, and MEMACK signals in a PCI bus design. The use of the PIRQ signal is described in Section 7.8, “Emulation Support.”

7.7.1 ISA_MASTER

The ISA_MASTER signal provides a mechanism for ISA devices (or a PCI-to-ISA bridge) that cannot generate a full 32-bit PCI address to access system memory, but only when using address map A.

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 ISA_MASTER indicates that an ISA master is requesting access to system memory.

The ISA_MASTER signal should be asserted coincident with the PCI-to-ISA bridge receiving a PCI bus grant. When the MPC106 detects ISA_MASTER asserted during the address phase, the MPC106 automatically asserts DEVSEL to claim the transaction regardless of the address in AD[31–0]. Due to the automatic assertion of DEVSEL when ISA_MASTER is detected, bus contention may occur if the current transaction is not truly intended for the MPC106 (system memory access).

If the PCI-to-ISA bridge can generate a full 32-bit address, the ISA_MASTER signal is unnecessary and may be tied to V_{DD} (high).

7.7.2 FLSHREQ and MEMACK

The FLSHREQ signal allows a PCI agent to request that the MPC106 flush its internal buffers. The MEMACK allows the MPC106 to acknowledge that it has flushed its internal buffers.

If a master on the PCI bus asserts FLSHREQ, the MPC106 stops accepting new transactions from the 60x bus (except snoop copyback operations), completes all outstanding transactions, and then asserts MEMACK. The MPC106 holds MEMACK asserted until two cycles after the master negates FLSHREQ. When FLSHREQ is negated, the master must wait until after MEMACK is negated before it can reassert FLSHREQ.

7.8 Emulation Support

When the ESCR1[EMULATION_MODE_EN] bit is set, the MPC106 operates in an emulation mode that is fully compliant with the PC emulation option. The features described in the following sections are provided by the MPC106 when operating in PC emulation mode.

7.8.1 PCI Address Decoding

When the MPC106 is operating in PC emulation mode, the MPC106 compares the addresses of PCI memory transactions to the ESCR1[TOP_OF_MEM] bits, and simultaneously tests whether the address falls within the PCI compatibility hole (configured by setting ESCR1[PCI_COMPATIBILITY_HOLE] while in PC emulation mode). If the address is not at the top of memory and does not fall within the PCI compatibility hole, the MPC106 claims the transaction by asserting DEVSEL. If the address is above the block address in ESCR1[TOP_OF_MEM], or if the address falls inside the PCI compatibility hole (with ESCR1[PCI_COMPATIBILITY_HOLE] set to 1) the MPC106 ignores the PCI transaction.

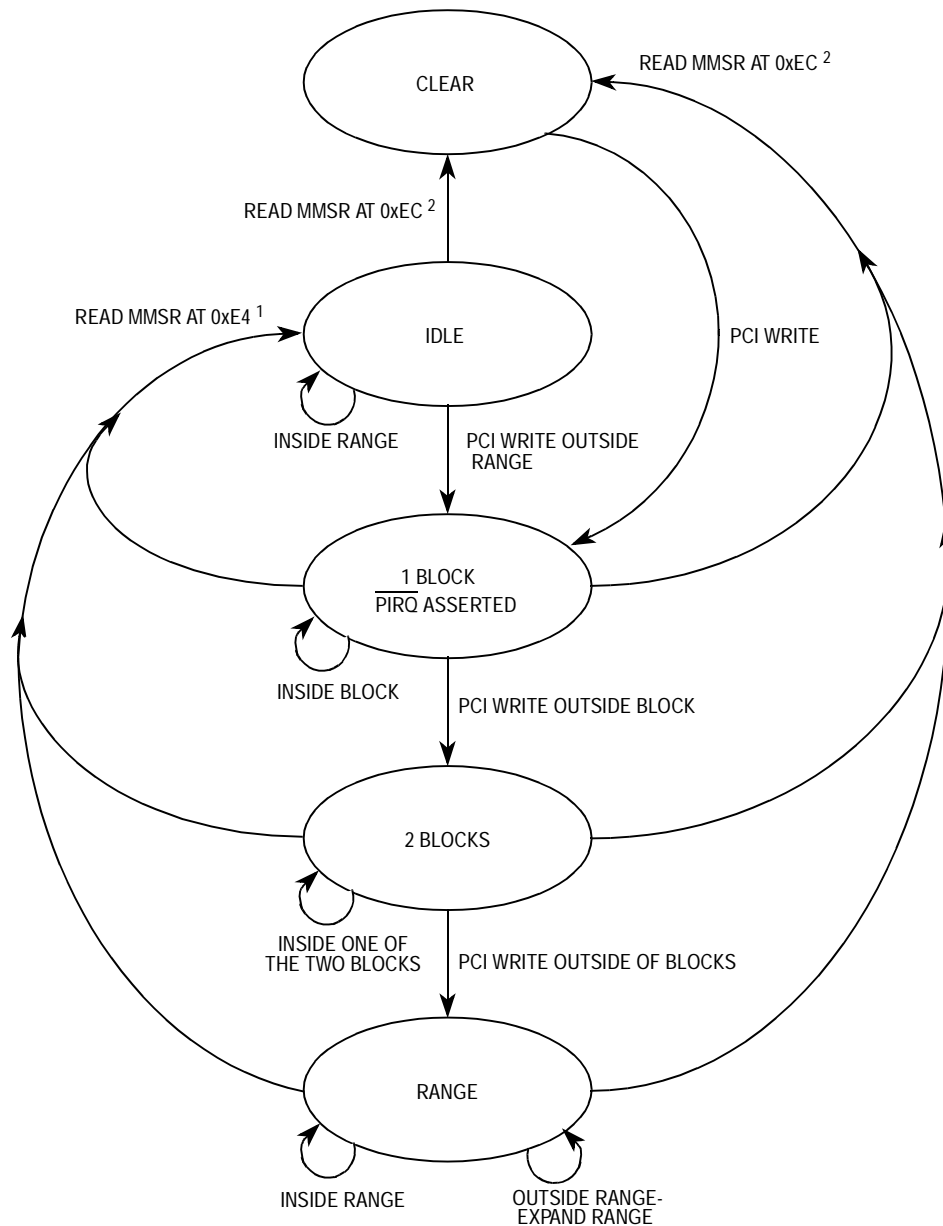
7.8.2 Interrupt Vector Relocation

The MPC106 is configured through the ESCR1[INT_VECTOR_RELOCATE] bits to relocate 60x memory accesses from 0xFFFF0_0000 to 0xFFFF_FFFF to a 1-Mbyte memory block lower in the memory array. The 12 most-significant bits of the address (0xFFF) of the 60x memory access are replaced by the contents of ESCR1[INT_VECTOR_RELOCATE], and the remaining 20 address bits are unchanged.

7.8.3 Modified Memory Status Register

When the MPC106 is operating in PC emulation mode, the modified memory status register (MMSR) tracks PCI writes to system memory so that emulation software can track system memory accesses. Note that systems that use the 3:2 or 5:2 clock mode should not use the modified memory tracking facility. These clock modes prevent the MPC106 from tracking memory properly. The MMSR contains two 15-bit fields reflecting the high (HWM) and low (LWM) memory block addresses, and a 2-bit MMSR[MOD_MEM_STATUS] field indicating the state of the register. Memory blocks are either 4 or 8 Kbytes in size, and the block size is configured by the ESCR2[MOD_MEM_SIZE] bits. If the emulated memory is less than 128 Mbytes the block size is 4 Kbytes, and 8 Kbytes if the emulated memory is larger than 128 Mbytes.

The MMSR tracks and signals the modifications to memory by PCI bus masters through the use of four states, with the current state indicated by the MMSR[MOD_MEM_STATUS] bits. The operation of the MMSR is shown in Figure 7-11.



Notes:

1. $\overline{\text{PIRQ}}$ negated, MMSR[MOD_MEM_STATUS] cleared, MMSR[HWM] and MMSR[LWM] not cleared.
2. $\overline{\text{PIRQ}}$ negated, MMSR[MOD_MEM_STATUS] cleared, MMSR[HWM] and MMSR[LWM] cleared.

Figure 7-11. Modified Memory Tracking States

When the MPC106 is reset, the MMSR[HWM], MMSR[LWM], and MMSR[MOD_MEM_STATUS] bits are cleared, shown as the CLEAR state in Figure 7-11. If a PCI-to-memory write occurs, the MMSR[MOD_MEM_STATUS] bits are set to 0b01, the block address of the modified system memory is loaded into both MMSR[HWM] and MMSR[LWM], and the $\overline{\text{PIRQ}}$ signal is asserted. This state is shown as the 1 BLOCK state in Figure 7-11. If a 60x processor subsequently performs a read access to the MMSR at address 0xE4, the $\overline{\text{PIRQ}}$ signal is negated, and the MMSR[MOD_MEM_STATUS] bits are cleared. However, the address in MMSR[HWM]

and MMSR[LWM] remains unchanged. This state is shown as the IDLE state in Figure 7-11. If a 60x processor performs a read access to the MMSR at address 0xEC, the $\overline{\text{PIRQ}}$ signal is negated, and MMSR[MOD_MEM_STATUS], MMSR[HWM], and MMSR[LWM] are cleared, returning the MMSR to the CLEAR state.

If a PCI write operation to memory occurs while the MMSR[MOD_MEM_STATUS] bits are set to 0b01 (1 BLOCK state), the MPC106 tests the memory block address against the contents of MMSR[HWM] and MMSR[LWM]. If the memory block address matches the previous memory block address, there is no state change. If the memory block address is different from the address for the first PCI write operation, the block address for the second PCI write operation is placed in either the MMSR[HWM] and MMSR[LWM] depending on whether the address is greater than or less than the block address of the first PCI write operation, and the MMSR[MOD_MEM_STATUS] bits are set to 0b10 (2 BLOCK state). A read access to the MMSR by a 60x processor through configuration address 0xE4 or 0xEC returns the MMSR to the IDLE or CLEAR state (MMSR[MOD_MEM_STATUS] = 0b00), and the MPC106 negates the $\overline{\text{PIRQ}}$ signal.

If a PCI write operation occurs while the MMSR[MOD_MEM_STATUS] bits are set to 0b10 (2 BLOCK state), the MPC106 tests the memory block address against the contents of MMSR[HWM] and MMSR[LWM]. If the memory block address matches one of the two previous memory block addresses, there is no state change. If the memory block address is different from the addresses for the previous PCI write operations, the block address for the current PCI write operation is placed in either the MMSR[HWM] or MMSR[LWM] depending on whether the address is greater than or less than the block address of the previous PCI write operations, and the MMSR[MOD_MEM_STATUS] bits are set to 0b11 (RANGE state). A read access to the MMSR by a 60x processor through configuration address 0xE4 or 0xEC returns the MMSR to the IDLE or CLEAR state (MMSR[MOD_MEM_STATUS] = 0b00), and the MPC106 negates the $\overline{\text{PIRQ}}$ signal.

If a PCI write operation occurs while the MMSR[MOD_MEM_STATUS] bits are set to 0b11 (RANGE state), the MPC106 tests the memory block address against the contents of MMSR[HWM] and MMSR[LWM]. If the memory block address falls between MMSR[HWM] and MMSR[LWM], there is no state change. If the memory block address is different from the addresses for the previous PCI write operations, the block address for the current PCI write operation is placed in either the MMSR[HWM] or MMSR[LWM] depending on whether the address is greater than or less than the block address of the previous PCI write operations. A read access to the MMSR by a 60x processor through configuration address 0xE4 or 0xEC returns the MMSR to the IDLE or CLEAR state (MMSR[MOD_MEM_STATUS] = 0b00), and the MPC106 negates the $\overline{\text{PIRQ}}$ signal.

If a PCI write operation occurs while the MMSR[MOD_MEM_STATUS] bits are cleared (IDLE state, following a 60x read access to configuration address 0xE4), the MPC106 tests the memory block address against the contents of MMSR[HWM] and MMSR[LWM]. If the memory block address equals the address in MMSR[HWM] or MMSR[LWM], or the address falls between MMSR[HWM] and MMSR[LWM] and the

MMSR[MOD_MEM_STATUS] bits were previously set to 0b11 (RANGE state), there is no state change, and the contents of MMSR[HWM] and MMSR[LWM] remain unchanged. If the new block address falls outside the addresses in MMSR[HWM] and MMSR[LWM], the block address is placed in both MMSR[HWM] and MMSR[LWM], the $\overline{\text{PIRQ}}$ signal is asserted, and the MMSR[MOD_MEM_STATUS] bits are set to 0b01 (1 BLOCK state). A 60x processor read operation to the configuration address 0xEC causes the MMSR[HWM] and MMSR[LWM] bits to be cleared, and the MMSR[MOD_MEM_STATUS] bits remain cleared (with a transition to the CLEAR state).

7.8.4 Curious Code Protection

Memory accesses from the 60x bus by programs running in emulation that cause memory select errors (through accesses where there is no physical memory), or that cause a master abort on the PCI bus will be returned 0xFFFF_FFFF_FFFF_FFFF on the data bus, and will not cause $\overline{\text{TEA}}$ to be asserted. Transactions initiated on the PCI bus that cause a memory select error will also be returned 0xFFFF_FFFF during the data tenure on the PCI bus.

7.9 Processor-to-PCI Transaction Examples

The figures in this section provide examples of signal timing for 60x processor-to-PCI transactions. Figure 7-12 shows a processor burst read from a device on the PCI bus. Figure 7-13 shows a processor burst write to a device on the PCI bus. Figure 7-14 shows a processor read from PCI that is terminated with a master abort because no device asserted $\overline{\text{DEVSEL}}$ within four clock cycles following the PCI address phase.

PCI-to system-memory transactions are shown in Section 6.4.12, “PCI-to-DRAM Transaction Examples.”

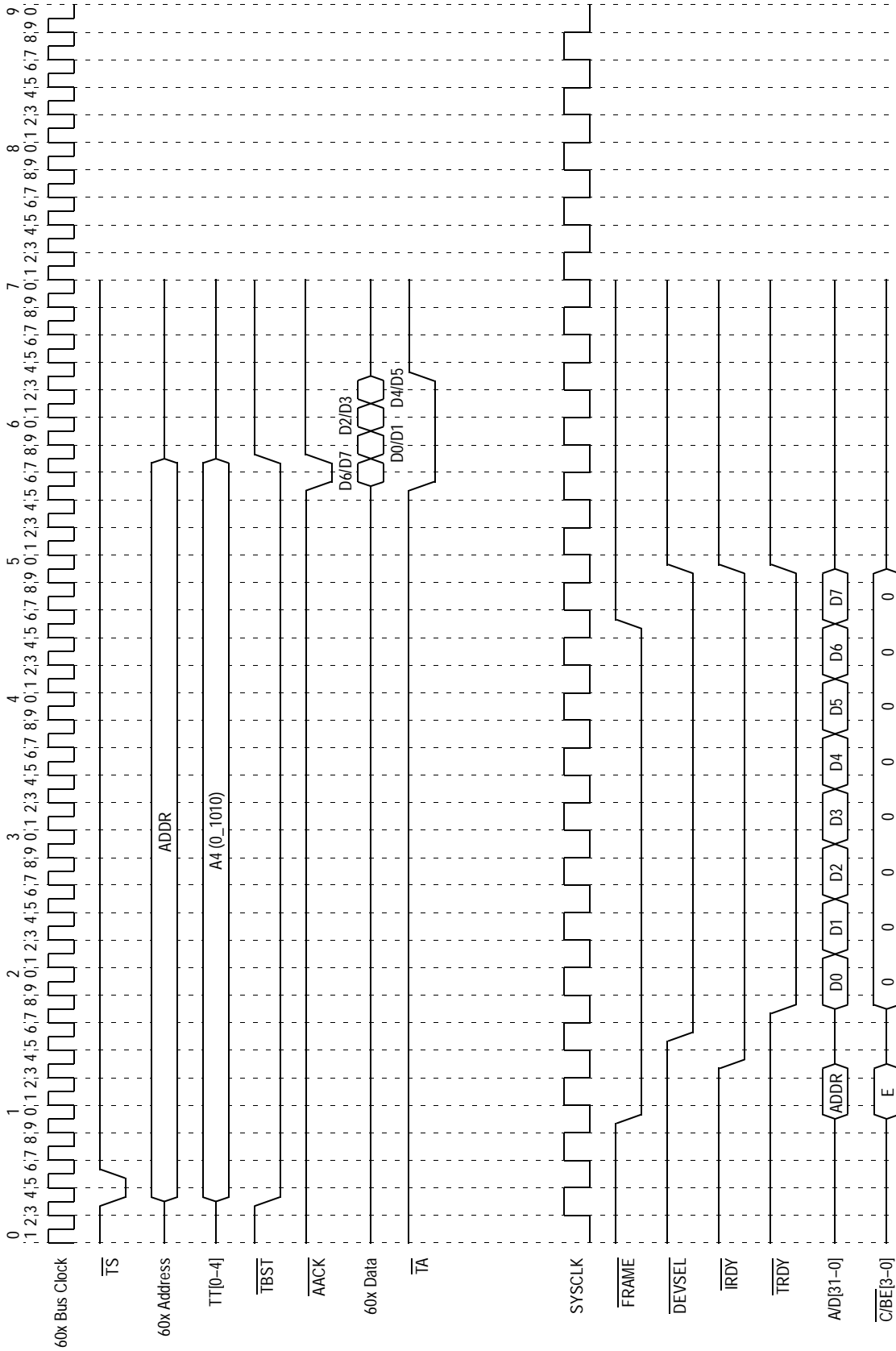


Figure 7-12. Processor Burst Read to Device on PCI Bus

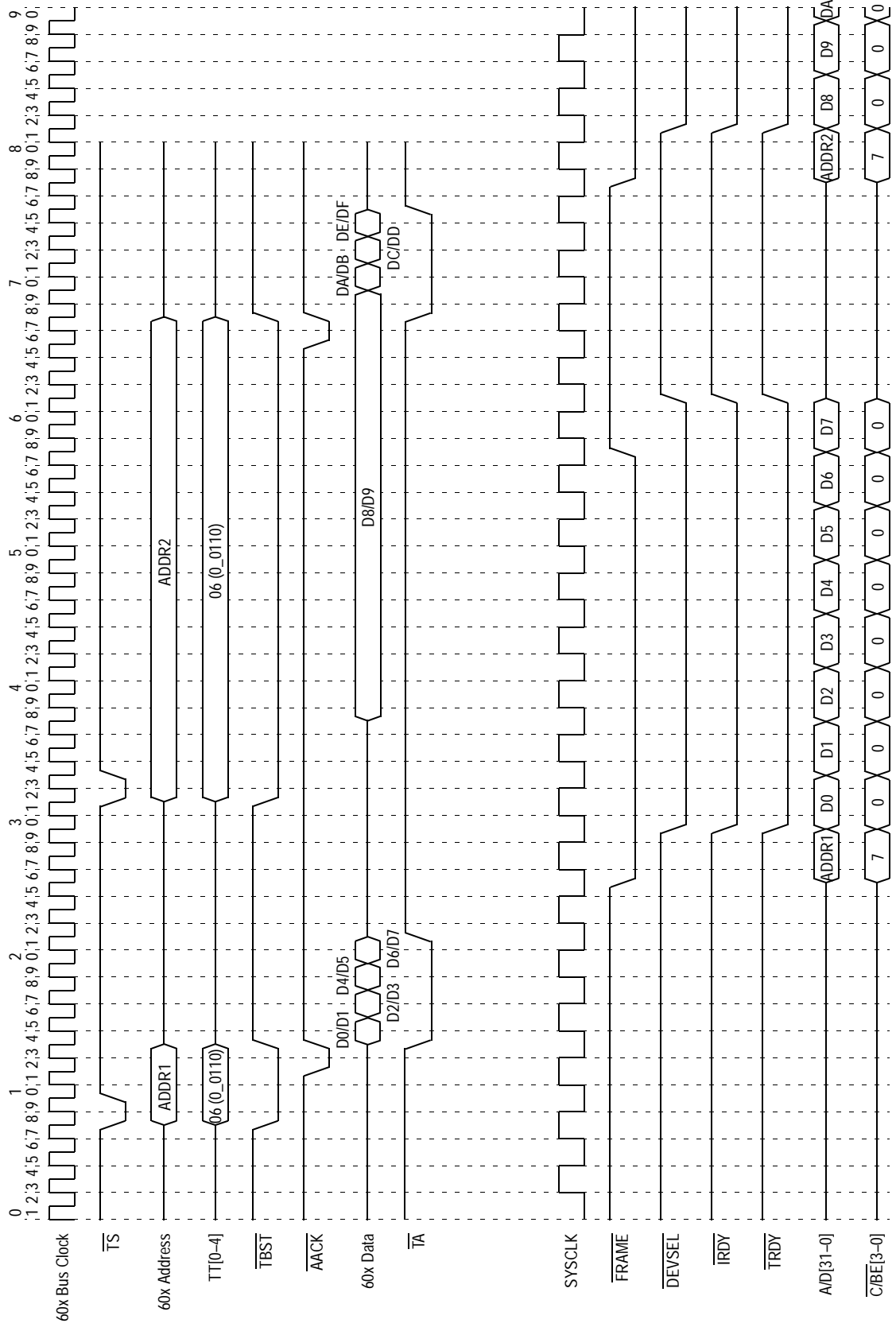


Figure 7-13. Processor Burst Write to Device on PCI Bus

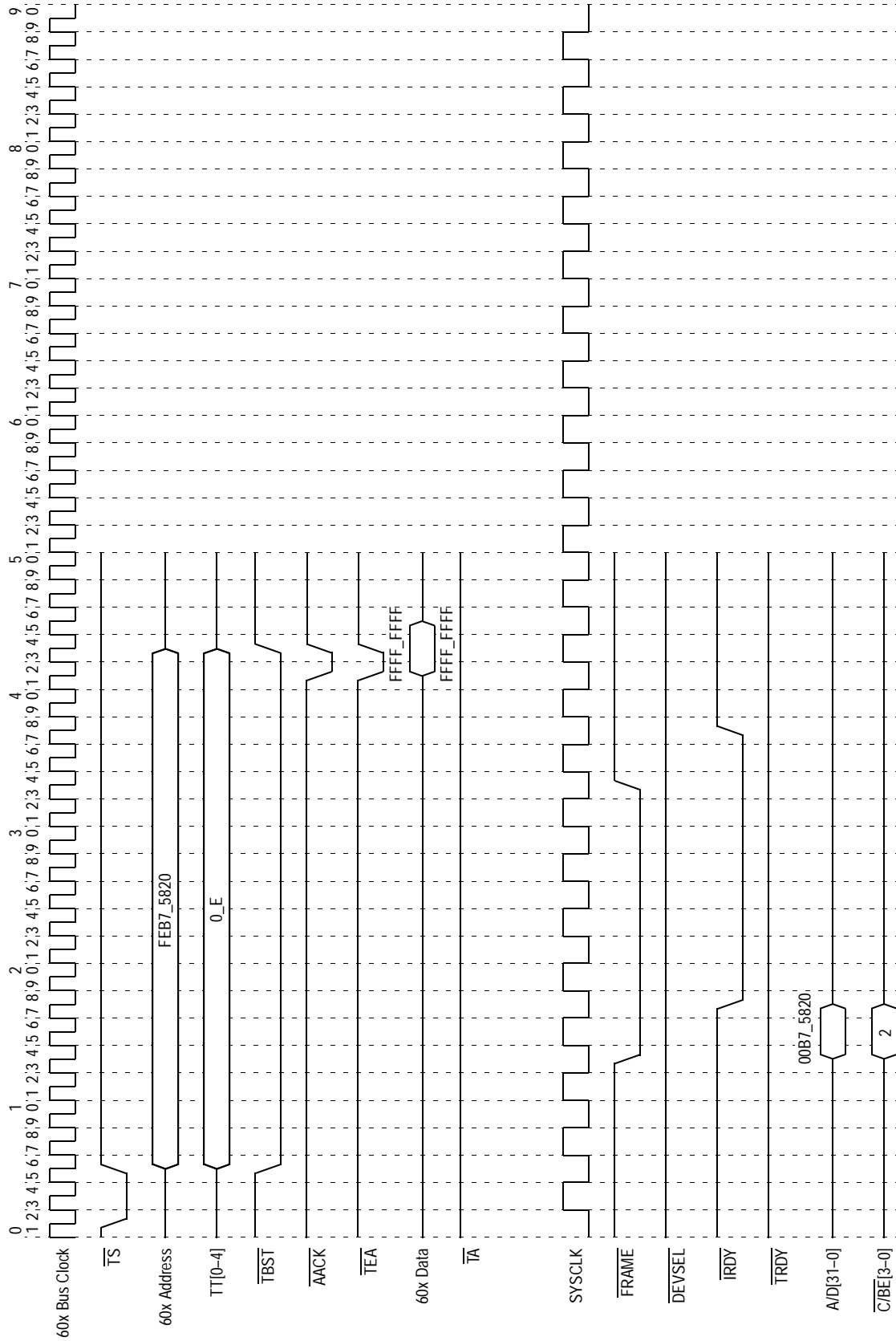


Figure 7-14. Processor Read from PCI with Master-Abort

Chapter 8

Internal Control

The MPC106 uses internal buffering to store addresses and data moving through it, and to maximize opportunities for concurrent operations. An internal control unit directs the flow of transactions through the MPC106, performing internal arbitration and coordinating the internal and external snooping. This chapter describes the internal buffering and arbitration logic of the MPC106.

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 copybacks and burst writes when ECC is enabled, occur directly on the shared data bus, so no internal data buffering is required for those transactions.

Each of the seven data buffers has a corresponding address buffer. An additional buffer stores the address of any 60x processor accesses to system memory. All transactions entering the MPC106 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 MPC106. This is especially important for write transactions that enter the MPC106 because memory can be updated out-of-order with respect to other transactions.

The MPC106 performs 60x bus snoop transactions (provided snooping is enabled) for each PCI access to system memory to enforce coherency between the PCI-initiated access and the L1 and L2 caches. All addresses are snooped in the order that they are received from the PCI bus. For systems that do not require hardware-enforced coherency, snooping can be disabled by setting the CF_NO_SNOOP parameter in PICR2. Note that if snooping is disabled, the PCI exclusive access mechanism (the $\overline{\text{LOCK}}$ signal) does not affect the transaction. That is, the transaction will complete, but the 60x processor will not be prohibited from accessing the cache line.

