MAC7100 Microcontroller Family Reference Manual

Devices Supported:

MAC7101 MAC7106
MAC7111 MAC7112
MAC7116 MAC7121
MAC7122 MAC7126
MAC7131 MAC7136
MAC7141 MAC7142

MAC7100RM
Rev. 1.0
10/2004

ARM POWERED®

Preliminary

freescale™ semiconductor

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Technical Information Center
3-20-1, Minami-Azabu, Minato-ku
Tokyo 106-0047, Japan
0120 191014 or +81 3 3440 3569
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 26668334
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

MAC7100RM
Rev. 1.0
10/2004

Introduction	1
Signal Description	2
Voltage Regulator Module (VREG)	3
System Clocks Module (OSC and CRG)	4
Resets	5
Exceptions	6
Modes of Operation	7
Device Memory Map	8
ARM7TDMI-S™ Processor Core	9
Interrupt Controller Module (INTC)	10
Miscellaneous Control Module (MCM)	11
Enhanced Direct Memory Access Controller Module (eDMA)	12
External Interface Module (EIM)	13
Cross-Bar Switch Module (XBS)	14
Common Flash Module (CFM)	15
AMBA to IP Bus Bridge Module (AIPS)	16
DMA Channel Multiplexer Module (DMAMux)	17
Port Integration Module (PIM)	18
Analog-to-Digital Converter Module (ATD)	19
Enhanced Modular I/O Subsystem Module (eMIOS)	20
Enhanced Serial Communications Interface Module (eSCI)	21
Deserial Serial Peripheral Interface Module (DSPI)	22
Controller Area Network Module (FlexCAN)	23
Inter-Integrated Circuit Bus Module (I^2C)	24
Periodic Interrupt Timer Module (PIT)	25
System Services Module (SSM)	26
Debug Interface	A
A7S Nexus 2 Module	B
Register Memory Map Quick Reference	C
Mask Set Differences Summary	D

1	Introduction
2	Signal Description
3	Voltage Regulator Module (VREG)
4	System Clocks Module (OSC and CRG)
5	Resets
6	Exceptions
7	Modes of Operation
8	Device Memory Map
9	ARM7TDMI-S™ Processor Core
10	Interrupt Controller Module (INTC)
11	Miscellaneous Control Module (MCM)
12	Enhanced Direct Memory Access Controller Module (eDMA)
13	External Interface Module (EIM)
14	Cross-Bar Switch Module (XBS)
15	Common Flash Module (CFM)
16	AMBA to IP Bus Bridge Module (AIPS)
17	DMA Channel Multiplexer Module (DMAMux)
18	Port Integration Module (PIM)
19	Analog-to-Digital Converter Module (ATD)
20	Enhanced Modular I/O Subsystem Module (eMIOS)
21	Enhanced Serial Communications Interface Module (eSCI)
22	Deserial Serial Peripheral Interface Module (DSPI)
23	Controller Area Network Module (FlexCAN)
24	Inter-Integrated Circuit Bus Module (I^2C)
25	Periodic Interrupt Timer Module (PIT)
26	System Services Module (SSM)
A	Debug Interface
B	A7S Nexus 2 Module
C	Register Memory Map Quick Reference
D	Mask Set Differences Summary

Contents

Paragraph Number	Title	Page Number
	Tables	xxix
	Figures	xxxix
	Preface	
	Document Structure	li
	How To Use This Document	li
	Conventions	lii
	Terminology	liii
	Register Descriptions	lvii
	Revision History	
	Content Changes by Document Version	lix
	Chapter 1 Introduction	
1.1	Overview	1-1
1.2	Block Diagram	1-2
1.3	Features	1-4
1.4	Modes of Operation	1-9
	Chapter 2 Signal Description	
2.1	External Signal Description	2-11
2.1.1	Clocks and Control	2-11
2.1.1.1	EXTAL, XTAL — Oscillator	2-11
2.1.1.2	XFC — PLL Loop Filter	2-11
2.1.1.3	$\overline{\text{RESET}}$ — External Reset	2-11
2.1.1.4	TCK — Test Clock	2-11
2.1.1.5	TMS — Test Mode	2-11
2.1.1.6	TDI — Test Data In	2-11
2.1.1.7	TDO — Test Data Out	2-12
2.1.1.8	$\overline{\text{TA}}$ / $\overline{\text{AS}}$ — Transfer Acknowledge / Address Strobe	2-12
2.1.1.9	TEST — Factory Test	2-12

Contents

Paragraph Number	Title	Page Number
2.1.1.10	Configuration and Optional Control Signals	2-12
2.1.2	General Purpose / Peripheral I/O	2-12
2.1.2.1	Port A Signal Group	2-13
2.1.2.2	Port B Signal Group	2-15
2.1.2.3	Port C Signal Group	2-16
2.1.2.4	Port D Signal Group	2-16
2.1.2.5	Port E Signal Group	2-17
2.1.2.6	Port F Signal Group	2-18
2.1.2.7	Port G Signal Group	2-18
2.1.2.8	Port H Signal Group	2-20
2.1.2.9	Port I Signal Group	2-20
2.2	Power Supply, Bypass and Reference	2-20
2.2.1	$V_{DD}X$,$V_{SS}X$ — I/O Drivers Power and Ground	2-20
2.2.2	$V_{DD}R$, $V_{SS}R$ — Internal Voltage Regulator Supply	2-20
2.2.3	$V_{DD}A$, $V_{SS}A$ — Analog Reference Supply	2-21
2.2.4	$V_{DD}2.5$, $V_{SS}2.5$ — Core Power Supply Bypass	2-21
2.2.5	$V_{DD}PLL$, $V_{SS}PLL$ — PLL Power Supply Bypass	2-21
2.2.6	V_{RH}, V_{RL} — ATD Reference Voltage	2-21
2.3	Signal Properties Summary	2-22
2.4	Packaging Options	2-27

Chapter 3
Voltage Regulator Module (VREG)

3.1	Overview	3-29
3.2	Features	3-29
3.3	Modes of Operation	3-31
3.4	Signal Description	3-31
3.4.1	$V_{DD}R$, $V_{SS}R$ — Regulator Power Input	3-32
3.4.2	$V_{DD}A$, $V_{SS}A$ — Regulator Reference Input	3-32
3.4.3	$V_{DD}2.5$, $V_{SS}2.5$ — Regulator Output 1 (Core Logic)	3-32
3.4.4	$V_{DD}PLL$, $V_{SS}PLL$ — Regulator Output 2 (PLL)	3-32
3.5	Memory Map / Register Definition	3-32
3.5.1	Register Descriptions	3-33
3.5.1.1	VREG High Temperature Control Register (VREGHTCL)	3-33
3.5.1.2	VREG Control Register (VREGCTRL)	3-33
3.5.1.3	VREG Autonomous Periodic Interrupt Control Register (VREGAPICL)	3-34
3.5.1.4	VREG Autonomous Periodic Interrupt Trimming Register (VREGAPITR)	3-35
3.6	Functional Description	3-36
3.6.1	REG – Regulator Core	3-36

Contents

Paragraph Number	Title	Page Number
3.6.1.1	Full Performance Mode	3-36
3.6.1.2	Reduced Power Mode	3-37
3.6.2	LVD – Low Voltage Detect	3-37
3.6.3	POR – Power-On Reset	3-37
3.6.4	LVR – Low Voltage Reset	3-37
3.6.5	CTRL – Regulator Control	3-37
3.6.6	API – Autonomous Periodic Interrupt	3-37
3.6.7	Resets	3-38
3.6.7.1	Power-On Reset (POR)	3-38
3.6.7.2	Low-Voltage Reset (LVR)	3-38
3.6.8	Interrupts	3-38
3.6.8.1	LVI – Low Voltage Interrupt	3-38
3.6.8.2	API – Autonomous Periodic Interrupt	3-39
3.7	Initialization / Application Information	3-39
3.7.1	Circuit Board Layout	3-39

Chapter 4
System Clocks Module (OSC and CRG)

4.1	Overview	4-45
4.2	On-Chip Oscillator (OSC) Module	4-47
4.2.1	OSC Overview	4-47
4.2.2	OSC Features	4-47
4.2.3	OSC Modes of Operation	4-48
4.2.4	OSC Signal Description	4-48
4.2.4.1	V_{DD}PLL, V_{SS}PLL	4-48
4.2.4.2	EXTAL, XTAL	4-48
4.2.4.3	CLKOUT / $\overline{\text{XCLKS}}$	4-49
4.2.5	OSC Functional Description	4-50
4.2.5.1	Gain control	4-50
4.2.5.2	Clock Monitor	4-50
4.3	Clock and Reset Generator (CRG) Module	4-50
4.3.1	CRG Overview	4-50
4.3.2	CRG Features	4-51
4.3.3	CRG Modes of Operation	4-52
4.3.4	CRG Signal Description	4-52
4.3.4.1	XFC	4-52
4.3.4.2	$\overline{\text{RESET}}$	4-53
4.3.4.3	CLKOUT / $\overline{\text{XCLKS}}$	4-53
4.3.5	CRG Memory Map / Register Definition	4-53

Contents

Paragraph Number	Title	Page Number
4.3.5.1	CRG Synthesizer Register (SYNR)	4-54
4.3.5.2	CRG Reference Divider Register (REFDV)	4-54
4.3.5.3	CRG Flags Register (CRGFLG)	4-55
4.3.5.4	CRG Interrupt Enable Register (CRGINT)	4-56
4.3.5.5	CRG Clock Select Register (CLKSEL)	4-57
4.3.5.6	CRG PLL Control Register (PLLCTL)	4-58
4.3.5.7	CRG Stop/Doze Control Register (SDMCTL)	4-59
4.3.5.8	CRG BDM Control Register (BDMCTL)	4-60
4.3.6	CRG Functional Description	4-60
4.3.6.1	Phase Locked Loop (PLL)	4-60
4.3.6.2	System Clocks Generator	4-63
4.3.6.3	Clock Monitor (CM)	4-64
4.3.6.4	Clock Quality Checker	4-64
4.3.6.5	Software Watchdog Timer (SWT)	4-66
4.3.6.6	Real Time Interrupt (RTI)	4-66
4.3.6.7	Resets	4-66
4.3.6.8	Software Watchdog Timer (SWT) Reset	4-69
4.3.6.9	Interrupts	4-70
4.3.6.10	CRG Operating Mode Details	4-70
4.4	System Clocks Summary	4-81

Chapter 5
Resets

5.1	Effects of Reset	5-83
5.1.1	I/O pins	5-83
5.1.2	Memory	5-83
5.2	Keyboard Wake-up on Port Pins	5-83

Chapter 6
Exceptions

6.1	Exception Vector Assignments	6-85

Chapter 7
Modes of Operation

7.1	Chip Hardware Configuration Summary	7-87
7.1.1	MCU Mode Selection	7-87

Contents

Paragraph Number	Title	Page Number
7.1.1.1	Expanded Modes	7-88
7.1.1.2	Single-Chip Modes	7-88
7.1.1.3	Data Flash Boot Mode	7-88
7.1.2	Oscillator Type Selection	7-88
7.1.3	External Bus Interface Configuration	7-88
7.1.4	Nexus Port Configuration	7-88
7.2	Security	7-89
7.2.1	Securing the Microcontroller	7-89
7.2.2	Operation of the Secured Microcontroller	7-89
7.2.2.1	Secured Single-Chip Mode	7-89
7.2.2.2	Secured Expanded Mode	7-89
7.2.3	Unsecuring the Microcontroller	7-90
7.2.3.1	Backdoor Access Key	7-90
7.2.3.2	Lockout Recovery Procedure	7-90
7.3	Power Consumption Considerations	7-90
7.3.1	Run Mode	7-91
7.3.2	Doze Mode	7-91
7.3.3	Stop Mode	7-91
7.3.4	Pseudo-Stop Mode	7-91
7.4	Mode and Configuration Identification	7-91

Chapter 8
Device Memory Map

8.1	Memory Map Details	8-93
8.1.1	Normal Single-Chip Mode	8-95
8.1.2	Secured Single-Chip Mode	8-95
8.1.3	Normal Expanded Mode	8-96
8.1.4	Secured Expanded Mode	8-97
8.1.5	Normal/Secured Data Flash Boot Mode	8-97
8.1.6	Peripheral Bus Memory Map	8-99
8.2	Accessing Registers	8-100
8.2.1	32-Bit Register Accesses	8-100
8.2.2	16-Bit Register Accesses	8-100
8.2.3	8-Bit Register Accesses	8-100

Chapter 9
ARM7TDMI-S™ Processor Core

9.1	Overview	9-101

Contents

Paragraph Number	Title	Page Number

Chapter 10
Interrupt Controller Module (INTC)

10.1 Overview 10-103
10.2 Features 10-104
10.3 Review of ARM7™ Interrupt Architecture 10-104
10.4 Signal Description 10-105
10.4.1 XIRQ 10-105
10.4.2 IRQ 10-105
10.5 Memory Map / Register Definition 10-105
10.5.1 Register Descriptions 10-108
10.5.1.1 INTC Interrupt Pending Register (IPRH, IPRL) 10-108
10.5.1.2 INTC Interrupt Mask Register (IMRH, IMRL) 10-109
10.5.1.3 INTC Force Interrupt Register (INTFRCH, INTFRCL) 10-110
10.5.1.4 INTC Module Configuration Register (ICONFIG) 10-111
10.5.1.5 INTC Set Interrupt Mask Register (SIMR) 10-112
10.5.1.6 INTC Clear Interrupt Mask Register (CIMR) 10-112
10.5.1.7 INTC Current Level Mask Register (CLMASK) 10-113
10.5.1.8 INTC Saved Level Mask Register (SLMASK) 10-114
10.5.1.9 INTC Interrupt Control Registers (ICR*n*) 10-115
10.5.1.10 INTC IRQ Acknowledge Register (IRQIACK) 10-115
10.5.1.11 INTC FIQ Acknowledge Register (FIQIACK) 10-116
10.6 Functional Description 10-116
10.6.1 Interrupt Recognition 10-117
10.6.2 Interrupt Prioritization and Level Masking 10-117
10.6.3 Vector Generation During IACK 10-117
10.7 Initialization / Application Information 10-119
10.7.1 Typical Applications 10-119
10.7.2 Interrupt Service Routines 10-119
10.7.3 Performance 10-121

Chapter 11
Miscellaneous Control Module (MCM)

11.1 Overview 11-123
11.2 Features 11-123
11.3 Memory Map / Register Definition 11-123
11.3.1 Register Descriptions 11-124
11.3.1.1 MCM Processor Core Type Register (PCT) 11-124
11.3.1.2 MCM Device Revision Register (REV) 11-125

Contents

Paragraph Number	Title	Page Number
11.3.1.3	MCM XBS Master Configuration Register (AMC)	11-126
11.3.1.4	MCM XBS Slave Configuration Register (ASC)	11-126
11.3.1.5	MCM IPS On-Platform Module Configuration Register (IOPMC)	11-126
11.3.1.6	MCM Reset Status Register (MRSR)	11-127
11.3.1.7	MCM Wake-up Control Register (MWCR)	11-128
11.3.1.8	MCM Software Watchdog Timer Control Register (MSWTCR)	11-129
11.3.1.9	MCM Software Watchdog Timer Service Register (MSWTSR)	11-131
11.3.1.10	MCM Software Watchdog Timer Interrupt Register (MSWTIR)	11-132
11.3.1.11	MCM XBS Address Map Register (AAMR)	11-132
11.3.1.12	MCM Core Fault Address Register (CFADR)	11-135
11.3.1.13	MCM Core Fault Location Register (CFLOC)	11-136
11.3.1.14	MCM Core Fault Attributes Register (CFATR)	11-137
11.3.1.15	MCM Core Fault Data Register (CFDTR)	11-138
11.4	Initialization / Application Information	11-138
11.4.1	Using The PCT And REV Registers	11-138

Chapter 12
Enhanced Direct Memory Access Controller Module (eDMA)

12.1	Overview	12-141
12.2	Features	12-143
12.3	Memory Map / Register Definition	12-143
12.3.1	Register Descriptions	12-144
12.3.1.1	eDMA Control Register (DMACR)	12-144
12.3.1.2	eDMA Error Status Register (DMAES)	12-145
12.3.1.3	eDMA Enable Request Register (DMAERQ)	12-147
12.3.1.4	eDMA Enable Error Interrupt Registers (DMAEEI)	12-148
12.3.1.5	eDMA Set Enable Request Register (DMASERQ)	12-149
12.3.1.6	eDMA Clear Enable Request Register (DMACERQ)	12-149
12.3.1.7	eDMA Set Enable Error Interrupt Register (DMASEEI)	12-150
12.3.1.8	eDMA Clear Enable Error Interrupt Register (DMACEEI)	12-150
12.3.1.9	eDMA Clear Interrupt Request Register (DMACINT)	12-151
12.3.1.10	eDMA Clear Error Register (DMACERR)	12-152
12.3.1.11	eDMA Set START Bit Register (DMASSRT)	12-152
12.3.1.12	eDMA Clear DONE Status Register (DMACDNE)	12-153
12.3.1.13	eDMA Interrupt Request Register (DMAINT)	12-153
12.3.1.14	eDMA Error Register (DMAERR)	12-154
12.3.1.15	eDMA Channel Priority Registers (DCHPRI*n*)	12-155
12.3.1.16	Transfer Control Descriptors (TCD*n*)	12-156
12.4	Functional Description	12-163

Contents

Paragraph Number	Title	Page Number
12.4.1	eDMA Microarchitecture	12-163
12.4.2	eDMA Basic Data Flow	12-164
12.5	Initialization / Application Information	12-167
12.5.1	eDMA Transfer Control Descriptor Header File Example	12-167
12.5.2	eDMA Basic Channel Operation	12-168
12.5.3	eDMA Initialization Sequence	12-171
12.5.4	eDMA Programming Errors	12-173
12.5.5	eDMA Arbitration Mode Considerations	12-173
12.5.5.1	Fixed-Priority Arbitration	12-174
12.5.5.2	Round-Robin Arbitration	12-174
12.5.6	eDMA Transfers	12-174
12.5.6.1	Single Request	12-174
12.5.6.2	Multiple Requests	12-175
12.5.6.3	Modulo Operation	12-176
12.5.7	eDMA TCD*n* Status Monitoring	12-177
12.5.7.1	Minor Loop Complete	12-177
12.5.7.2	Active Channel TCD*n* Reads	12-178
12.5.7.3	Preemption Status	12-178
12.5.8	Channel Linking	12-178
12.5.9	Dynamic Channel Linking and Scatter/Gather Operation	12-180