Figure 8-1 depicts the organization of the internal buffers.

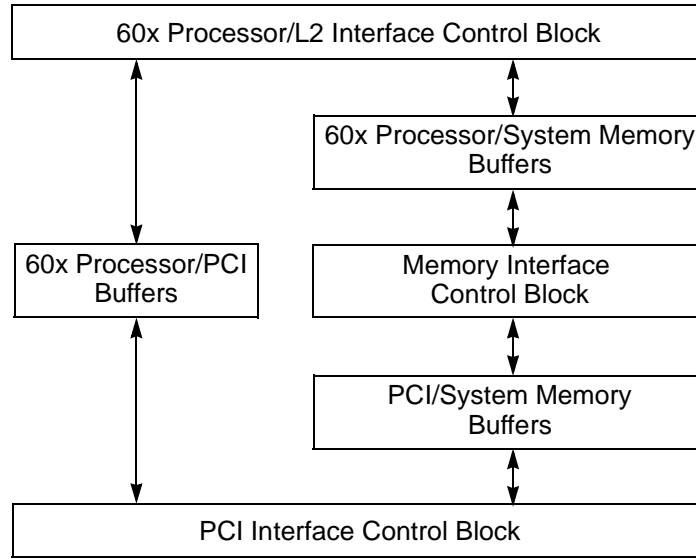


Figure 8-1. MPC106 Internal Buffer Organization

8.1.1 60x Processor/System Memory Buffers

Because systems using the MPC106 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 copyback buffer which is used for temporary storage of L1 copyback or external L2 copyback operations due to snooping PCI-initiated reads from memory, L2 castouts, and processor burst writes when ECC is enabled. The copyback 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.

Figure 8-2 shows the address and data buffers between the 60x processor bus and the system memory bus.

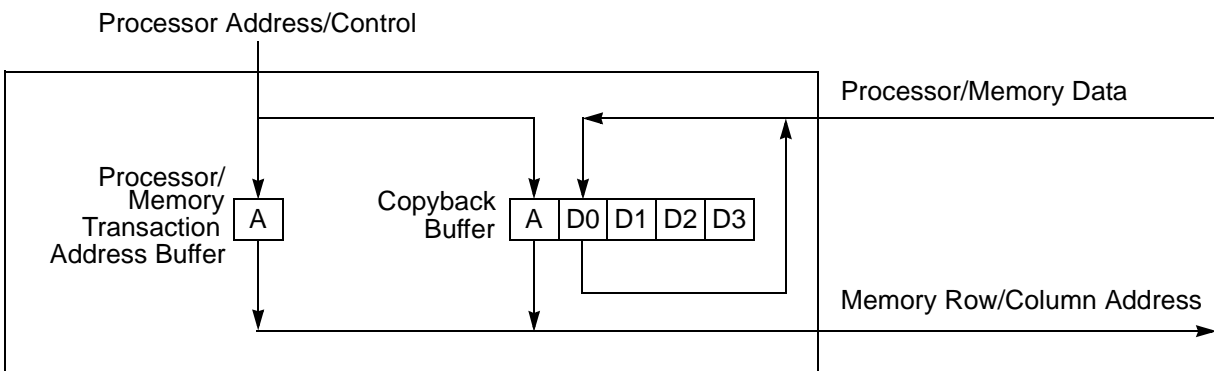


Figure 8-2. 60x Processor/System Memory Buffers

In the case of a snoop for a PCI read from system memory that causes an L1 or external L2 copyback, the copyback data is simultaneously latched in the copyback buffer and the PCI-read-from-system-memory buffer (PCMRB). Once the L1 or external L2 copyback is complete, the data is forwarded to the PCI agent from the PCMRB. The MPC106 flushes the data in the copyback buffer to system memory at the earliest available opportunity.

L2 castouts 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 MPC106 latches the modified data from the L2 to minimize the latency of the original 60x processor/system memory transaction. The MPC106 flushes the data in the copyback buffer to system memory at the earliest available opportunity. Note that the copyback buffer is also used for L2 cache flush operations.

For processor burst writes to memory with ECC enabled, the MPC106 uses the copyback buffer as a temporary holding area while it generates the appropriate ECC codes to send to memory.

Once the copyback buffer has been filled, the data remains in the buffer until the system memory bus is available to flush the copyback buffer contents to system memory. During the time that modified data waits in the copyback buffer, all transactions to system memory space are snooped against the copyback buffer. If a 60x processor burst write to system memory hits in the copyback buffer, the copyback buffer is invalidated. Also, since the L1 cache in the 60x processor can contain a more recently modified version of a cache line than that in the copyback buffer, all PCI-initiated transactions that hit in the copyback buffer cause a snoop broadcast on the 60x processor bus (provided snooping is enabled).

8.1.2 60x Processor/PCI Buffers

There are three data buffers for processor accesses to PCI—one 32-byte processor-to-PCI-read buffer (PRPRB) for processor reads from PCI, and two 16-byte processor-to-PCI-write buffers (PRPWBs) for processor writes to PCI. Figure 8-3 shows the address and data buffers between the 60x processor bus and the PCI bus.

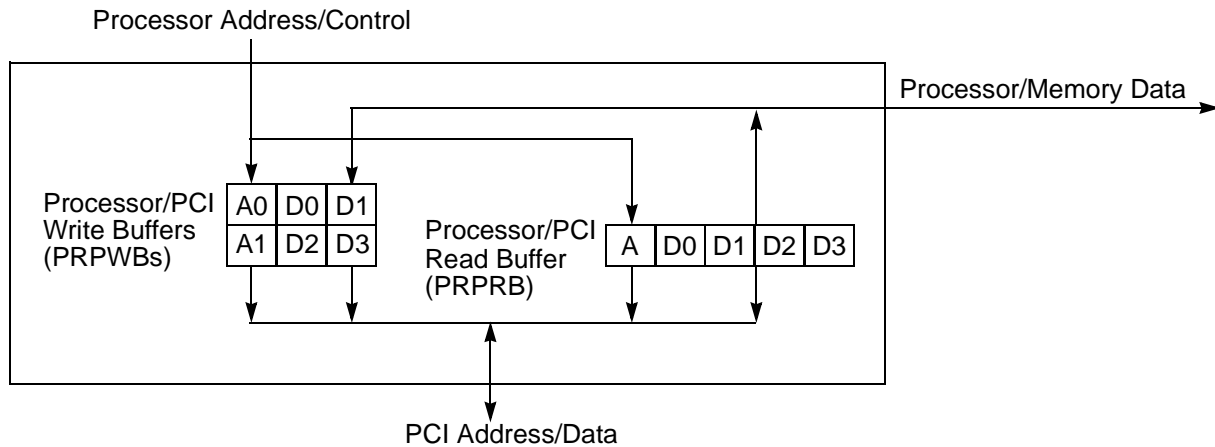


Figure 8-3. 60x Processor/PCI Buffers

8.1.2.1 Processor-to-PCI-Read 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 MPC106 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 MPC106 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 MPC106 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 system memory access, the MPC106 retries the 60x processor so that the incoming PCI transaction can be snooped. A PCI-initiated access to system memory may require a snoop transaction on the 60x processor bus, and a copyback may be necessary. The MPC106 does not provide the data to the 60x bus (for the processor to PCI read transaction) until all outstanding snoops for PCI writes to system memory have completed. Note that if a processor read from PCI transaction is waiting for a PCWMB snoop to complete, all subsequent requests for PCI writes to system memory will be retried on the PCI bus.

The PCI interface of the MPC106 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, and all PCI write to system memory snoops have completed, the MPC106 asserts \overline{AACK} and \overline{DBGn} (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 MPC106 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 MPC106 is waiting for the PCI bus grant, the MPC106 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 PRPRB to ensure that the processor is requesting the same data. Once all data requested by the processor is latched in the PRPRB, 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 MPC106 to support both burst transactions and streams of single-beat transactions. The MPC106 performs store gathering (if enabled) of sequential accesses 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 MPC106 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 MPC106 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 sequential bytes in 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/O 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 MPC106 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. While the PCI interface is busy with the first transfer, any sequential processor single-beat writes within 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 PCI-to-system memory read buffer (PCMRB) for PCI reads from system memory and two 32-byte PCI-to-system memory write buffers (PCMWBs) for PCI writes to system memory. Figure 8-4 shows the address and data buffers between the PCI bus and the system memory.

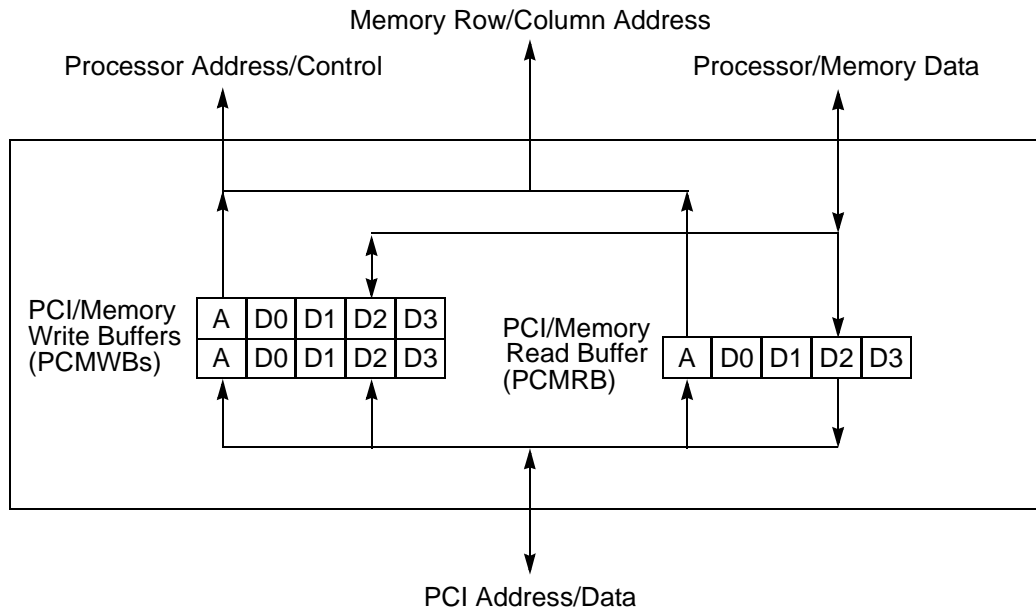


Figure 8-4. PCI/System Memory Buffers

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). All snoops for PCI accesses to system memory are performed strictly in-order. Table 8-1 summarizes the snooping behavior of PCI accesses to system memory 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	copyback	Yes
Read (not locked)	PCMRB	No
Read (first access of a locked transfer)*	PCMRB	Yes
Read (not locked)	PCMWB	No
Read (first access of a locked transfer)*	PCMWB	Yes
Write	copyback	Yes
Write	PCMRB	Yes
Write	PCMWB	No

* 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-to-System-Memory-Read Buffer (PCMRB)

When a PCI device initiates a read from system memory, the address is snooped on the 60x processor bus (provided snooping is enabled). If the memory interface is available, the memory access is started simultaneously with the snoop. If the snoop results in a hit in either the L1 or L2 cache, the MPC106 cancels the system memory access.

Depending on the outcome of the snoop, the requested data is latched into either the 32-byte PCI-to-system-memory-read buffer (PCMRB), or into both the copyback buffer and the PCMRB (as described in Section 8.1.1, “60x Processor/System Memory Buffers”).

- If the snoop hits in the L1 or externally-controlled L2, the copyback data is written to both the copyback buffer and to the PCMRB. The data is forwarded to the PCI bus from the PCMRB, and to system memory from the copyback buffer.
- If the snoop hits in the internally-controlled 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 copyback 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. If the PCI agent requested a cache wrap mode transfer, the beginning of the cache line is then loaded into the PCMRB.

The data is forwarded to the PCI bus 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 to system memory hits in the PCMRB, the PCMRB 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.

8.1.3.1.1 Speculative PCI Reads from System Memory

To minimize the latency for large block transfers, the MPC106 provides the ability to perform speculative PCI reads from system memory. When speculative reading is enabled, the MPC106 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 the MPC106 has completed the current PCI read, the data at the speculative address is fetched from system memory and loaded into the PCMRB in anticipation of the next PCI request.

Note that the assertion of \overline{CAS}_n for the speculative operation is delayed until PCI is finished reading the data currently latched in the PCMRB. If a different address is requested, the speculative operation is halted and any data latched in the PCMRB is invalidated.

Speculative PCI reads are enabled on a per access basis by using the PCI memory-read-multiple command. Speculative PCI reads can be enabled for all PCI memory read commands (memory-read, memory-read-multiple, and memory-read-line) by setting bit 2 in PICR1.

The MPC106 starts the speculative read operation only under the following conditions:

- PICR1[2] = 1 or the current PCI read access is from a memory read-multiple command.
- The current PCI read access started at the beginning of the cache line.
- The MPC106 is accessing the third double word of the cache line in the PCMRB for the current PCI read.
- No internal buffer flushes are pending.
- The access is to system memory space. (The MPC106 does not perform speculative reads on system ROM space.)

Speculative PCI read operations must not be allowed to occur past the last cache line of the physical memory boundaries as set by the memory boundary registers or the MPC106 will hang.

Note that speculative read operations and memory-read-multiple commands are incompatible with the MPC106's implementation of the PCI exclusive access protocol. See Section 7.5.5, "Exclusive Access and the MPC106," for more information.

8.1.3.2 PCI-to-System-Memory-Write Buffers (PCMWBs)

For PCI write transactions to system memory, the MPC106 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 (if snooping is enabled). 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 copyback 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 copyback.

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 MPC106 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 MPC106 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.

8.2 Internal Arbitration

The arbitration for the PCI bus is performed externally. All processor to PCI transactions are performed strictly in-order. Also, all snoops for PCI accesses to system memory are performed in order (if snooping is enabled). However, the MPC106 performs arbitration internally for the shared processor/memory data bus. The arbitration for the processor/memory data bus employs the priority scheme shown in Table 8-2.

Table 8-2. Internal Arbitration Priorities

Priority	Operation
1	A high-priority copyback buffer flush due to one of the following: <ul style="list-style-type: none"> • A PCI access to system memory hits in the copyback buffer. • An L2 castout occurs simultaneously with a PCI access to system memory and both the castout and the PCI access are to the same address in system memory. • An L2 castout hits a nonsnooped address in the PCMWB. • A 60x processor burst write to system memory with ECC enabled hits a nonsnooped address in the PCMWB. • A 60x processor burst write to system memory with ECC enabled hits a nonsnooped address in the PCMRB.
2	A PCI read from system memory (with snoop complete)
3	Priority 60x processor or L2 cache transfers including the following: <ul style="list-style-type: none"> • An L2 cache copyback (or PCMRB data transfer) due to a PCI read snoop hit • A 60x processor read from system memory • A 60x processor to L2 cache transfer • A fast L2 cache castout
4	A high priority PCMWB flush due to one of the following: <ul style="list-style-type: none"> • A PCI read hits in the PCMWB • The PCMWB is full and another PCI write to system memory starts • A 60x processor to system memory read hits in the PCMWB • A 60x processor to system memory single-beat write hits in the PCMWB
5	A medium priority copyback buffer flush due to one of the following: <ul style="list-style-type: none"> • A 60x processor read hits in the copyback buffer. • A 60x processor single-beat write hits in the copyback buffer. • The copyback buffer is full and new data needs to be written to it.
6	Normal 60x processor or L2 cache transfers including the following: <ul style="list-style-type: none"> • A 60x processor write to system memory • A snoop copyback due to a PCI write snoop • A 60x processor read from PCI • A 60x processor write to PCI • A copyback buffer fill
7	A PCI read from system memory (with snoop not complete)
8	A low-priority copyback buffer flush
9	A low-priority PCMWB flush
10	A PCMRB prefetch from system memory due to a speculative PCI read operation

Chapter 9

Error Handling

The MPC106 provides error detection and reporting on the three primary interfaces (60x processor interface, memory interface, and PCI interface). This chapter describes how the MPC106 handles different error (or interrupt) conditions.

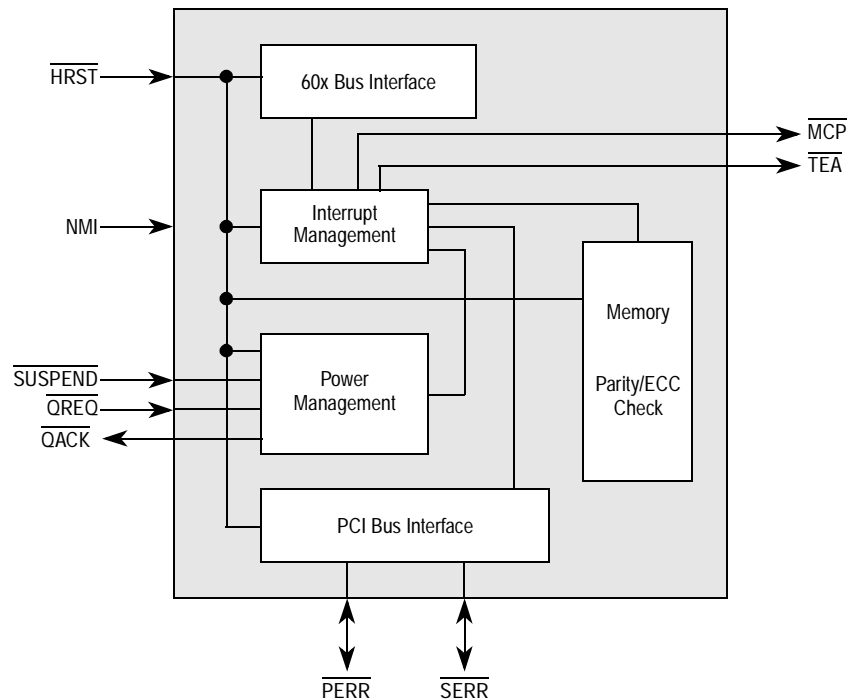
Errors detected by the MPC106 are reported to the 60x processor by asserting the machine check ($\overline{\text{MCP}}$) or transfer error acknowledge ($\overline{\text{TEA}}$) signal. The system error ($\overline{\text{SERR}}$) and parity error ($\overline{\text{PERR}}$) signals are used to report errors on and to the PCI bus. The MPC106 provides the NMI signal for ISA bridges to report errors on the ISA bus. The MPC106 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 MPC106 detects illegal transfer types from the 60x processor, illegal L2 copyback errors, illegal Flash write transactions, PCI address and data parity errors, accesses to memory addresses out of the range of physical memory, memory parity errors, memory refresh overflow errors, L2 cache parity errors, ECC errors, PCI master-abort cycles, and PCI received target-abort errors.

The MPC106 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 3.2.7.4, “Error Status Registers,” for more information. Chapter 2, “Signal Descriptions,” contains the signal definitions for the interrupt signals.

Figure 9-1 provides the internal interrupt management block diagram.


Figure 9-1. Internal Interrupt Management Block Diagram

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 (MCP or $\overline{\text{TEA}}$)	Unsupported 60x bus transaction, illegal L2 copyback operation, or Flash write error	Nonrecoverable in most cases
2	Machine check (MCP)	PCI address parity error ($\overline{\text{SERR}}$) or PCI data parity error ($\overline{\text{PERR}}$) when the MPC106 is acting as the PCI target	Nonrecoverable in most cases
3	Machine check (MCP or $\overline{\text{TEA}}$)	Memory select error, memory data read parity error, memory refresh overflow, L2 parity error, or ECC error	Nonrecoverable in most cases
4	Machine check (MCP or $\overline{\text{TEA}}$)	PCI address parity error ($\overline{\text{SERR}}$) or PCI data parity error ($\overline{\text{PERR}}$) when the MPC106 is acting as the PCI master, PCI master-abort, or received PCI target-abort	Nonrecoverable in most cases
5	Machine check (MCP)	NMI (nonmaskable interrupt)	Nonrecoverable in most cases

Note that for priority 1 through 5, the exception is the same. The machine check exception and the priority are related to additional error information provided by the MPC106 (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 MPC106 aborts all current internal and external transactions, releases all bidirectional I/O signals to a high-impedance state, ignores the input signals (except for SYSCLK, and the configuration signals $\overline{\text{DBG0}}$, $\overline{\text{FOE}}$, $\overline{\text{RCS0}}$, and PLL[0–3]), and drives most of the output signals to an inactive state. (Table 2-2 shows the states of the output-only signals during system reset.) The MPC106 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 MPC106 are given in the MPC106 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 MPC106 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 MPC106 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 MPC106 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] bit in the 60x processor. The programmable parameter PICR1[MCP_EN] is used to enable or disable the assertion of $\overline{\text{MCP}}$ by the MPC106.

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 MPC106 holds $\overline{\text{MCP}}$ asserted until the 60x processor has taken the exception. The MPC106 decodes an interrupt acknowledge cycle by detecting 60x processor reads from the two possible machine check exception addresses at 0x0000_0200–0x0000_0207 and 0xFFFF0_0200–0xFFFF0_0207.

The $\overline{\text{MCP}}$ signal can be configured, by programming PMCR2[SHARED_MCP], as an output-only or as an open drain output signal. When configured for output-only, the MPC106 always drives the $\overline{\text{MCP}}$ signal. When configured for open drain output, the MPC106 drives the $\overline{\text{MCP}}$ signal only when signalling a detected error. Otherwise, the MPC106 releases $\overline{\text{MCP}}$ to a high-impedance state. The open drain configuration allows other devices to signal $\overline{\text{MCP}}$ to the 60x processor without contention.

9.2.2.2 Transfer Error Acknowledge ($\overline{\text{TEA}}$)

The MPC106 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 MPC106 are set to report the specific error.

Assertion of $\overline{\text{TEA}}$ terminates the data transaction in progress; that is, it is not necessary to assert $\overline{\text{TA}}$ because it will be ignored by the target processor. An unsupported transaction causes the assertion of $\overline{\text{TEA}}$ (provided TEA is enabled). Unsupported transactions include:

- A direct-store access
- A graphics read or write (**eciwx** or **ecowx**)
- A write to the PCI interrupt-acknowledge space (map A or map B)
- A write to system ROM space, when Flash writes are disabled
- An aborted processor-to-PCI transaction

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 the MSR[ME] bit in 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 MPC106 asserts $\overline{\text{TEA}}$ for only one clock. 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 programmable parameter PICR1[TEA_EN] is used to enable or disable the assertion of $\overline{\text{TEA}}$ by the MPC106. If PICR1[TEA_EN] is programmed to disable the assertion of $\overline{\text{TEA}}$, and a 60x processor data transfer error occurs, then the MPC106 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 MPC106 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 AD[31–0] 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 1s on AD[31–0], $\overline{\text{C/BE}}[3–0]$, and PAR equals an even number.

The $\overline{\text{SERR}}$ 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 $\overline{\text{SERR}}$ is activated due to an address parity error.

Bits 8 and 6 of the PCI command register control whether the MPC106 asserts $\overline{\text{SERR}}$ upon detecting one of the error conditions. Bit 14 of the PCI status register reports when the MPC106 has asserted the $\overline{\text{SERR}}$ signal.

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

The $\overline{\text{PERR}}$ signal is used to report PCI data parity errors during all PCI transactions, except for a PCI special-cycle command. The agent responsible for driving AD[31–0] 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 1s on AD[31–0], $\overline{\text{C/BE}}[3–0]$, and PAR equals an even number.

The $\overline{\text{PERR}}$ 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 MPC106 ignores $\overline{\text{PERR}}$. Bit 15 and bit 8 of the PCI status register are used to report when the MPC106 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 MPC106. The NMI signal is driven by the PCI-to-ISA bridge to report any nonrecoverable error detected on the ISA bus (normally, through the $\overline{\text{IOCHCK}}$ 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 MPC106 reports the NMI error to the 60x processor by asserting $\overline{\text{MCP}}$.

9.3 Error Reporting

Error detection on the MPC106 is designed to log the occurrence of an error and also log information related to the error condition. The individual error detection bits are contained in the PCI status register, error detection register 1 (ErrDR1), and error detection register 2 (ErrDR2). These bits indicate which specific error has been detected. (The error detection bits are specifically bits 15,13, and 12 in the PCI status register, bits 7–4 and 2–0 in ErrDR1, and bits 5,4,3, and 0 in ErrDR2.)

The intent of error reporting is to log the information pertaining to the first error that occurs and prevent additional errors from being reported until the first error is acknowledged (and cleared). In order for additional errors to be reported, all error detection bits must be cleared. Once one of the error detection bits is set, the MPC106 will not report additional errors until all of the error detection bits are cleared. As a programming note, it is possible for more than one of the error detection bits to be set. This could happen if simultaneous errors are detected by the MPC106. Therefore, software must check to see whether more than one bit is set before trying to determine information about the error.

The 60x/PCI error address register, the 60x bus error status register, and the PCI bus error status register, together with ErrDR1[3] (60x/PCI cycle) and ErrDR2[7] (invalid error address), are used to provide additional information about a detected error condition. When an error is detected, the associated information is latched inside these registers until all the error detection bits are cleared. Subsequent errors will set the appropriate error detection bits, but the bus error status and error address registers retain the information for the initial error until all error detection bits are cleared.

As described in Section 9.2.2.1, “Machine Check (MCP),” the MPC106 asserts $\overline{\text{MCP}}$ to the 60x processor when an enabled error condition has occurred during system operation. The assertion of $\overline{\text{MCP}}$ depends upon whether the error handling registers of the MPC106 are set to report the specific error. Once asserted, the MPC106 continues to assert $\overline{\text{MCP}}$ until the MPC106 decodes a read from the processor to the machine check exception vector (0xnnn0_0200). When it decodes a processor read from the machine check exception vector, the MPC106 negates $\overline{\text{MCP}}$. However, until all the error detection bits are cleared, the MPC106 will not report subsequent errors by re-asserting $\overline{\text{MCP}}$.

In addition to the error detection bits, the MPC106 reports the assertion of NMI to the 60x processor by asserting $\overline{\text{MCP}}$ (if enabled). Note that NMI assertion is not recorded in the MPC106's error detection bits. Reporting NMI assertion (by $\overline{\text{MCP}}$) can be masked by any error detection bits that are set.

9.3.1 60x Processor Interface

The 60x processor interface of the MPC106 detects unsupported 60x bus transaction errors, illegal L2 copyback errors, and Flash write errors. In these cases, 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 MPC106 asserts either \overline{TEA} or \overline{TA} (depending on the value of PICR1[TEA_EN]) to terminate the data tenure.

9.3.1.1 Unsupported 60x Bus Transaction Error

When an unsupported 60x bus transaction error occurs, ErrDR1[1–0] is set to reflect the error type. Unsupported 60x bus transactions include \overline{XATS} -initiated transactions, writes to the PCI interrupt-acknowledge space (0xBFFF_FFFn using address map A or 0xFEFn_nnnn using address map B), and transactions with unsupported transfer attributes. Unsupported transfer attributes include the illegal and reserved transfer types defined in Section 4.3.2.1, “Transfer Type Signal Encodings.”

9.3.1.2 Illegal L2 Copyback Error

The MPC106 does not support L2 cache copyback operations to the PCI address space or to the system ROM space. If the L2 attempts a copyback operation to one of these address spaces, ErrDR2[5] is set.

9.3.1.3 Flash Write Error

The MPC106 allows data bus width writes to the system ROM space when PICR1[FLASH_WR_EN] is set and PICR2[FLASH_WR_LOCKOUT] is cleared. Otherwise, any 60x processor write transaction to the system ROM space results in a Flash write error. When a Flash write error occurs, ErrDR2[0] is set.

The MPC106 accommodates only single-beat, data path sized (8- or 64-bit depending on the configuration) writes to Flash memory. Software must partition larger data into individual data path sized (8- or 64-bit) write operations. Attempts to write to Flash with a data size other than the full data path size will cause a Flash write error (ErrDR2[0] is set).

9.3.2 Memory Interface

The memory interface of the MPC106 detects read parity, ECC, memory select, and refresh overflow errors. The MPC106 detects parity errors on the data bus during DRAM/EDO read cycles or during L2 cache read cycles. When ECC is enabled, the memory controller can detect single-bit and multibit errors for system memory read transactions. Since the ECC logic corrects single-bit errors, they are reported only when the number of errors in the ECC single-bit error counter register equals the threshold value in the ECC single-bit error trigger register. A memory select error occurs when the address for a system memory

transaction falls outside of the physical memory boundaries. A refresh overflow error occurs when there is no refresh transaction within a period that is equivalent to 16 refresh cycles.

In all cases, if the memory transaction was initiated by a PCI master, ErrDR1[3] is set; if the memory 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. If the ECC single-bit error trigger threshold is reached, then the error address will indicate the address of the most recent ECC single-bit error. When a parity or ECC error occurs on the last beat of a transaction and another transaction to the same page has started, ErrDR2[7] is set to indicate that the error address in the 60x/PCI error address register is not valid. Note that for L2 data parity errors and refresh overflow errors, the MPC106 cannot provide the error address and the corresponding bus status. In these cases, ErrDR2[7] is set to indicate that the error address in the 60x/PCI error address register is not valid.

If the 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.

9.3.2.1 System Memory Read Data Parity Error

When MCCR1[PCKEN] is set, the MPC106 checks memory parity on every memory read cycle and generates the parity on every memory write cycle that emanates from the MPC106. When a read parity error occurs, ErrDR1[2] is set.

The MPC106 does not check parity for transactions in the system ROM address space. Note that the processor should not check parity for system ROM space transactions as the parity data will be incorrect for these accesses.

9.3.2.2 L2 Cache Read Data Parity Error

When ErrEnR2[4] and MCCR1[PCKEN] are set, the MPC106 checks L2 cache parity on every L2 cache read cycle that is not in the system ROM address space. This allows ROM/Flash data to be cached in the L2. See Section 6.6.1, “ROM/Flash Cacheability,” for more information. When an L2 cache read parity error occurs, ErrDR2[4] is set. Note that the processor should not check parity for system ROM space transactions as the parity data will be incorrect for these accesses.

9.3.2.3 System Memory ECC Error

When MCCR2[ECC_EN] is set, the MPC106 performs an ECC check on every memory read cycle and generates the ECC check data on every memory write cycle. When a single-bit ECC error occurs, the ECC single-bit error counter register is incremented by 1 and its value is compared to the value in the ECC single-bit error trigger register. If the

values are equal, ErrDR1[2] is set. In addition to single-bit errors, the MPC106 detects all 2-bit errors, all errors within a nibble (one-half byte), and any other multibit error that does not alias to either a single-bit error or no error. When a multibit ECC error occurs, ErrDR2[3] is set.

9.3.2.3.1 External ECM Errors

When configured to use an external ECM (see Section 6.5.9, “External Error Checking Module (ECM) Support,” for more information), the MPC106 cannot report ECC single-bit errors—these are handled externally by the ECM. Write parity errors are reported in ErrDR1[2] (memory read parity error/ECC single-bit error exceeded). Read parity/multibit ECC errors are reported in ErrDR2[3] (ECC multibit error). The MPC106 will assert $\overline{\text{MCP}}$ (provided PICR1[MCP_EN] = 1) if either of these flags are set.

9.3.2.4 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, the MPC106 returns 0xFFFF_FFFF (all 1s). No $\overline{\text{RAS}}$ signals are asserted in either case.

9.3.2.5 System Memory Refresh Overflow Error

When there are no refresh transactions for a period equal to 16 refresh cycles, the MPC106 reports the error as a refresh overflow. When the MPC106 detects a refresh overflow, ErrDR1[4] is set.

9.3.3 PCI Interface

The MPC106 supports the error detection and reporting mechanism as specified in the *PCI Local Bus Specification*, Revision 2.1. The MPC106 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 MPC106 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 MPC106 is acting as a PCI master, and the target detects and reports (by asserting $\overline{\text{SERR}}$) a PCI address parity error, then the MPC106 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. Note that for the MPC106 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.

If the MPC106 is acting as a PCI target and detects a PCI address parity error, the PCI interface of the MPC106 sets the status bit in the PCI status register (bit 15). If bits 8 and 6 of the PCI command register are set, the MPC106 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, if PICR1[MCP_EN] is set, the MPC106 reports the error to the 60x processor by asserting $\overline{\text{MCP}}$.

9.3.3.2 Data Parity Error

If the MPC106 is acting as a PCI master and a data parity error occurs, the MPC106 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 MPC106 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 MPC106 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 MPC106 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 MPC106 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 MPC106 asserts $\overline{\text{MCP}}$ to report the error to the 60x processor. These actions also occur if the MPC106 is the master and detects the assertion of $\overline{\text{PERR}}$ by the target (for a write).

If the MPC106 is acting as a PCI target when the data parity error occurs (on a write), the MPC106 asserts $\overline{\text{PERR}}$, and sets ErrDR1[6] (PCI target $\overline{\text{PERR}}$). If the data had been transferred, the MPC106 completes the operation but discards the data. Also, if PICR1[MCP_EN] is set, the MPC106 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 address of the transfer will be logged in the error address register and $\overline{\text{MCP}}$ is optionally asserted.

9.3.3.3 Master-Abort Transaction Termination

If the MPC106, acting as a master, initiates a PCI bus transaction (excluding special-cycle 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 MPC106 terminates the transaction with a master-abort and sets the master-abort flag (bit 13) in the PCI status register. Special-cycle transactions are normally terminated with a master-abort, but these terminations do not set the master-abort flag in the PCI status register.

If ErrEnR1[1] and PICR1[MCP_EN] are both set and the MPC106 terminates a transaction with a master-abort, the MPC106 reports the error to the 60x processor by asserting $\overline{\text{MCP}}$ and $\overline{\text{TEA}}$.

In some system designs, attempts to clear the master-abort flag in the PCI status register may not succeed. The following procedure has been found to reliably clear the master-abort flag in most systems:

1. Disable $\overline{\text{MCP}}$ (PICR1[MCP_EN] = 0b0)
2. Disable master-abort error detection (ErrEnR1[1] = 0b0)
3. Write 0xFF to ErrDR1 (at offset 0xC1)
4. Write 0xFF to ErrDR2 (at offset 0xC5)
5. Write 0xFFFF to the PCI status register (at offset 0x06)
6. Read from the PCI status register (at offset 0x06)
7. If the value returned from the PCI status register is not 0x0080, then repeat from step 3; if the value returned from the PCI status register is 0x0080, then continue to step 8.
8. Enable master-abort error detection (ErrEnR1[1] = 0b1)
9. Enable $\overline{\text{MCP}}$ (PICR1[MCP_EN] = 0b1)

Note that the MPC106 may reassert $\overline{\text{MCP}}$ (possibly causing a checkstop) if software fails to clear the master-abort flag when master-abort error detection is enabled. The procedure above avoids this problem by disabling $\overline{\text{MCP}}$ and master-abort error detection before attempting to clear the master-abort flag and re-enabling master-abort error detection and $\overline{\text{MCP}}$ after the flag has been successfully cleared.

9.3.3.4 Received Target-Abort Error

If a PCI transaction initiated by the MPC106 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 MPC106 receives a target-abort, the MPC106 reports the error to the 60x processor by asserting $\overline{\text{MCP}}$ and $\overline{\text{TEA}}$.

Note that any data transferred in a target-aborted transaction may be corrupt.

9.3.3.5 NMI (Nonmaskable Interrupt)

If PICR1[MCP_EN] is set and a PCI agent (typically the system interrupt controller) asserts the NMI signal to the MPC106, the MPC106 reports the error to the 60x processor by asserting MCP.

When the NMI signal is asserted, no error flags are set in the status registers of the MPC106. 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 MPC106 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 MPC106, and an interrupt controller on the PCI bus. Typically the interrupt controller is integrated into the PCI-to-ISA bridge. Figure 9-2 shows a MPC603 microprocessor- or MPC604 microprocessor-based system design. Figure 9-3 shows a MPC601 microprocessor-based system design.

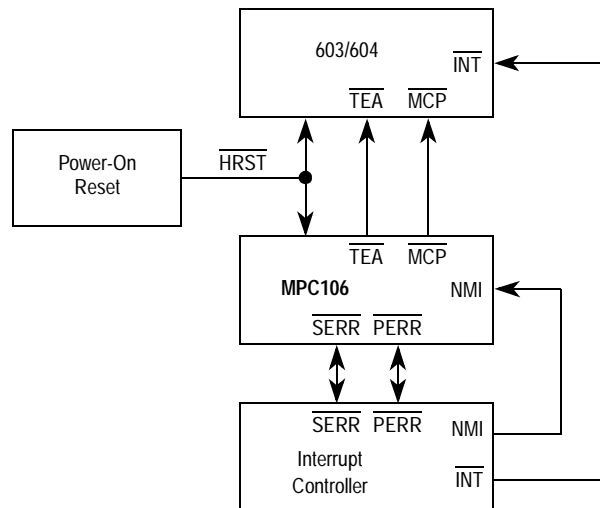


Figure 9-2. Example Interrupt Signal Configuration—603-/604-/740-/750-Based Systems

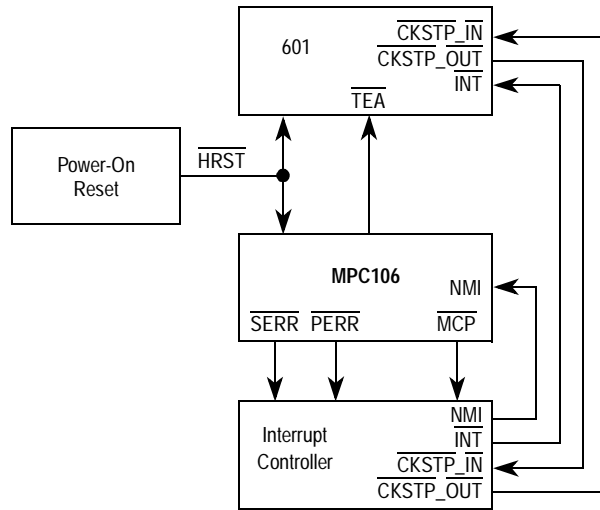


Figure 9-3. Example Interrupt Signal Configuration—601-Based Systems



Chapter 10

Performance Monitor

The MPC106 includes a performance monitor facility that allows it to record/monitor selected system behaviors. Four 32-bit performance monitor counters (PMC0, PMC1, PMC2, and PMC3) in the MPC106 count the occurrence of software-selectable events. The benefits of an on-board performance monitor are numerous, and include the following:

- Since some systems or software environments are not easily characterized by signal traces or benchmarks, the performance monitor can be used to understand the MPC106's behavior in any system or software environment.
- The performance monitor facility can be used to aid system developers when bringing up and debugging systems.
- In multiple processor (MP) systems, system performance can be increased by monitoring memory hierarchy behavior. This can help to optimize algorithms used to schedule or partition tasks and to refine the data structures and distribution used by each task.

The performance monitor command register (CMDR) and the performance monitor mode control register (MMCR) control the operation of all four counters. See Section 3.2.4, "Performance Monitor Registers," for more information.

10.1 Performance Monitor Operation

Using the configuration registers discussed in Section 3.2.4, "Performance Monitor Registers," the four performance monitor counters can be used to count the occurrences of specific events. Up to four different events can be counted simultaneously using the four different counters. The CMDR is used to select a specific counter and the events to be counted. The MMCR is used to control the behavior of the counters.

There is only one CMDR to set up all four counters. The CMDR needs to be programmed once for each counter. The CMDR[COUNTER] parameter specifies which counter is being configured. If all four counters need to start at the same time, each counter is configured by writing to the CMDR, and then counting is enabled by setting MMCR[ENABLE].

CMDR[THRESHOLD_U] and CMDR[THRESHOLD_L] are used to specify the threshold value for PMC0 and PMC1 threshold events. PMC2 and PMC3 do not support

threshold events, so a threshold value is meaningless for these counters. See Section 10.2.3, “Threshold Events,” for more information.

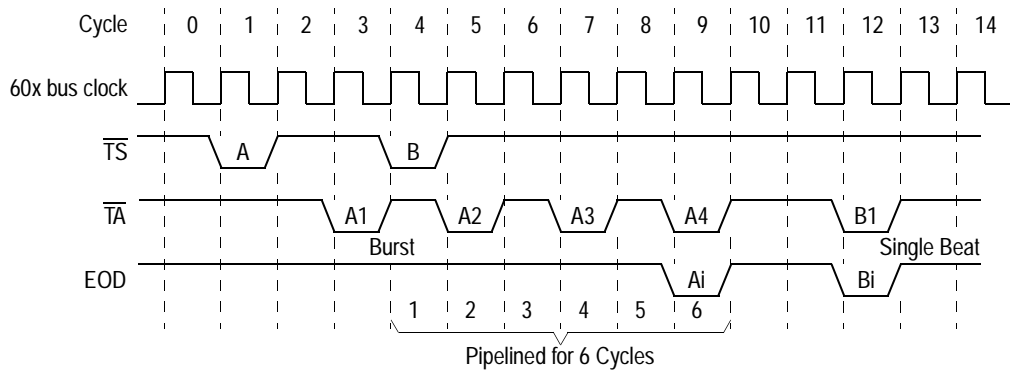
MMCR[OVFLOW 0_1] can be used to link PMC0 to PMC1 effectively turning these two 32-bit counters into one 64-bit counter. Similarly, MMCR[OVFLOW 2_3] can be used to link PMC2 to PMC3. See Section 10.2.5, “Counter Overflow,” for more information.

10.2 Performance Monitor Events

There are two types of events that can be counted—command type 0 and command type 1. The command type is selected by the CMDR[CMD_TYPE] parameter. The eight bit event parameter, composed of CMDR[EVENT_U] || CMDR[EVENT_L], specifies which events are counted. Depending on the command type selected, the meaning of the event parameter varies.

Before describing performance monitor events, it is important to note how some terms are defined. The following terms are used throughout this section:

End-of-data (EOD) EOD is the clock cycle of the last data transfer for a transaction. For burst transactions, EOD occurs on the fourth assertion of \overline{TA} . For single-beat transactions EOD occurs on the first (and only) assertion of \overline{TA} . Figure 10-1 shows example of EOD for both burst and single-beat transactions.



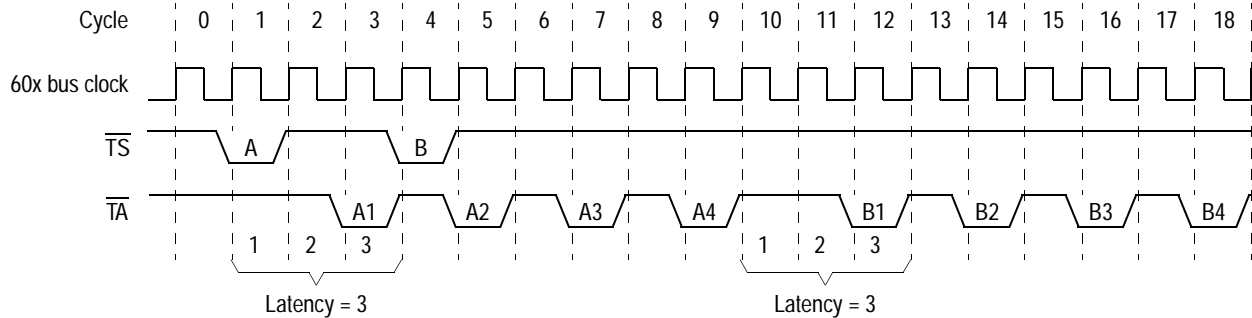
Transaction A is a burst transaction, therefore the EOD occurs on the fourth \overline{TA} .
 Transaction B is a single beat transaction, so the EOD occurs at the first and only \overline{TA} .
 (Note that transaction B is a pipelined transaction, because the \overline{TS} of transaction B comes before the EOD of the previous transaction, A.)

Figure 10-1. Pipelined Processor Transaction and End-of-Data (EOD)

Pipelined For processor transactions, pipelining occurs when \overline{TS} is asserted before the previous transaction’s EOD. Figure 10-1 shows an example of a pipelined processor transaction.

For memory transactions, pipelining occurs when the MPC106 internally starts a system memory transaction before the previous memory transaction has completed.

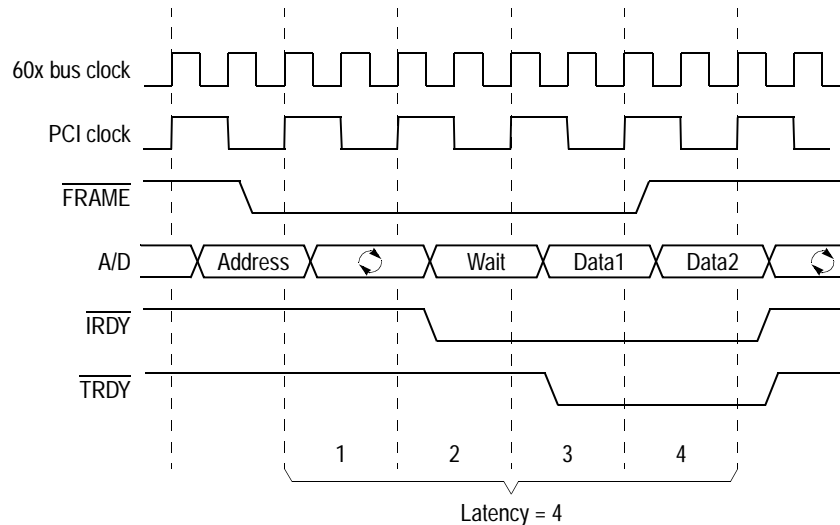
Processor latency Processor latency is the number of clock cycles between, and including, the assertion of \overline{TS} and the first assertion of \overline{TA} . For pipelined transactions, latency is the number of clock cycles between, and including, the clock cycle following the previous transaction's EOD and the first assertion of \overline{TA} ; refer to Figure 10-2.



Transaction A shows a non-pipelined transaction. Transaction B shows a pipelined transaction.

Figure 10-2. Processor Latency

PCI latency PCI latency is the number of PCI clock cycles between and including, the assertion of \overline{FRAME} and the first PCI clock cycle that data is valid (\overline{IRDY} and \overline{TRDY} asserted); refer to Figure 10-3.



Latency is counted starting with the cycle \overline{FRAME} is asserted and counted until the first data is valid.

Figure 10-3. PCI Latency

10.2.1 Command Type 0 Events

Command type 0 events are specified with `CMDR[CMD_TYPE] = 0`. Command type 0 events can only be counted in PMC0 and PMC1. (PMC2 and PMC3 cannot be used because all command type 0 events are threshold events—see Section 10.2.3, “Threshold Events,” for more information.) All command type 0 events are processor transactions.

There are 135 valid command type 0 events. Each of these events is defined by the transaction type, destination, and transaction size bits shown in Table 10-1. There are three possible transaction types (read, write, and read/write), fifteen possible destinations (L2, system memory, PCI, and/or ROM), and three possible transaction sizes (burst, single-beat, and burst/single-beat). Every command type 0 event must have at least one transaction type, one destination, and one transaction size defined. Note that processor snoop copybacks and retried processor transactions are not counted.

Table 10-1. Command Type 0—Processor Transactions

CMDR Bit		Name	Description
EVENT_U	4	READ	Read transaction type bit. Enables counting of read transactions. 0 Disable counting of read transactions 1 Enable counting of read transactions
	3	WRITE	Write transaction type bit. Enables counting of write transactions. 0 Disable counting of write transactions 1 Enable counting of write transactions
	2	L2	L2 destination bit. Enables counting of transactions with the L2 as the target. 0 Disable counting events targeted at the L2 1 Enable counting events targeted at the L2 Note that writes to the L2, in write-through cache mode, will only be counted as transactions to memory.
	1	MEMORY	Memory destination bit. Enables counting of transactions with system memory as the target. 0 Disable counting events targeted at system memory 1 Enable counting events targeted at system memory
	0	PCI	PCI destination bit. Enables counting of transactions with PCI as the target. 0 Disable counting events targeted at PCI 1 Enable counting events targeted at PCI
EVENT_L	15	ROM	ROM destination bit. Enables counting of transactions with ROM as the target. 0 Disable counting events targeted at ROM 1 Enable counting events targeted at ROM
	14	BURST	Burst transaction size bit. Enables counting of burst transactions. 0 Disable counting of burst transactions 1 Enable counting of burst transactions
	13	SINGLE_BT	Single-beat transaction size bit. Enables counting of single beat transactions. 0 Disable counting of single beat transactions 1 Enable counting of single beat transactions

10.2.2 Command Type 1 Events

Command type 1 events are specified with `CMDR[CMD_TYPE] = 1`. Command type 1 events are defined in Table 10-2. Most command type 1 events can be counted by all four counters. However, as specified in Table 10-2, some events can only be counted in PMC0 and PMC1, and others are only countable in PMC2 and PMC3. Note that all events are counted in 60x bus cycles, except PCI events which are counted in PCI cycles.