Chapter 13
External Interface Module (EIM)

13.1	Overview	13-181
13.2	Features	13-182
13.3	Modes of Operation	13-182
13.4	Signal Description	13-183
13.4.1	CLKOUT	13-183
13.4.2	Address Bus (ADDR[21:0])	13-183
13.4.3	Data Bus (DATA[15:0])	13-183
13.4.4	Read/Write Signal (R/$\overline{W}$)	13-184
13.4.5	Address Strobe ($\overline{AS}$)	13-184
13.4.6	Transfer Acknowledge ($\overline{TA}$)	13-184
13.4.7	Output Enable ($\overline{OE}$)	13-184
13.4.8	Chip Selects ($\overline{CS}$[2:0])	13-184
13.4.9	Byte Selects ($\overline{BS}$[1:0])	13-184
13.5	Memory Map / Register Definition	13-185
13.5.1	Register Descriptions	13-185
13.5.1.1	EIM Chip Select Address Registers (CSAR*n*)	13-185

Contents

Paragraph Number	Title	Page Number
13.5.1.2	EIM Chip Select Mask Registers (CSMR*n*)	13-186
13.5.1.3	EIM Chip Select Control Registers (CSCR*n*)	13-187
13.6	Functional Description	13-189
13.6.1	Chip Select Operation	13-189
13.6.1.1	8- and 16-Bit Port Sizing	13-191
13.6.1.2	Global Chip Select	13-191
13.6.2	External Bus Operation	13-192
13.6.2.1	Data Transfers	13-192
13.6.2.2	Bus Cycle Execution	13-193
13.6.2.3	Data Transfer Cycle States	13-194
13.6.2.4	Read Cycle	13-195
13.6.2.5	Write Cycle	13-196
13.6.2.6	Fast Termination Cycles	13-197
13.6.2.7	Back-to-Back Bus Cycles	13-198
13.6.2.8	Burst Cycles	13-198
13.7	Initialization / Application Information	13-201
13.7.1	Using Global Chip Select Mode	13-201
13.7.2	Configuring Chip Selects	13-201
13.7.3	Dynamic Chip Select Configuration	13-202

Chapter 14
Cross-Bar Switch Module (XBS)

14.1	Overview	14-203
14.2	Features	14-203
14.3	Modes of Operation	14-204
14.4	Memory Map / Register Definition	14-204
14.4.1	Register Descriptions	14-205
14.4.1.1	XBS Priority Registers (PR_*port*)	14-205
14.4.1.2	XBS Control Registers (CR_*port*)	14-206
14.5	Functional Description	14-207
14.5.1	Arbitration	14-207
14.5.1.1	Fixed-Priority Operation	14-207
14.5.1.2	Round-Robin Priority Operation	14-208
14.5.1.3	Priority Assignment	14-208
14.6	Initialization / Application Information	14-208

Contents

Paragraph Number	Title	Page Number

Chapter 15
Common Flash Module (CFM)

15.1 Overview 15-209
15.2 Features 15-211
15.3 Memory Map / Register Definition 15-212
15.3.1 Register Descriptions 15-214
15.3.1.1 CFM Module Configuration Register (CFMMCR) 15-214
15.3.1.2 CFM Clock Divider Register (CFMCLKD) 15-216
15.3.1.3 CFM Security Register (CFMSEC) 15-217
15.3.1.4 CFM Program Flash Protection Register (CFMPROT) 15-218
15.3.1.5 CFM Data Flash Protection Register (CFMDFPROT) 15-220
15.3.1.6 CFM Program Flash Supervisor Access Register (CFMSACC) 15-221
15.3.1.7 CFM Data Flash Supervisor Access Register (CFMDFSACC) 15-221
15.3.1.8 CFM Program Flash Data Access Register (CFMDACC) 15-222
15.3.1.9 CFM Data Flash Data Access Register (CFMDFDACC) 15-223
15.3.1.10 CFM User Status Register (CFMUSTAT) 15-223
15.3.1.11 CFM Command Register (CFMCMD) 15-225
15.3.1.12 CFM Data Registers (CFMDATA1/0) 15-225
15.3.1.13 CFM Disable Upper Block Register (CFMDISU) 15-226
15.3.1.14 CFM Clock Select Register (CFMCLKSEL) 15-227
15.4 Functional Description 15-227
15.4.1 Flash Normal Mode 15-227
15.4.1.1 Read Operation 15-228
15.4.1.2 Write Operation 15-228
15.4.1.3 Command Launch Sequence 15-228
15.4.1.4 Initializing the CFMCLKD Register 15-229
15.4.1.5 Program, Erase, and Verify Operations 15-231
15.4.1.6 Flash Normal Mode Illegal Operations 15-246
15.4.1.7 Stop Mode 15-246
15.4.2 Flash Security Operation 15-247
15.4.2.1 Backdoor Access Sequence 15-247
15.4.2.2 Blank Check 15-248
15.4.2.3 JTAG Lockout Recovery 15-248
15.5 Initialization / Application Information 15-248
15.5.1 Using The Data Signature Command 15-248

Contents

Paragraph Number	Title	Page Number

Chapter 16
AMBA to IP Bus Bridge Module (AIPS)

16.1 Overview 16-251
16.2 Features 16-251
16.3 Modes of Operation 16-251
16.4 Memory Map / Register Definition 16-253
16.4.1 Register Descriptions 16-255
16.4.1.1 AIPS Master Protection Registers (MPR*x*) 16-255
16.4.1.2 AIPS Peripheral Access Control Registers (PACR*x*) 16-255
16.4.1.3 AIPS Off-Platform Peripheral Access Control Registers (OPACR*x*) 16-256
16.5 Functional Description 16-257
16.5.1 Access Protections 16-257
16.5.2 Access Support 16-257
16.5.3 Read Cycles 16-257
16.5.4 Write Cycles 16-257
16.5.5 Aborted Cycles 16-257
16.6 Initialization / Application Information 16-258

Chapter 17
DMA Channel Multiplexer Module (DMAMux)

17.1 Overview 17-259
17.2 Features 17-259
17.3 Modes of Operation 17-260
17.4 Memory Map / Register Definition 17-260
17.4.1 Register Descriptions 17-261
17.4.1.1 DMAMux Channel Configuration Registers (CHCONFIG*n*) 17-261
17.5 Functional Description 17-262
17.5.1 eDMA Channels 0 to 7 17-263
17.5.2 eDMA Channels 8 to 15 17-264
17.5.3 Always Enabled DMA Request Sources 17-265
17.6 Initialization / Application Information 17-266
17.6.1 Simple Setup 17-266
17.6.1.1 Configure eDMA Channel 0 to Service eSCI_A Transmit Requests 17-266
17.6.2 Using the “Always Enabled” Feature to Periodically Drive GPIO Pins 17-267
17.6.3 Disabling a Source 17-267
17.6.4 Enable Source With Periodic Triggering 17-268
17.6.4.1 DSPI Channel Configured For Periodic Service 17-268
17.6.5 Enable Source With Transparent Triggering 17-268

Contents

Paragraph Number	Title	Page Number
17.6.5.1	DSPI Channel Configured for Immediate Service	17-269
17.6.6	Switching DMA Request Source to eDMA Channel Assignment	17-269
17.6.6.1	Switch eDMA Channel 9 from DSPI_A Transmit to eSCI_D Transmit	17-270

Chapter 18
Port Integration Module (PIM)

18.1	Overview	18-271
18.2	Features	18-273
18.3	Modes of Operation	18-273
18.4	Signal Description	18-273
18.5	Memory Map / Register Definition	18-276
18.5.1	Register Descriptions	18-286
18.5.1.1	PIM Port *x* Pin Configuration Registers (CONFIG*n*_*x*)	18-286
18.5.1.2	PIM Port *x* Interrupt Flag Register (PORTIFR_*x*)	18-287
18.5.1.3	PIM Port *x* Data Register (PORTDATA_*x*)	18-288
18.5.1.4	PIM Port *x* Input Register (PORTIR_*x*)	18-288
18.5.1.5	PIM Port *x* Pin Data Registers (PINDATA*n*_*x*)	18-289
18.5.1.6	PIM Global Interrupt Status Register (GLBINT)	18-289
18.5.1.7	PIM Global Configuration Register (PIMCONFIG)	18-290
18.5.1.8	PIM Configure TDI Pin Register (CONFIG_TDI)	18-291
18.5.1.9	PIM Configure TDO Pin Register (CONFIG_TDO)	18-292
18.5.1.10	PIM Configure TMS Pin Register (CONFIG_TMS)	18-293
18.5.1.11	PIM Configure TCK Pin Register (CONFIG_TCK)	18-294
18.5.1.12	PIM Configure $\overline{\text{TA}}$ / $\overline{\text{AS}}$ Pin Register (CONFIG_TA)	18-294
18.5.1.13	PIM Port *x*/*x* 32-bit Input Registers (PORT32IR_*xx*)	18-295
18.6	Functional Description	18-296
18.6.1	Reset	18-296
18.6.2	Peripheral Mode	18-296
18.6.3	General Purpose Input Mode	18-298
18.6.3.1	Interrupts	18-299
18.6.4	General Purpose Output Mode	18-301
18.7	Initialization / Application Information	18-303
18.7.1	Using a Pin in Peripheral Mode	18-303
18.7.1.1	PIM Example — Enable the I^2C module	18-303
18.7.2	Using a Pin in GPIO Mode	18-304
18.7.2.1	GPIO Mode Initialization	18-304
18.7.2.2	PIM Example — Use PB[1:0] in GPO/GPI Mode	18-304
18.7.2.3	Accessing Data	18-305
18.7.2.4	Using Port Interrupts	18-309

Contents

Paragraph Number	Title	Page Number
18.7.3	PD2 / CLKOUT Configuration	18-313
18.7.3.1	Using PD2 GPI Functionality	18-313
18.7.3.2	Using CLKOUT Functionality	18-313
18.7.4	$\overline{TA}$ / $\overline{AS}$ Configuration	18-314
18.7.5	E-ICE JTAG Port Configuration	18-314
18.7.6	Using the PIMCONFIG Register	18-314
18.7.6.1	Using the PORTHSEL Bit	18-315
18.7.6.2	Using the EIMCLKEN Bit	18-315
18.7.7	Minimizing Power Consumption	18-315

Chapter 19
Analog-to-Digital Converter Module (ATD)

19.1	Overview	19-317
19.2	Features	19-318
19.3	Modes of Operation	19-319
19.4	Signal Description	19-319
19.4.1	AN*n_x*	19-320
19.4.2	V_{RH} / V_{RL}	19-320
19.4.3	$V_{DD}A$ / $V_{SS}A$	19-320
19.5	Memory Map / Register Definition	19-320
19.5.1	Register Descriptions	19-321
19.5.1.1	ATD Trigger Control Register (ATDTRIGCTL)	19-321
19.5.1.2	ATD External Trigger Channel Register (ATDETRIGCH)	19-322
19.5.1.3	ATD Prescaler Register (ATDPRE)	19-323
19.5.1.4	ATD Operating Mode Register (ATDMODE)	19-324
19.5.1.5	ATD Interrupt Register (ATDINT)	19-325
19.5.1.6	ATD Flag Register (ATDFLAG)	19-326
19.5.1.7	ATD Command Word Register (ATDCW)	19-328
19.5.1.8	ATD Result Register (ATDRR)	19-330
19.6	Functional Description	19-333
19.6.1	General	19-333
19.6.2	Analog Sub-Module	19-333
19.6.2.1	Analog Input Multiplexer	19-333
19.6.2.2	Sample Buffer Amplifier	19-333
19.6.2.3	Sample and Hold Machine	19-334
19.6.2.4	DAC	19-334
19.6.2.5	Comparator	19-334
19.6.2.6	Schmitt Trigger	19-334
19.6.3	Digital Sub-Module	19-334

Contents

Paragraph Number	Title	Page Number
19.6.3.1	Mode / Timing Control	19-334
19.6.3.2	Clock Prescaler	19-334
19.6.3.3	IPS Bus Interface	19-335
19.6.3.4	SYSTRIG0, SYSTRIG1, External Trigger Input	19-335
19.6.4	ATD Operating Mode Details	19-335
19.6.4.1	ATD Normal Mode	19-335
19.6.4.2	ATD Debug Mode	19-335
19.6.4.3	ATD Disabled Mode	19-336
19.6.4.4	ATD Doze Mode	19-336
19.6.4.5	ATD Stop Mode	19-337
19.6.5	Conversion process	19-337
19.6.6	Reset	19-340
19.6.7	Interrupts	19-340
19.7	Initialization / Application Information	19-341
19.7.1	ATD Initialization Sequence	19-341
19.7.2	ATD Example 1 — Simple Conversion	19-341
19.7.3	ATD Example 2 — Three Consecutive Conversions	19-342
19.7.4	ATD Example 3 — Interrupted Continuous Conversion	19-343
19.7.5	ATD Example 4 — Edge Triggered Conversion	19-343
19.7.6	ATD Example 5 — Level Triggered Conversion	19-344
19.7.7	ATD Example 6 — Using External Triggers	19-345
19.7.8	ATD Example 7 — Using System Triggers	19-345
19.7.9	Conversion Mechanism — CWCH, CWNF, CWGI and CWSC Bits	19-346
19.7.10	Conversion Mechanism — CWSL, CWSB and CW8 Bits	19-347
19.7.11	Measuring Internal Reference Voltages	19-348

Chapter 20
Enhanced Modular I/O Subsystem Module (eMIOS)

20.1	Overview	20-351
20.2	Features	20-352
20.3	Modes of Operation	20-352
20.4	Signal Description	20-353
20.4.1	emiosi*n* — eMIOS Unified Channel *n* Input Signal	20-353
20.4.2	emioso*n* — eMIOS Unified Channel *n* Output Signal	20-353
20.5	Memory Map / Register Definition	20-353
20.5.1	Register Descriptions	20-354
20.5.1.1	eMIOS Module Configuration Register (MCR)	20-354
20.5.1.2	eMIOS Global Flag Register (GFLAG)	20-355
20.5.1.3	eMIOS Output Update Disable Register (OUDIS)	20-356

Contents

Paragraph Number	Title	Page Number
20.5.1.4	eMIOS Channel Disable Register (UCDIS)	20-356
20.5.1.5	eMIOS Channel A Data Registers (UCA*n*)	20-357
20.5.1.6	eMIOS Channel B Data Registers (UCB*n*)	20-358
20.5.1.7	eMIOS Channel Counter Registers (UCCNT*n*)	20-359
20.5.1.8	eMIOS Channel Control Registers (UCCR*n*)	20-359
20.5.1.9	eMIOS Channel Status Registers (UCSR*n*)	20-363
20.6	Functional Description	20-364
20.6.1	eMIOS Operating Mode Details	20-364
20.6.1.1	eMIOS Normal Mode	20-364
20.6.1.2	eMIOS Debug Mode	20-365
20.6.1.3	eMIOS Disabled Mode	20-365
20.6.1.4	eMIOS Doze Mode	20-365
20.6.1.5	eMIOS Stop Mode	20-365
20.6.2	IP Bus Interface Unit (BIU)	20-365
20.6.2.1	Effect of Debug Mode on the BIU	20-366
20.6.3	Global Clock Prescaler (GCP) Submodule	20-366
20.6.3.1	Effect of Debug Mode on the GCP	20-366
20.6.4	Unified Channel (UC)	20-366
20.6.4.1	Effect of Debug Mode on Unified Channels	20-366
20.6.5	Clock Prescaler (CP)	20-367
20.6.6	Input Programmable Filter (IPF)	20-368
20.6.7	UC Modes of Operation	20-368
20.6.7.1	General Purpose Input/Output (GPIO) Mode	20-369
20.6.7.2	Single Action Input Capture (SAIC) Mode	20-369
20.6.7.3	Single Action Output Compare (SAOC) Mode	20-369
20.6.7.4	Input Pulse Width Measurement (IPWM) Mode	20-370
20.6.7.5	Input Period Measurement (IPM) Mode	20-371
20.6.7.6	Double Action Output Compare (DAOC) Mode	20-372
20.6.7.7	Pulse/Edge Accumulation (PEA) Mode	20-373
20.6.7.8	Pulse/Edge Counting (PEC) Mode	20-375
20.6.7.9	Quadrature Decode (QDEC) Mode	20-376
20.6.7.10	Windowed Programmable Time Accumulation (WPTA) Mode	20-378
20.6.7.11	Modulus Counter (MC) Mode	20-379
20.6.7.12	Output Pulse Width and Frequency Modulation (OPWFM) Mode	20-380
20.6.7.13	Center Aligned Output Pulse Width Modulation (OPWMC) Mode	20-383
20.6.7.14	Output Pulse Width Modulation (OPWM) Mode	20-385
20.6.7.15	Modulus Counter, Buffered (MCB) Mode	20-386
20.6.7.16	Output Pulse Width and Frequency Modulation, Buffered (OPWFMB) Mode	20-389
20.6.7.17	Center Aligned Output Pulse Width Modulation, Buffered (OPWMCB) Mode	20-393
20.6.7.18	Output Pulse Width Modulation, Buffered (OPWMB) Mode	20-398
20.7	Initialization / Application Information	20-401

Contents

Paragraph Number	Title	Page Number
20.7.1	Changing UC Mode Considerations	20-402
20.7.2	Correlated Output Signal Generation	20-402
20.7.3	Time Base Generation	20-402

Chapter 21
Enhanced Serial Communications Interface Module (eSCI)