Table 10-2. Command Type 1—Event Encodings

Event (decimal)	Event (hex)	Counter PMC n	Description
Processor Transactions			
0	00	0, 1, 2, or 3	Counter holds current value
1	01	0, 1, 2, or 3	Number of 60x cycles
2	02	0 or 1	TA overlap. Number of cycles from the \overline{TS} to the last \overline{TA} of the previous transaction in pipelined situations. This is the period of time marked 'Pipelined for 6 cycles' in Figure 10-1. Note that the counter will increment only if the number of \overline{TA} overlap cycles is greater than the threshold value.
3	03	0, 1, 2, or 3	Cache-inhibited transactions that are not retried
4	04	—	Reserved
5	05	0, 1, 2, or 3	L2 castouts with no retry
6	06	0, 1, 2, or 3	Total L2 castouts
7	07	0, 1, 2, or 3	sync and eieio instructions that are not retried
8	08	0, 1, 2, or 3	Address-only transactions (sync , eieio , Kill, icbi , Clean, Flush, tlbi , lwarx , stwcx , tlbsync) that are not retried
9	09	0 or 1	Number of cycles between $\overline{BR0}$ and $\overline{BG0}$ (qualified). Note that the counter will increment only if the number of cycles from a bus request until a bus grant is qualified, including the cycles the signals are asserted, is greater than the threshold value.
10	0A	0 or 1	Number of cycles between $\overline{BR1}$ and $\overline{BG1}$ (qualified). Note that the counter will increment only if the number of cycles from a bus request until a bus grant is qualified, including the cycles the signals are asserted, is greater than the threshold value.
11	0B	0, 1, 2, or 3	Number of cycles that the 60x address bus is busy (includes all address phases)
12	0C	0, 1, 2, or 3	Number of cycles that the 60x data bus is busy (includes all data transfers)
13	0D	0, 1, 2, or 3	Number of retries issued by the MPC106 on the 60x bus
14	0E	0, 1, 2, or 3	Number of retries issued by an alternate master (not including the L2) on the 60x bus
15	0F	0, 1, 2, or 3	Number of retries issued to the MPC106 on the 60x bus
16	10	0, 1, 2, or 3	Number of cycles that the memory interface is busy reading from ROM
17	11	0, 1, 2, or 3	Number of cycles that the memory interface is busy reading from system memory
18	12	0, 1, 2, or 3	Number of cycles that the memory interface is busy performing reads and writes (from both system memory and ROM)

Table 10-2. Command Type 1—Event Encodings (continued)

Event (decimal)	Event (hex)	Counter PMC _n	Description
19	13	0 or 1	Number of cycles that an L2 castout or a processor burst write with ECC enabled has to wait for the L2 castout buffer Note that the counter will increment only if the number of cycles an L2 castout or a processor burst write with ECC enabled has to wait for the L2 castout buffer is greater than the threshold value.
20	14	0 or 1	Number of cycles that the processor waits while a PCI transaction is occurring Note that the counter will increment if the number of cycles the processor waits while a PCI transaction is occurring is greater than the threshold value.
21	15	0 or 1	Burstiness for processor transactions. PMC0 will count the number of times there are X transactions in a row with an acceptable latency of Y cycles. Note that this event involves PMC0 and PMC1. See Section 10.2.4, "Burstiness," for more information.
PCI Transactions			
32	20	0, 1, 2, or 3	Number of PCI cycles
33	21	0 or 1	Number of PCI read from memory commands (memory-read, memory-read-line, and memory-read-multiple) Note that the counter will increment if the number of cycles from the start of the transaction until the first data is greater than the threshold value.
34	22	0, 1, 2, or 3	Number of PCI read and write to memory commands (memory-read, memory-read-line, and memory-read-multiple, memory-write and memory-write-invalidate)
35	23	0, 1, 2, or 3	Number of data beats read by external PCI master
36	24	0, 1, 2, or 3	Number of data beats read and written by external PCI master
37	25	0, 1, 2, or 3	Number of PCI memory-read-line commands
38	26	0, 1, 2, or 3	Number of PCI memory-read-multiple commands
39	27	0, 1, 2, or 3	Number of PCI reads from ROM space
40	28	0, 1, 2, or 3	Number of PCI memory-write-and-invalidate commands
41	29	0, 1, 2, or 3	Number of speculative PCI read snoops
42	2A	0, 1, 2, or 3	Number of PCI read and speculative PCI read snoops
43	2B	0, 1, 2, or 3	Number of PCI write, PCI read, and speculative PCI read snoops
44	2C	0, 1, 2, or 3	Number of PCI reads that hit in the PCI-to system-memory-read-buffer (PCMRB) while the MPC106 is still trying to fill it after a disconnect. When a PCI read is attempted but the latency exceeds 32 PCI cycles, the MPC106 must disconnect as required by the PCI specification. The MPC106 assumes that the PCI master will return with a request to the same address and continues the line fill from system memory. Event 44 occurs if the PCI master issues a read command to the same cache line, while the line fill is still in progress

Table 10-2. Command Type 1—Event Encodings (continued)

Event (decimal)	Event (hex)	Counter PMC n	Description
45	2D	0, 1, 2, or 3	Number of PCI reads that hit in the PCMRB while the MPC106 is still busy performing a speculative fetch of that line
46	2E	0, 1, 2, or 3	Number of PCI reads that hit in the PCMRB
47	2F	0, 1, 2, or 3	Number of PCI reads that hit modified cache lines in a processor's L1 cache
48	30	0, 1, 2, or 3	Number of PCI reads and PCI writes that hit modified cache lines in a processor's L1 cache
49	31	0, 1, 2, or 3	Number of PCI transactions that disconnected at the end of a cache line
50–59	32–3B	—	Reserved
60	3C	0, 1, 2, or 3	Number of cycles that $\overline{\text{FRAME}}$ is asserted
61	3D	0, 1, 2, or 3	Number of cycles that $\overline{\text{IRDY}}$ is asserted
62	3E	0, 1, 2, or 3	Number of cycles that $\overline{\text{TRDY}}$ is asserted
System Memory Transactions			
80	50	2 or 3	Number of pipelined read misses to page 0
81	51	2 or 3	Number of pipelined read and write misses to page 0
82	52	2 or 3	Number of non-pipelined read misses to page 0
83	53	2 or 3	Number of non-pipelined read and write misses to page 0
84	54	2 or 3	Number of pipelined read hits to page 1—SDRAM only
85	55	2 or 3	Number of pipelined read and write misses to page 1—SDRAM only
86	56	2 or 3	Number of non-pipelined read misses to page 1—SDRAM only
87	57	2 or 3	Number of non-pipelined read and write misses to page 1—SDRAM only
88	58	2 or 3	Number of pipelined read hits to page 0
89	59	2 or 3	Number of pipelined read and write hits to page 0
90	5A	2 or 3	Number of non-pipelined read hits to page 0
91	5B	2 or 3	Number of non-pipelined read and write hits to page 0
92	5C	2 or 3	Number of pipelined read hits to page 1—SDRAM only
93	5D	2 or 3	Number of pipelined read and write hits to page 1—SDRAM only
94	5E	2 or 3	Number of non-pipelined read hits to page 1—SDRAM only
95	5F	2 or 3	Number of non-pipelined read and write hits to page 1—SDRAM only
96	60	2 or 3	Number of forced closings for page 0 (excluding refreshes)
97	61	2 or 3	Number of forced closings for page 1 (excluding refreshes)—SDRAM only
98	62	2 or 3	Total number of misses to page 0 and page 1, pipelined and non-pipelined

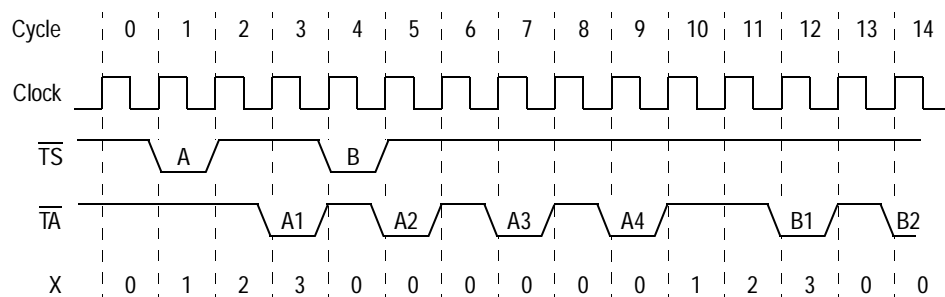
Table 10-2. Command Type 1—Event Encodings (continued)

Event (decimal)	Event (hex)	Counter PMC _n	Description
99	63	2 or 3	Total number of hits to page 0 and page 1, pipelined and non-pipelined
100	64	2 or 3	Total number of forced closings for page 0 and page 1

10.2.3 Threshold Events

The intent of threshold support is to be able to characterize events that can take a variable number of cycles to occur. Threshold events are only counted if the latency is greater than the threshold value (CMDR[THRESHOLD_U] || CMDR[THRESHOLD_L]). Figure 10-4 shows an example of a command type 0 event with threshold.

Note that PMC0 and PMC1 are the only counters that support counting threshold events. PMC2 and PMC3 do not support threshold events



These two transactions (a non-pipelined transaction 'A' and a pipelined transaction 'B') both have a latency of 3. Therefore they will both be counted as events if the threshold value is 0, 1, or 2 (that is, $X > \text{CMDR}[\text{THRESHOLD_U}] \parallel \text{CMDR}[\text{THRESHOLD_L}]$).

Figure 10-4. Command Type 0 Threshold Example

There are seven command type 1 threshold events (events 2, 9, 10, 19, 20, 21, and 33). The event descriptions in Table 10-2 describe how threshold is used for those events.

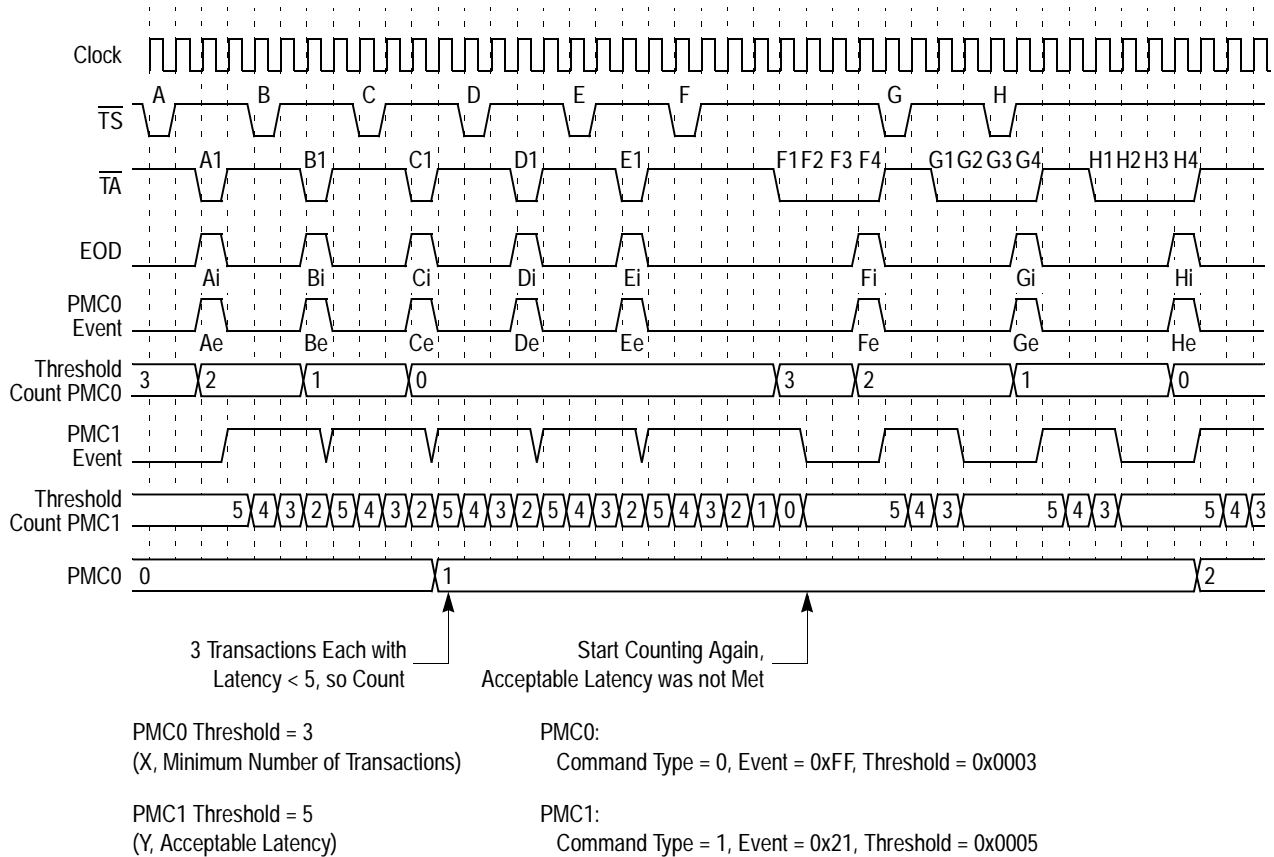
10.2.4 Burstiness

Burstiness is a special command type 1 event that involves both PMC0 and PMC1. When configured correctly, it enables PMC0 to count the number of times there are at least X transactions in a row with an acceptable latency of Y clock cycles between any two transactions. For example, burstiness can count the number of times there are 1,000 writes to PCI in a row, each with less than or equal to 9 cycles of latency between them.

To count burstiness, PMC0 is configured to count the command type 0 processor transaction(s) of interest (see Table 10-1). The threshold of PMC0 is set to be the minimum number of consecutive transactions desired (referred to as X above). PMC1 is configured to count command type 1 event 21 (see Table 10-2). The threshold of PMC1 is set to the

acceptable latency desired (referred to previously as Y). Note that the latency to the first transaction is not considered.

Figure 10-5 shows a simple example that counts the number of times at least three processor transactions are in a row with an acceptable latency of five between them. In this example, X is 3 and Y is 5 (that is, the acceptable latency is ≤ 5 cycles). For this example, X is the threshold of PMC0 and Y is the threshold of PMC1. Since burstiness does not actually use PMC1, it may be useful to set MMCR[OVFLOW 0_1] to allow overflow counting into PMC1. PMC0 operates as a normal command type 0 event until PMC1 is configured for command type 1 event 21.



NOTE: Since X is the minimum number of transactions, even if 2,000 transactions occur each with an acceptable latency of 5, then the value in PMC0 will still only be 1. The acceptable latency has to be violated before the counter can be eligible to increment again.

Figure 10-5. Processor Burstiness Example

10.2.5 Counter Overflow

The counters can be configured as 64-bit counters by setting MMCR[OVFLOW 0_1] for PMC0 and PMC1 or MMCR[OVFLOW 2_3] for PMC2 and PMC3. Figure 10-6 shows a diagram of PMC0 and PMC1 configured to utilize overflow. Note that using counter overflow requires MMCR[DISCOUNT] = 0 and MMCR[PMCTRG] = 0.

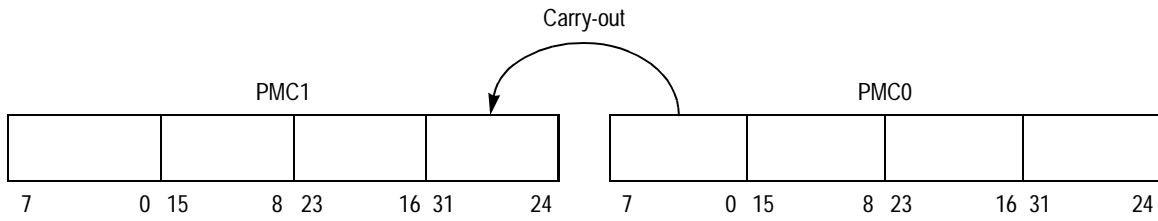


Figure 10-6. Overflow Example

Appendix A

Power Management

The MPC106 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 MPC106 for power management.

A.1 MPC106 Power Modes

The MPC106 implements four levels of power reduction—doze, nap, sleep, and suspend, with power consumption reduced with each step from doze to suspend. Refer to Section A.1.2, “Full-On Mode through Section A.1.6, “Suspend Mode for further details. 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 3.2.5, “Power Management Configuration Registers (PMCRs).” The suspend mode is entered by the assertion of the $\overline{\text{SUSPEND}}$ signal, as described in Section 2.2.7.6, “Suspend (SUSPEND)—Input.” All of the power management modes are enabled by the configuration of the global power management bit, PMCR[PM].

A.1.1 MPC106 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 MPC106 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 $\overline{\text{QREQ}}$.

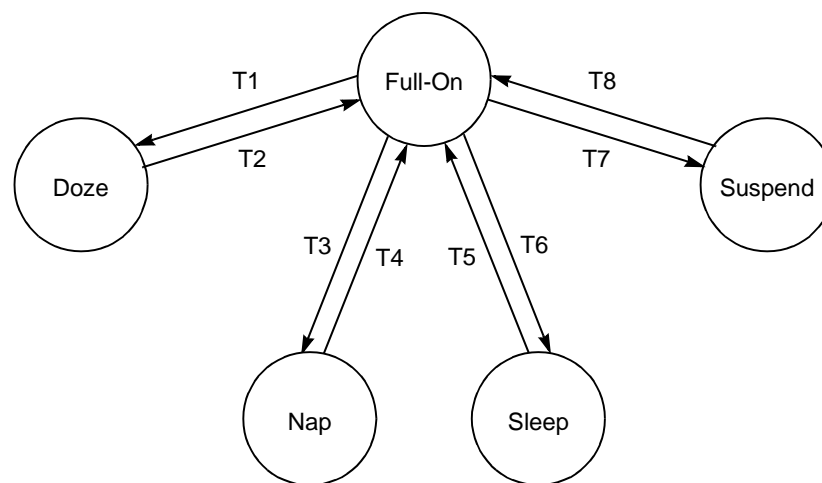
The configuration of the MPC106 and the processor signals asserted differ depending on which processor (MPC601 microprocessor, MPC603 microprocessor, MPC604 microprocessor, MPC740 microprocessor, or MPC750 microprocessor) the MPC106 is connected to. The response of the MPC106 is configured through the setting of the processor type bits in PICR1[PROC_TYPE].

In a system designed using the 601, the MPC106 can be configured to ignore the state of the \overline{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{QREQ} , and when set to 1 requires the assertion of \overline{QREQ} by system power control logic to enter the desired power management mode.

In systems designed using the 603, 740, or 750, the power control signals \overline{QREQ} and \overline{QACK} are connected to the corresponding signals on the MPC106, and the doze, nap, or sleep mode is entered following the configuration of the required bit in the PMCR, and the assertion of \overline{QREQ} (in nap and sleep modes) to the MPC106 by the 603.

In systems designed using the 604, the MPC106's \overline{QREQ} signal is connected to the 604's HALT signal, and the MPC106's \overline{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{QREQ} (in nap and sleep modes) to the MPC106 by the 604. Configuring the processor type bits in PICR1[PROC_TYPE] for the 604 causes the signal levels sampled and driven by the MPC106's \overline{QREQ} and \overline{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 MPC106, and the conditions required for entering and exiting those modes.



- T1: PMCR[DOZE] = 1 & PMCR[PM] = 1
- T2: hard reset, $\overline{BRx} = 0$, PCI address hit, NMI
- T3: PMCR[NAP] = 1 & PMCR[PM] = 1 & $\overline{QREQ} = 0$ (or HALT = 1 in 604 system)
- T4: hard reset, $\overline{BRx} = 0$, PCI address hit, NMI
- T5: PMCR[SLEEP] = 1 & PMCR[PM] = 1 & $\overline{QREQ} = 0$ (or HALT = 1 in 604 system)
- T6: hard reset, $\overline{BRx} = 0$, NMI
- T7: $\overline{suspend} = 0$ & PMCR[PM] = 1
- T8: $\overline{suspend} = 1$

Figure A-1. MPC106 Power Modes

A.1.2 Full-On Mode

This is the default power mode of the MPC106. In this mode, the MPC106 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 MPC106'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 MPC106 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 MPC106'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 MPC106's nap mode, both the MPC106 and the processor should be programmed to enable the nap mode. The processor may also be programmed to enter sleep mode while the MPC106 enters nap mode.

As in doze mode, all the MPC106'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 MPC106 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 MPC106 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 MPC106 will not automatically re-enter the nap mode. For PCI bus-initiated transactions, $PMCR[PM]$ will not be cleared, and the MPC106 will return to nap mode after the transaction has been serviced.

While the MPC106 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 MPC106. 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{QACK} signal from the MPC106) is asserted. This response by the 604 is enabled by clearing PMCR[NO_604_RUN] to 0. If the MPC106 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{QREQ} from 603 or HALT from 604 will be sampled active by the MPC106, which will then respond with a \overline{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 MPC106 and the processor should be configured to enable the sleep mode (although the processor may optionally be configured for nap mode while the MPC106 is in sleep mode). While the MPC106 is in sleep mode, no functional units are operating except the system RAM refresh logic (optional), processor bus request monitoring (through $\overline{BR0}$ or $\overline{BR1}$), and NMI signal monitoring. A hard reset, a bus request from $\overline{BR0}$, a bus request from $\overline{BR1}$ (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 MPC106 from the sleep mode. The PMCR[PM] bit will always be cleared after the MPC106 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. Note that SYSCLK must continue for at least three clock cycles after setting the PLL in bypass mode before SYSCLK can be disabled. 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 relock 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 third 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 MCCR1[SREN]. If the LP_REF_EN bit is cleared to 0, there will be no memory refresh operation when the MPC106 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 MPC106 is in a low-power mode. In this case, MCCR1[SREN] is used to determine whether the refresh is a self refresh (MCCR1[SREN] set to 1) or a CBR refresh (MCCR1[SREN] cleared to 0).

When the MPC106 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{QREQ} from 603 or HALT from 604 should be sampled active. The MPC106 will then respond with a \overline{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{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 MPC106.

A.1.6 Suspend Mode

Suspend mode provides the greatest reduction of power consumption. It is activated through the assertion of the 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 SUSPEND signal. The MPC106 will remain in the suspend mode until the 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. Note that SYSCLK must continue for at least three clock cycles after setting the PLL in bypass mode before SYSCLK can be disabled. 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 μ s).

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 MPC106. Note that RTC refresh is not supported for SDRAM configurations. However, JEDEC-compliant SDRAM devices allow self-refresh, which consumes less power than RTC refresh. 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 MCCR1[SREN]. If PMCR[LP_REF_EN] is cleared to 0, there will be no memory refresh operation when the MPC106 is in suspend mode. If PMCR[LP_REF_EN] is set to 1, memory refresh will be carried out even when the MPC106 is in suspend mode.

In this case, MCCR1[SREN] will be used to determine whether the refresh is a self refresh (MCCR1[SREN] set to 1) or a low-frequency refresh (MCCR1[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, with the exception of $\overline{\text{HRST}}$ and 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 MPC106 Power Management Support

The MPC106 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 MPC106.

A.2.1 Power Management Configuration Registers

The PMCRs provide software access to the power management modes, enables, and configurations for different processors. Refer to Section 3.2.5, “Power Management Configuration Registers (PMCRs),” 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 MPC106 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 MPC106 before MPC106 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 MPC106 is in the doze or nap mode. A PCI transaction to system memory awakens the MPC106 from the doze or nap power saving mode.

After servicing the PCI transaction, the MPC106 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 MPC106 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 Transactions," for a description of PCI special-cycle operations.

In nap mode, if PMCR[NO_NAP_MSG] is cleared to 0, the MPC106 broadcasts the halt message over the PCI bus. If PMCR[NO_NAP_MSG] is set to 1, the MPC106 does not broadcast any message to the PCI bus.

In sleep mode, if PMCR[NO_SLEEP_MSG] is cleared to 0, the MPC106 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 MPC106 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 MPC106 monitors the $\overline{BR0}$ signal. When $\overline{BR0}$ is asserted, (for example, due to the processor's time base interrupt service routine), the MPC106 exits its power saving mode and returns to the full-on mode to service the request.

Additionally, in a multiprocessor system, $\overline{BR}[1-3]$ can be used to awaken the MPC106. In nap or sleep mode, the assertion of $\overline{BR1}$, $\overline{BR2}$, or $\overline{BR3}$ 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 MPC106 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 MPC106 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 MPC106 is in the sleep or suspend mode. If PMCR[LP_REF_EN] is cleared to 0, memory refresh operations will cease when the MPC106 enters the sleep or suspend mode. For additional detail on memory refresh operations, refer to Section 6.4.10.2, "DRAM/EDO Refresh and Power Saving Modes."

A.2.7 PCI Bus Parking in Sleep and Suspend Modes

Section 3.4.3. "Arbitration Parking," of the *PCI Local Bus Specification*, Revision 2.1 states:

"When the arbiter asserts an agent's GNT# and the bus is in the Idle state, that agent must enable its AD[31::00], C/BE#[3::0], and (one clock later) PAR output buffers within eight clocks (required) . . . This requirement ensures that the arbiter can safely park the bus at some agent and know that the bus will not float."

In sleep or suspend mode, all functional units, including the PCI bus interface, do not operate. The MPC106 releases the PCI bus to high-impedance when it enters sleep or suspend modes, regardless of the state of the $\overline{\text{GNT}}$ pin.

System designs should not expect the MPC106 to sustain the AD[31-0], $\overline{\text{C/BE}}$ [3-0], and PAR pins during sleep or suspend modes. This can be accomplished by parking the bus with a PCI bus master other than the MPC106, or by passively sustaining (that is, pull-up) the AD[31-0], $\overline{\text{C/BE}}$ [3-0], and PAR signals.

A.2.8 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 MPC106 supports both big-endian and little-endian formatted data on the PCI bus. This appendix provides examples of the big- and little-endian modes of operation. PICR1[LE_MODE] controls the endian mode of the MPC106. LE_MODE is also accessible from the external configuration register at port 0x092. Note that the 60x processor and the MPC106 should be set for the same endian mode before accessing the devices on PCI bus.

When designing little-endian or big-endian systems using the MPC106, system designers and programmers must consider the following:

- The PCI bus uses a little-endian bit format (the most-significant bit (msb) 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 (LSB) 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 Chapter 3, “Operand Conventions” in the *Programming Environments Manual*, 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 MPC106 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 operating in big-endian mode, no address modification is performed by the processor. In big-endian mode, the MPC106 maintains the big-endian

byte ordering on the PCI bus during the data phase(s) of PCI transactions. The byte lane translation for big-endian mode is shown in Table B-1. Note that the bit ordering on the PCI bus remains little-endian.

Table B-1. Byte Lane Translation in Big-Endian Mode

60x Byte Lane	60x Data Bus Signals	PCI Byte Lane	PCI Address/Data Bus Signals During PCI Data Phase
0	DH[0-7]	0	AD[7-0]
1	DH[8-15]	1	AD[15-8]
2	DH[16-23]	2	AD[23-16]
3	DH[24-31]	3	AD[31-24]
4	DL[0-7]	0	AD[7-0]
5	DL[8-15]	1	AD[15-8]
6	DL[16-23]	2	AD[23-16]
7	DL[24-31]	3	AD[31-24]

Figure B-1 shows a 4-byte write to PCI memory space in big-endian mode.

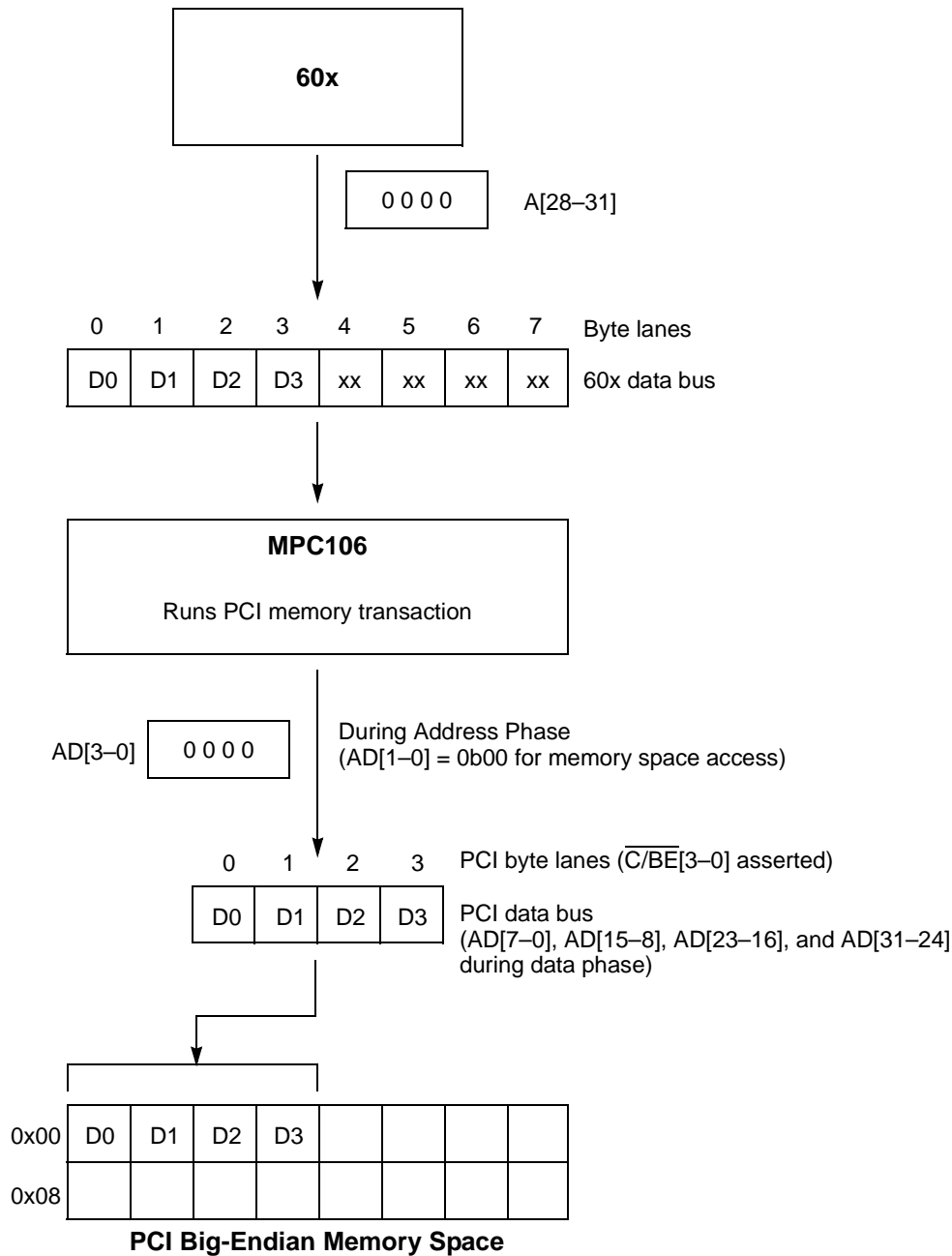


Figure B-1. Four-Byte Transfer to PCI Memory Space—Big-Endian Mode

Note that the most-significant byte on the 60x bus, D0, is placed on byte lane 0 (AD[7-0]) on the PCI bus. This occurs so that D0 appears at address 0xnnnn_nn00 and not at address 0xnnnn_nn03 in the PCI space.

The following example demonstrates the operation of a system in big-endian mode. Starting with a program that does the following:

```
store string ("hello, world") at 0x000
store pointer (0xFEDCBA98) at 0x010
```

store half word (0d1234) at 0x00E

store byte (0x55) at 0x00D

If the data is stored into system memory, it appears as shown in Figure B-2.

Contents	'h'	'e'	'l'	'l'	'o'	','	''	'w'
Address	00	01	02	03	04	05	06	07
Contents	'o'	'r'	'l'	'd'		0x55	12	34
Address	08	09	0A	0B	0C	0D	0E	0F
Contents	0xFE	0xDC	0xBA	0x98				
Address	10	11	12	13	14	15	16	17

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.

If the data is stored to the PCI memory space, it appears as shown in Figure B-3.

Contents	'l'	'l'	'e'	'h'
Address	03	02	01	00
Contents	'w'	''	','	'o'
Address	07	06	05	04
Contents	'd'	'l'	'r'	'o'
Address	0B	0A	09	08
Contents	34	12	0x55	0x00
Address	0F	0E	0D	0C
Contents	0x98	0xBA	0xDC	0xFE
Address	13	12	11	10
Contents				
Address	17	16	15	14

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 is stored to the desired location 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-2.

Table B-2. Processor Address Modification for Individual Aligned Scalars

Data Length (in Bytes)	Address Modification A[29–31]
8	No change
4	XOR with 0b100
2	XOR with 0b110
1	XOR with 0b111

Note that munging makes it appear to the processor that individually aligned scalars are stored in little-endian byte order, when in fact, they are stored in big-endian order, but at different byte addresses within double words. Only the address is modified, not the byte order.

The munged address is used by the memory interface of the MPC106 to access system memory. To provide true little-endian byte-ordering to the PCI bus, the MPC106 unmunges the address to its original value and the byte lanes are reversed. The MPC106 unmunges aligned addresses by exclusive-ORing the three low-order address bits with a three-bit value that depends on the length of the operand (1, 2, 3, 4, or 8 bytes), as shown in Table B-3.

Table B-3. MPC106 Address Modification for Individual Aligned Scalars

Data Length (in Bytes)	Address Modification A[29–31]
8	No change
4	XOR with 0b100
3	XOR with 0b101
2	XOR with 0b110
1	XOR with 0b111

The MPC106 also supports misaligned 2-byte transfers that do not cross word boundaries in little-endian mode. The MPC106 exclusive-ORs the address with 0x100. Note that the

MPC106 does not support 2-byte transfers that cross word boundaries in little-endian mode.

The byte lane translation for little-endian mode is shown in Table B-4.

Table B-4. Byte Lane Translation in Little-Endian Mode

60x Byte Lane	60x Data Bus Signals	PCI Byte Lane	PCI Address/Data Bus Signals During PCI Data Phase
0	DH[0-7]	3	AD[31-24]
1	DH[8-15]	2	AD[23-16]
2	DH[16-23]	1	AD[15-8]
3	DH[24-31]	0	AD[7-0]
4	DL[0-7]	3	AD[31-24]
5	DL[8-15]	2	AD[23-16]
6	DL[16-23]	1	AD[15-8]
7	DL[24-31]	0	AD[7-0]

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 in little-endian mode, the MPC106 stores the data to the 60x-munged addresses as shown in Figure B-4.

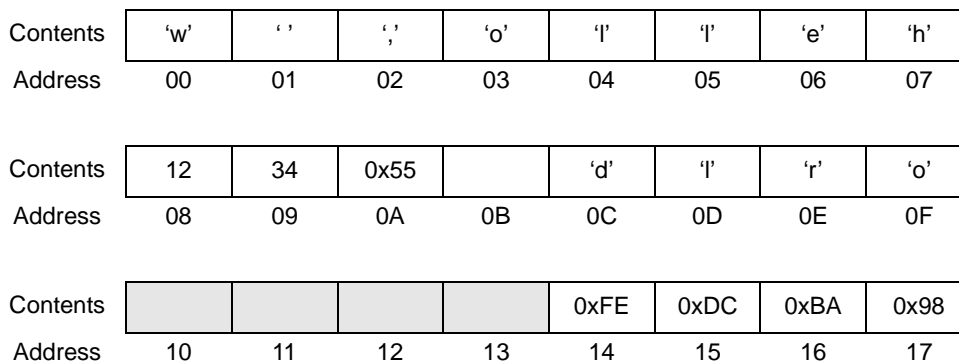


Figure B-4. Munged Memory Image in System Memory

The image shown in Figure B-4 is not a true little-endian mapping. Note how munging has changed the addresses of the data. The 'h' is now at address 0x007. Also note that the byte ordering of the word-length pointer, 0xFEDCBA98, is big-endian. Only the address has

been modified. However, since the 60x processor munges the address when accessing system memory, the mapping appears little-endian to the processor.

If the data is stored to the PCI memory space, or if a PCI agent reads from system memory, the MPC106 unmunges the addresses and reverses the byte-ordering before sending the data out to the PCI bus. The data is stored to little-endian PCI memory space as shown in Figure B-5.

Contents	'l'	'l'	'e'	'h'
Address	03	02	01	00
Contents	'w'	' '	','	'o'
Address	07	06	05	04
Contents	'd'	'l'	'r'	'o'
Address	0B	0A	09	08
Contents	12	34	0x55	0x00
Address	0F	0E	0D	0C
Contents	0xFE	0xDC	0xBA	0x98
Address	13	12	11	10
Contents				
Address	17	16	15	14

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 is stored to the desired location with true 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.