21.1	Overview	21-405
21.2	Features	21-406
21.2.1	LIN support	21-406
21.3	Modes of Operation	21-407
21.4	Signal Description	21-407
21.4.1	TXD_*x* — SCI Transmit Data	21-407
21.4.2	RXD_*x* — SCI Receive Data	21-407
21.5	Memory Map / Register Definition	21-407
21.5.1	Register Descriptions	21-408
21.5.1.1	eSCI Baud Rate Registers (ESCIBDH, ESCIBDH)	21-408
21.5.1.2	eSCI Control Registers (ESCICR1 through ESCICR4)	21-409
21.5.1.3	eSCI Data Registers (ESCIDRH, ESCIDRL)	21-414
21.5.1.4	eSCI Status Registers (ESCISR1, ESCISR2)	21-415
21.5.1.5	LIN Status Registers (LINSTAT1, LINSTAT2)	21-417
21.5.1.6	LIN Control Registers (LINCTRL1, LINCTRL2, LINCTRL3)	21-418
21.5.1.7	LIN TX Register (LINTX)	21-420
21.5.1.8	LIN RX Register (LINRX)	21-423
21.5.1.9	LIN CRC Polynomial Registers (LINCRCP1, LINCRCP2)	21-423
21.6	Functional Description	21-424
21.6.1	Overview	21-424
21.6.2	Data Format	21-424
21.6.3	Baud Rate Generation	21-425
21.6.4	Transmitter	21-426
21.6.5	Transmitter Character Length	21-427
21.6.6	Character Transmission	21-427
21.6.7	Break Characters	21-428
21.6.8	Idle Characters	21-429
21.6.9	Fast Bit Error Detection	21-429
21.6.10	Receiver	21-430
21.6.11	Receiver Character Length	21-430
21.6.12	Character Reception	21-431
21.6.13	Data Sampling	21-431
21.6.14	Framing Errors	21-435

Contents

Paragraph Number	Title	Page Number
21.6.15	Baud Rate Tolerance	21-435
21.6.15.1	Slow Data Tolerance	21-436
21.6.15.2	Fast Data Tolerance	21-437
21.6.16	Receiver Wake-up	21-437
21.6.16.1	Idle Input Line Wake-up (WAKE = 0)	21-438
21.6.16.2	Address Mark Wake-up (WAKE = 1)	21-438
21.6.17	Single-Wire Operation	21-438
21.6.18	Loop Operation	21-439
21.6.19	eSCI Operating Mode Details	21-439
21.6.19.1	Run Mode	21-439
21.6.19.2	Doze Mode	21-439
21.6.19.3	Module Disable	21-440
21.6.19.4	Stop Mode	21-440
21.6.20	Interrupt Operation	21-440
21.6.20.1	Interrupt Flags and Masks	21-440
21.7	Initialization / Application Information	21-441
21.7.1	Using the eSCI in 9-bit Data Mode	21-441
21.7.2	Using the LIN hardware	21-442
21.7.2.1	LIN Setup	21-443
21.7.2.2	Features of the LIN Hardware	21-443
21.7.2.3	Generating a TX frame	21-444
21.7.2.4	Generating an RX frame	21-445
21.7.2.5	LIN Error Handling	21-446
21.7.2.6	LIN Wake-up	21-447
21.7.2.7	System Wake-up on LIN Bus Activity	21-447

Chapter 22
Deserial Serial Peripheral Interface Module (DSPI)

22.1	Overview	22-449
22.2	Features	22-450
22.3	Modes of Operation	22-451
22.4	Signal Description	22-451
22.4.1	PCS0_*x* / $\overline{\text{SS}}$_*x* — Peripheral Chip Select or Slave Select	22-452
22.4.2	PCS[1:4, 6:7]_*x* — Peripheral Chip Selects 1–4, 6–7	22-452
22.4.3	PCS5_*x* / $\overline{\text{PCSS}}$_*x* — Peripheral Chip Select 5 or Chip Select Strobe	22-452
22.4.4	SIN_*x* — Serial Input	22-452
22.4.5	SOUT_*x* — Serial Output	22-452
22.4.6	SCK_*x* — Serial Clock	22-452
22.5	Memory Map / Register Definition	22-452

Contents

Paragraph Number	Title	Page Number
22.5.1	Register Descriptions	22-453
22.5.1.1	DSPI Module Configuration Register (DSPI*x*_MCR)	22-453
22.5.1.2	DSPI Transfer Count Register (DSPI*x*_TCR)	22-456
22.5.1.3	DSPI Clock and Transfer Attributes Registers (DSPI*x*_CTAR*n*)	22-457
22.5.1.4	DSPI Status Register (DSPI*x*_SR)	22-462
22.5.1.5	DSPI DMA/Interrupt Request Select / Enable Register (DSPI*x*_RSER)	22-464
22.5.1.6	DSPI PUSH TX FIFO Register (DSPI*x*_PUSHR)	22-466
22.5.1.7	DSPI POP RX FIFO Register (DSPI*x*_POPR)	22-467
22.5.1.8	DSPI Transmit FIFO Registers (DSPI*x*_TXFR*n*)	22-468
22.5.1.9	DSPI Receive FIFO Registers (DSPI*x*_RXFR*n*)	22-468
22.6	Functional Description	22-469
22.6.1	DSPI Operating Mode Details	22-470
22.6.1.1	DSPI Master Mode	22-470
22.6.1.2	DSPI Slave Mode	22-471
22.6.1.3	DSPI Disabled Mode	22-471
22.6.1.4	DSPI Stop Mode	22-471
22.6.1.5	DSPI Debug Mode	22-471
22.6.2	Start and Stop of DSPI Transfers	22-472
22.6.3	Serial Peripheral Interface (SPI) Configuration	22-472
22.6.3.1	Master Mode	22-473
22.6.3.2	Slave Mode	22-473
22.6.3.3	FIFO Disable Operation	22-473
22.6.3.4	Transmit First In First Out (TX FIFO) Buffering Mechanism	22-473
22.6.3.5	Receive First In First Out (RX FIFO) Buffering Mechanism	22-474
22.6.4	DSPI Baud Rate and Clock Delay Generation	22-475
22.6.4.1	Baud Rate Generator	22-475
22.6.4.2	PCS to SCK Delay (t_{CSC})	22-476
22.6.4.3	After SCK Delay (t_{ASC})	22-476
22.6.4.4	Delay after Transfer (t_{DT})	22-476
22.6.4.5	Peripheral Chip Select Strobe Enable ($\overline{PCSS}$)	22-477
22.6.5	Transfer Formats	22-478
22.6.5.1	Classic SPI Transfer Format (CPHA = 0)	22-479
22.6.5.2	Classic SPI Transfer Format (CPHA = 1)	22-479
22.6.5.3	Modified SPI Transfer Format (MTFE = 1, CPHA = 0)	22-480
22.6.5.4	Modified SPI Transfer Format (MTFE = 1, CPHA = 1)	22-481
22.6.5.5	Continuous Selection Format	22-482
22.6.6	Continuous Serial Communications Clock	22-483
22.6.7	Interrupts/DMA Requests	22-484
22.6.7.1	End of Queue Interrupt Request	22-485
22.6.7.2	Transmit FIFO Fill Interrupt or DMA Request	22-485
22.6.7.3	Transfer Complete Interrupt Request	22-485

Contents

Paragraph Number	Title	Page Number
22.6.7.4	Transmit FIFO Underflow Interrupt Request	22-485
22.6.7.5	Receive FIFO Drain Interrupt or DMA Request	22-485
22.6.7.6	Receive FIFO Overflow Interrupt Request	22-486
22.7	Initialization / Application Information	22-486
22.7.1	Changing Queues	22-486
22.7.2	Baud Rate Settings	22-487
22.7.3	Delay Settings	22-487
22.7.4	Calculation of FIFO Pointer Addresses	22-488
22.7.4.1	Address Calculation for First-in Entry / Last-in Entry in RX FIFO	22-489

Chapter 23
Controller Area Network Module (FlexCAN)

23.1	Overview	23-491
23.1.1	References	23-491
23.2	Features	23-492
23.3	Modes of Operation	23-493
23.4	Signal Description	23-494
23.4.1	CNRX_*x*	23-494
23.4.2	CNTX_*x*	23-494
23.5	Memory Map / Register Definition	23-494
23.5.1	Message Buffer Structure	23-495
23.5.2	Register Descriptions	23-498
23.5.2.1	FlexCAN Module Configuration Register (MCR)	23-498
23.5.2.2	FlexCAN Control Register (CTRL)	23-501
23.5.2.3	FlexCAN Timer Register (TIMER)	23-504
23.5.2.4	Rx Mask Registers	23-504
23.5.2.5	FlexCAN Error Counter Register (ECR)	23-506
23.5.2.6	FlexCAN Error and Status Register (ESR)	23-507
23.5.2.7	FlexCAN Interrupt Mask Register (IMASK)	23-510
23.5.2.8	FlexCAN Interrupt Flags Register (IFLAG)	23-511
23.6	Functional Description	23-511
23.6.1	Overview	23-511
23.6.2	Transmit Process	23-512
23.6.3	Arbitration Process	23-512
23.6.4	Receive Process	23-513
23.6.5	Matching Process	23-514
23.6.6	Data Coherence	23-514
23.6.6.1	Message Buffer Deactivation	23-514
23.6.6.2	Message Buffer Lock Mechanism	23-515

Contents

Paragraph Number	Title	Page Number
23.6.7	CAN Protocol Related Features	23-516
23.6.7.1	Remote Frames	23-516
23.6.7.2	Overload Frames	23-516
23.6.7.3	Time Stamp	23-516
23.6.7.4	Protocol Timing	23-517
23.6.7.5	Arbitration and Matching Timing	23-518
23.6.8	FlexCAN Operating Mode Details	23-519
23.6.8.1	Freeze Mode	23-519
23.6.8.2	Module Disabled Mode	23-520
23.6.8.3	Doze Mode	23-520
23.6.8.4	Stop Mode	23-521
23.6.9	Interrupts	23-522
23.6.10	Bus Interface	23-523
23.7	Initialization / Application Information	23-523
23.7.1	FlexCAN Initialization Sequence	23-523
23.7.2	FlexCAN Addressing and RAM Size Configurations	23-524

Chapter 24
Inter-Integrated Circuit Bus Module (I^2C)

24.1	Overview	24-525
24.2	Features	24-525
24.2.1	DMA Request Interface	24-526
24.3	Modes of Operation	24-526
24.4	Signal Description	24-526
24.4.1	SCL	24-526
24.4.2	SDA	24-526
24.5	Memory Map / Register Definition	24-527
24.5.1	Register Descriptions	24-527
24.5.1.1	I^2C Bus Address Register (IBAD)	24-527
24.5.1.2	I^2C Bus Frequency Divider Register (IBFD)	24-528
24.5.1.3	I^2C Bus Control Register (IBCR)	24-533
24.5.1.4	I^2C Bus Status Register (IBSR)	24-534
24.5.1.5	I^2C Bus Data I/O Register (IBDR)	24-535
24.6	Functional Description	24-536
24.6.1	I^2C Operating Mode Details	24-536
24.6.1.1	I^2C Module Normal Mode	24-536
24.6.1.2	I^2C Module Debug Mode	24-536
24.6.1.3	I^2C Module Disabled Mode	24-536
24.6.1.4	I^2C Module Doze Mode	24-537

Contents

Paragraph Number	Title	Page Number
24.6.1.5	I²C Module Stop Mode	24-537
24.6.2	I²C Bus Protocol	24-537
24.6.2.1	START Signal	24-538
24.6.2.2	Slave Address Transmission	24-538
24.6.2.3	Data Transfer	24-538
24.6.2.4	STOP Signal	24-539
24.6.2.5	Repeated START Signal	24-539
24.6.2.6	Arbitration Procedure	24-539
24.6.2.7	Clock Synchronization	24-539
24.6.2.8	Handshaking	24-540
24.6.2.9	Clock Stretching	24-540
24.6.3	Interrupts	24-540
24.7	Initialization / Application Information	24-541
24.7.1	I²C Programming Examples	24-541
24.7.1.1	Initialization Sequence	24-541
24.7.1.2	Generation of START	24-541
24.7.1.3	Post-Transfer Software Response	24-542
24.7.1.4	Generation of STOP	24-543
24.7.1.5	Generation of Repeated START	24-544
24.7.1.6	Slave Mode	24-544
24.7.1.7	Arbitration Lost	24-544
24.7.2	DMA Application Information	24-544
24.7.2.1	DMA Mode, Master Transmit	24-546
24.7.2.2	DMA Mode, Master Receive	24-547

Chapter 25
Periodic Interrupt Timer Module (PIT)

25.1	Overview	25-549
25.2	Features	25-550
25.3	Modes of Operation	25-550
25.4	Memory Map / Register Definition	25-550
25.4.1	Register Descriptions	25-551
25.4.1.1	PIT RTI / Timer Load Value Registers (TLVAL*n*)	25-552
25.4.1.2	PIT RTI / Timer Current Value Registers (TVAL*n*)	25-553
25.4.1.3	PIT Interrupt Flags Register (PITFLG)	25-553
25.4.1.4	PIT Interrupt Enable Register (PITINTEN)	25-554
25.4.1.5	PIT Interrupt/DMA Select Register (PITINTSEL)	25-555
25.4.1.6	PIT Timer Enable Register (PITEN)	25-555
25.4.1.7	PIT Control Register (PITCTRL)	25-556

Contents

Paragraph Number	Title	Page Number
25.5	Functional Description	25-557
25.5.1	General	25-557
25.5.2	PIT Operating Mode Details	25-557
25.5.2.1	PIT Module Normal Mode	25-557
25.5.2.2	PIT Module Debug Mode	25-557
25.5.2.3	PIT Module Disabled Mode	25-558
25.5.2.4	PIT Module Doze Mode	25-558
25.5.2.5	PIT Module Stop Mode	25-558
25.5.3	Timer / RTI	25-558
25.5.4	Interrupts	25-559
25.5.4.1	General	25-559
25.5.4.2	Description of Interrupt Operation	25-560
25.6	Initialization / Application Information	25-560
25.6.1	Example Configuration	25-560

Chapter 26
System Services Module (SSM)

26.1	Overview	26-563
26.2	Features	26-563
26.3	Modes of Operation	26-563
26.4	Memory Map / Register Definition	26-564
26.4.1	Register Descriptions	26-564
26.4.1.1	SSM Current System Status Register (STATUS)	26-564
26.4.1.2	SSM System Memory Configuration Register (MEMCONFIG)	26-566
26.4.1.3	SSM Wake-up Source Register (WAKEUP)	26-567
26.4.1.4	SSM Error Configuration Register (ERROR)	26-568
26.4.1.5	SSM Port Select Register (PORTSEL)	26-569
26.4.1.6	SSM Debug Status Port Control Register (DEBUGPORT)	26-570
26.5	Functional Description	26-572
26.5.1	System Configuration / Status	26-572
26.5.2	Wake-up Source Identification	26-573
26.6	Initialization / Application Information	26-573
26.6.1	Using the STATUS Register	26-573
26.6.2	Using the MEMCONFIG Register	26-574
26.6.3	Using the ERROR Register	26-574
26.6.4	Using the DEBUGPORT Register	26-574
26.6.5	Using the WAKEUP Register	26-575
26.6.6	Using the PORTSEL Register	26-576

Contents

Paragraph Number	Title	Page Number

Appendix A
Debug Interface

A.1 Overview A-577
A.2 E-ICE Interface A-578
A.3 NEXUS Interface A-578

Appendix B
A7S Nexus 2 Module

B.1 Terminology and Introduction B-579
B.1.1 A7S Nexus 2 Overview B-579
B.1.2 Feature List B-580
B.1.3 Modes of Operation B-581
B.1.3.1 Reset B-581
B.1.3.2 Normal B-582
B.1.3.3 Disabled B-582
B.1.4 TCODEs supported B-582
B.2 External Signal Description B-584
B.2.1 Pins Implemented B-584
B.2.2 Pin Protocol B-586
B.2.3 Rules for Output Messages B-588
B.2.4 Examples B-588
B.3 A7S Nexus 2 Programmers Model B-590
B.3.1 JTAG ID Register (ID) B-590
B.3.2 A7S Nexus 2 Memory Map / Register Definition B-591
B.3.3 A7S Nexus 2 Register Descriptions B-591
B.3.3.1 Client Select Control Register (CSC) B-591
B.3.3.2 Development Control Register (DC) B-592
B.3.3.3 Development Status Register (DS) B-593
B.3.3.4 User Base Address Register (UBA) B-594
B.3.3.5 Read/Write Access Control / Status Register (RWCS) B-595
B.3.3.6 Read/Write Access Data Register (RWD) B-596
B.3.3.7 Read/Write Access Address Register (RWA) B-596
B.3.3.8 Watchpoint Trigger Register (WT) B-596
B.3.4 Nexus Register Access via JTAG B-597
B.3.5 Programming Considerations ($\overline{\text{RESET}}$) B-599
B.4 Functional Description B-599
B.4.1 Ownership Trace B-599
B.4.1.1 Ownership Trace Messaging (OTM) B-599

Contents

Paragraph Number	Title	Page Number
B.4.1.2	OTM Error Messages	B-600
B.4.1.3	OTM Flow	B-600
B.4.2	Program Trace	B-601
B.4.2.1	Branch Trace Messaging (BTM)	B-601
B.4.2.2	Branch Trace Message Formats (History and Traditional)	B-602
B.4.2.3	BTM Operation	B-607
B.4.2.4	Program Trace Timing Diagrams (2 MDO / 1 $\overline{\text{MSEO}}$ configuration)	B-609
B.4.3	Watchpoint Support	B-610
B.4.3.1	Watchpoint Messaging	B-610
B.4.3.2	Watchpoint Error Message	B-611
B.4.3.3	Watchpoint Timing Diagram (2 MDO / 1 $\overline{\text{MSEO}}$ configuration)	B-611
B.4.4	Read/Write Access	B-611
B.4.4.1	Functional Description	B-611
B.4.4.2	Read/Write Access to Internal Nexus Registers	B-612
B.4.4.3	Memory Mapped Register Access via JTAG	B-612
B.4.4.4	Error Handling	B-615
B.4.4.5	Timing Diagram	B-616
B.5	Electrical Characteristics	B-617
B.5.1	Maximum Ratings / DC Electrical Specifications	B-617
B.5.2	A7S Nexus 2 Auxiliary Pin Timing Specifications	B-617
B.6	IEEE 1149.1 State Machine and RD/WR Sequences	B-619
B.6.1	JTAG State Machine	B-619
B.6.2	JTAG Sequence for Accessing Internal Nexus Registers	B-620
B.6.3	JTAG Sequence for Read Access of Memory-Mapped Resources	B-620
B.6.4	JTAG Sequence for Write Access of Memory-Mapped Resources	B-620

Appendix C
Register Memory Map Quick Reference

C.1	Peripherals Register Memory Map	C-621

Appendix D
Mask Set Differences Summary

D.1	Differences Between L49P, L47W, L61W and L38Y Mask Set Devices	D-635

Tables

Table Number	Title	Page Number
1-1	MAC7100 Family Device Derivatives	1-3
2-1	Pins Not Bonded Out By Device Derivative	2-13
2-2	Signal Properties Summary	2-22
2-3	Power Supply, Voltage Regulator and Reference Summary	2-26
3-1	VREG Signal Properties	3-31
3-2	VREG Memory Map	3-32
3-3	VREGHTCL Field Descriptions	3-33
3-4	VREGCTRL Field Descriptions	3-34
3-5	VREGCTRL Field Descriptions	3-34
3-6	VREG Selectable Autonomous Periodic Interrupt Periods	3-35
3-7	VREGAPITR Field Descriptions	3-36
3-8	VREG Reset Sources	3-38
3-9	VREG Interrupt Vectors	3-38
3-10	VREG Recommended Circuit Board Component Values – LQFP	3-40
3-11	VREG Recommended Circuit Board Component Values – 208 MAP BGA	3-40
4-1	CRG Signal Properties	4-52
4-2	CRG Memory Map	4-53
4-3	CRGFLG Field Descriptions	4-55
4-4	CRGINT Field Descriptions	4-56
4-5	CLKSEL Field Descriptions	4-57
4-6	PLLCTL Field Descriptions	4-58
4-7	SDMCTL Field Descriptions	4-59
4-8	BDMCTL Field Description	4-60
4-9	CRG Reset Source Summary	4-67
4-10	CRG Reset Vector Selection	4-67
4-11	CRG Interrupt Sources	4-70
4-12	MCU Configuration During Doze Mode	4-71
4-13	Outcome of Clock Loss in Doze Mode	4-74
4-14	Outcome of Clock Loss in Pseudo-Stop Mode	4-78
4-15	CRG Modes Active Device Clocks Summary	4-81
4-16	CRG Modes Entry Sequences	4-81
6-1	ARM7 Exception Table	6-85
6-2	MAC7100 Family Interrupt Vector Assignments	6-85
7-1	MCU Mode Selection	7-87
7-2	Clock Selection based on $\overline{\text{XCLKS}}$	7-88
8-1	Available Address Map Configurations	8-93
8-2	Reset Memory Map in Normal Single-Chip Mode	8-95
8-3	Reset Memory Map in Secured Single-Chip Mode	8-95
8-4	Available Address Map Configurations in Secured Single-Chip Mode	8-96

Tables

Table Number	Title	Page Number
8-5	Reset Memory Map in Normal Expanded Mode	8-96
8-6	Reset Memory Map in Secured Expanded Mode	8-97
8-7	Available Address Map Configurations in Secured Expanded Mode	8-97
8-8	Reset Memory Map in Normal/Secured Data Flash Boot Mode	8-98
8-9	Available Address Map Configurations in Data Flash Boot Modes	8-98
8-10	Peripheral Bus Memory Map	8-99
10-1	ARM7 Interrupt Exception Summary	10-105
10-2	INTC Interrupt Source-to-ICR*n* Assignments	10-106
10-3	INTC Memory Map	10-106
10-4	IPRH/IPRL Field Descriptions	10-109
10-5	IMRH/IMRL Field Descriptions	10-110
10-6	INTFRCH/INTFRCL Field Descriptions	10-111
10-7	ICONFIG Field Descriptions	10-111
10-8	SIMR Field Descriptions	10-112
10-10	CLMASK Field Descriptions	10-113
10-9	CIMR Field Descriptions	10-113
10-11	SLMASK Field Descriptions	10-114
10-12	ICR*n* Field Descriptions	10-115
10-13	IRQIACK Field Descriptions	10-116
10-14	FIQIACK Field Descriptions	10-116
11-1	MCM Memory Map	11-124
11-2	PCT Field Descriptions	11-125
11-3	REV Field Descriptions	11-125
11-4	REV Values by Mask Set	11-125
11-5	AMC Field Descriptions	11-126
11-6	ASC Field Descriptions	11-126
11-7	IOPMC Field Descriptions	11-127
11-8	MRSR Field Descriptions	11-128
11-9	MWCR Field Descriptions	11-129
11-10	MSWTCR Field Descriptions	11-130
11-11	MSWTIR Field Descriptions	11-132
11-12	AAMR Field Descriptions	11-133
11-13	AAMR Reset Values	11-133
11-14	AAMR[7:0] Allowed Values	11-135
11-15	CFADR Field Descriptions	11-136
11-16	CFLOC Field Descriptions	11-137
11-17	CFATR Field Descriptions	11-137
11-18	CFDTR Field Descriptions	11-138
11-19	Processor Type and Device Revision Summary	11-139
11-20	PCT and REV Values Summary	11-139
12-1	eDMA Request Sources	12-142

Tables

Table Number	Title	Page Number
12-2	eDMA Memory Map	12-144
12-3	DMACR Field Descriptions	12-145
12-4	DMAES Field Descriptions	12-146
12-5	DMAERQ Field Descriptions	12-148
12-6	DMAEEI Field Descriptions	12-148
12-7	DMASERQ Field Descriptions	12-149
12-8	DMACERQ Field Descriptions	12-149
12-9	DMASEEI Field Descriptions	12-150
12-10	DMACEEI Field Descriptions	12-151
12-11	DMACINT Field Descriptions	12-151
12-12	DMACERR Field Descriptions	12-152
12-13	DMASSRT Field Descriptions	12-152
12-14	DMACDNE Field Descriptions	12-153
12-15	DMAINT Field Descriptions	12-154
12-16	DMAERR Field Descriptions	12-155
12-17	DCHPRI*n* Field Descriptions	12-155
12-18	TCD*n* Memory Map Detail	12-156
12-19	TCD*n*[SADDR] Field Description	12-157
12-20	TCD*n*[SMOD, SSIZE, DMOD, DSIZE, SOFF] Field Descriptions	12-157
12-21	TCD*n*[NBYTES] Field Description	12-158
12-22	TCD*n*[SLAST] Field Description	12-159
12-23	TCD*n*[DADDR] Field Description	12-159
12-24	TCD*n*[DOFF, CITER] Field Descriptions	12-160
12-25	TCD*n*[DLAST_SGA] Field Description	12-161
12-26	TCD*n*[BITER, Control/Status] Field Descriptions	12-162
12-27	eDMA TCD Primary Control and Status Fields	12-171
12-28	eDMA Memory Array Terms	12-173
12-29	eDMA Modulo Feature Example	12-177
12-30	Channel Linking Parameters	12-179
13-1	EIM Signal Properties	13-183
13-2	EIM Memory Map	13-185
13-3	CSAR*n* Field Description	13-186
13-4	CSMR*n* Field Descriptions	13-186
13-5	CSMR*n*[BAM] / CSAR*n* Configuration Examples	13-187
13-6	CSCR*n* Field Descriptions	13-188
13-7	Chip Select Module Control Signal Functions	13-190
13-8	EIM Byte Select Signals Generation	13-190
13-9	PA[15:14] Global Chip Select Configuration	13-191
13-10	Accesses by Matches in CSCR*n*s	13-193
13-11	Bus Cycle States	13-194
13-12	Allowable Line Access Patterns	13-199

Tables

Table Number	Title	Page Number
14-1	XBS Memory Map	14-204
14-2	PR_*port* Field Descriptions	14-205
14-3	CR_*port* Descriptions	14-206
15-1	CFM Flash Blocks Address Map Summary	15-212
15-2	CFM Flash Configuration Field	15-213
15-3	CFM Memory Map	15-214
15-4	CFMMCR Field Descriptions	15-215
15-5	CFMCLKD Field Descriptions	15-216
15-6	CFMSEC Field Descriptions	15-217
15-7	CFMPROT Field Descriptions	15-218
15-8	CFMDFPROT Field Descriptions	15-220
15-9	CFMSACC Field Descriptions	15-221
15-10	CFMDFSACC	15-222
15-11	CFMDACC Field Descriptions	15-223
15-12	CFMDFDACC Field	15-223
15-13	CFMUSTAT Field Descriptions	15-224
15-14	CFMCMD Field Descriptions	15-225
15-15	CFMDISU Field Descriptions	15-227
15-16	CFMCLKSEL Field Descriptions	15-227
15-17	CFMCLKD Register Values For 40 MHz and 50 MHz f_{SYS}	15-230
15-18	CFM Flash Memory Commands	15-231
15-19	CFM Multiple-Block Data Signature Operations	15-243
16-1	AIPS Memory Map	16-253
16-2	AIPS Access Control Fields Module Assignments	16-254
16-3	AIPS MPR*x* MPROT*n* Bit Descriptions	16-255
16-4	AIPS PACR*x* PAC*n* Bit Descriptions	16-256
17-1	DMAMux Memory Map	17-260
17-2	CHCONFIG*n* Field Descriptions	17-261
17-3	DMAMux Channel to PIT Timer Assignments	17-264
18-1	GPIO and Peripheral Signal Associations	18-271
18-2	PIM / Peripheral Signal Properties	18-274
18-3	PIM Memory Map — Global Registers	18-276
18-4	PIM Memory Map — Port *x* Registers	18-277
18-5	CONFIG*n_x* Field Descriptions	18-286
18-6	PORTIFR_*x* Field Descriptions	18-288
18-7	PORTDATA_*x* Field Descriptions	18-288
18-8	PORTIR_*x* Field Descriptions	18-289
18-9	PINDATA*n_x* Field Descriptions	18-289
18-10	GLBINT Field Descriptions	18-290
18-11	PIMCONFIG Field Descriptions	18-291
18-12	CONFIG_TDI Field Descriptions	18-292

Tables

Table Number	Title	Page Number
18-13	CONFIG_TDO Field Descriptions	18-292
18-14	CONFIG_TMS Field Descriptions	18-293
18-15	CONFIG_TCK Field Descriptions	18-294
18-16	CONFIG_TA Field Descriptions	18-295
18-17	PORT32IR_*xx* Field Descriptions	18-296
18-18	PIM Register Behavior in Peripheral Mode	18-297
18-19	PIM Register Behavior in GPI Mode	18-299
18-20	PIM GPI Interrupt Polarity Configuration	18-299
18-21	PIM Input Glitch Filter Characteristics	18-300
18-22	PIM Register Behavior in GPO Mode	18-301
18-23	PD2 / CLKOUT Mode Selection	18-313
19-1	ATD Signal Properties	19-319
19-2	ATD Memory Map	19-320
19-3	ATDTRIGCTL Field Descriptions	19-322
19-4	ATDETRIGCH Field Descriptions	19-323
19-5	ATDPRE Field Descriptions	19-324
19-6	ATDMODE Field Descriptions	19-324
19-7	ATDINT Field Descriptions	19-325
19-8	ATDFLAG Field Descriptions	19-327
19-9	ATDCW Field Descriptions	19-328
19-10	ATDRR Field Descriptions	19-331
19-11	Numeric Examples of Result Values	19-332
19-12	ATD Debug Modes	19-335
19-13	ATD Interrupt Sources	19-340
20-1	eMIOS Signal Properties	20-353
20-2	eMIOS Memory Map	20-353
20-3	eMIOS UC*n* Memory Map Detail	20-354
20-4	MCR Field Descriptions	20-355
20-5	OUDIS Field Descriptions	20-356
20-6	UCDIS Field Descriptions	20-357
20-7	UCA*n* and UCB*n* Access Assignment	20-358
20-8	UCCR*n* Register Field Descriptions	20-360
20-9	UCCR*n* MODE Field Definitions	20-362
20-10	UCSR*n* Field Descriptions	20-364
20-11	eMIOS OPWFM Output Waveforms Examples	20-383
21-1	eSCI Memory Map	21-407
21-2	ESCIBDH Field Descriptions	21-409
21-3	ESCIBDL Field Descriptions	21-409
21-4	ESCICR1 Field Descriptions	21-410
21-5	ESCICR2 Field Descriptions	21-412
21-6	ESCICR3 Field Descriptions	21-412

Tables

Table Number	Title	Page Number
21-7	ESCICR4 Field Descriptions	21-413
21-8	ESCIDRH Field Descriptions	21-414
21-9	ESCIDRL Field Descriptions	21-415
21-10	ESCISR1 Field Descriptions	21-416
21-11	ESCISR2 Field Descriptions	21-416
21-12	LINSTAT1 Field Descriptions	21-417
21-13	LINSTAT2 Field Descriptions	21-418
21-14	LINCTRL1 Field Descriptions	21-419
21-15	LINCTRL2 Field Descriptions	21-420
21-16	LINCTRL3 Field Description	21-420
21-17	LINTX First Byte Field Descriptions	21-421
21-18	LINTX Second Byte Field Descriptions	21-422
21-19	LINTX Third Byte Field Descriptions	21-422
21-20	LINTX Rx Frame Fourth Byte Field Descriptions	21-422
21-21	LINTX Tx Frame Fourth+ Byte / Rx Frame Fifth+ Byte Field Description	21-423
21-22	LINRX Field Descriptions	21-423
21-23	LINCRCP*n* Field Description	21-424
21-24	Example of 8-bit Data Formats	21-425
21-25	Example of 9-Bit Data Formats	21-425
21-26	Baud Rates (Example: Module Clock = 10.2 Mhz)	21-426
21-27	Start Bit Verification	21-431
21-28	Data Bit Recovery	21-432
21-29	Stop Bit Recovery	21-432
21-30	eSCI Interrupt Sources Summary	21-441
22-1	DSPI Signal Properties	22-451
22-2	DSPI Memory Map	22-453
22-3	DSPI*x*_MCR Field Descriptions	22-454
22-4	DSPI*x*_TCR Field Descriptions	22-457
22-5	DSPI*x*_CTAR*n* Field Descriptions	22-458
22-6	DSPI*x*_SR Field Descriptions	22-463
22-7	DSPI*x*_RSER Field Descriptions	22-465
22-8	DSPI*x*_PUSHR Field Descriptions	22-466
22-9	DSPI*x*_POPR Field Descriptions	22-468
22-10	DSPI*x*_TXFR0 Field Descriptions	22-468
22-11	DSPI*x*_RXFR*n* Field Description	22-469
22-12	State Transitions for Start and Stop of DSPI Transfers	22-472
22-13	Baud Rate Computation Examples	22-476
22-14	PCS to SCK Delay Computation Examples	22-476
22-15	After SCK Delay Computation Examples	22-476
22-16	Delay After Transfer Computation Examples	22-477
22-17	Peripheral Chip Select Strobe Assert Computation Examples	22-477

Tables

Table Number	Title	Page Number
22-18	Peripheral Chip Select Strobe Negate Computation Examples	22-478
22-19	DSPI Interrupt and DMA Request Conditions	22-484
22-20	Example DSPI Baud Rate Values (Hz), f_{IPS} = 20 MHz	22-487
22-21	Example DSPI Delay Values (ns), f_{IPS} = 20 MHz	22-488
23-1	FlexCAN Signal Properties	23-494
23-2	FlexCAN Memory Map	23-494
23-3	FlexCAN Message Buffer MB0 Memory Mapping	23-495
23-4	FlexCAN MB Field Descriptions	23-495
23-5	FlexCAN Message Buffer Code for Rx buffers	23-496
23-6	FlexCAN Message Buffer Code for Tx buffers	23-497
23-7	MCR Field Descriptions	23-498
23-8	CTRL Field Descriptions	23-502
23-9	Mask examples for Standard/Extended Message Buffers	23-505
23-10	RXGMASK Field Descriptions	23-506
23-11	ESR Field Descriptions	23-508
23-12	IMASK Field Descriptions	23-510
23-13	IFLAG Field Descriptions	23-511
23-14	Time Segment Syntax	23-518
23-15	CAN Standard Compliant Bit Time Segment Settings	23-518
23-16	Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate	23-519
23-17	FlexCAN Wake-up from Doze Mode	23-521
23-18	FlexCAN Wake-up from Stop Mode	23-522
24-1	I^2C Memory Map	24-527
24-2	IBAD Field Descriptions	24-527
24-3	IBFD Field Descriptions	24-528
24-4	I^2C Divider and Hold Values	24-530
24-5	IBCR Field Descriptions	24-533
24-6	IBSR Field Descriptions	24-534
25-1	PIT Memory Map	25-550
25-2	TLVAL*n* Field Descriptions	25-552
25-3	TVAL*n* Field Descriptions	25-553
25-4	PITFLG Field Descriptions	25-554
25-5	PITINTEN Field Descriptions	25-554
25-6	PITINTSEL Field Description	25-555
25-7	PITEN Field Descriptions	25-556
25-8	PITCTRL Field Descriptions	25-556
25-9	PIT Timer Feature Assignments	25-557
25-10	PIT Interrupt Sources	25-559
26-1	SSM Memory Map	26-564
26-2	STATUS Field Descriptions	26-565
26-3	MEMCONFIG Field Descriptions	26-566