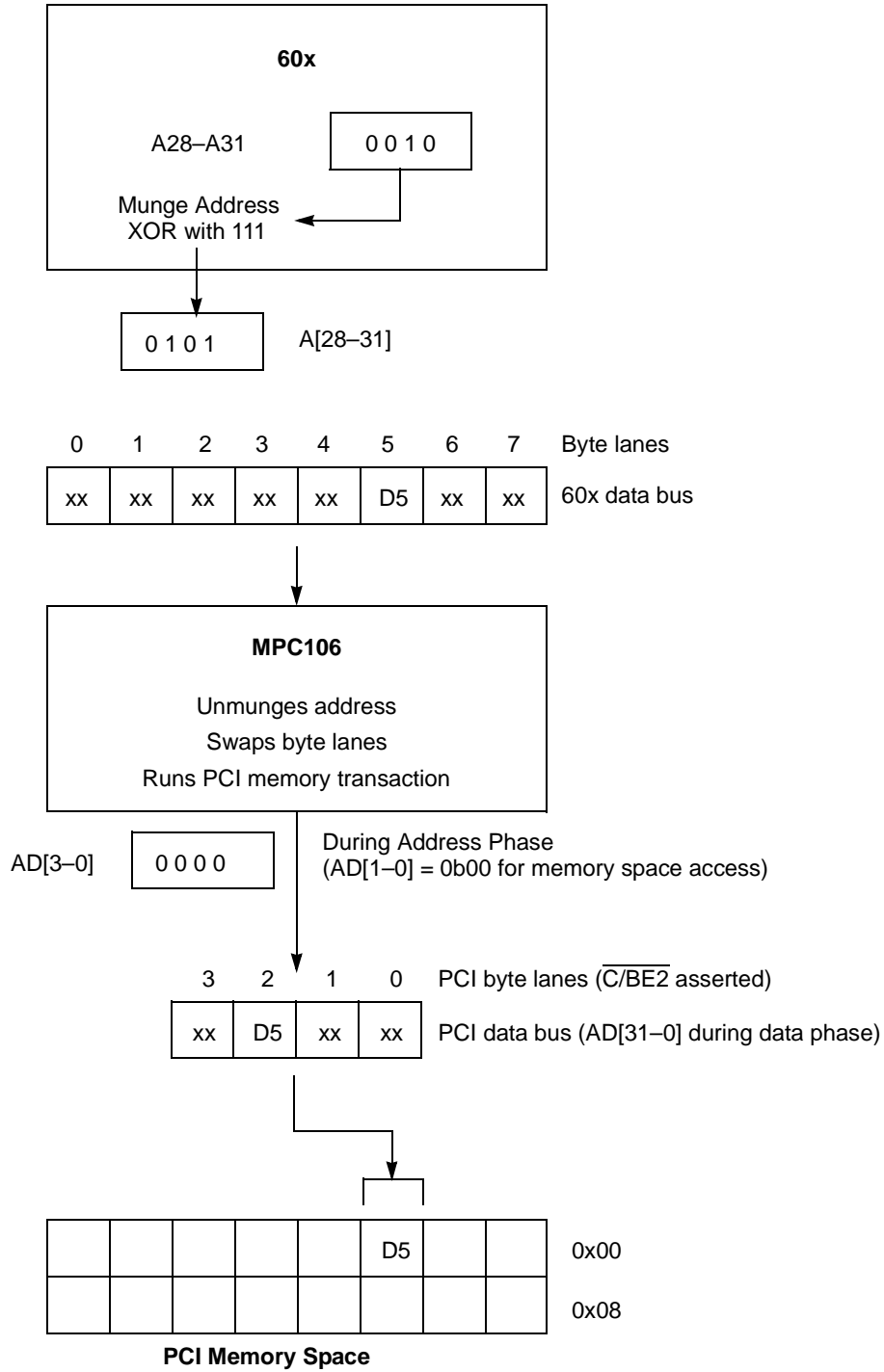


Figure B-6. One-Byte Transfer to PCI Memory Space—Little-Endian Mode

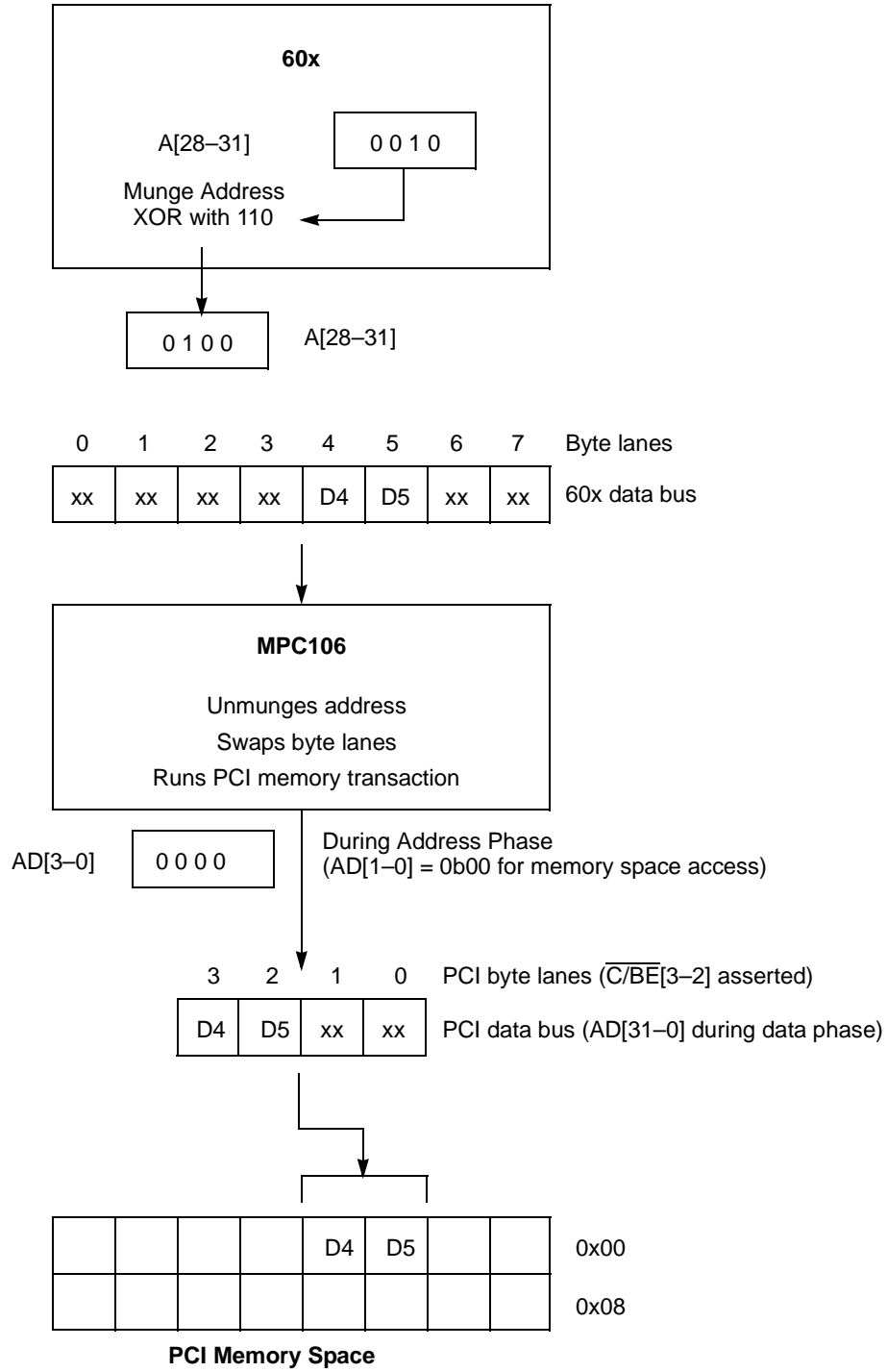


Figure B-7. Two-Byte Transfer to PCI Memory Space—Little-Endian Mode

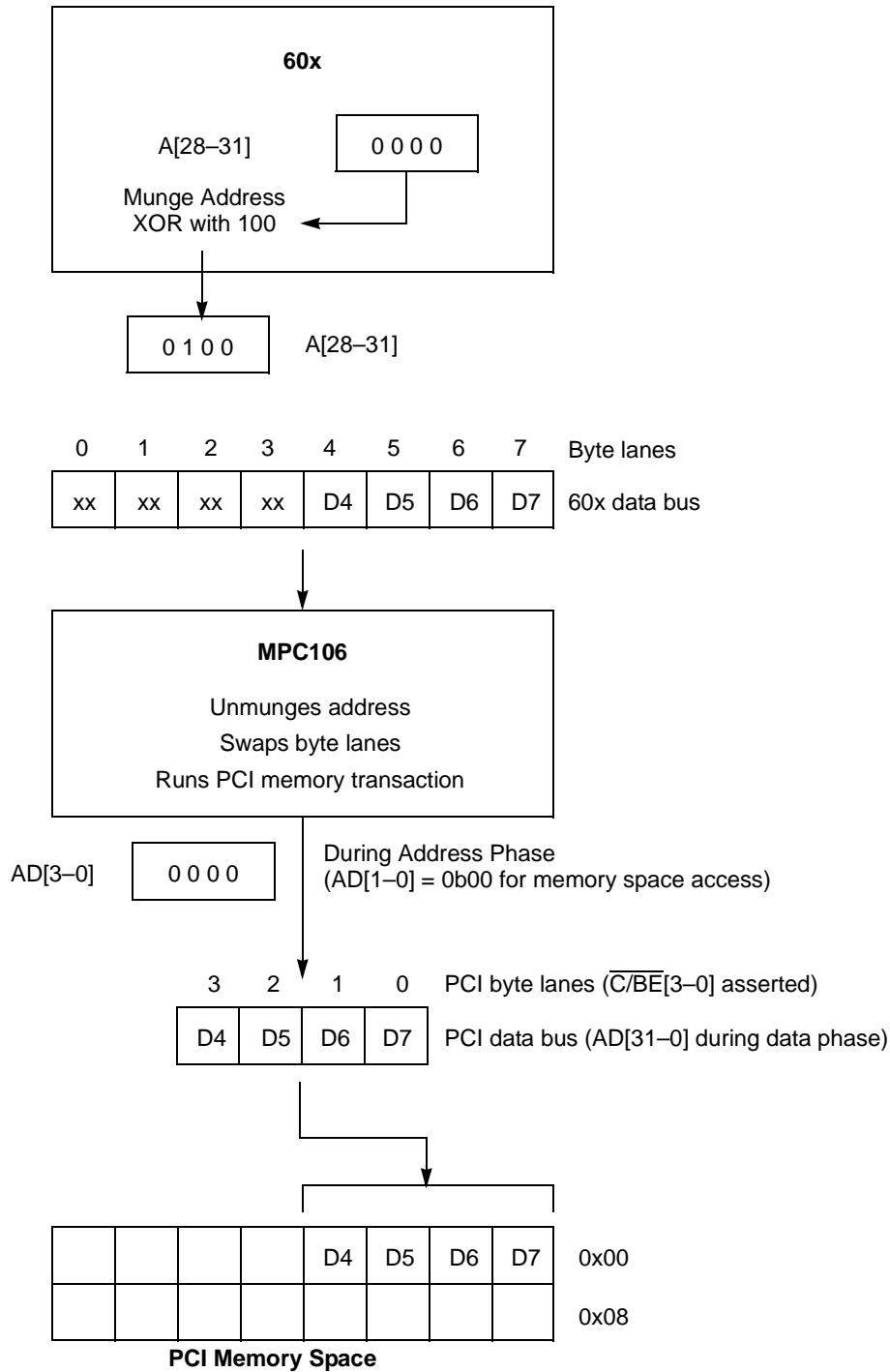


Figure B-8. Four-Byte Transfer to PCI Memory Space—Little-Endian Mode

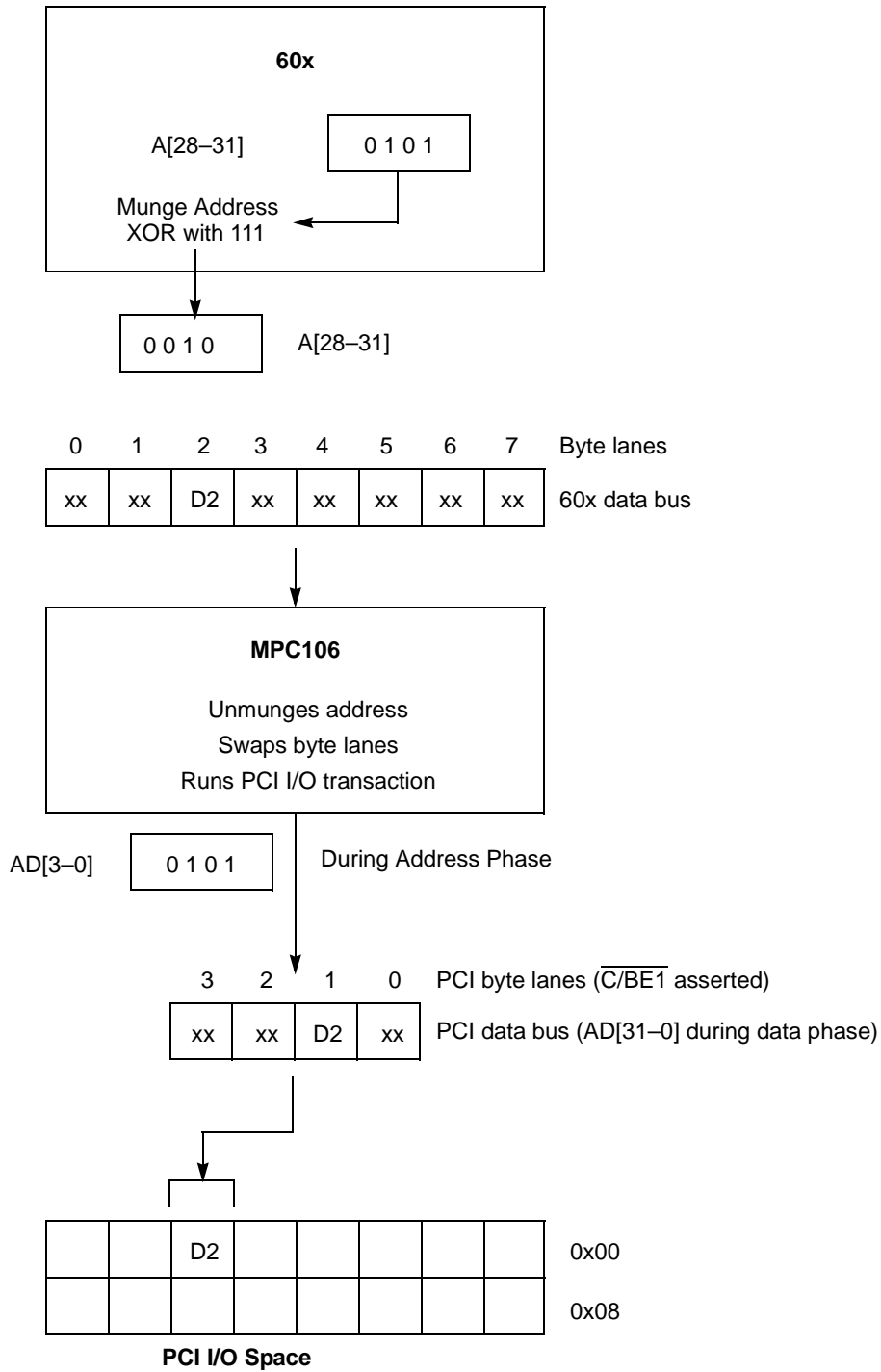


Figure B-9. One-Byte Transfer to PCI I/O Space—Little-Endian Mode

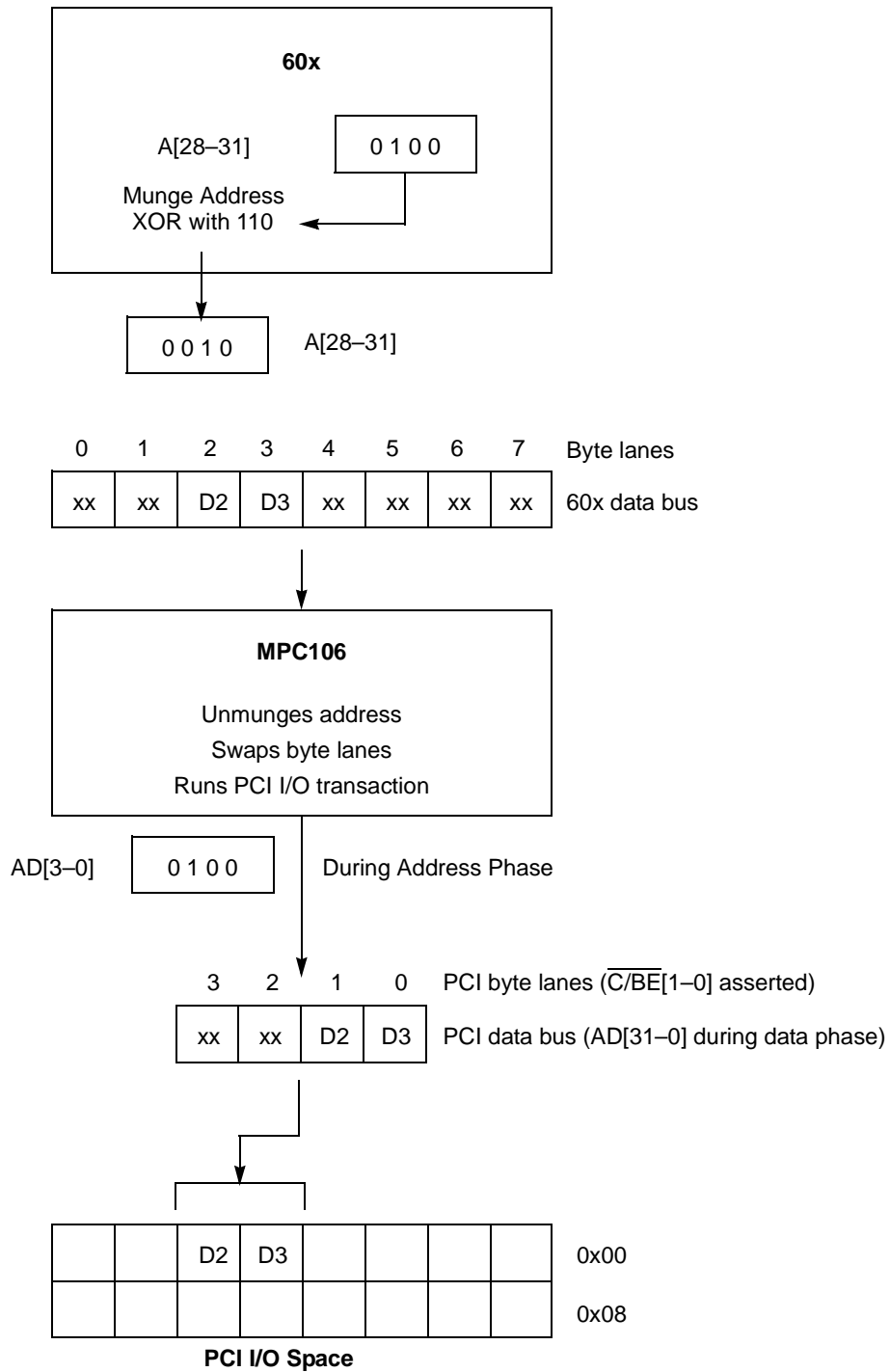


Figure B-10. Two-Byte Transfer to PCI I/O Space—Little-Endian Mode

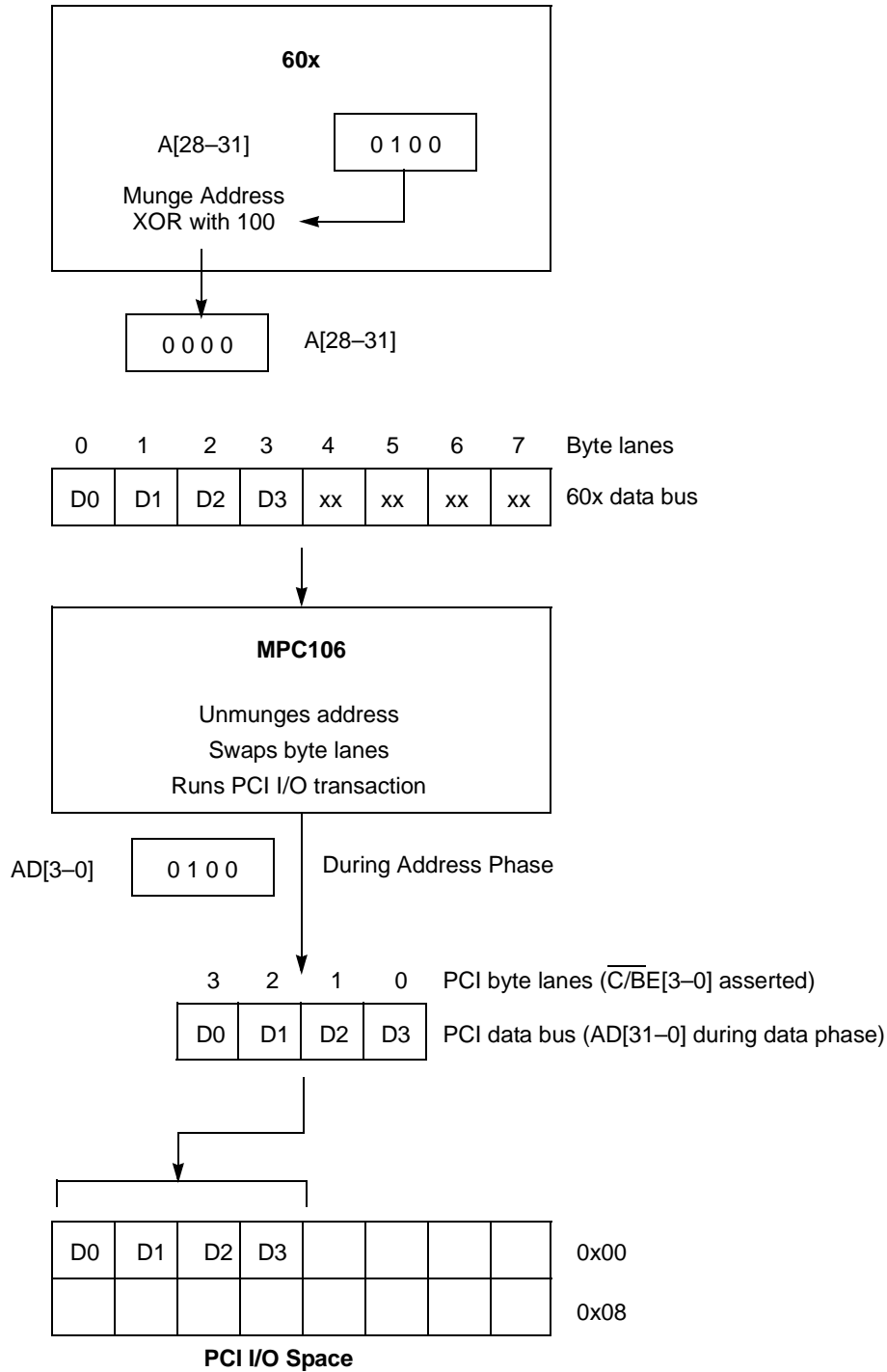


Figure B-11. Four-Byte Transfer to PCI I/O Space—Little-Endian Mode



Appendix C

JTAG/Testing Support

The MPC106 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.

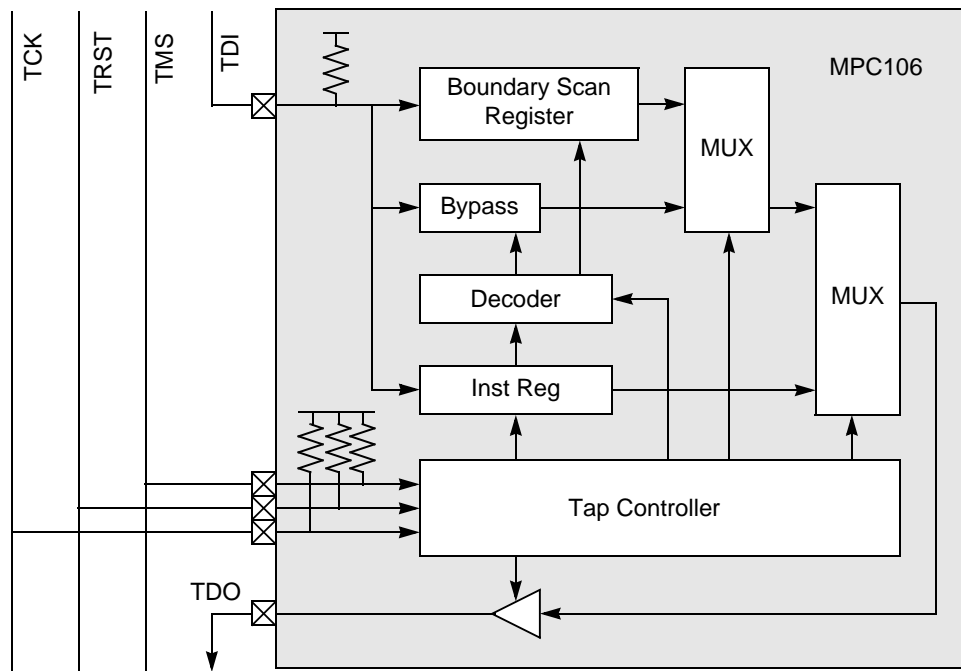


Figure C-1. JTAG Interface Block Diagram

C.1.1 JTAG Signals

The MPC106 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 MPC106.

Section 2.2.8, “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 MPC106. 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 MPC106 during board-level boundary-scan operations involving components other than the MPC106. 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 MPC106. 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 MPC106'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 MPC106 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 an example PowerPC assembly language initialization routine for an MPC106-based system using address map A.

```

;***** Contains MPC106 initialization *****
#include "./asmPowerPC.h"
#include "./asmppc.h"

        .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_ADDR
;# r2 = 0x 8000 0cfc  CONFIG_DATA
;# r3 is used to define which register number is to be stored at CONFIG_ADDR
;# r4 is used for loading and storing data to/from CONFIG_DATA
;#
;=====
;*****CYFS*****
#define  WHICHDRAM initmdc2pdram
#define  WHICHDRAM initmdc2edo
;*****END CYFS*****
.main:
        lis     r0, MPC106_REG      # r0 = 0x 8000 0000  BASE_ADDRESS
        ori     r1, r0, 0x0cf8     # r1 = 0x 8000 0cf8  CONFIG_ADDR
        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 VERY FAST ROM or SRAM.
;#
lis      r3, MPC106_REG      # start building new register number
ori      r3, r3, MCCR1      # 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.
;# -+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
;#
;# This section of code initializes the MPC106's PCI Interface Registers
;#

initpci:

;#
lis      r3, MPC106_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, MPC106_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 MPC106's Processor Interface Registers
;# for use with the PDC4 (single 603/604 and 512 kB of pipeline
;# or flowthrough L2) at 60 - 66 MHz.
;#

```

```

initproc:

;#
    lis        r3, MPC106_REG        # start building new register number
    ori        r3, r3, PICR2         # register number 0xac
    stwbrx    r3, 0, r1              # write this value to CONFIG_ADDR

    lwbrx     r4, 0, r2              # load r4 from CONFIG_DATA

    lis        r0, 0x2800            # Reserved bits are 29, 27, and 11
    ori        r0, r0, 0x0800        # bit 31 is MSb, bit 0 is LSB
    and        r4, r4, r0            # clears all 0xac reg except reserved bits

;#    lis        r0, 0x800f            #flow through type
;#    lis        r0, 0x814f            #pipelined type, fast l2 mode
    lis        r0, 0x804f            #pipelined type, slow l2 mode

;#    ori        r0, r0, 0x068e        #1 bank 256 k
;#    ori        r0, r0, 0x068f        #1 bank 256 k with adsp mode on
;#    ori        r0, r0, 0x069e        #2 banks 512 k, fast castout, aphase = 3 wait
                                        #hit delay =3
    ori        r0, r0, 0x0296        #2 banks 512 k, fast castout, aphase = 1 wait
                                        #hit delay=1

    or         r4, r4, r0            # sets the desired bits
    stwbrx    r4, 0, r2              # write value to CONFIG_DATA

;# These next five lines may not be necessary if the above OR pattern = 0x c29e 650e
    lwbrx     r4, 0, r2
    lis        r0, 0x4000            # Now that we've written to the 0x ac register,
    ori        r0, r0, 0x4000        # 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 value to CONFIG_DATA

;#
    lis        r3, MPC106_REG        # start building new register number
    ori        r3, r3, PICR1         # register number 0xa8
    stwbrx    r3, 0, r1              # write this value to CONFIG_ADDR
    lwbrx     r4, 0, r2              # load r4 from CONFIG_DATA

    lis        r0, 0x0000            # bit 14 is reserved bit
    ori        r0, r0, 0x4000        # bit 31 is MSb
    and        r4, r4, r0            # clears all 0xa8 reg bits except reserved bits
;#    lis        r0, 0x3f75            #603 in drtry mode
    lis        r0, 0x3f37            #604 type processor
    ori        r0, r0, 0x0498        # LEAVE THE L2 CACHE OFF
;#    ori        r0, r0, 0x0698        # LEAVE THE L2 CACHE OFF but turn on dpark
    or         r4, r4, r0            # sets the desired bits

```

```

stwbrx    r4, 0, r2          # write value to CONFIG_DATA
;#
lis       r3, MPC106_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     #
or        r4, r4, r0         # sets the desired bits
stb       r4, 2(r2)          # write the modified data to CONFIG_DATA

;#
lis       r3, MPC106_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     #
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 MPC106's memory configuration registers
;# for use with the MDC2 (64 MB PDRAM/EDO and 2 MB SRAM BootROM) at 60 - 66 MHz.
;#
;# This will branch to either initmdc2pdram for page mode type dram or
;# it will branch to initmdc2edo for edo dram
        b          WHICHDRAM
;#
;#*****Normal page mode DRAM initialization*****
initmdc2pdram:

;#
lis       r3, MPC106_REG     # start building new register number
ori       r3, r3, MCCR1     # 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, MPC106_REG     # start building new register number

```

```

ori      r3, r3, MCCR2      # 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, 0x0c34     # 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, MPC106_REG     # start building new register number
ori     r3, r3, MCCR3     # 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

lis     r3, MPC106_REG     # start building new register number
ori     r3, r3, MCCR4     # register number 0xfc
stwbrx  r3, 0, r1          # write this value to CONFIG_ADDR
lwbrx   r4, 0, r2          # load r4 from CONFIG_DATA

lis     r0, 0xffef         # all BUT bit 20 are reserved!!!
ori     r0, r0, 0xffff     # bit 31 is MSb
and     r4, r4, r0         # clears all 0xfc reg bits except reserved bits
;#   lis     r0, 0x0000     # set RCBUF=0 for flow thru data buffers
lis     r0, 0x0010         # set RCBUF=1 for clocked/latched data buffers
ori     r0, r0, 0x0000     #
or      r4, r4, r0         # sets the desired bits
stwbrx  r4, 0, r2          # write value to CONFIG_DATA

;#   lis     r0, 0x0000     # set RCBUF=0 for flow thru data buffers
;#   ori     r0, r0, 0x0000     #
;#   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 PAGE MODE 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, MPC106_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, MPC106_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, MPC106_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, MPC106_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, MPC106_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, MPC106_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, MPC106_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, MPC106_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, MPC106_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
b        endmdc2init    #jump to enable memories
;#*****edo initialization*****
initmdc2edo:

;#
lis      r3, MPC106_REG  # start building new register number
ori      r3, r3, MCCR1   # 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, MPC106_REG     # start building new register number
ori      r3, r3, MCCR2     # register number 0xf4
stwbrx   r3, 0, r1        # write this value to CONFIG_ADDR

lwbrx    r4, 0, r2          # load r4 from CONFIG_DATA
lis      r0, 0x0001        # Self-Refresh value (not used for MDC2 or MDC3) EDO mode set
;# lis    r0, 0x0000        # Self-Refresh value (not used for MDC2 or MDC3) EDO mode NOT set
ori      r0, r0, 0x0c34    # 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, MPC106_REG     # start building new register number
ori      r3, r3, MCCR3     # 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

lis      r3, MPC106_REG     # start building new register number
ori      r3, r3, MCCR4     # register number 0xfc
stwbrx   r3, 0, r1        # write this value to CONFIG_ADDR
lwbrx    r4, 0, r2          # load r4 from CONFIG_DATA

lis      r0, 0xffef        # all BUT bit 20 are reserved!!!
ori      r0, r0, 0xffff    # bit 31 is MSb
and      r4, r4, r0        # clears all 0xfc reg bits except reserved bits
;# lis    r0, 0x0000        # set RCBUF=0 for flow thru data buffers
lis      r0, 0x0010        # set RCBUF=1 for clocked/latched data buffers
ori      r0, r0, 0x0000    #
or       r4, r4, r0        # sets the desired bits
stwbrx   r4, 0, r2        # write value to CONFIG_DATA

;# lis    r0, 0x0000        # set RCBUF=0 for flow thru data buffers

```



```

;#   ori      r0, r0, 0x0000   #
;#   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, 32MB total of EDO DRAM
;#
;# Bank0 = 0x 0000 0000 - 0x 007f ffff start=00 ext=00 end=07 extend=00
;# Bank1 = not implemented
;# Bank2 = 0x 0080 0000 - 0x 00ff ffff start=08 ext=00 end=0f extend=00
;# Bank3 = not implemented
;# Bank4 = 0x 0100 0000 - 0x 017f ffff start=10 ext=00 end=17 extend=00
;# Bank5 = not implemented
;# Bank6 = 0x 0180 0000 - 0x 01ff ffff start=18 ext=00 end=1f extend=00
;# Bank7 = not implemented
;#
    lis      r3, MPC106_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, 0xed08        # Each bank on MDC2 is 8MB
    ori      r4, r4, 0xed00    # (no reserved bits)
    stwbrx   r4, 0, r2        # write the modified data to CONFIG_DATA

;#
    lis      r3, MPC106_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, 0xed18        # Each bank on MDC2 is 8MB
    ori      r4, r4, 0xed10    # (no reserved bits)
    stwbrx   r4, 0, r2        # write the modified data to CONFIG_DATA

;#
    lis      r3, MPC106_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, MPC106_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, MPC106_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, 0xed0f       # Each bank on MDC2 is 8MB
ori      r4, r4, 0xed07   # (no reserved bits)
stwbrx   r4, 0, r2       # write the modified data to CONFIG_DATA

;#
lis      r3, MPC106_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, 0xed1f       # Each bank on MDC2 is 8MB
ori      r4, r4, 0xed17   # (no reserved bits)
stwbrx   r4, 0, r2       # write the modified data to CONFIG_DATA

;#
lis      r3, MPC106_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, MPC106_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, MPC106_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 0,2,4,6 banks of edo 8 mb each
    ori      r4, r4, 0x0055    # (no reserved bits)
    stb      r4, 0(r2)        # write the modified data to CONFIG_DATA

endmdc2init:

;#
    lis      r3, MPC106_REG    # start building new register number
    ori      r3, r3, MEM_PGMAX # register number 0xa3
    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, 0x0020    # this byte numberX64
    ori      r0, r0, 0x0000    # this byte numberX64
    or       r4, r4, r0        # sets the desired bits
    stb      r4, 3(r2)        # write the modified data to CONFIG_DATA
    lbz      r4, 3(r2)        # load r4 from CONFIG_DATA look at contents

;#***** end of edo or pdram initialization*****
;#
;# DRAM SHOULD NOW BE CONFIGURED AND ENABLED - MUST WAIT 200 us BEFORE ACCESSING
;#
    li       r0, 0x3800        #
    mtctr    r0
wait200us:
    bdnz     wait200us
;#
    lis      r3, MPC106_REG    # start building new register number
    ori      r3, r3, MCCR1     # 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
;#
;#=====
;#
;# Now let's turn on the L2 cache ; uncomment the desired mode below
;#
    lis     r3, MPC106_REG  # start building new register number
    ori     r3, r3, PICR1  # 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

```

Appendix E

Revision History

This appendix provides a list of the major differences between the *MPC106 PCI Bridge/Memory Controller User's Manual*, Revision 1 and the *MPC106 PCI Bridge/Memory Controller User's Manual*, Revision 2.

E.1 Revision Changes From Revision 1 to Revision 2

Major changes to the *MPC106 PCI Bridge/Memory Controller User's Manual* from Revision 1 to Revision 2 are as follows:

Section, Page	Changes
Book	Added trademark information for PowerPC. Added Appendix E, "Revision History."
2.2, 2-7	In Table 2-1, The entry for the TV signal should state "O" in the I/O column indicating that TV is an output-only signal.
2.2.2.11, 2-16	Delete the sentence, "Note that $\overline{\text{LBCLAIM}}$... power-on reset." and add the following: Note that when PLL[0:3] corresponds to one of the half-bus clock settings (5:2 and 3:2), $\overline{\text{LBCLAIM}}$ functionality is disabled. Thus, local bus slaves cannot be used in conjunction with half-clock bus modes.
2.2.4.22, 2-36	Add the following to the Timing Comments for $\overline{\text{WE}}$: For Flash, the MPC106 asserts $\overline{\text{WE}}$ one clock cycle after $\overline{\text{RCSn}}$ is asserted and negates $\overline{\text{WE}}$ one clock cycle after $\overline{\text{RCSn}}$ is negated.
2.2.9.4, 2-48	Replace the entire section text with the following: The 60x clock phase configuration signal uses $\overline{\text{LBCLAIM}}$ as a configuration input. Following is the state meaning for the $\overline{\text{LBCLAIM}}$ configuration signal.
State Meaning	High/Low—60x bus clock input to resolve clock phasing with the PCI bus clock (SYSCLK) in 3:2 and 5:2 clock modes. The 60x clock's phase is only sampled during power-on reset, hard reset, and when coming out of the sleep and suspend power saving modes.

- Note that this signal is only necessary when the PLL is configured for one of the half-bus clock modes (5:2 and 3:2). When PLL[0:3] corresponds to a half-bus clock mode, local bus slave functionality is disabled. Thus, local bus slaves cannot be used in conjunction with half-bus clock modes.
- 2.3, 2-50 In the first paragraph on the page, add the text, “Because the LBCLAIM signal is used for clock phasing in half-bus clock modes, the local bus slave functionality is disabled. Thus, local bus slaves cannot be used in conjunction with half-bus clock modes.” just before the sentence that begins, “Note that systems that use...”
- 3.2.2, 3-19 In Table 3-10, replace MAX GNT configuration register with MAX LAT, which has address offset of 3F in Hex.
- 3.2.2, 3-19 In Table 3-10, remove Special cycle address configuration register which has address offset of 44 (in Hex) from the table.
- 3.2.2, 3-21 In Figure 3-7, replace MAX GNT (at offset 0x3F) with MAX LAT
- 3.2.2, 3-21 In Figure 3-7, replace Special Cycle Address (at offset 0x44) with a reserved row.
- 4.3.2, 4-12 Add the following after the first sentence: “However, the MPC106 does not check address bus parity)”
- 4.4.5, 4-23 Add the following after the first sentence, “Note that local bus slaves cannot be used in conjunction with half-bus clock modes (3:2 and 5:2). See Section 2.3, ‘Clocking,’ for more information.”
- 6.6.6, 6-84 Prepend the following sentence to the fourth paragraph: “MCCR1[MEMGO] must be set before any writes to Flash are attempted.”
- 8.1.3.1.1, 8-8 Prepend the following sentence to the last paragraph in the section: “Speculative PCI read operations must not be allowed to occur past the last cache line of the physical memory boundaries as set by the memory boundary registers or the MPC106 will hang.”
- 9.3.2.5, 9-9 In the last sentence of the section, replace “ErrDR1[3],” with “ErrDR1[4]”
- Glossary-5 For the entry for “Reserved,” replace the last two sentences with the following: “For the MPC106, reserved bits are not guaranteed to have predictable values. However, software must preserve the values of reserved bits when writing to a register. Also, when reading from a register, software should not rely on the value of any reserved bit remaining consistent.”

Glossary of Terms and Abbreviations

The glossary contains an alphabetical list of terms, phrases, and abbreviations used in this book. Some of the terms and definitions included in the glossary are reprinted from *IEEE Std 754-1985, IEEE Standard for Binary Floating-Point Arithmetic*, copyright ©1985 by the Institute of Electrical and Electronics Engineers, Inc. with the permission of the IEEE.

Note that some terms are defined in the context of how they are used in this book.

A **Architecture.** A detailed specification of requirements for a processor or computer system. It does not specify details of how the processor or computer system must be implemented; instead it provides a template for a family of compatible *implementations*.

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/stwex** instruction pair, which asserts the TTO 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. *See* Little-endian.

Block. An area of memory that ranges from 128 Kbyte to 256 Mbyte, whose size, translation, and protection attributes are controlled by the block address translation (BAT) mechanism.

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 flush. An operation that removes from a cache any data from a specified address range. This operation ensures that any modified data within the specified address range is written back to main memory. This operation is generated typically by a Data Cache Block Flush (**dcbf**) instruction.

Caching-inhibited. A memory update policy in which the *cache* is bypassed and the load or store is performed to or from main memory.

Cache line. A small region of contiguous memory that is copied from memory into a *cache*. The size of a cache line may vary among processors; the maximum block size is one *page*. Note that the term ‘cache line’ is often used interchangeably with ‘cache block’.

Castouts. *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.

Clear. To cause a bit or bit field to register a value of zero. *See also* Set.

Copyback operation. A cache operation in which a cache line is copied back to memory. Copyback operations consist of *snoop push-out* operations and cache *castout* operations.

D

Direct-store. Interface available on processors only to support direct-store devices from the POWER architecture.

Disconnect. The termination of a PCI cycle initiated by the target because it is unable to respond within eight PCI clock cycles. Note that the term ‘disconnect’ is often used interchangeably with ‘target-disconnect’.

E

Exception. An unusual or error condition encountered by the processor that results in special processing.

F **Fetch.** Retrieving instructions from either the *cache* or main memory and placing them into the instruction queue.

Flush. An operation that causes a modified *cache line* to be invalidated and the data to be written to memory.

I **Implementation.** A particular processor that conforms to the PowerPC architecture, but may differ from other architecture-compliant implementations for example in design, feature set, and implementation of *optional* features. The PowerPC architecture has many different implementations.

In-order. An aspect of an operation that adheres to a sequential model. An operation is said to be performed in-order if, at the time that it is performed, it is known to be required by the sequential execution model. *See* Out-of-order.

Interrupt. An external signal that causes the 60x to suspend current execution and take a predefined exception.

K **Kill.** An operation that causes a *cache line* to be invalidated.

L **L1 cache.** *See* Primary cache.

L2 cache. *See* Secondary cache.

Latency. The number of clock cycles necessary to execute an instruction and make ready the results of that instruction.

Least-significant byte (LSB). The byte of least value in an address, register, data element, or instruction encoding.

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*. *See* Big-endian.

M **Most-significant bit (msb).** The highest-order bit in an address, registers, data element, or instruction encoding.

Most-significant byte (MSB). The highest-order byte in an address, registers, data element, or instruction encoding.

Multiplex. To combine two or more signals into a single physical pin. The pin may alternate between the multiple signal functions (such as the PCI $\overline{C}/\overline{BE}[3-0]$ signals), or may provide only one function based on the configuration (such as the $\overline{BA0}/\overline{BR3}$ signal).

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

Nibble. A sequence of four bits. In the context of ECC, a nibble is either the high-order four bits (bit 0–3) or the low-order four bits (bits 4–7) in a byte.

O

Optional. A feature, such as an instruction, a register, or an exception, that is defined by the PowerPC architecture but not required to be implemented.

Out-of-order. An aspect of an operation that allows it to be performed ahead of one that may have preceded it in the sequential model, for example, speculative operations. An operation is said to be performed out-of-order if, at the time that it is performed, it is not known to be required by the sequential execution model. *See* In-order.

Overflow. An error condition that occurs during arithmetic operations when the result cannot be stored accurately in the destination register(s). For example, if two 32-bit numbers are multiplied, the result may not be representable in 32 bits.

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.

Physical memory. The actual memory that can be accessed through the system's memory bus.

Pipelining. A technique that breaks operations, such as instruction processing or bus transactions, into smaller distinct stages or tenures (respectively) so that a subsequent operation can begin before the previous one has completed.

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.

Reservation. The processor establishes a reservation on a *cache line* of memory space when it executes an **lwarx/stwcx** instruction to read a memory semaphore into a GPR.

Reserved. In a register, a reserved field is one that is not assigned a function. A reserved field may be a single bit. The handling of reserved bits is implementation-dependent. For the MPC106, reserved bits are not guaranteed to have predictable values. However, software must preserve the values of reserved bits when writing to a register. Also, when reading from a register, software should not rely on the value of any reserved bit remaining consistent.