Tables

Table Number	Title	Page Number
26-4	WAKEUP Field Descriptions	26-568
26-5	WAKEUP SOURCE Field Detail	26-568
26-6	ERROR Field Descriptions	26-569
26-7	ERROR[BABORT] Protected Address Ranges	26-569
26-8	PORTSEL Field Descriptions	26-570
26-9	DEBUGPORT Field Descriptions	26-571
26-10	Port F Debug Status Mode Signal Assignments	26-571
26-11	MEMCONFIG Field Details	26-572
26-12	Nexus Port Configuration Summary	A-577
B-1	Terms and Definitions	B-579
B-2	Public TCODEs Supported	B-582
B-3	Error Code Encoding (TCODE = 8)	B-584
B-4	Resource Code Encoding (TCODE = 27)	B-584
B-5	Event Code Encoding (TCODE = 33)	B-584
B-6	JTAG Pins for A7S Nexus 2	B-585
B-7	A7S Nexus 2 Auxiliary Pins	B-585
B-8	$\overline{\text{MSEO}}$ Pin(s) Protocol	B-586
B-9	Indirect Branch Message Example (2 MDO / 1 $\overline{\text{MSEO}}$)	B-588
B-10	Indirect Branch Message Example (8 MDO / 2 $\overline{\text{MSEO}}$)	B-589
B-11	Direct Branch Message Example (2 MDO / 1 $\overline{\text{MSEO}}$)	B-589
B-12	Direct Branch Message Example (8 MDO / 2 $\overline{\text{MSEO}}$)	B-589
B-13	ID Field Descriptions	B-590
B-14	A7S Nexus 2 Memory Map	B-591
B-15	CSC Field Description	B-591
B-16	DC Field Description	B-592
B-17	DS Field Description	B-594
B-18	RWCS Field Description	B-595
B-19	Read / Write Access Status Bit Encoding	B-596
B-20	WT Field Description	B-597
B-21	ARM7 JTAG Instructions	B-598
B-22	JTAG DR Field Values for Nexus Register Access	B-598
B-23	Indirect Branch / Branch History Message Instructions	B-601
B-24	Direct Branch Message Instructions	B-602
B-25	Program Trace Exception Summary	B-606
B-26	Watchpoint Source Description	B-610
B-27	JTAG Nexus3 Register Select	B-612
B-28	JTAG Sequence for Accessing Internal Nexus Registers	B-620
B-29	JTAG Sequence for Read Access of Memory-Mapped Resources	B-620
B-30	JTAG Sequence for Write Access of Memory-Mapped Resources	B-620
C-1	MAC7100 Family Peripheral Memory Map	C-621
C-2	AIPS Memory Map	C-622

Tables

Table Number	Title	Page Number
C-3	XBS Memory Map	C-622
C-4	EIM Memory Map	C-623
C-5	MCM Memory Map	C-623
C-6	eDMA Memory Map	C-624
C-6a	eDMA TCD*n* Memory Map Detail	C-624
C-7	INTC Memory Map	C-625
C-8	SSM Memory Map	C-626
C-9	DMAMux Memory Map	C-626
C-10	CRG Memory Map	C-626
C-11	PIT Memory Map	C-627
C-12	VREG Memory Map	C-628
C-13	FlexCAN Memory Map	C-628
C-13a	FlexCAN MB*n* Memory Map Detail	C-628
C-14	I^2C Memory Map	C-629
C-15	DSPI Memory Map	C-629
C-16	eSCI Memory Map	C-630
C-17	eMIOS Memory Map	C-631
C-17a	eMIOS UC*n* Memory Map Detail	C-631
C-18	ATD Memory Map	C-631
C-19	PIM Memory Map — Port Control Registers	C-632
C-20	PIM Memory Map — Global Control Registers	C-633
C-21	CFM Memory Map	C-634
D-1	MAC7100 Family Mask Set to Part Number Correspondence	D-635
D-2	MAC7100 Microcontroller Family Mask Set Differences Summary	D-636

Tables

Figures

Figure Number	Title	Page Number
1-1	MAC7100 Family Block Diagram	1-2
1-2	Order Part Number Example	1-4
3-1	VREG Block Diagram for L49P Mask Devices	3-30
3-2	VREG Block Diagram for non-L49P Mask Devices	3-30
3-3	VREG High Temperature Control Register (VREGHTCL)	3-33
3-4	VREG Control Register (VREGCTRL)	3-33
3-5	VREG Autonomous Periodic Interrupt Control Register (VREGAPICL)	3-34
3-6	VREG Autonomous Periodic Interrupt Trimming Register (VREGAPITR)	3-36
3-7	Recommended PCB Layout for 144 QFP	3-41
3-8	Recommended PCB Layout for 112 LQFP	3-42
3-9	Recommended PCB Layout for 100 LQFP	3-43
3-10	Recommended PCB Layout for 208 MAP BGA – Ground Plane	3-44
3-11	Recommended PCB Layout for 208 MAP BGA – Decoupling	3-44
4-1	MAC7100 Family Clock Connections	4-46
4-2	Oscillator Block Diagram	4-47
4-3	Loop Controlled Pierce Oscillator Connections	4-49
4-4	Full Swing Pierce Oscillator Connections	4-49
4-5	External Clock Connections	4-49
4-6	Clock and Reset Generator Block Diagram	4-50
4-7	PLL Loop Filter Connections	4-53
4-8	CRG Synthesizer Register (SYNR)	4-54
4-9	CRG Reference Divider Register (REFDV)	4-54
4-10	CRG Flags Register (CRGFLG)	4-55
4-11	CRG Interrupt Enable Register (CRGINT)	4-56
4-12	CRG Clock Select Register (CLKSEL)	4-57
4-13	CRG PLL Control Register (PLLCTL)	4-58
4-14	CRG Stop/Doze Control Register (SDMCTL)	4-59
4-15	CRG BDM Control Register (BDMCTL)	4-60
4-16	PLL Block Diagram	4-61
4-17	System Clocks Generator Block Diagram	4-63
4-18	System Clock and Peripheral Bus Clock Relationship	4-63
4-19	Check Window Example	4-64
4-20	Sequence for Clock Quality Check on Non-Mask Set L49P Devices	4-65
4-21	Sequence for Clock Quality Check on Mask Set L49P Devices	4-66
4-22	$\overline{\text{RESET}}$ Timing	4-68
4-23	$\overline{\text{RESET}}$ Timing controlled by JTAG	4-68
4-24	Power-Up Sequence — $\overline{\text{RESET}}$ Pin Tied to V_{DD}2.5 via a Pull-up Resistor	4-69
4-25	Power-Up Sequence — $\overline{\text{RESET}}$ Pin Driven Low Externally	4-70
4-26	Doze Mode Entry/Exit Sequence	4-73

Figures

Figure Number	Title	Page Number
4-27	Stop Mode Entry/Exit Sequence	4-76
4-28	Fast Wake-up from Full Stop mode: Example 1	4-80
4-29	Fast Wake-up from Full Stop mode: Example 2	4-80
8-1	MAC71*x*1 Family Memory Map Examples	8-94
10-1	Interrupt Controller Block Diagram	10-103
10-2	INTC Interrupt Pending Register High (IPRH)	10-108
10-3	INTC Interrupt Pending Register Low (IPRL)	10-108
10-4	INTC Interrupt Mask Register High (IMRH)	10-109
10-5	INTC Interrupt Mask Register Low (IMRL)	10-109
10-6	INTC Force Interrupt Register High (INTFRCH)	10-110
10-7	INTC Force Interrupt Register Low (INTFRCL)	10-110
10-8	INTC Interrupt Configuration Register (ICONFIG)	10-111
10-9	INTC Set Interrupt Mask Register (SIMR)	10-112
10-10	INTC Clear Interrupt Mask Register (CIMR)	10-112
10-11	INTC Current Level Mask Register (CLMASK)	10-113
10-12	INTC Saved Level Mask Register (SLMASK)	10-114
10-13	INTC Interrupt Control Registers (ICR*n*)	10-115
10-14	INTC IRQ Acknowledge Register (IRQIACK)	10-115
10-15	INTC FIQ Acknowledge Register (FIQIACK)	10-116
10-16	INTC Interrupt Service Routine and Masking	10-120
11-1	MCM Processor Core Type Register (PCT)	11-124
11-2	MCM Device Revision Register (REV)	11-125
11-3	MCM XBS Master Configuration Register (AMC)	11-126
11-4	MCM XBS Slave Configuration Register (ASC)	11-126
11-5	MCM IPS On-Platform Module Configuration Register (IOPMC)	11-127
11-6	MCM Reset Status Register (MRSR)	11-127
11-7	MCM Wake-up Control Register (MWCR)	11-128
11-8	MCM Software Watchdog Timer Control Register (MSWTCR)	11-130
11-9	MCM SWT Service Register (MSWTSR)	11-131
11-10	MCM SWT Timer Interrupt Register (MSWTIR)	11-132
11-11	MCM XBS Address Map Register (AAMR)	11-133
11-12	MCM Core Fault Address Register (CFADR)	11-136
11-13	MCM Core Fault Location Register (CFLOC)	11-136
11-14	MCM Core Fault Attributes Register (CFATR)	11-137
11-15	MCM Core Fault Data Register (CFDTR)	11-138
12-1	eDMA Block Diagram	12-142
12-2	eDMA Control Register (DMACR)	12-145
12-3	eDMA Error Status Register (DMAES)	12-146
12-4	eDMA Enable Request Register (DMAERQ)	12-148
12-5	eDMA Enable Error Interrupt Registers (DMAEEI)	12-148
12-6	eDMA Set Enable Request Register (DMASERQ)	12-149

Figures

Figure Number	Title	Page Number
12-7	eDMA Clear Enable Request Register (DMACERQ)	12-149
12-8	eDMA Set Enable Error Interrupt Register (DMASEEI)	12-150
12-9	eDMA Clear Enable Error Interrupt Register (DMACEEI)	12-151
12-10	eDMA Clear Interrupt Request Register (DMACINT)	12-151
12-11	eDMA Clear Error Register (DMACERR)	12-152
12-12	eDMA Set START Bit Register (DMASSRT)	12-152
12-13	eDMA Clear DONE Status Register (DMACDNE)	12-153
12-14	eDMA Interrupt Request Register (DMAINT)	12-154
12-15	eDMA Error Registers (DMAERR)	12-154
12-16	eDMA Channel Priority Registers (DCHPRI*n*)	12-155
12-17	TCD*n* Structure Detail	12-156
12-18	TCD*n* Word 0 (TCD*n*[SADDR])	12-157
12-19	TCD*n* Word 0 (TCD*n*[SMOD, SSIZE, DMOD, DSIZE, SOFF])	12-157
12-20	TCD*n* Word 2 (TCD*n*[NBYTES])	12-158
12-21	TCD*n* Word 3 (TCD*n*[SLAST])	12-159
12-22	TCD*n* Word 4 (TCD*n*[DADDR])	12-159
12-23	TCD*n* Word 5 (TCD*n*[CITER, DOFF])	12-160
12-24	TCD*n* Word 6 (TCD*n*[DLAST_SGA])	12-161
12-25	TCD*n* Word 7 (TCD*n*[BITER, Control/Status])	12-161
12-26	eDMA Operation – Part 1	12-165
12-27	eDMA Operation – Part 2	12-166
12-28	eDMA Operation – Part 3	12-166
12-29	eDMA Transfer Control Descriptor Header File Example	12-167
12-30	eDMA Example – Multiple Loop Iterations	12-172
13-1	MAC7100 Family Bus Architecture Block Diagram	13-181
13-2	EIM Chip Select Address Registers (CSAR*n*)	13-185
13-3	EIM Chip Select Mask Registers (CSMR*n*)	13-186
13-4	EIM Chip Select Control Registers (CSCR*n*)	13-188
13-5	EIM Connections for External Memory Port Sizes	13-191
13-6	EIM Signal Relationship to CLKOUT	13-192
13-7	EIM Strobe Output Timing Diagram	13-193
13-8	EIM Data Transfer State Transition Diagram	13-194
13-9	EIM Read Cycle Flowchart	13-195
13-10	EIM Basic Read Bus Cycle	13-196
13-11	EIM Write Cycle Flowchart	13-196
13-12	EIM Basic Write Bus Cycle	13-197
13-13	EIM Read Cycle with Fast Termination	13-197
13-14	EIM Write Cycle with Fast Termination	13-198
13-15	EIM Back-to-Back Bus Cycles	13-198
13-16	EIM Line Read Burst (2-1-1-1-1-1-1-1), External Termination	13-199
13-17	EIM Line Read Burst (2-1-1-1-1-1-1-1), Internal Termination	13-200

Figures

Figure Number	Title	Page Number
13-18	EIM Line Write Burst (2-1-1-1-1-1-1-1), External Termination	13-200
13-19	EIM Line Write Burst (2-1-1-1-1-1-1-1), Internal Termination	13-201
14-1	MAC7100 Family Bus Architecture Block Diagram	14-203
14-2	XBS Priority Registers (PR_*port*)	14-205
14-3	XBS Control Registers (CR_*port*)	14-206
15-1	CFM Block Diagram, MAC71*x*1 and MAC71*x*2 Devices	15-210
15-2	CFM Block Diagram, MAC71*x*6 Devices	15-210
15-3	CFM Flash Memory Map Overview (MAC71*x*6 Devices)	15-212
15-4	CFM Flash Memory Map Overview (MAC71*x*1 Devices)	15-213
15-5	CFM Flash Memory Map Overview (MAC71*x*2 Devices)	15-213
15-6	CFM Module Configuration Register (CFMMCR)	15-215
15-7	CFM Clock Divider Register (CFMCLKD)	15-216
15-8	CFM Security Register (CFMSEC)	15-217
15-9	CFM Program Flash Protection Register (CFMPROT)	15-218
15-10	CFMPROT Protection Diagram, MAC71*x*6 Devices	15-219
15-11	CFMPROT Protection Diagram, MAC71*x*1 Devices	15-219
15-12	CFMPROT Protection Diagram, MAC71*x*2 Devices	15-219
15-13	CFM Data Flash Protection Register (CFMDFPROT)	15-220
15-14	CFMDFPROT Protection Diagram	15-220
15-15	CFM Program Flash Supervisor Access Register (CFMSACC)	15-221
15-16	CFM Data Flash Supervisor Access Register (CFMDFSACC)	15-222
15-17	CFM Program Flash Data Access Register (CFMDACC)	15-222
15-18	CFM Data Flash Data Access Register (CFMDFDACC)	15-223
15-19	CFM User Status Register (CFMUSTAT)	15-224
15-20	CFM Command Register (CFMCMD)	15-225
15-21	CFM Data Registers (CFMDATA1/0)	15-226
15-22	CFM Disable Upper Register (CFMDISU)	15-226
15-23	CFM Clock Select Register (CFMCLKSEL)	15-227
15-24	CFM f_{NVMOP} Generation Logical Block Diagram	15-229
15-25	CFM Example Blank Check Command Flow	15-232
15-26	CFM Flash Logical Page and Security Sector Mapping	15-233
15-27	CFM Example Page Erase Verify Command Flow	15-235
15-28	CFM Example Program Command Flow	15-237
15-29	CFM Example Page Erase Command Flow for Program Flash	15-239
15-30	CFM Example Page Erase Command Flow for Data Flash	15-240
15-31	CFM Example Mass Erase Command Flow	15-242
15-32	CFM Example Data Signature Command Flow	15-246
16-1	AIPS Interface Block Diagram	16-252
16-2	AIPS Peripheral Module Address Assignment	16-252
16-3	AIPS MPR*x* Master Protection Fields (MPROT*n*)	16-255
16-4	AIPS PACR*x* Peripheral Access Control Fields (PAC*n*)	16-256

Figures

Figure Number	Title	Page Number
17-1	DMAMux Block Diagram	17-259
17-2	DMAMux Channel Configuration Registers (CHCONFIG*n*)	17-261
17-3	DMAMux Channel 0 to 7 Block Diagram	17-263
17-4	DMAMux Channel Triggering: Normal Operation	17-263
17-5	DMAMux Channel Triggering: Gated Request	17-264
17-6	DMAMux Channel 8 to 15 Block Diagram	17-265
18-1	Port Integration Module Block Diagram	18-272
18-2	PIM Port *x* Pin Configuration Registers (CONFIG*n_x*)	18-286
18-3	PIM Port *x* Interrupt Flag Register (PORTIFR_*x*)	18-287
18-4	PIM Port *x* Data Register (PORTDATA_*x*)	18-288
18-5	PIM Port *x* Input Register (PORTIR_*x*)	18-289
18-6	PIM Port *x* Pin Data Registers (PINDATA*n_x*)	18-289
18-7	PIM Global Interrupt Status Register (GLBINT)	18-290
18-8	PIM Global Configuration Register (PIMCONFIG)	18-291
18-9	PIM Configure TDI Pin Register (CONFIG_TDI)	18-291
18-10	PIM Configure TDO Pin Register (CONFIG_TDO)	18-292
18-11	PIM Configure TMS Pin Register (CONFIG_TMS)	18-293
18-12	PIM Configure TCK Pin Register (CONFIG_TCK)	18-294
18-13	PIM Configure $\overline{\text{TA}}$ / $\overline{\text{AS}}$ Pin Register (CONFIG_TA)	18-295
18-14	PIM Port *x*/*x* 32-bit Input Register (PORT32IR_*xx*)	18-296
18-15	PIM Pad in Peripheral Mode Logical Diagram	18-298
18-16	PIM Pad in General Purpose Input Mode	18-299
18-17	PIM External Interrupt Detection Requirements	18-301
18-18	PIM Pad in GPO Mode (except PD2)	18-302
18-19	PIM Pad in GPO Mode (PD2 only)	18-302
19-1	ATD Block Diagram	19-318
19-2	ATD Trigger Control Register (ATDTRIGCTL)	19-321
19-3	ATD External Trigger Channel Register (ATDETRIGCH)	19-322
19-4	ATD Prescaler Register (ATDPRE)	19-323
19-5	ATD Operating Mode Register (ATDMODE)	19-324
19-6	ATD Interrupt Register (ATDINT)	19-325
19-7	ATD Flag Register (ATDFLAG)	19-326
19-8	ATD Command Word Register (ATDCW)	19-328
19-9	ATD Result Register (ATDRR)	19-331
19-10	ATD Right Justified Unsigned 8/10-Bit Result Format	19-331
19-11	ATD Right Justified Signed 8/10-Bit Result Format	19-331
19-12	ATD Left Justified Unsigned 8/10-Bit Result Format	19-332
19-13	ATD Left Justified Signed 8/10-Bit Result Format	19-332
19-14	ATD Command Processing Flow Diagram	19-339
19-15	ATD Result Processing and Command Fetching Flow Diagrams	19-339
19-16	ATD Command Word Buffering	19-340

Figures

Figure Number	Title	Page Number
19-17	ATD Edge-Based Trigger Example	19-344
19-18	ATD Level-Based Trigger Examples	19-345
19-19	ATD 10-bit Conversion Timing Example	19-348
19-20	ATD 8-bit Conversion Timing, Amplifier Bypass Example	19-348
19-21	ATD Internal Reference Voltage Selection Logical Diagram	19-349
20-1	eMIOS Block Diagram	20-351
20-2	eMIOS Module Configuration Register (MCR)	20-354
20-3	eMIOS Global Flag Register (GFLAG)	20-355
20-4	eMIOS Output Update Disable Register (OUDIS)	20-356
20-5	eMIOS Channel Disable Register (UCDIS)	20-357
20-6	eMIOS Channel A Data Registers (UCA*n*)	20-357
20-7	eMIOS Channel B Data Registers (UCB*n*)	20-358
20-8	eMIOS Channel Counter Registers (UCCNT*n*)	20-359
20-9	eMIOS Channel Control Registers (UCCR*n*)	20-359
20-10	eMIOS Channel Status Registers (UCSR*n*)	20-363
20-11	eMIOS Unified Channel Block Diagram	20-367
20-12	eMIOS Input Programmable Filter Submodule Diagram	20-368
20-13	eMIOS Input Programmable Filter Example	20-368
20-14	eMIOS SAIC Mode Example	20-369
20-15	eMIOS SAOC Mode Example — EDPOL Transferred to Output	20-370
20-16	eMIOS SAOC Mode Example — Toggle Output	20-370
20-17	eMIOS PWM Mode Example	20-371
20-18	eMIOS IPM Mode Example	20-372
20-19	eMIOS DAOC Mode Example — FLAG Set On Second Match	20-373
20-20	eMIOS DAOC Mode Example — FLAG Set On Both Matches	20-373
20-21	eMIOS PEA Mode Example — Continuous Operation	20-374
20-22	eMIOS PEA Mode Example — Single Shot Operation	20-375
20-23	eMIOS PEC Mode Example — Continuous Operation	20-376
20-24	eMIOS PEC Mode Example — Single Shot Operation	20-376
20-25	eMIOS QDEC Mode Example — Count & Direction Encoder	20-377
20-26	eMIOS QDEC Mode Example — Phase_A & Phase_B Encoder	20-378
20-27	eMIOS WPTA Mode Example	20-379
20-28	eMIOS MC Mode Example — Up Operation	20-380
20-29	eMIOS MC Mode Example — Up/Down Operation	20-380
20-30	eMIOS OPWFM Mode Example — Immediate Update Operation	20-382
20-31	eMIOS OPWFM Mode Example — Next-Period Update Operation	20-382
20-32	eMIOS OPWMC Example — Leading Edge Dead Time	20-384
20-33	eMIOS OPWMC Example — Trailing Edge Dead Time	20-385
20-34	eMIOS OPWM Mode Example — Immediate Update	20-386
20-35	eMIOS OPWM Mode Example — Next Period Update	20-386
20-36	eMIOS MCB Mode Example — Up Operation	20-387

Figures

Figure Number	Title	Page Number
20-37	eMIOS MCB Mode Example — Up/Down Operation	20-388
20-38	eMIOS MCB Mode Example — Up Operation A1 Register Update	20-388
20-39	eMIOS MCB Mode Example — Up/Down Operation A1 Register Update	20-389
20-40	eMIOS OPWFMB Mode Example — A1/B1 Match to Output Register Delay	20-390
20-41	eMIOS OPWFMB Mode Example — A1 = 0 (0% Duty Cycle)	20-391
20-42	eMIOS OPWFMB Mode Example — A1/B1 Updates and Flags	20-392
20-43	eMIOS OPWFMB Mode Example — Active Output Disable	20-392
20-44	eMIOS OPWFMB Mode Example — 100% to 0% Duty Cycle	20-393
20-45	eMIOS OPWMCB Mode Example — A1/B1 Register Loading	20-394
20-46	eMIOS OPWMCB Mode Example — Lead Dead Time Insertion	20-395
20-47	eMIOS OPWMCB Mode Example — Trailing Dead Time Insertion	20-396
20-48	eMIOS OPWMCB Mode Example — 100% Duty Cycle (A1 = 4, B1 = 3)	20-397
20-49	eMIOS OPWMB Mode Example — Matches and Flags	20-399
20-50	eMIOS OPWMB Mode Example — 0% Duty Cycle	20-400
20-51	eMIOS OPWMB Mode Example — Active Output Disable	20-401
20-52	eMIOS OPWMB Mode Example — 100% to 0% Duty Cycle	20-401
20-53	eMIOS Time Base Generation Block Diagram	20-402
20-54	eMIOS Time Base Example — Fastest Prescaler Ratio	20-403
20-55	eMIOS Time Base Example — Prescale Ratio = 3, Match Value = 3	20-403
20-56	eMIOS Time Base Example — Prescale Ratio = 2, Match Value = 5	20-403
21-1	eSCI Block Diagram	21-405
21-2	eSCI Baud Rate Register High (ESCIBDH)	21-408
21-3	eSCI Baud Rate Register Low (ESCIBDL)	21-409
21-4	eSCI Control Register 1 (ESCICR1)	21-409
21-5	eSCI Control Register 2 (ESCICR2)	21-410
21-6	eSCI Control Register 3 (ESCICR3)	21-410
21-7	eSCI Control Register 4 (ESCICR4)	21-410
21-8	eSCI Data Register High (ESCIDRH)	21-414
21-9	eSCI Data Register Low (ESCIDRL)	21-414
21-10	eSCI Status Register 1 (ESCISR1)	21-415
21-11	eSCI Status Register 2 (ESCISR2)	21-415
21-12	LIN Status Register 1 (LINSTAT1)	21-417
21-13	LIN Status Register 2 (LINSTAT2)	21-417
21-14	LIN Control Register 1 (LINCTRL1)	21-418
21-15	LIN Control Register 2 (LINCTRL2)	21-419
21-16	LIN Control Register 3 (LINCTRL3)	21-419
21-17	LIN TX Register (LINTX)	21-421
21-18	LIN RX Register (LINRX)	21-423
21-19	LIN CRC Polynomial Register 1 (LINCRCP1)	21-423
21-20	LIN CRC Polynomial Register 2 (LINCRCP2)	21-423
21-21	eSCI Block Diagram	21-424

Figures

Figure Number	Title	Page Number
21-22	SCI Data Formats	21-425
21-23	eSCI Transmitter Block Diagram	21-426
21-24	Fast Bit Error Detection on a LIN Bus	21-429
21-25	Fast Bit Error Detection Timing Diagram	21-430
21-26	eSCI Receiver Block Diagram	21-430
21-27	Receiver Data Sampling	21-431
21-28	Start Bit Search Example 1	21-433
21-29	Start Bit Search Example 2	21-433
21-30	Start Bit Search Example 3	21-434
21-31	Start Bit Search Example 4	21-434
21-32	Start Bit Search Example 5	21-435
21-33	Start Bit Search Example 6	21-435
21-34	Slow Data Tolerance Example	21-436
21-35	Fast Data Tolerance Example	21-437
21-36	Single-Wire Operation (LOOPS = 1, RSRC = 1)	21-439
21-37	Loop Operation (LOOPS = 1, RSRC = 0)	21-439
21-38	Typical LIN Frame	21-443
21-39	LIN Frame With CRC Bytes	21-444
21-40	DMA Transfer of a TX Frame	21-445
21-41	DMA Transfer of an RX Frame	21-446
22-1	DSPI Block Diagram	22-449
22-2	DSPI with Queues and DMA	22-450
22-3	DSPI Module Configuration Register (DSPI*x*_MCR)	22-454
22-4	DSPI Transfer Count Register (DSPI*x*_TCR)	22-456
22-5	DSPI Clock and Transfer Attributes Registers (DSPI*x*_CTAR*n*)	22-457
22-6	DSPI Status Register (DSPI*x*_SR)	22-462
22-7	DSPI DMA/Interrupt Request Select and Enable Register (DSPI*x*_RSER)	22-464
22-8	DSPI PUSH TX FIFO Register (DSPI*x*_PUSHR)	22-466
22-9	DSPI POP RX FIFO Register (DSPI*x*_POPR)	22-467
22-10	DSPI Transmit FIFO Register (DSPI*x*_TXFR0)	22-468
22-11	DSPI Receive FIFO Registers (DSPI*x*_RXFR*n*)	22-469
22-12	SPI Serial Protocol Overview	22-470
22-13	DSPI Start and Stop State Diagram	22-472
22-14	Serial Communications Clock Prescalers and Scalers	22-475
22-15	Peripheral Chip Select Strobe Timing	22-477
22-16	DSPI Transfer Timing Diagram (MTFE=0, CPHA=0, FMSZ=8)	22-479
22-17	DSPI Transfer Timing Diagram (MTFE=0, CPHA=1, FMSZ=8)	22-480
22-18	DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{SCK} = f_{IPS} \div 4$)	22-481
22-19	DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{SCK} = f_{IPS} \div 4$)	22-482
22-20	DSPI Example of Non-Continuous Format (CPHA=1, CONT=0)	22-482
22-21	DSPI Example of Continuous Transfer (CPHA=1, CONT=1)	22-483

Figures

Figure Number	Title	Page Number
22-22	DSPI Continuous SCK Timing Diagram (CONT=0)	22-484
22-23	DSPI Continuous SCK Timing Diagram (CONT=1)	22-484
22-24	DSPI TX FIFO Pointers and Counter	22-488
23-1	FlexCAN Block Diagram	23-492
23-2	FlexCAN Message Buffer Structure	23-495
23-3	FlexCAN Module Configuration Register (MCR)	23-498
23-4	FlexCAN Control Register (CTRL)	23-501
23-5	FlexCAN Timer (TIMER)	23-504
23-6	FlexCAN Rx Global Mask Register (RXGMASK)	23-505
23-7	FlexCAN Error Counter Register (ECR)	23-507
23-8	FlexCAN Error and Status Register (ESR)	23-508
23-9	FlexCAN Interrupt Mask Register (IMASK)	23-510
23-10	FlexCAN Interrupt Flags Register (IFLAG)	23-511
23-11	CAN Engine Clocking Scheme	23-517
23-12	Segments Within the Bit Time	23-518
23-13	Arbitration, Match, and Move Time Windows	23-519
24-1	I^2C Module Block Diagram	24-525
24-2	I^2C Bus Address Register (IBAD)	24-527
24-3	I^2C Bus Frequency Divider Register (IBFD)	24-528
24-4	SCL Divider and SDA Hold	24-529
24-5	I^2C Bus Control Register (IBCR)	24-533
24-6	I^2C Bus Status Register (IBSR)	24-534
24-7	I^2C Bus Data I/O Register (IBDR)	24-535
24-8	I^2C Bus Transmission Signals	24-537
24-9	I^2C Bus Start and Stop conditions	24-538
24-10	I^2C Bus Clock Synchronization	24-540
24-11	I^2C Typical Interrupt Routine Flow Chart	24-542
24-12	I^2C Master Transmit in DMA Mode	24-546
24-13	I^2C Master Receive in DMA Mode	24-547
25-1	PIT Block Diagram	25-549
25-2	PIT RTI/Timer Load Value Registers (TLVAL*n*)	25-552
25-3	PIT Current RTI / Timer Values (TVAL*n*)	25-553
25-4	PIT Interrupt Flags Register (PITFLG)	25-553
25-5	PIT Interrupt Enable Register (PITINTEN)	25-554
25-6	PIT Interrupt/DMA Select Register (PITINTSEL)	25-555
25-7	PIT Timer Enable Register (PITEN)	25-555
25-8	PIT Control Register (PITCTRL)	25-556
25-9	Stopping and Starting a Timer	25-559
25-10	Modifying a Running Timer Period	25-559
25-11	Dynamically Setting a New Load Value	25-559
25-12	PIT Trigger Connections to Other Modules	25-562

Figures

Figure Number	Title	Page Number
26-1	SSM Block Diagram	26-563
26-2	SSM Current System Status Register (STATUS)	26-564
26-3	SSM System Memory Configuration Register (MEMCONFIG)	26-566
26-4	SSM Wake-up Source Register (WAKEUP)	26-567
26-5	SSM Error Configuration Register (ERROR)	26-568
26-6	SSM Port Select Register (PORTSEL)	26-570
26-7	SSM Debug Status Port Control Register (DEBUGPORT)	26-570
B-1	MAC7100 Family Nexus 2 Functional Block Diagram	B-580
B-2	Single Pin $\overline{\text{MSEO}}$ Transfers	B-586
B-3	Two Pin $\overline{\text{MSEO}}$ Transfers	B-587
B-4	JTAG ID Register (ID)	B-590
B-5	Client Select Control Register (CSC)	B-591
B-6	Development Control Register (DC)	B-592
B-7	Development Status Register (DS)	B-593
B-8	User Base Address Register (UBA)	B-594
B-9	Read / Write Access Control Register (RWCS)	B-595
B-10	Read/Write Access Data Register (RWD)	B-596
B-11	Read/Write Access Address Register (RWA)	B-596
B-12	Watchpoint Trigger Register (WT)	B-597
B-13	JTAG DR for Nexus Register Access	B-598
B-14	Ownership Trace Message Format	B-599
B-15	Error Message Format	B-600
B-16	Indirect Branch Message (History) Format	B-603
B-17	Indirect Branch Message (Traditional) Format	B-603
B-18	Direct Branch Message Format	B-603
B-19	Resource Full Message Format	B-604
B-20	Debug Status Message Format	B-604
B-21	Program Correlation Message Format	B-604
B-22	Error Message Format	B-605
B-23	Indirect Branch History w/ Sync. Message Format	B-606
B-24	Direct/Indirect Branch with Sync. Message Format (traditional)	B-606
B-25	Relative Address Generation and Re-creation	B-608
B-26	Program Trace - Indirect Branch Message (Traditional)	B-609
B-27	Program Trace - Indirect Branch Message (History)	B-609
B-28	Program Trace - Direct Branch (Traditional) and Error Messages	B-609
B-29	Program Trace - Indirect Branch with Sync. Message (Traditional)	B-610
B-30	Watchpoint Message Format	B-610
B-31	Error Message Format	B-611
B-32	Watchpoint Message and Watchpoint Error Message	B-611
B-33	Error Message Format	B-616
B-34	A7S Nexus 3 DMA clock relationships	B-616

Figures

Figure Number	Title	Page Number
B-35	MCKO Related Timing	B-617
B-36	TCK Related Timing	B-617
B-37	MCKO Timing	B-618
B-38	MDO, EVTO Timing	B-618
B-39	EVTI Timing	B-618
B-40	JTAG State Machine	B-619

Figures

Figure Number	Title	Page Number

Preface

This Reference Manual provides information about the MAC7100 Family of microcontroller devices, which are made up of standard System-on-a-Chip modules and an ARM7TDMI-S™ processor core.

Document Structure

This document is part of the documentation needed to complete a design using a MAC7100 Family device. A complete set of device manuals also includes the ARM7TDMI-S core manuals and the *MAC7100 Microcontroller Family Hardware Specifications*:

- *ARM Architecture Reference Manual* (ARM DDI-0100)
- *ARM7TDMI-S (Rev 4) Technical Reference Manual* (ARM DDI 0234A)
- *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC)

How To Use This Document

If the reader is new to the MAC7100 family of devices, it is recommended that the following list of sections be read before bringing up a MAC7100 Family device:

- Chapter 1, “Introduction,” — Describes the features of the MAC7100 Family.
- Chapter 2, “Signal Description,” — Describes the functionality of MAC7100 Family device pins
- Chapter 4, “System Clocks Module (OSC and CRG),” — Describes clock generation and distribution to modules on MAC7100 Family devices.
- Chapter 5, “Resets,” — Describes the reset functionality of the MAC7100 Family.
- Chapter 6, “Exceptions,” — Describes the system and interrupt exceptions of the MAC7100 Family.
- Chapter 7, “Modes of Operation,” — Describes the operational modes of the MAC7100 Family.
- Chapter 8, “Device Memory Map,” — Describes the memory map of the MAC7100 Family devices in various operating modes.

If the functionality of particular peripheral is of interest, refer to the appropriate module description (Chapter 9, “ARM7TDMI-S™ Processor Core,” through Chapter 26, “System Services Module (SSM)”).

Conventions

The following table gives conventions for terms used throughout this document.