RISC (reduced instruction set computing). An *architecture* characterized by fixed-length instructions with nonoverlapping functionality and by a separate set of load and store instructions that perform memory accesses.

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 (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.

Set (v). To write a nonzero value to a bit or bit field, the opposite of *clear*. The term 'set' may also be used to generally describe the updating of a bit or bit field.

Set (*n*). A subdivision of a *cache*. Cacheable data can be stored in a given location in any one of the sets, typically corresponding to its lower-order address bits. Because several memory locations can map to the same location, cached data is typically placed in the set whose *cache line* corresponding to that address was used least recently.

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.

Synchronization. A process to ensure that operations occur strictly *in order*.

System memory. The physical memory available to a processor.

T

Target-disconnect. The termination of a PCI cycle initiated by the target because it is unable to respond within eight PCI clock cycles. Note that the term ‘target-disconnect’ is often used interchangeably with ‘disconnect’.

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.

Throughput. The measure of the number of instructions that are processed per clock cycle.

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

Word. A 32-bit data element.

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 *copyback operation* is required.

Write-through. A memory update policy in which all processor write cycles are written to both the cache and memory.



Index

Numerics

60x address bus, *see* Address bus, 60x
 60x data bus, *see* Data bus, 60x
 60x processor interface
 60x local bus slave timing, 4-23
 address *retry*, 2-11, 4-10
 address tenure operations, 4-10
 alternate *bus master*, 4-1
 burst ordering and data transfers, 4-15
 bus accesses, 4-7
 bus configuration, 4-1
 bus error signals, 9-3
 bus error status register, 3-33, 3-38, 3-45, 9-6
 bus interface support, 4-1
 bus interface unit (BIU), B-1
 bus protocol, 4-7
 bus request monitoring, power management, A-7
 byte ordering, B-1
 configuration registers, 3-57, 4-7
 configuring power management, 3-28
 data tenure operations, 4-20
 error checking and correction (ECC), 6-26
 error detection, 9-7
 multiprocessor configuration, 4-4
 external L2 cache, 4-5
 overview, 1-4, 4-1
 PCI buffers, 8-4
 PCI bus operations, 4-14
 processor-to-system-memory, 6-33
 programmable parameters
 parking, 3-57, 4-7
 PICR1/PICR2 registers, 3-57, 4-7
 signal timing examples, 6-33
 signals, 2-9
 single-processor configuration, 4-1
 slave support, 4-22
 system memory buffer, 8-2
 unsupported bus transactions error, 4-22, 9-7

A

AACK (address acknowledge) signal, 2-10, 4-18
 Acronyms and Abbreviations, xxix

Address, 3-8
 Address bus, 60x
 address *tenure*
 bus protocol overview, 4-7
 operations, 4-10
 timing configuration, 4-19
 arbitration
 bus arbitration, 4-10
 dual processor arbitration, 4-11
 signals, 4-9
 L2 cache address operations, 5-9
 snoop operation, 4-18
 transfer attribute signals, 4-12
 transfer termination, 4-18
 Address maps
 address map A
 contiguous map, 3-4
 discontiguous map, 3-5
 overview, 3-1
 PCI I/O map, 3-7
 PCI I/O master view, 3-3
 PCI memory map, 3-6
 PCI memory master view, 3-2
 processor view, 3-2
 address map B
 emulation mode
 address map, 3-11
 PCI I/O master view, 3-12
 PCI memory master view, 3-11
 processor view, 3-11
 overview, 3-7
 PCI I/O master view, 3-9
 PCI memory master view, 3-8
 processor view, 3-8
 addressing on PCI bus, 7-6
 DBG0 signal, 3-1
 description, 3-1
 emulation mode address map, 3-1, 3-11, 3-69, 7-28
 ESCR1 register, 3-1, 3-69, 7-28
 examples, configuration sequences, 3-15
 Addressing
 L2 cache addressing, 5-9
 memory addressing, 6-13

- PCI bus
 - configuration space, 7-7
 - I/O space, 7-7
 - memory space, 7-6
 - AD_n (PCI address/data bus) signals, 2-37, 7-6
 - ADS (address strobe) signal, 2-22, 5-39
 - Alignment
 - aligned data transfers, 4-16
 - byte alignment, 6-80, 7-8, B-1
 - Alternate bus master, usage, 4-1
 - Alternate OS-visible parameters registers, 3-67
 - An (60x address bus) signals, 2-10
 - Arbitration
 - 60x bus
 - address bus arbitration, 4-10
 - address tenure, 4-8
 - arbitration signals, 4-9
 - data bus, 4-20
 - data tenure, 4-8
 - dual processor arbitration, 4-11
 - internal arbitration, 8-9
 - PCI bus arbitration, 7-3
 - Architecture, PowerPC, xxv
 - AR_n (ROM address) signals, 2-29, 6-80
 - ARTRY (address retry) signal, 2-11, 4-18
 - Asynchronous SRAMs
 - CF_DOE bit, 3-67, 5-25
 - CF_WDATA bit, 3-67, 5-25
 - CF_WMODE bit, 3-64, 5-26
 - description, 5-6
 - L2 cache timing examples, 5-38
- ## B
- BA0 (burst address 0) signal, 2-22
 - BAA (bus address advance) signal, 2-23
 - BCTL_n (buffer control) signals, 2-30, 6-2
 - BGL2 (external L2 bus grant) signal, 2-26, 5-44
 - BG_n (bus grant) signals, 2-12, 2-28, 4-9
 - Big-endian mode
 - accessing configuration registers, 3-16
 - byte lane translation, B-1
 - byte ordering, B-1
 - LE_MODE bit, 3-61, B-1
 - PCI memory space, B-3, B-5
 - BIST (built-in self test) control register, 3-22
 - Boundary-scan registers, C-2
 - BRL2 (external L2 bus request) signals, 2-27, 5-44
 - BR_n (bus request) signals, 2-13, 2-28, 4-9
 - Buffers
 - configurations, parameter settings, 6-2
 - internal buffers
 - copy-back buffer, 8-2
 - PCI-to-system memory read buffer (PCMRB), 8-6
 - PCI-to-system memory write buffers (PCMWBs), 8-6
 - processor-to-PCI-read buffer (PRPRB), 8-4
 - processor-to-PCI-write buffers (PRPWBs), 8-5
 - memory buffers
 - configurations, parameter settings, 6-2
 - flow-through buffers, 6-3
 - mode/control signals, 6-2
 - registered buffers, 6-5
 - (16501 type), 6-8
 - (16601 type), 6-7
 - (16952 type), 6-6
 - transparent latch buffers, 6-4
 - Burst operations
 - 60x data bus transfers
 - description, 4-15
 - normal termination, 4-20
 - 64-bit data path, 6-23
 - burst ordering
 - 60x data transfers, 4-15
 - PCI cache wrap mode, 7-6
 - PCI linear incrementing, 7-6
 - PCI bus transfer, 7-3
 - SDRAM-based systems, 6-57
 - Bus error status registers, 60x, 3-33, 3-38, 3-45, 9-6
 - Bus interface unit (BIU), 60x, B-1
 - Bus operations
 - 60x processor interface
 - address tenure operations, 4-10
 - bus protocol, 4-7
 - data tenure operations, 4-20
 - L2 cache response, 5-12
 - PCI bus transactions, 7-9
 - Bypass register, C-2
 - Byte
 - alignment, 6-80, 7-8, B-1
 - byte enable signals, 2-37
 - C/BE_n (command/byte enable) signals, 7-26
 - C/BE_n signals, 7-8
 - ordering
 - 60x bus, B-1
 - big-endian mode, B-1
 - least-significant byte (LSB), B-1
 - little-endian mode, B-5
 - most-significant byte/bit (MSB/msb), B-1
 - PCI bus, 7-2, B-1
 - PCI alignment, 7-8, B-2
- ## C
- C/BE_n (command/byte enable) signals, 2-37
 - C/BE_n signals, 7-8, 7-26
 - Cache wrap mode, PCI, 7-6
 - CAS_n (column address strobe) signals, 2-31, 6-12, 6-33

- CHRP (common hardware reference platform), 1-1, 3-1
 - CI (caching-inhibited) signal, 2-13
 - CK0, output (test clock) signal, 2-45
 - CKE (SDRAM clock enable) signal, 2-31
 - Clock configuration
 - LBCLAIM, 2-49
 - PLL n (clock mode) signals, 2-49
 - power management support, A-6
 - Clock signals, 2-44
 - Commands
 - PCI commands
 - C/BE n signals, 2-37, 7-4
 - interrupt-acknowledge transaction, 7-22
 - PCI command register, 3-22, 7-16
 - special-cycle command, 7-22
 - SDRAM commands
 - command encodings, 6-62
 - JEDEC standard commands, 6-61
 - mode-set commands, 6-61
 - Common hardware reference platform (CHRP), 1-1, 3-1
 - Completion, PCI transaction, 7-12
 - Configuration
 - configuration registers
 - 60x bus error status registers, 3-33, 3-38, 3-45, 9-6
 - 60x/PCI error address register, 3-40, 9-6
 - accessing registers, 3-14-3-17
 - alternate OS-visible parameters registers, 3-67
 - ECC single-bit error registers, 3-33, 9-7
 - emulation support, 3-1, 3-69, 7-28
 - error detection registers, 3-36, 9-6
 - error enabling registers, 3-34
 - error status registers, 3-38, 7-25
 - external configuration registers, 3-72
 - memory bank enable register, 3-44, 6-15
 - memory boundary registers, 3-40-3-44, 6-15, 6-59
 - memory control configuration registers, 3-46, 6-2
 - memory interface configuration registers, 3-40
 - memory page mode register, 3-45
 - modified memory status register, 3-71
 - PCI command register, 3-22, 7-16
 - PCI status register, 3-24, 7-16
 - power management registers, 3-28-3-30, A-1
 - processor interface configuration registers, 3-57
 - register access, 3-14-3-17
 - reserved* bits, 3-14
 - summary of registers, list, 3-17
 - configuration signals, 2-48, 2-49
 - configuration space
 - MPC106 configuration space, 3-21
 - PCI addressing, 7-7
 - DRAM/EDO configurations, parameter settings, 6-15
 - external configuration registers
 - port registers, 3-72
 - PCI configuration
 - configuration cycles
 - CONFIG_ADDR register, 3-8, 3-14, 7-18
 - CONFIG_DATA register, 3-8, 3-14, 7-18
 - configuration space header, 7-15
 - type 0 and 1 accesses, 7-17
 - configuration header summary, 3-21, 7-16
 - SDRAM device configurations, 6-54
 - Conventions, xxix
 - CS n (SDRAM command select) signals, 2-31
- ## D
- Data bus, 60x
 - address tenure timing configuration, 4-21
 - arbitration signals, 4-9
 - bus arbitration, 4-20
 - bus transaction errors, 4-22, 9-7
 - data *tenure*
 - bus protocol overview, 4-7
 - operations, 4-20
 - data transfer, 4-15, 4-20
 - shared data bus, 8-2
 - termination by TEA, 4-21, 9-11
 - Data transfers, 60x
 - alignment, 4-16
 - burst ordering, 4-15
 - normal termination, 4-20
 - DBGL2 (external L2 data bus grant) signal, 2-27, 5-44
 - DBGLB
 - slave timing, 4-23
 - DBGLB (data bus grant local bus slave) signal, 2-14
 - DBG n (data bus grant) signals, 2-13, 2-29, 3-1, 4-9
 - DCS (data RAM chip select) signal, 2-23, 5-3-5-6
 - Device drivers
 - modifying for power management, A-8
 - posted writes, 8-5
 - DEVSEL (device select) signal, 2-38, 7-7
 - DH n /DL n (data bus) signals, 2-14, 6-11
 - Direct-store access, 4-22
 - DIRTY_IN signal, 2-24, 5-10
 - DIRTY_OUT signal, 2-24, 5-10
 - Disconnect, 7-2, 7-12, 8-4
 - DOE (data RAM output enable) signal, 2-24, 3-67, 5-25
 - Doze mode, 1-7, 6-32, A-3
 - DQM n (SDRAM data qualifier) signals, 2-32
 - DRAM/EDO interface
 - burst wrap, 6-23
 - CAS n for byte lane selection, 6-12
 - DRAM system with parity, 16 Mbyte, 6-11
 - DRAM/EDO device configurations, 6-12
 - ECC single-bit error description, 6-26
 - interface operation, 6-10
 - interface timing, 6-16

- latency, 6-23
- organizations supported, 6-12
- page mode, 6-24
- parity support, 6-25, 6-69
- power-on initialization, 6-15
- programmable parameters, 3-40, 6-15
- refresh, 6-30
- RMW parity, 6-25
- suggested DRAM timing configurations, 6-17
- timing parameters, 6-17
- use in MPC106, 6-1

DWEn (data RAM write enable) signals, 2-25, 5-3

E

ECC

- additional errors, 6-26
- single-bit error description, 6-26

ECC single-bit error

- registers, 3-33, 9-7

Emulation, 3-11

Emulation mode

- address map overview, 3-11
- emulation support, 7-28
- ESCR1/ESCR2 registers, 3-1, 3-69, 7-28
- L2 cache and vector relocation, 5-12

ErrDR1/ErrDR2 (error detection) registers, 3-36, 7-25, 9-6

ErrEnR1/ErrEnR2 (error enabling) registers, 3-34

Errors

- error detect, external ECM, input BERR, 2-44
- error detection registers, 3-36, 7-25, 9-6
- error enabling registers, 3-34
- error handling
 - overview, 9-1
 - registers, 3-33, 7-25, 9-3-9-4, 9-6
- error reporting
 - 60x processor interface, 9-7
 - address/data error, 7-14, 9-10
 - error detection registers, 3-36, 7-25, 9-6
 - errors within a *nibble*, 6-26, 9-9
 - Flash write error, 9-7
 - illegal L2 copyback error, 9-7
 - L2 cache read data parity error, 9-8
 - master-abort transaction termination, 7-12, 9-10
 - nonmaskable interrupt, 2-45, 9-12
 - PCI bus, 7-25, 9-5
 - PERR and SERR signals, 7-27, 9-5
 - system memory errors, 6-1, 9-8
 - TEA and MCP signals, 2-16, 2-18, 4-21, 9-1
 - unsupported bus transaction error, 4-22, 9-7
- error status registers, 3-38, 7-25, 9-6
- interrupt and error signals, 2-16, 2-45, 9-3
- overflow* condition, 3-34, 9-9

- ESCR1/ESCR2 (emulation support configuration) registers, 3-1, 3-69, 7-28

Exceptions

- bus errors, 9-4
- interrupt and error signals, 2-45, 9-3
- interrupt latencies, 9-12
- interrupt priorities, 9-2
- system reset interrupt, 9-3

- Exclusive access, PCI, 7-23, 7-25

- External configuration registers, 3-72

- External error checking (ECM)

- BERR, 6-71
- block diagram, 6-71
- parameter settings, 6-71
- support, 6-70

F

- FLSHREQ (flush request) signal, 2-43, 7-27

- FNR (ROM bank 0 data path width) signal, 2-49

- FOE (flash output enable) signal, 2-32, 6-94

- FRAME signal, 2-39, 7-3

- Full-on mode, 1-7, 6-32, A-3

G

- GBL (global) signal, 2-16

- GNT (PCI bus grant) signal, 2-39, 7-3

H

- HIT signal, 2-25, 2-27, 5-10

- HRST (hard reset) signal, 2-45, 9-3, A-2

I

- IEEE 1149.1 specifications

- signals, 2-47, C-2
- specification compliance, C-2

- Implementation* of the PowerPC architecture, 1-4

Initialization

- DRAM power-on initialization, 6-15
- initialization code example, C-1
- SDRAM power-on initialization, 6-59

Interface

- 60x processor interface
 - 60x local bus slave timing, 4-23
 - address *retry*, 2-11, 4-10
 - address tenure operations, 4-10
 - alternate *bus master*, 4-1
 - burst ordering and data transfers, 4-15
 - bus accesses, 4-7
 - bus configuration, 4-1
 - bus error signals, 9-3
 - bus error status register, 3-33, 3-38, 3-45, 9-6

- bus interface support, 4-1
- bus interface unit (BIU), B-1
- bus protocol, 4-7
- bus request monitoring, power management, A-7
- byte ordering, 60x bus, B-1
- configuration registers, 3-57, 4-6
- configuring power management, 3-28
- data tenure operations, 4-20
- error detection, 9-7
- multiprocessor configuration, 4-4
- overview, 1-4, 4-1
- PCI buffers, 8-4
- PCI bus operations, 4-14
- signals, 2-9
- single-processor configuration, 4-1
- slave* support, 4-22
- system memory buffer, 8-2
- unsupported bus transactions error, 4-22, 9-7
- JTAG interface
 - block diagram, C-1
 - description, C-1
 - registers, C-2
 - signals, 2-47, C-2
 - see also JTAG interface
- L2 interface
 - 60x address bus, 5-9
 - cache configurations, 5-2
 - cache control parameters, 5-22, 5-44
 - cache flush, 3-63, 3-74
 - cache initialization parameters, 5-23
 - cache line status, 2-24, 5-10
 - cache operation, 5-44
 - cache tag lookup, 2-24, 5-10
 - castouts*, 5-11
 - CF_L2_HIT_DELAY timing configuration, 3-66, 4-7, 4-23, 5-25
 - configuration registers, 3-61
 - copyback operation*, 5-10
 - data RAM write enable
 - signals, 2-25
 - timings, 5-26
 - dirty bit, 5-2
 - external cache controller operation, 5-42
 - features list, 1-2
 - illegal L2 copyback error, 9-7
 - internal cache controller operation, 5-9
 - overview, 1-4, 5-1
 - parity support, 5-11
 - read data parity error, 9-8
 - response to bus operations, 5-12
 - signals, 2-22
 - SRAMs, 5-2-5-7, 5-30-5-41
 - tag RAM and data RAM addressing, 5-10
 - timing examples, 5-29
 - vector relocation, 5-12
 - write-back*, 5-2, 5-13
 - write-through*, 5-2, 5-19
- memory interface
 - buffer mode parameters, 6-2
 - configuration registers, 3-40
 - DRAM/EDO
 - address *multiplexing*, 6-13
 - interface operation, 6-10
 - see also DRAM/EDO interface
 - ECC error, 9-8
 - error detection, 9-7
 - errors within a *nibble*, 9-8
 - features list, 1-3
 - fetch*, 6-82
 - Flash interface
 - cacheability restrictions, 6-81
 - description, 6-78
 - interface timing, 6-87
 - Flash write error, 9-7
 - MCCRs (memory control configuration) registers, 6-15, 6-59
 - MICR (memory interface configuration) registers, 6-15
 - overview, 1-5, 6-1
 - physical memory*, 9-8
 - power management support, 6-32
 - read data parity error, 9-8
 - refresh overflow error, 9-9
 - registers, 3-40-3-46, 6-15
 - ROM interface
 - cacheability restrictions, 6-81
 - description, 6-78
 - ROM/Flash interface
 - 16-Mbyte ROM system, 6-79
 - 1-Mbyte Flash system, 6-80
 - burst read timing, 6-83
 - interface timing, 6-82
 - memory write timing, 6-95
 - parity/ECC signals, 6-8
 - write operations, 6-94
 - SDRAM interface operation
 - overview, 6-52
 - select error, 9-9
 - signal buffering, 6-2
 - signals, 2-29
 - single-byte read timing, 6-87
 - system memory*, 6-1, 9-8
- PCI interface
 - address bus decoding, 7-6, A-7
 - address/data parity error, 7-14, 9-10
 - big-endian mode, B-3
 - burst operation, 7-3
 - bus arbitration, 7-3

- bus commands, 7-4
- bus error signals, 9-5
- bus protocol, 7-3
- bus transactions, 7-9, 7-22
- byte alignment, 7-8, B-2
- byte ordering, 7-2, B-1
- C/BE_n signals, 7-8, 7-26
- cache wrap mode, 7-6
- configuration cycles, 7-15
- configuration header, 7-16
- configuration header summary, 7-16
- configuration space addressing, 7-7
- data transfers, 7-3, 7-8
- direct-access PCI configuration transaction, 7-21
- error detection and reporting, 7-25, 9-5, 9-9
- error transactions, 7-25
- exclusive access, 7-23
- fast back-to-back transactions, 7-14
- features list, 1-3
- I/O space addressing, 7-7
- interrupt-acknowledge transaction, 7-22
- legend for timing diagrams, 7-9
- linear incrementing, 7-6
- locked operations, 7-25
- master-abort transaction termination, 7-12, 9-10
- memory space addressing, 7-6
- MPC106 as PCI bus master, 7-2
- MPC106 as PCI target, 7-2
- nonmaskable interrupt, 2-45, 9-12
- overview, 1-6, 7-1
- PCI command encodings, 7-4
- PCI Local Bus Specification*, xxvii, 3-21
- PCI special-cycle operations, 7-23, A-7
- PCI System Design Guide*, xxvii, 3-21
- PCI-to-ISA bridge, 7-27
- PCI-to-system memory read buffer (PCMRB), 8-6
- PCI-to-system memory write buffers (PCMWBs), 8-6
- processor-to-PCI read buffer (PRPRB), 8-4
- processor-to-PCI-write buffers (PRPWBs), 8-5
- read transactions, 7-9
- registers, 3-21, 7-16, 9-10
- retry PCI transactions, 7-13
- signals, 2-36, 7-3-7-8
- special-cycle transaction, 7-22
- target-abort error, 7-13, 9-11
- target-disconnect, 7-2, 7-12
- target-disconnect*, 8-4
- target-initiated termination, 7-13
- transaction termination, 7-11
- turnaround cycle, 7-8
- write transactions, 7-10

- Internal control
 - arbitration
 - in-order* execution, 8-9
 - out-of-order* execution, 8-1
 - buffers, 8-1
- Interrupts*, see Exceptions
- IRDY (initiator ready) signal, 2-39, 7-3
- ISA_MASTER signal, 2-43, 7-27

J

- JTAG interface
 - block diagram of JTAG interface, C-1
 - boundary-scan registers, C-2
 - bypass register, C-2
 - description, C-1
 - instruction register, C-3
 - JTAG registers, C-2
 - JTAG signals, 2-47, C-2
 - scan interface*, 1-4
 - status register, C-3
 - TAP controller, C-3

K

- Kill* operation, 3-65

L

- L2 (secondary cache)*, see L2 interface
- L2 interface
 - 60x address bus, 5-9
 - cache configurations, 5-2
 - cache control parameters, 5-22, 5-44
 - cache flush, 3-63, 3-74
 - cache initialization parameters, 5-23
 - cache line status, 2-24, 5-10
 - cache operation, 5-44
 - cache tag lookup, 2-24, 5-10
 - castouts*, 5-11
 - CF_L2_HIT_DELAY timing configuration, 3-66, 4-7, 4-23, 5-25
 - configuration registers, 3-61
 - copyback operation*, 5-10
 - data RAM write enable
 - signals, 2-25
 - timings, 5-26
 - dirty bit, 5-2
 - external cache controller operation, 5-42
 - features list, 1-2
 - illegal L2 copyback error, 9-7
 - internal cache controller operation, 5-9
 - overview, 1-4, 5-1
 - parity support, 5-11
 - read data parity error, 9-8

response to bus operations, 5-12
 signals, 2-22
 SRAMs
 asynchronous SRAMs, 5-6, 5-38
 pipelined burst SRAMs, 5-4
 synchronous burst SRAMs, 5-2, 5-30
 two-bank support, 5-7
 tag RAM and data RAM addressing, 5-10
 timing diagrams
 burst read, 5-39
 burst read line update, 5-40
 burst write, 5-41
 cache line cast-out, 5-33
 castout timing, 5-34
 castout timing with no ARTRY, 5-35
 hit following PCI read snoop, 5-36
 invalidate following PCI read snoop, 5-38
 L2 cache line cast-out, 5-33
 legend for timing diagrams, 5-30
 push following PCI write snoop, 5-37
 read hit timing, 5-30
 update timing, 5-33
 write hit timing, 5-32
 vector relocation, 5-12
write-back, 5-2, 5-13
write-through, 5-2, 5-19
 Latency, DRAM/EDO, 6-23
 LBCLAIM (local bus slave claim signal), 2-49
 LBCLAIM (local bus slave claim) signal, 2-16, 4-23
 Little-endian mode
 PCI bus, 7-2
Little-endian mode
 accessing configuration registers, 3-14
 aligned scalars, address modification, B-5
 byte lane translation, B-6
 byte ordering, B-5
 LE_MODE bit, 3-61, B-5
 PCI bus, B-1
 PCI I/O space, B-12
 PCI memory space, B-9
 Local bus slave timing, 60x, 4-23
 LOCK signal, 2-40, 7-23

M

Major functional units
 memory interface
 error detection, 1-5
 MAn (memory address) signals, 2-32
 Master-abort, PCI, 7-12, 9-10
 MCCRn (memory control configuration) registers,
 3-46-3-55
 MCCRs (memory control configuration) registers, 6-15
 MCP (machine check) signal, 2-16, 4-21, 9-3

MDLE (memory data latch enable) signal, 2-33, 6-4
 MEMACK (memory acknowledge) signal, 2-44, 7-27
 Memory interface, 6-15
 buffer mode parameters, 6-2
 configuration registers, 3-40
 DRAM/EDO
 address *multiplexing*, 6-13
 interface operation, 6-10
 ECC error, 9-8
 error detection, 9-7
 errors within a *nibble*, 9-8
 features list, 1-3
 Flash write error, 9-7
 memory configurations, 6-16, 6-54
 overview, 1-5, 6-1
 error detection, 6-1
 parity checking, 6-1
physical memory, 9-8
 power management support, 6-32, 6-77
 processor-to-DRAM transaction examples, 6-33
 read data parity error, 9-8
 refresh overflow error, 9-9
 registers
 MCCRs (memory control configuration) registers,
 6-15
 memory bank enable register, 3-44
 memory boundary registers, 3-40-3-44
 memory control configuration registers, 3-46
 memory page mode register, 3-45
 MICRs (memory interface configuration) registers,
 6-15, 6-59
 ROM/Flash interface
 16-Mbyte ROM system, 6-79
 1-Mbyte Flash system, 6-80
 burst read timing, 6-83
 cacheability restrictions, 6-81
 description, 6-78
 interface timing, 6-82, 6-87
 memory write timing, 6-95
 parity/ECC signals, 6-8
 ROM address signals, 2-29
 single-byte read timing, 6-87
 write operations, 6-94
 select error, 9-9
 system memory, 6-1, 9-8
 Memory page mode register, 3-45
 MICR (memory interface configuration) registers, 6-59
 Misaligned 60x data transfer, 4-17
 Mode-set command, SDRAM, 6-61
 MPC106
 aligned scalars, address modification, B-6
 block diagram, 1-2
 default mode, A-3
 device programming, 3-1