Table i. Conventions

Terms	Description
logic level one	The voltage level that corresponds to a Boolean true (1) state.
logic level zero	The voltage level that corresponds to a Boolean false (0) state.
ACTIVE_HIGH	Names for signals that are active high are shown in uppercase text without an overbar. Signals that are active high are referred to as asserted when they are logic 1 and negated when they are logic 0.
$\overline{\text{ACTIVE\_LOW}}$	A bar over a signal name indicates that the signal is active low. Active-low signals are referred to as asserted when they are logic 0 and negated when they are logic 1.
asserted	Signal is in the active logic state. In active high logic, the signal is asserted when it changes to logic level one; in active low logic, the signal is asserted when it changes to logic zero.
negated	Signal is in the inactive logic state. In active high logic, the signal is negated when it changes to logic level zero; in active low logic, the signal is negated when it changes to logic level one.
set	To establish logic level one on a bit or bits
clear	To establish logic level zero on a bit or bits
0x0000	Hexadecimal numbers
0b0011	Binary numbers
n	Indicates a numeric place holder. In register field contexts, indicates a value that may be written or read. In register names, indicates any one of a set of multiple, identical registers. For example, UCCR*n* indicates a reference to any one of the eMIOS Channel Control Registers, UCCR0 through UCCR15.
x	In certain contexts, such as bit or signal encoding, this indicates a don't care. For example, if a four-bit binary field is represented as 0bx001, the state of the first bit is a don't care. In other contexts, such as module or register names, this is a place holder for a letter to designate a module instantiation. For example, ATD_*x* indicates a reference to either ATD_A or ATD_B.
b	Bit place holder

Terminology

The following table lists definitions for abbreviations and names used throughout this document

Table ii. Terminology

Terms	Description
AIPS	AMBA to IPS interface unit
ALC	Amplitude Limitation Control
ATD	Analog-to-digital converter with DMA interface with 16 channels, resolution of 10 bits
Baud Rate	Rate of data transmission in bits per second.
Byte	8 bits
CAN	Controller Area Network, a serial communication protocol defined in Reference 1 and Reference 2.
CFM	Common Flash Module. Acronym used throughout this document to reference the Flash memory module. The CFM includes the Common Flash bus interface, IP bus interface, Flash command controller, Flash memory controller, and Flash arrays.
Clock Phase	Determines when the data should be sampled relative to the active edge of SCK
Clock Polarity	Determines the idle state of the SCK signal.
coherency / coherent access	Coherent access is used to indicate an action to guarantee data consistency, preventing data from being accessed simultaneously using different methods in such a way that it is not completely updated before being used.
Command Write Sequence	A three-stop command instruction sequence to program, erase, or verify the Flash memory.
CPI	CAN Protocol Interface, a FlexCAN sub-module containing the CAN protocol engine.
CPU	Central Processor Unit
CRC	Cyclic Redundancy Check.
CRG	Clock and Reset Generator module
CS	Chip Select. In Master Mode, the CS signal is used to select which slave device to talk to.
DAC	Digital to Analog Converter: Converts a binary value into a voltage
DAIC	Double Action Input Capture
DAOC	Double Action Output Compare
Debug Mode	This is a system mode intended for debugging operations. When this mode is triggered, a global Debug Mode Request signal is sent to all modules, so that they can prepare themselves with debugging capabilities.
DMA	Direct memory access
DMA Mux	Direct Memory Access Multiplexer module
Dominant Bit	A dominant bit wins the arbitration on the CAN bus. It is transmitted as ‘0.’
Doze Mode	This is a system low power mode in which the CPU bus is kept alive and a global Doze Mode request is sent to all peripherals asking them to enter low power mode. Typically, when Doze Mode is requested, each peripheral can be enabled individually to enter or not low power mode.
DSPI	(Deserialized) Serial Peripheral Interface

Table ii. Terminology (continued)

Terms	Description
EIM	External Interface Module
eDMA	Enhanced Direct Memory Access controller module
eMIOS	Enhanced Modular Input/Output Subsystem
EOQ	End of Queue
Erase State	Flash array bit state that reads as a “1.”
eSCI	Enhanced SCI module with LIN hardware and DMA support
Field	Two or more register bits grouped together.
Flash Array	A non-volatile SuperFlash® memory array used to build the Program Flash blocks, which includes a Flash memory core with built-in high voltage generation and parametric features.
Flash Logical Page	4096 bytes of contiguous Flash memory consisting of two interleaved Flash physical blocks representing the smallest section of the Flash memory that can be erased.
Flash Logical Sector	Section of contiguous Flash memory that can be protected from program, erase, and unauthorized access.
Flash User Mode	Flash module operations defined for User/Normal mode.
Frame	The data content of a serial transmission. Also referred to as DSPI Data.
FSM	Finite State Machine
GPIO	General purpose Input/Output
Half word	16 bits
Hard Reset	Reset coming from external pin and/or following power-on. It resets everything.
Host	Refers to the MCU or other bus master module
input capture	Sampling of a time base value upon the occurrence of an input signal transition.
IPF	Input Programmable Filter
IPS	Intelligent Peripheral Subsystem bus interface
IPM	Input Period Measurement
IPWM	Input Pulse Width Measurement
LC	Loop Control
LIN	Local Interconnect Network – A protocol for low-cost automobile networks
LIN FSM	LIN Finite State Machine – The control logic of the LIN hardware
LSB	Least Significant Bit
LVR	Low Voltage Reset
match	Match an event that occurs when the value of a match register becomes equal to the value of the selected time-base.
MC	Modulus Counter
MCU	Microcontroller Unit.
Message Buffer (MB)	Internal FlexCAN data structure containing bytes received or to be transmitted to the CAN line, as well as information about this data.

Table ii. Terminology (continued)

Terms	Description
MSB	Most Significant Bit
MUX	Multiplexer
NSM	New command word state machine
OpAmp	Operational Amplifier
OPWFM	Output Pulse Width and Frequency Modulation
OPWM	Output Pulse Width Modulation
OPWMC	Center Aligned Output Pulse Width Modulation
OSCCLK	Oscillator clock
Output Compare	The modification of an output signal due to a time base match.
PCS	Peripheral Chip Select
PEA	Pulse/Edge Accumulation
PEC	Pulse/Edge Counting
Pipeline	Act of initiating a bus cycle while another bus cycle is in progress. Thus the bus can have multiple bus cycles pending at one time.
PIT	Periodic Interrupt Timer
PLL	Phase Locked Loop
POR	Power on Reset
Program State	Flash array bit state that reads as a “0.”
QDEC	Quadrature Decode
RC	Resistor-Capacitor
Receive or RX FIFO	First-In-First-Out buffer for received data
Recessive Bit	A recessive bit loses the arbitration on the CAN bus. It is transmitted as ‘1.’
Reset Sequence	Coming out of reset, the CFM will read the Flash configuration field and load specific registers.
RTI	Real Time Interrupt - A timer with an independent clock which can run in system Doze or Pseudo-Stop mode, and can be used for system wakeup.
SAIC	Single Action Input Capture
SAOC	Single Action Output Compare
SAR	Successive Approximation Register: A method to adjust a reference voltage to an input voltage
SCI	Serial Communications Interface
SCM	Self Clock Mode
Serialize	To convert data from a parallel format to a serial format.
Slave	A bus slave is a device that responds to a bus transaction, but never initiates a cycle on the bus.
Soft Reset	Global reset typically used by peripherals to re-initialize some of its registers, but not all of them.
SPI	Serial Peripheral Interface

Table ii. Terminology (continued)

Terms	Description
SS	Slave Select. Signal from the SPI master to the SPI slave indicating which SPI slave device the Master want to communicate with.
Stop Mode	This is a system low power mode in which all MCU clocks are stopped for maximum power savings. Typically, when Stop Mode is requested, each module will put itself in a known state and then send a Stop Acknowledge signal to inform the CPU that it can stop the clocks.
SWT	Software Watchdog Timer
SYSCLK	System clock (f_{SYS}) is the clock used by the core CPU (peripherals operate at $f_{SYS} \div 2$)
Transaction	A bus transaction consists of an address transfer (address phase) and one or more data transfer(s) (data phase).
Transmit or TX FIFO	First-In-First-Out buffer for transmit data
UC*n*	Unified channel *n*, submodule that performs timed input or output functions supported by the eMIOS
VCO	Voltage Controlled Oscillator
VREG	Voltage regulator
V_{RH}	High reference voltage
V_{RL}	Low reference voltage
Word	32 bits
WPTA	Windowed Programmable Time Accumulation

Register Descriptions

Each peripheral module chapter (Chapter 3, Chapter 4 and Chapter 9 through Chapter 26) contains a register description subsection that details the location and definition of the user-accessible control and status bits and fields for the peripheral. All register descriptions in this manual use bit 31 to represent the most significant bit and bit 0 to represent the least significant bit. Refer to “Accessing Registers” on page 8-100 for details on how registers may be accessed.

Each register description subsection includes a register diagram figure showing bit field mnemonic names and locations followed by a table containing the full name for each bit field, a short description of operational characteristics and exact bit value definitions. The figure and tables below show an example of the format used for register figures, field descriptions and the conventions used to specify bit fields.

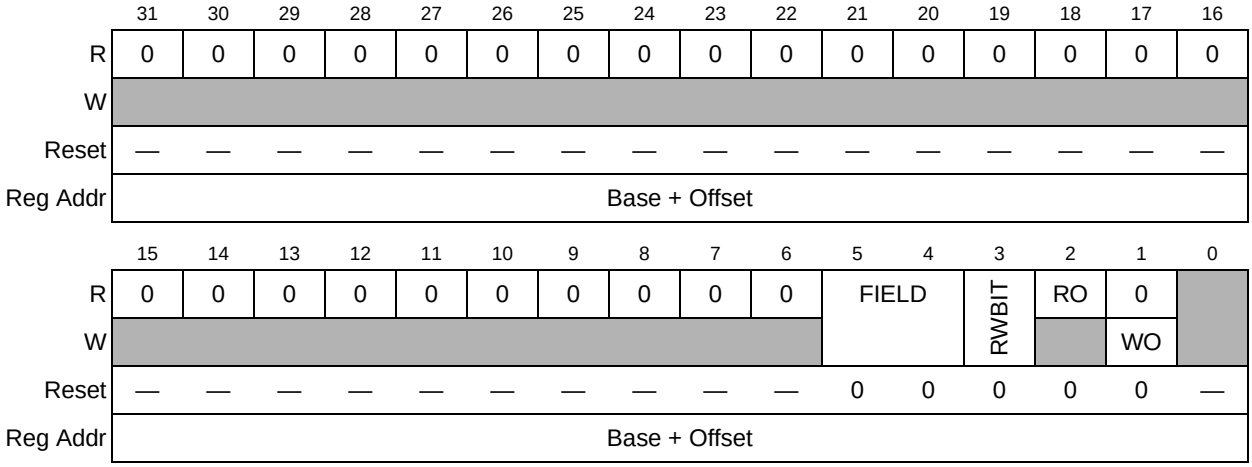


Figure i. Example Register (EXREG)

Table iii. Register Diagram Conventions

Row Label	Column(s) Content
xx	Bit number, specifies the location of the bit or field within the register.
R	Behavior for read accesses. If named, the description table below the register diagram specifies the definition. If zero, the bit will always read as zero. If shaded, the read value is undefined and the bit position is reserved and must be ignored for future compatibility.
W	Behavior for write accesses. If named, the description table below the register diagram specifies the definition. If shaded, the bit position is reserved and must be written as zero for future compatibility.
Reset	Bit state immediately after a reset operation. Zero or one indicates how the bit state is affected by reset, an emdash indicates that the bit state is not affected by reset.
Reg Addr	Specifies the address of the most significant byte of the register (for a 16- or 32-bit register). Base is the module address as specified in Chapter 8, “Device Memory Map,” Table 8-10 on page 8-99. Offset is a hexadecimal number in the format 0x0000, or a formula used to calculate the offset, that is added to Base to calculate the address of the register.

Table iv. EXREG Field Descriptions

Bits	Name	Description
15–6	—	Reserved; always reads as zero, must be written as zero.
5–4	FIELD[1:0]	Bit field. This is a two-bit read/write field. 00 Behavior when field is written as 0b00, or status indicated when read as 0b00 01 Behavior when field is written as 0b01, or status indicated when read as 0b01 10 Behavior when field is written as 0b10, or status indicated when read as 0b10 11 Behavior when field is written as 0b11, or status indicated when read as 0b11
3	RWBIT	Read/write bit. This is a writable control bit that is used to specify certain behavior of the module. Reading the bit position will return the last value written. 0 Behavior when bit is cleared 1 Behavior when bit is set
2	RO	Read-only. Normally used for status bits that reflect operating characteristics at the time of the read access. Writes are ignored, but for future compatibility zero should always be written. 0 Status definition if bit is clear 1 Status definition if bit is set
1	WO	Write-only bit. Normally used for control bits that trigger an event when written as one, but always read as zero. 0 Writing zero has no effect 1 Writing one triggers event
0	—	Undefined bit. Normally indicates bits that are used only for factory testing and must not be modified by customer code, and whose read contents are indeterminate. Writing non-zero values may cause erratic device behavior.

Table v. Register Field Description Conventions

Column Label	Row Content
Bits	Bit number, specifies the exact location of the bit field within the register. Always listed from the most significant bit in the register to the least significant. For bit fields, the most significant and least significant bit numbers are shown.
Name	Mnemonic name of the bit or field. For bit fields, the mnemonic is followed by bracketed numbers specifying the size of the field.
Description	Full name of the bit or field. Includes a short definition of operational characteristics. Often contains cross-references to detailed functional descriptions of module behaviors that are affected by a control field or that are reflected in a status field. Followed by paragraphs listing definitions for each possible value of the bit or field. For more complex definitions, or where bits or fields interact with other bits or fields in various modes, tables are often embedded within the field description cell for more detailed information. 0 single-bit value definitions 1 single-bit value definitions 0..0 first in a list of multi-bit field value definitions . . . 1..1 last in a list of multi-bit-field value definitions

Revision History

Content Changes by Document Version

Version No. Release Date	Description of Changes	Page Numbers
v0.5 31-Oct-03	**First public customer release (preliminary).**	
v0.6 21-May-04	General: • Miscellaneous updates for presentation consistency. • Added blue highlight to cross-references for easier PDF navigation.	
	Preface • Added Register Descriptions conventions detail.	lvii
	Chapter 1, "Introduction." • Added MAC71x2 descriptions. • Corrected TMS pin direction and added $\overline{AS}$ to Figure 1-1 ($\overline{TA}$ / $\overline{AS}$ pin). • Expanded Table 1-1 to clarify module availability. • Updated Section 1.1, Table 1-1, and Section 1.3	 1-2 1-3 1-1, 1-3, 1-4
	Chapter 2, "Signal Description." • Added PD2 functionality descriptions as needed. • Added $\overline{TA}$ / $\overline{AS}$ description as Section 2.1.1.8, pull-up/down added to Table 2-2. • Corrected Table 2-1 (CAN and eSCI were identified in Port B). • Removed pin assignments from Table 2-2 (content moved to MAC7100EC/D). • Added new PIM/SSM functions (JTAG, $\overline{AS}$, debug status) to Table 2-2 • Reworked Section 2.2 and Table 2-3 to match the VREG chapter and to correctly identify $V_{DD}A/V_{SS}A$ as inputs.	 2-12 2-13 2-22 2-22 2-20, 2-26
	Chapter 3, "Voltage Regulator Module (VREG)." • Corrected Table 3-1 and Section 3.4.2 to show $V_{DD}A$ as an input. • Added Section 3.5.1.1, Section 3.6.8.2. • Moved circuit board layout information from Section 3.4 to Section 3.7 • Added 208 MAP BGA package layout information (Figure 3-10 and Figure 3-11).	 3-31, 3-32 3-33, 3-39 3-39 3-44, 3-44
	Chapter 4, "System Clocks Module (OSC and CRG)." • Corrected CAUTION text in Section 4.3.6.1 to use "parameters" versus "dividers." • Corrected SYNR register field name to SYN in Figure 4-8. • Added stop-entry flag information to Section 4.3.5.3. • Added fast wake-up information to Section 4.3.5.6 and Section 4.3.6.10.7. • Added minimum POR cycle count toSection 4.3.6.7.1 and Figure 4-22. • Updated or added Figure 4-20, Figure 4-23, Figure 4-27, Figure 4-28 and Figure 4-29. • Added JTAG reset information to Section 4.3.6.7.2.	 4-60 4-54 4-55 4-58, 4-79 4-67, 4-68 4-65, 4-68, 4-76, 4-80, 4-80 4-68
	Chapter 6, "Exceptions." • Added "Refer To" column to Table 6-2 to aid in document navigation. • Clarified VREG and FlexCAN vector source descriptions. • Some wake-up sources now share vectors with previously defined exceptions: – RTI added to vector 0x0013. **NOTE: Incorrect, see v0.6.1 revisions below.** – FlexCAN wake-up added to vectors 0x001A, 0x001D, 0x0020 and 0x0023.	 6-85

Version No. Release Date	Description of Changes	Page Numbers
v0.6 21-May-04 (cont'd)	**Chapter 7, "Modes of Operation."** • Expanded Section 7.1, adding Section 7.1.1 through Section 7.1.4. • Clarified Section 7.2.2.1 • Added Section 7.2.3.1 and Section 7.2.3.2. • Expanded Section 7.3	 7-87 7-89 7-90, 7-90 7-90
	Chapter 8, "Device Memory Map." • Updated program Flash and SRAM sizes for MAC71x2 devices. • Content restructuring for clarity and to remove redundant information: – Added Table 8-1 to clarify AAMR interaction with memory map. – Re-worked Figure 8-1 to be more accurate, added A*n*Slave notes. – Re-worked Section 8.1.1 through Section 8.1.5, added AAMR limitations. – Moved all peripheral map information to Section 8.1.6 / Table 8-10. – Corrected CFM register address range in Table 8-2 through Table 8-6.	 8-93 8-94 8-95 to 8-97 8-99 8-95 to 8-97
	Chapter 9, "ARM7TDMI-S™ Processor Core." • Added implementation version note and reference to external documentation.	 9-101
	Chapter 10, "Interrupt Controller Module (INTC)." • Added Table 10-2 to show peripheral-to-IRC*n* correspondence.	 10-106
	Chapter 11, "Miscellaneous Control Module (MCM)." • Updated Table 11-3 with new mask set identifiers. • Added Section 11.4 and Section 11.4.1.	 11-125 11-138
	Chapter 12, "Enhanced Direct Memory Access Controller Module (eDMA)." • Consistent use of "major" and "minor," removed redundant "outer" and "inner." • Added new feature descriptions (fixed-priority preemption, round-robin arbitration, channel linking, and scatter/gather). See Section 12.3.1.1, Section 12.3.1.2, Section 12.3.1.15, Section 12.3.1.16, Section 12.3.1.16.6, Section 12.3.1.16.7 and Section 12.3.1.16.8. • Clarified TCDn initialization requirement in Section 12.3.1.16. • Minor clarifications in Section 12.4.1. • Updated example header code in Section 12.5.1 for changed/new TCD*n* fields. • Added Section 12.5.5, Section 12.5.7.3, Section 12.5.8 and Section 12.5.9.	 12-144, 12-145, 12-155, 12-156, 12-160, 12-161, 12-161 12-156 12-163 12-167 12-173, 12-178, 12-178, 12-180
	Chapter 13, "External Interface Module (EIM)." • Corrected CSMR*n*[V] bit reset definition in Figure 13-3. • Mask set L47W and L61W updates: – Added note at front of chapter regarding L61W. – Modified body text and added footnotes for L47W changes. • Clarified CSMR*n*[BAM] / CSAR*n* function with example in Table 13-5. • Clarified global chip select mode description in Section 13.6.1.2. • Added access alignment restriction to Section 13.6.2.1. • Add $\overline{AS}$ to descriptive text, flowcharts and timing diagrams, with appropriate $\overline{TA}$ / $\overline{AS}$ footnotes: – Section 13.6.2.4, Figure 13-9, Figure 13-10 – Section 13.6.2.5, Figure 13-11, Figure 13-12 – Section 13.6.2.6, Figure 13-13, Figure 13-14 – Section 13.6.2.7, Figure 13-15 – Section 13.6.2.8.2, Figure 13-16, Figure 13-17 – Section 13.6.2.8.3, Figure 13-18, Figure 13-19	 13-186 13-187 13-191 13-192 13-195 13-196 13-197 13-198 13-199 13-200
	Chapter 14, "Cross-Bar Switch Module (XBS)." • Added MAC7112, MAC7122 and MAC7142 part numbers to Section 14.1.	 14-203