- major functional units, 1-4
- PCI bus master, 7-2
- PCI bus/memory controller features, 1-1
- PCI sideband signals, 7-27
- PCI target, 7-2
- Multiplexing, 6-55
- Multiprocessor implementations
 - address pipelining/split-bus capability, 4-9
 - multiprocessor configuration, 4-4
- Munging*
 - for 60x processors, B-1
 - munged memory image, LE mode, B-7
- N**
- Nap mode
 - description, 1-7, A-3
 - memory refresh, 6-32
 - PMCR bit settings, 3-29
 - QREQ signal, A-1
 - special cycle, PCI, 7-22
- Nibble*, see Errors
- NMI (nonmaskable interrupt) signal, 2-45, 9-5, 9-12
- O**
- Overview, 1-1
- P**
- PAR (PCI parity) signal, 2-40, 7-26
- Parity/ECC path read control, 6-8
- PAR_n (data parity/ECC) signals, 2-33
- PCI interface
 - address bus decoding, 7-6, A-7
 - address/data parity error, 7-14, 9-10
 - big-endian mode, four-byte transfer, B-3
 - burst operation, 7-3
 - bus arbitration, 7-3
 - bus commands, 7-4
 - bus error signals, 9-5
 - bus protocol, 7-3
 - bus transactions
 - fast back-to-back transactions, 7-14
 - interrupt-acknowledge transaction, 7-22
 - legend for timing diagrams, 7-9
 - read transactions, 7-9
 - special-cycle transaction, 7-22
 - transaction termination, 7-11
 - write transactions, 7-10
 - byte alignment, 7-8, B-2
 - byte ordering, 7-2, B-1
 - C/BE_n signals, 7-8, 7-26
 - cache wrap mode, 7-6
 - configuration cycles, 7-15
 - configuration header, 7-16
 - configuration header summary, 7-16
 - configuration space addressing, 7-7
 - data transfers, 7-3
 - direct-access PCI configuration transaction, 7-21
 - error detection and reporting, 7-25, 9-5, 9-9
 - exclusive access, 7-23
 - features list, 1-3
 - I/O space addressing, 7-7
 - linear incrementing, 7-6
 - little-endian mode transfers to I/O space, B-12
 - little-endian mode transfers to memory space, B-9
 - locked operations, 7-25
 - master-abort transaction termination, 7-12, 9-10
 - memory space addressing, 7-6
 - MPC106 as PCI bus master, 7-2
 - MPC106 as PCI target, 7-2
 - nonmaskable interrupt, 2-45, 9-12
 - overview, 1-6, 7-1
 - PCI bus parking, A-7
 - PCI command encodings, 7-4
 - PCI commands
 - interrupt-acknowledge transaction, 7-22
 - special-cycle command, 7-22
 - PCI Local Bus Specification*, xxvii, 3-21
 - PCI special-cycle operations, 7-23, A-7
 - PCI System Design Guide*, xxvii, 3-21
 - PCI-to-DRAM transaction examples, 6-40
 - PCI-to-ISA bridge, 7-27, 9-5
 - PCI-to-system memory read buffer (PCMRB), 8-6
 - PCI-to-system memory write buffers (PCMWBs), 8-6
 - processor-to-PCI read buffer (PRPRB), 8-4
 - processor-to-PCI-write buffers (PRPWBs), 8-5
 - registers, 9-10
 - bus error status register, 3-39
 - CONFIG_ADDR register, 3-8, 3-14, 7-18
 - CONFIG_DATA register, 3-8, 3-14, 7-18
 - configuration header summary, 3-21
 - PCI commands register, 3-22, 7-16
 - PCI status register, 7-16
 - status register, 3-24
 - retry PCI transactions, 7-13
 - signal timing examples for a DRAM configuration, 6-40
 - signals
 - C/BE_n (command/byte enable) signals, 7-8, 7-26
 - DEVSEL, 2-38, 7-7
 - error reporting signals, 9-5
 - FLSHREQ, 2-43, 7-27
 - FRAME, 2-39, 7-3
 - GNT, 2-39, 7-3
 - IRDY, 2-39, 7-3
 - ISA_MASTER, 2-43, 7-27
 - LOCK, 2-40, 7-23

- MEMACK, 2-44, 7-27
- PERR, 2-41, 7-27, 9-5
- REQ, 2-41, 7-3
- SERR, 2-42, 7-27, 9-5
- TRDY, 2-42, 7-3
- special-cycle transaction, 7-22
- target-abort error, 7-13, 9-11
- target-disconnect, 7-2, 7-12, 8-4
- target-initiated termination, 7-13
- turnaround cycle, 7-8
- PCIB/MC bridge/memory controller
 - overview, xxv
- Performance monitor, 1-6
 - burstiness, 10-8
 - example, 10-10
 - command register (CMDR), 3-25, 10-1
 - counter overflow, 10-10
 - counter overflow, example, 10-10
 - events, 10-2
 - command type 0, 10-4
 - command type 1, 10-5
 - encodings, 10-5
 - threshold, 10-8
 - mode control (MMCR) register, 3-26, 10-1
 - operation, 3-28, 10-1
 - pipelined processor transaction, 10-2
 - terminology, 10-2
 - end-of-data (EOD), 10-2
 - PCI *latency*, 10-3
 - pipelined, 10-2
 - processor latency, 10-3
- PERR (PCI parity error) signal, 2-41, 7-27, 9-5
- PICRs (processor interface configuration registers)
 - PICR1 register
 - address map A contiguity, 3-1, 3-59
 - bit settings/overview, 3-58, 4-7
 - CF_BREAD_WS bit, 3-58, 4-7, 4-21
 - FLASH_WR_EN bit, 3-60, 9-7
 - LE_MODE (endian mode) bit, 3-61, 3-73, B-5
 - MCP_EN bit, 4-21, 9-3
 - speculative PCI reads bit, 3-61
 - ST_GATH_EN bit, 3-60
 - TEA_EN bit, 2-18, 3-60, 4-21, 9-4
 - XIO_MODE bit, 3-1, 3-59
 - PICR2 register
 - bit settings/overview, 3-63, 4-7
 - CF_APARK bit, 3-61, 4-7
 - CF_APHASE_WS bit, 3-66, 4-7
 - CF_DOE bit, 3-67, 5-25
 - CF_FAST_CASTOUT bit, 3-66, 5-24
 - CF_HOLD bit, 3-65, 5-24
 - CF_L2_HIT_DELAY bit, 3-66, 4-23, 5-24
 - CF_SNOOP_WS bit, 3-65, 4-7
 - CF_TWO_BANKS bit, 3-66, 5-7
 - CF_WDATA bit, 3-67, 5-25
 - CF_WMODE bit, 3-64, 5-26
 - FLASH_WR_LOCKOUT bit, 3-64, 9-7
 - L2_EN bit, 3-63, 3-73, 5-23
 - TEA_EN bit, 3-73
- Pipelined burst SRAMs
 - CF_WDATA bit, 3-67, 5-25
 - description, 5-4
- Pipelining
 - address pipelining, 4-9, 4-12
 - memory *latency*, 4-9
 - split-bus transactions, 4-9
 - throughput*, 4-9
- PLL
 - core frequency, 2-50
- PLL_n (clock mode) signals, 2-49
- PMCR registers
 - refresh during power saving modes, 6-77
- PMCR registers, *see* Power management
- Power management
 - clock configuration, A-6
 - doze mode, 6-32, A-3
 - DRAM refresh, 6-32
 - full-on mode, 6-32, A-3
 - memory interface, support, 6-32, 6-77
 - memory refresh operation, 6-30
 - memory refresh operations, A-8
 - modifying device drivers, A-8
 - MPC106 default mode, A-3
 - MPC106 support, A-6
 - nap mode, 6-32, 7-23, A-3, A-7
 - overview, 1-7
 - PCI address bus decoding, A-7
 - PCI special-cycle operations, 7-22
 - PMCR registers
 - DRAM/EDO refresh configuration, 6-32
 - LP_REF_EN bit, A-8
 - overview, 3-28-3-31
 - PM bit, A-1
 - PMCR1
 - bit settings, 3-29
 - PM bit, 3-30
 - PMCR2 bit settings, 3-31
 - power management support, A-6
 - power modes, A-1
 - refresh and power saving modes, 6-32
 - power mode transition, A-1
 - processor bus request monitoring, A-7
 - QREQ signal, A-1
 - SDRAM power saving modes, 6-77
 - signals, 2-44
 - sleep mode, 3-29, 6-32, 7-23, A-4
 - suspend mode, 2-35, 6-33, A-5
 - systems using 601, 3-29, A-2

- systems using 603, 3-59, A-2
- systems using 604, 3-29, A-2
- Power mode transition with the PICR1, PROC_TYPE bit, A-1
- Power-on initialization
 - power-on reset (POR), 3-40, 6-13, 9-3
 - SDRAM power-on initialization
 - overview, 6-59
 - programmable parameters, 6-59
 - setting up MICR parameters, 6-15
- PowerPC
 - common hardware reference platform (CHRP), 1-1, 3-1
 - PowerPC 601 microprocessor, 3-29, A-2
 - PowerPC 603 microprocessor, 3-59, A-2
 - PowerPC 604 microprocessor
 - power management, 3-29, A-2
 - PowerPC Reference Platform Specification*, 3-1
 - programming the MPC106, 3-1
- PPEN (parity path read enable) signal, 2-33, 6-8
- Precharge-all-banks command, SDRAM, 6-61
- Precharge-bank command, SDRAM, 6-61
- Processor interface, *see* 60x processor interface
- processor-to-SDRAM transactions, 6-33
- Programming the MPC106, 3-1

Q

- QACK (quiesce acknowledge) signal, 2-46, 3-29, A-1
- QREQ (quiesce request) signal, 2-46, 3-29, A-1

R

- RAM access time, 3-49
- RAS_n (row address strobe) signals, 2-34, 6-10, 6-33
- RCS0 (ROM location configuration) signal, 2-50, 6-2, 6-81
- RCS_n (ROM bank select) signals, 2-34
- Real-time clock signal, 2-35, 6-33, A-5
- Refresh
 - DRAM/EDO refresh, 6-30
 - power management, refresh operations, 6-32, 6-77, A-8
 - SDRAM refresh
 - command, 6-61
 - overview, 6-74
 - timing diagram, 6-76
- Registers
 - 60x bus error status registers, 3-33, 3-38, 3-45, 9-6
 - 60x/PCI error address register, 3-40, 9-6
 - alternate OS-visible parameters registers, 3-67
 - clearing* bits, 3-24
 - CONFIG_ADDR register, 7-18
 - CONFIG_DATA register, 7-18
 - configuration registers, 3-14

- ECC single-bit error registers, 3-33, 9-7
- error detection registers, 3-36, 9-6
- error enabling registers, 3-34
- error handling registers, 3-33, 7-25, 9-3-9-4, 9-6
- error status registers, 3-38, 7-25
- ESCRs (emulation support configuration) registers, 3-1, 3-69
- external configuration registers, 3-72
 - port registers, 3-72
- JTAG
 - boundary-scan registers, C-2
 - bypass register, C-2
 - instruction register, C-3
 - status register, C-3
- MCCR_n (memory control configuration) registers, 3-46-3-55
- MCCRs (memory control configuration) registers, 6-15
- memory bank enable register, 3-44
- memory boundary registers, 3-40-3-44
- memory interface configuration
 - memory page mode, 3-31
- memory page mode register, 3-45
- MICR_n (memory interface configuration) registers, 3-40
- MICRs (memory interface configuration) registers, 6-15, 6-59
 - optional* register, BIST control, 3-22
- output driver control (ODCR), 0x73, 3-31
- PCI command register, 3-22, 7-16
 - bit settings, 3-23
- PCI registers, 3-21
- PCI status register, 3-24, 7-16
 - reads and writes, 3-24
- performance monitor registers
 - command (CMDR), 3-25
 - counters, 3-28
 - mode control (MMCR), 3-26
- PICR_n (processor interface configuration) registers, 3-57, 4-6
 - power management registers, 3-28-3-30
 - registers, 6-15
 - setting* bits, 3-24
- REQ (PCI bus request) signal, 2-41, 7-3
- Reservation* set, lwarx/stwxx., 4-13
- ROM/Flash interface
 - 16-Mbyte ROM system, 6-79
 - 1-Mbyte Flash system, 6-80
 - burst read timing, 6-83
 - cacheability restrictions, 6-81
 - description, 6-78
 - interface timing, 6-82, 6-87
 - memory write timing, 6-95
 - parity/ECC signals, 6-8

ROM address signals, 2-29
 single-byte read timing, 6-87
 write operations, 6-94
 RTC signal, 2-35, 6-33, A-5

S

SDBA0 (SDRAM internal bank select) signal, 2-35
 SDCAS (SDRAM column address strobe) signal, 2-35
 SDMA_n (SDRAM address) signals, 2-35
 SDRAM device configurations
 EDO DRAM device configurations, 6-54
 SDRAM interface operation
 128-Mbyte SDRAM system, 6-53
 burst operations, 6-57
 CAS latency, 6-61
 commands
 bank-activate command, 6-61
 command encodings, 6-62
 JEDEC interface commands, 6-61
 mode-set command, 6-61
 precharge-all-banks command, 6-61
 precharge-bank command, 6-61
 read command, 6-61
 self-refresh command, 6-62
 write command, 6-61
 configurations supported, 6-54
 external error checking (ECM), 6-70
 overview, 6-52
 power saving modes, 6-77
 power-on initialization, 6-59
 programmable parameters, 6-59
 refresh
 command, 6-61
 description, 6-74
 timing diagrams
 burst read followed by burst write timing, 6-67
 burst read timing, 6-65
 burst write timing, 6-65
 self-refresh entry timing, 6-77
 self-refresh exit timing, 6-78
 single-beat read timing, 6-67
 single-beat write timing, 6-68
 write command, 6-61
 SDRAM power-on initialization, 6-59
 SDRAS (SDRAM row address strobe) signal, 2-36
Secondary (L2) cache, see L2 interface
 Self-refresh command, SDRAM, 6-62
 SERR (system error) signal, 2-42, 7-27, 9-5
 Signal buffering
 flow-through buffers, 6-3
 memory interface buffer configurations, 6-2
 parity/ECC path read control, 6-8
 registered buffers, 6-5

(16501 type), 6-8
 (16601 type), 6-7
 (16952 type), 6-6
 transparent latch buffers, 6-4

Signals

60x address/data bus arbitration, 4-9
 AACK, 2-10, 4-18
 AD_n, 2-37, 7-6
 ADS, 2-22, 5-39
 alternate functions, list, 2-4
 A_n, 2-10
 AR_n, 2-29, 6-80
 ARTRY, 2-11, 4-18
 BA0, 2-22
 BAA, 2-23
 BCTL_n, 2-30, 6-2, 6-71
 BERR, 6-71
 BERR (input), 2-44
 BGL2, 2-26, 5-44
 BG_n, 2-12, 2-28, 4-9
 BRL2, 2-27, 5-44
 BR_n, 2-13, 2-28, 4-9
 C/BEN, 2-37
 C/BEN (command/byte enable) signals, 7-8
 C/BEN signals, 7-26
 CAS_n, 2-31, 6-12, 6-33
 CI, 2-13
 CK0, 2-45
 CKE, 2-31
 configuration signal
 501 mode, input ($\overline{\text{BCTLO}}$), 2-48
 configuration signals
 501-mode ($\overline{\text{BCTLO}}$), 2-48
 CS_n, 2-31
 DBGL2, 2-27, 5-44
 DBGLB, 2-14, 4-23
 DBG_n, 2-13, 2-29, 3-1, 4-9
 DCS, 2-23, 5-3-5-6
 DEVSEL, 2-38, 7-7
 DH_n/DL_n, 2-14, 6-11
 DIRTY_IN, 2-24, 5-10
 DIRTY_OUT, 2-24, 5-10
 DOE, 2-24, 3-67, 5-25
 DQM_n, 2-32
 DWEN, 2-25, 5-3
 error signals, 2-45, 9-3
 FLSHREQ, 2-43, 7-27
 FNR, 2-49
 FOE, 2-32, 6-94
 FRAME, 2-39, 7-3
 GBL, 2-16
 GNT, 2-39, 7-3
 HIT, 2-25, 2-27, 5-10
 HRST, 2-45, 9-3, A-2

- IEEE 1149.1 interface, 2-47, C-2
- interrupt and error signals, 2-45, 9-3
- interrupt signal connections, examples, 9-12
- interrupt signals, 2-45, 9-3
- interrupt, clock, and power management, 2-44
- IRDY, 2-39, 7-3
- ISA_MASTER, 2-6, 2-43, 6-71, 7-27
- JTAG signals, 2-47, C-2
- L2 cache interface signals, 2-22
- LBCLAIM, 2-6, 2-16, 2-49, 4-23
- LOCK, 2-40, 7-23
- MA_n, 2-32
- MCP, 2-16, 4-21, 9-3
- MDLE, 2-33, 6-4
- MEMACK, 2-44, 7-27
- memory interface, 2-29
- NMI, 2-45, 9-5, 9-12
- PAR (PCI parity), 2-40, 7-26
- PAR_n (data parity/ECC), 2-33
- PCI interface signals, 2-36
- PCI sideband signals, 2-43, 7-27
- PERR, 2-41, 7-27, 9-5
- PLL_n, 2-49
- PPEN, 2-33, 6-8
- QACK, 2-46, 3-29, A-1
- QREQ, 2-46, 3-29, A-1
- RAS_n, 2-34, 6-10, 6-33
- RCS0 (ROM location configuration), 2-50, 6-2, 6-81
- RCS_n (ROM bank select), 2-34
- REQ, 2-41, 7-3
- RTC, 2-35, 6-33, A-5
- SDBA0, 2-35
- SDCAS, 2-35
- SDMA_n, 2-35
- SDRAS, 2-36
- SERR, 2-42, 7-27, 9-5
- signal connections, examples, 9-12
- signal groupings, 2-3
- signal timing examples, 6-33, 6-40
- STOP, 2-42, 7-8
- SUSPEND, 2-46
- SYSCLK, 2-47, 2-50
- TA, 2-17
- TBST, 2-18, 4-15
- TCK (JTAG test clock), 2-47, C-2
- TDI (JTAG test data input), 2-47, C-2
- TDO (JTAG test data output), 2-47, C-2
- TEA, 2-18, 4-21, 9-4
- TMS (JTAG test mode select), 2-48, C-2
- TOE, 2-25
- TRDY, 2-42, 7-4
- TRST (JTAG test reset), 2-48, C-2
- TS, 2-19
- TSIZ_n, 2-19, 4-15
- TT_n, 2-20, 4-12
- TV, 2-26, 5-10
- TWE, 2-26, 5-24
- WE, 2-36, 6-11
- WT, 2-20, 5-12
- XATS, 2-8, 2-21, 3-37, 4-22
- Single-beat operations, 4-20
- Single-beat transactions
 - SDRAM-based systems, 6-57, 6-67, 6-68
- Single-beat transfer, *see* Transfer
- Sleep mode
 - description, 1-7, A-4
 - memory refresh, 6-32
 - PCI bus parking, A-8
 - PCI interface support, 7-23
 - PMCR bit settings, 3-29
 - QREQ signal, 2-46, 3-29, A-1
- Snooping*
 - snoop push*, 4-18
 - snoop response, 4-14, 4-18, 8-9
- Split-bus transactions, 4-9
- SRAMs
 - asynchronous SRAMs, 5-6, 5-38
 - pipelined burst SRAMs, 5-4
 - synchronous burst SRAMs, 5-2, 5-30
 - timing examples, 5-23-5-41
 - two-bank support, 5-7
- Status register, PCI, 3-24, 7-16
- STOP signal, 2-42, 7-8
- Suspend mode
 - description, A-5
 - PCI bus parking, A-8
 - refresh, 2-35, 6-33, A-5
 - RTC signal, 2-35, 6-33, A-5
- SUSPEND signal, 2-46
- Synchronous burst SRAMs
 - CF_DOE bit, 3-67, 5-25
 - CF_WDATA bit, 3-67, 5-25
 - description, 5-2
 - L2 cache timing examples, 5-30
 - two-bank support, 5-7
- SYSCLK
 - input with internal multiples, 2-50
- SYSCLK (system clock) signal, 2-47, 2-50
- System reset
 - HRST signal, 2-45, 9-3, A-2
 - initialization sequence, 6-60
 - system reset interrupt, 9-3

T

- TA (transfer acknowledge) signal, 2-17
- Target-abort error, 7-13, 9-11
- Target-disconnect*, *see* PCI interface

- Target-initiated termination
 - description, 7-2, 7-12, 8-4
 - PCI status register, 7-13
 - TBST (transfer burst) signal, 2-18, 4-15
 - TCK (JTAG test clock) signal, 2-47, C-2
 - TDI (JTAG test data input) signal, 2-47, C-2
 - TDO (JTAG test data output) signal, 2-47, C-2
 - TEA (transfer error acknowledge) signal, 2-18, 4-21, 9-4
 - Termination
 - 60x address tenure, 4-8
 - 60x data tenure, 4-8
 - completion, PCI transaction, 7-12
 - master-abort, PCI, 7-12, 9-10
 - normal termination, 4-20
 - target-disconnect, PCI, 7-2, 7-12, 8-4
 - target-initiated termination, 7-13
 - termination by TEA, 4-21, 9-11
 - termination of PCI transaction, 7-11
 - timeout*, PCI transaction, 7-12
 - transfer termination*, 4-18
 - Timeout*, PCI transaction, 7-12
 - Timing diagrams
 - 60x single-beat/burst data transfers, 4-21
 - acronyms in DRAM timing diagrams, 6-17
 - burst read followed by burst write timing, 6-67
 - CBR refresh timing, SDRAM, 6-76
 - CF_L2_HIT_DELAY, 3-66, 4-7, 4-23, 5-25
 - CF_WMODE, 5-26
 - DRAM RTC refresh in suspend mode, 6-33
 - DRAM/EDO
 - interface timing, 6-16
 - interface timing with ECC, 6-26
 - refresh, 6-31
 - self-refresh in sleep and suspend modes, 6-33
 - L2 cache
 - burst read, 5-39
 - burst read line update, 5-40
 - burst write, 5-41
 - castout timing, 5-34
 - castout timing with no ARTRY, 5-35
 - hit following PCI read snoop, 5-36
 - invalidate following PCI read snoop, 5-38
 - push following PCI write snoop, 5-37
 - read hit timing, 5-30
 - update timing, 5-33
 - write hit timing, 5-32
 - legend for L2 interface timing, 5-30
 - ROM/Flash
 - burst read timing, 6-83
 - Flash memory write, 6-95
 - interface timing, 6-87
 - single-byte read, 6-87
 - SDRAM
 - burst read followed by burst write timing, 6-67
 - burst read timing, 6-65
 - burst write timing, 6-65
 - self-refresh entry, 6-77
 - self-refresh exit, 6-78
 - single-beat read, 6-67
 - single-beat write, 6-68
 - TMS (JTAG test mode select) signal, 2-48, C-2
 - TOE (tag output enable) signal, 2-25
 - Transactions
 - direct-access PCI configuration transaction, 7-21
 - fast back-to-back transactions, PCI bus, 7-14
 - PCI bus transactions, 7-9
 - PCI-to-DRAM
 - PCI reads from memory, 6-44
 - PCI writes to memory, 6-52
 - processor-to-DRAM memory, 6-33
 - processor burst reads from memory, 6-37
 - processor-to-SDRAM
 - burst write timing, 6-65
 - mode-set command timing, 6-69
 - single-beat read timing, 6-67
 - Transactions*
 - PCI transaction termination, 7-11
 - Transfer
 - 60x address/data tenure, 4-8
 - aligned data transfer, 60x, 4-16
 - transfer termination*, 4-18
 - TRDY (target ready) signal, 2-42, 7-3, 7-4, 7-11
 - TRST (JTAG test reset) signal, 2-48, C-2
 - TS (transfer start) signal, 2-19
 - TSIZ_n (transfer size) signals, 2-19, 4-15
 - TT_n (transfer type) signals, 2-20, 4-12
 - Turnaround cycle and PCI bus, 7-8
 - TV (tag valid) signal, 2-26, 5-10
 - TWE (tag write enable) signal, 2-26, 5-24
- ## W
- WE (write enable) signal, 2-36
 - Write command, SDRAM, 6-61
 - Write-back*
 - L2 cache response, 5-13
 - support for L2 cache operation, 5-2
 - write-back cache with MPC106, 5-2
 - Write-through*
 - L2 cache response, 5-19
 - support for L2 cache operation, 5-2
 - write-through cache with MPC106, 5-2
 - WT (write-through) signal, 2-20, 5-12



X-X

X

XATS (extended address transfer) signal, 2-21, 3-37,
4-22

Overview	1
Signal Descriptions	2
Device Programming	3
Processor Bus Interface	4
Secondary Cache Interface	5
Memory Interface	6
PCI Bus Interface	7
Internal Control	8
Error Handling	9
Performance Monitor	10
Power Management	A
Bit and Byte Ordering	B
JTAG/Testing Support	C
Initialization Example	D
Revision History	E
Glossary of Terms and Abbreviations	GLO
Index	IND

1 Overview

2 Signal Descriptions

3 Device Programming

4 Processor Bus Interface

5 Secondary Cache Interface

6 Memory Interface

7 PCI Bus Interface

8 Internal Control

9 Error Handling

10 Performance Monitor

A Power Management

B Bit and Byte Ordering

C JTAG/Testing Support

D Initialization Example

E Revision History

GLO Glossary of Terms and Abbreviations

IND Index