Version No. Release Date	Description of Changes	Page Numbers
v0.6 21-May-04 (cont'd)	**Chapter 15, "Common Flash Module (CFM)."** • Added Figure 15-26 and supporting text to Section 15.4.1.5.4 to clarify page-to-sector relationship. • Added WARNING paragraph. • Added new feature descriptions: – Section 15.1 updated for CFMCLKSEL register feature. – Section 15.2 updated for data signature feature. – Table 15-3 updated to show CFMDATA and CFMCLKSEL registers. – MRDS field added to Figure 15-6 and Table 15-4. – Data Signature added to command list in Table 15-14 and Table 15-18. – Added Section 15.3.1.12, Section 15.3.1.14, Section 15.4.1.5.6 and Section 15.5.	15-238 15-236 15-209 15-211 15-214 15-215 15-225 15-225, 15-243, 15-248
	Chapter 16, "AMBA to IP Bus Bridge Module (AIPS)." • Simplified Section 16.4 introduction • Reworded Section 16.4.1.1, Section 16.4.1.2 and Section 16.4.1.3 for clarity.	16-253 16-255, 16-255, 16-256
	Chapter 18, "Port Integration Module (PIM)." • Updated to describe mask set L47W and L61W functionality: – Added PD2 functionality descriptions as needed, footnoted for L49P – Added $\overline{TA}$ / $\overline{AS}$ and JTAG pin control info to Section 18.1 – Added $\overline{TA}$ / $\overline{AS}$, JTAG. debug status and EIM control info to Section 18.2 – Modified Figure 18-1 to include $\overline{TA}$ / $\overline{AS}$ and JTAG pins – Added register descriptions: Section 18.5.1.6, Section 18.5.1.7, Section 18.5.1.8, Section 18.5.1.9, Section 18.5.1.10, Section 18.5.1.11 and Section 18.5.1.12. – Updated Section 18.6.1 with $\overline{TA}$/$\overline{AS}$ and JTAG descriptions. – Updated Section 18.7.2.4 to include GLBINT utilization. – Updated Section 18.7.3.2 and Section 18.7.4 to clarify L61W capability. • Added Applications Information: Section 18.7.3, Section 18.7.4, Section 18.7.5, Section 18.7.6 and Section 18.7.7. • Clarified code examples.	18-271 18-273 18-271 18-289, 18-291, 18-292, 18-293, 18-294, 18-294 18-296 18-309 18-313, 18-314 18-313, 18-314, 18-314, 18-315
	Chapter 19, "Analog-to-Digital Converter Module (ATD)." • Removed reference to software reset in Section 19.1. • Corrected Table 19-9 bit 30 sub-table definition for CWSC=0 & CWCH[3:0] = xxxx (previously indicated "Reserved"). • Changed all references to $(V_{RH} - V_{RL}) \div 2$ to $(V_{RH} + V_{RL}) \div 2$ • Added Section 19.7.11. • Added "not recommended" notes to all references to amplifier bypass (Table 19-9, Section 19.6.2.2 and Section 19.7.10).	19-317 19-328 19-328, 19-334, 19-346, 19-348 19-348 19-328, 19-333, 19-347
	Chapter 20, "Enhanced Modular I/O Subsystem Module (eMIOS)." • Corrected UCDIS[31:24] reset value in Figure 20-5. • Clarified Table 20-8 regarding EDSEL and EDPOL. • Corrected Figure 20-13. • Clarified Section 20.6.7.11. • Updated Section 20.6.7.12 to reflect L47W changes, adding Table 20-11. • Clarified Section 20.7.2.	20-357 20-360 20-368 20-379 20-380, 20-383 20-402

Version No. Release Date	Description of Changes	Page Numbers
v0.6 21-May-04 (cont'd)	**Chapter 21, "Enhanced Serial Communications Interface Module (eSCI)."** • Structure changes for consistency (register groups under single headings). • Removed Tx IRQ generation and LIN/DMA linkage arrows from Figure 21-1. • Removed reference to software reset in Section 21.1. • Updated for L47W and L61W mask set devices: – ESCICR4 added to memory map, Table 21-1. – Register descriptions updated: Section 21.5.1.4, Section 21.5.1.5, Section 21.5.1.6 and Section 21.5.1.7. – Added Figure 21-7. – Added Figure 21-29 thru Figure 21-33 and supporting text – Updated Section 21.6.19.4 and Section 21.6.20. – Added Section 21.7.2.6 and Section 21.7.2.7. • FBR, BESM13 and SBSTP bit descriptions footnoted for L61W-specific features: – Figure 21-6 / Table 21-6 and Figure 21-7 / Table 21-7. • Added Section 21.6.9 including Figure 21-24 and Figure 21-25 • Corrected equation values in Section 21.6.15.1 and Section 21.6.15.2. • Added Section 21.7.1 • Removed "Timeout" from Figure 21-40 data structure.	 21-405 21-405 21-407 21-415, 21-417, 21-418, 21-420 21-410 21-433 – 21-435 21-440, 21-440 21-447, 21-447 21-410, 21-410 21-429 21-436,21-437 21-441 21-445
	Chapter 22, "Deserial Serial Peripheral Interface Module (DSPI)." • Removed reference to software reset in Section 22.1. • Added DSPI*x*_CTAR2 – DSPI*x*_CTAR5 and DSPI*x*_TXFR1 – DSPI*x*_TXFR3 to Table 22-2. • Updated Section 22.1, Figure 22-1, Section 22.2, Section 22.5.1.3, Figure 22-8, Section 22.5.1.8 and Section 22.6.3.4. Added DBR to Figure 22-5 and Figure 22-14, added DBR description to Table 22-5, changed BR[3:0] description equation in Table 22-5. • Updated Table 22-20 and Table 22-21 values for f_{SYS} = 40 MHz vs. 100 MHz.	 22-449 22-453 22-449, 22-450, 22-457, 22-466, 22-468, 22-473 22-457, 22-475 22-487, 22-488
	Chapter 23, "Controller Area Network Module (FlexCAN)." • Removed reference to software reset in Section 23.1. • Added footnotes for L49P / L47W wake-up handling differences. • Removed non-MAC71xx content from Table 23-8 (CLK_SRC), Section 23.6.1 (reception queue), Section 23.6.8.4 (low-pass filter disclaimer). • Unused MB note made more specific.	 23-491 23-501/522/523 23-502, 23-511, 23-521 23-523
	Chapter 25, "Periodic Interrupt Timer Module (PIT)." • Removed reference to software reset in Section 25.1. • Modified Table 25-9 and Section 25.5.4.2.1 for mask set L47W. • Added Figure 25-10 and supporting text. • Added Section 25.5.2.	 25-549 25-557, 25-560 25-559 25-557
	Chapter 26, "System Services Module (SSM)." • Updated for L47W mask set: – Section 26.1, Section 26.2, Table 26-1, Section 26.4.1.2, Table 26-3, Section 26.4.1.3, Table 26-7, Section 26.4.1.5, Section 26.5.1, Section 26.6.5 and Section 26.6.6 • Added for L47W mask set: – Section 26.4.1.4, Section 26.4.1.6, Section 26.6.3 and Section 26.6.4 • Removed redundant previous Table 26-3 (DMAACTCH Field Detail). • Added preliminary Section 26.6.4.	 26-563, 26-563, 26-564, 26-566, 26-566, 26-567, 26-569, 26-572, 26-575, 26-576 26-568, 26-570, 26-574, 26-574 26-565 26-574

Version No. Release Date	Description of Changes	Page Numbers
v0.6 21-May-04 (cont'd)	**Appendix A, "Debug Interface."** • Added optional debug port information to Section A.1.	A-577
	Appendix B, "A7S Nexus 2 Module." • Updated Section B.3.3.8 for L47W.	B-596
	Appendix C, "Register Memory Map Quick Reference." (Initial version previously released as part of Addendum, MAC7100RMAD/D.) • Appendix added; all memory map tables in Chapters standardized for terminology and presentation. • Updated for L47W and L61W mask sets: – Noted new fields in eDMA TCD detail (Table C-6a) – Added two registers to SSM map (Table C-8) – Added three registers to VREG map (Table C-12) – Added seven registers to DSPI map (Table C-15) – Added one register to eSCI map (Table C-16) – Added seven registers, footnotes to PIM map (Table C-19, Table C-20) – Added two registers to CFM map (Table C-21)	**C-621** C-624 C-626 C-628 C-629 C-630 C-632, C-633 C-634
	Appendix D, "Mask Set Differences Summary." • Initial version	D-635
v0.6.1 25-May-04	Corrected RTI interrupt assignment descriptions (shares vector with Timer 4): • Chapter 6, "Exceptions," Table 6-2 • Chapter 10, "Interrupt Controller Module (INTC)," Table 10-2 • Chapter 25, "Periodic Interrupt Timer Module (PIT)," Table 25-9, Section 25.5.4.2 • Appendix D, "Mask Set Differences Summary," Table D-2	6-85, 6-86 10-106 25-557, 25-560 D-635
v1.0 19-Oct-04	General: • Freescale Semiconductor, Inc. identity conversion. • MAC71x6 information added, as appropriate. • Miscellaneous other updates or corrections as noted below.	
	Chapter 1, "Introduction." • Clarified GPIO interrupt capability in introductory text • Added $\overline{AS}$, NEXPS/NEXPR, Port I and new DSPI chip selects to Figure 1-1	1-1 1-2
	Chapter 2, "Signal Description." • Reversed polarity of $\overline{XCLKS}$ description in Section 2.1.2.4.2 • Added Section 2.1.2.9 • Added Port I to Table 2-2.	2-16 2-20 2-22
	Chapter 3, "Voltage Regulator Module (VREG)." • Added 32-bit access restriction to Section 3.5.1	3-33
	Chapter 4, "System Clocks Module (OSC and CRG)." • Clarified clock name conventions throughout. • Reversed polarity of $\overline{XCLKS}$ description in Section 4.2.4.3 and Section 4.3.4.3 • Clarified clock names and functions in Table 4-15	4-49, 4-53 4-81
	Chapter 8, "Device Memory Map." • Updated Table 8-2, Table 8-3, Table 8-5, Table 8-6, Table 8-8 and Table 8-10 for MAC71*x*6 device Flash and SRAM sizes	8-95 through 8-99
	Chapter 11, "Miscellaneous Control Module (MCM)." • Added new mask set information to Table 11-4 and Table 11-20 • Clarified $\overline{RESET}$ in Figure 11-9, Figure 11-12, Figure 11-13, Figure 11-14, Figure 11-15 • Updated Section 11.3.1.11 to show AAMR[31:8] as read-only • Updated example in Section 11.4.1	11-125, 11-139 11-131 – 11-138 11-132 11-138

Version No. Release Date	Description of Changes	Page Numbers
v1.0 19-Oct-04 (cont'd)	**Chapter 13, "External Interface Module (EIM)."** • Corrected CSMR*n*[BAM] example in Table 13-5, adding footnote regarding maximum physical block size	13-187
	Chapter 15, "Common Flash Module (CFM)." • Changed all instances of FCLK to f_{NVMOP} and all clock references to be more specific (now uses only f_{IPS} and f_{NVMOP} throughout, except where f_{SYS} is needed for clarity) • Clarification of Section 15.1 introductory paragraph • Corrected Figure 15-3, Figure 15-4 and Figure 15-5 array designations • Added CFMDISU to Table 15-3 and added Section 15.3.1.13 • Corrected LOCK bit behavior description in Section 15.3.1.1 • Added footnotes to Figure 15-6 and modified Figure 15-4 to clarify write restrictions • Clarified clocking description in Table 15-5 • Restructured / clarified Section 15.4.1 through Section 15.4.1.5 for clarity. Promoted Section 15.4.1.6 up one heading level • Corrected Section 15.4.1.4 for consistent use of f_{NVMOP} • Added Table 15-17 • Updated Figure 15-27, Figure 15-28, Figure 15-29, Figure 15-30 and Figure 15-32 • Data Signature description (Section 15.4.1.5.6) clarified • Clarified Section 15.4.2 introductory paragraph to support Table 15-18 updates • Clarified Section 15.4.2.3 with cross reference to the new Appendix B content	15-209 15-212, 15-213 15-214, 15-226 15-214 15-215 15-216 15-227 – 15-246 15-229 15-230 15-235 – 15-246 15-243 15-247 15-248
	Chapter 17, "DMA Channel Multiplexer Module (DMAMux)." • Added PIT signals and router symbols to Figure 17-1	17-259
	Chapter 18, "Port Integration Module (PIM)." • Added Port I information to Table 18-1, Figure 18-1, Table 18-2, Table 18-3, Table 18-4 • Added 32-bit input port description (Section 18.5.1.13)	18-271 – 18-277 18-295
	Chapter 20, "Enhanced Modular I/O Subsystem Module (eMIOS)." • Clarified f_{IPS} references in Figure 20-1, Figure 20-12, Section 20.6.3 and Section 20.6.6 • Removed Global Time Base information from Figure 20-1, Figure 20-2 and Table 20-4 Added new buffered operating modes for MAC71*x*6 devices: • Section 20.3, Table 20-9, Section 20.6.7, Section 20.6.7.15, Section 20.6.7.16, Section 20.6.7.17, Section 20.6.7.18	20-351 – 20-368 20-351 – 20-355 20-352, 20-362, 20-368, 20-386, 20-389, 20-393, 20-398
	Chapter 21, "Enhanced Serial Communications Interface Module (eSCI)." • Updated Section 21.7.2.1 (added steps h, i, j)	21-443
	Chapter 22, "Deserial Serial Peripheral Interface Module (DSPI)." • Updated Section 22.1 and Section 22.2 for new chip select options • Added PCSIS[3:4, 6:7] to Table 22-1, Section 22.4.2, Section 22.4.3, Figure 22-3, Table 22-3, Figure 22-8, Table 22-8 • Removed reference to DCONT (DSI Continuous Peripheral Chip Select Enable) from Table 22-5 CPOL bit description • Changed references to "system clock" or f_{SYS} to "peripheral bus clock" or f_{IPS}: – Table 22-5 (DBR, PBR[1:0], CSSCK[3:0], ASC[3:0], DT[3:0], BR[3:0]), – Section 22.6.4 (all subsections and tables), – Figure 22-18, Figure 22-19 • Recalculated examples in Table 22-13, Table 22-14, Table 22-15, Table 22-16, Table 22-17, Table 22-18, Table 22-21 for f_{IPS} • Changed continuous mode descriptions Section 22.6.5.5, Section 22.6.6 • Added DBR = 1 condition for Table 22-20	22-449, 22-450 22-451 – 22-466 22-458 22-458 22-475 22-481, 22-482 22-476 – 22-488 22-483, 22-484 22-487

Version No. Release Date	Description of Changes	Page Numbers
v1.0 19-Oct-04 (cont'd)	**Chapter 25, "Periodic Interrupt Timer Module (PIT)."** • Added Figure 25-12 and descriptive text to Section 25.6 • Corrected example values in Section 25.6.1 for f_{IPS} versus f_{SYS}	 25-560, 25-562 25-560
	Chapter 26, "System Services Module (SSM)." • Nexus and EIM status bits added to STATUS register (Section 26.4.1.1) • Updated Table 26-3 and Table 26-11 for MAC71x6 devices • SRHOLE and new memory sizes added to Section 26.4.1.2	 26-564 26-566, 26-572 26-566
	Appendix B, "A7S Nexus 2 Module." • Appendix renamed to reflect expanded content (Venation Description) • Changed pin names on Figure B-1 to reflect MAC7100 implementation • Expanded Section B.1.3.2 and Section B.1.3.3 • Added Packet Name column to Table B-2 • Added MAC7100 implementation footnotes • Added WT usage note • Added Section B.4	 B-580 B-582 B-582 B-581, B-584, B-585, B-591, B-597 B-597 B-599 – B-616
	Appendix C, "Register Memory Map Quick Reference." • Added Port I offset to Table C-19 • Added 32-bit input registers to PIM map (Table C-20)	 C-632 C-633
	Appendix D, "Mask Set Differences Summary." • Updated to reflect MAC71*x*6 devices and latest mask set errata.	 **D-635**

Chapter 1
Introduction

1.1 Overview

MAC7100 microcontrollers (MCUs) are members of a pin-compatible family of 32-bit Flash-memory-based devices developed specifically for embedded automotive applications. The pin-compatible family concept enables users to select between different memory and peripheral options for scalable designs. All MAC7100 devices are composed of a 32-bit ARM7TDMI-S™ central processing unit, up to 1 Mbyte of high performance embedded Flash memory for program storage, an optional 32 Kbytes of embedded Flash for data and/or program storage, and up to 48 Kbytes of RAM.

The family is implemented with an enhanced DMA (eDMA) controller, which executes in parallel with the CPU to improve the performance of data transfers between memory and many of the on-chip peripherals. DMA transfers may be triggered by various peripheral events, such as data frame transmission or reception, elapsed timer periods, and analog-to-digital conversion completions. The peripheral set includes enhanced asynchronous serial communications interfaces (eSCI) with Local Interconnect Network (LIN) support hardware to reduce interrupt overhead, serial peripheral interfaces (DSPI) with flexible chip selects and fast baud rate switching, inter-integrated circuit (I^2C™) bus controllers, FlexCAN interfaces with flexible message buffering, an enhanced modular I/O subsystem (eMIOS) with sixteen high-performance 16-bit timers, one or two sixteen-channel 10-bit analog-to-digital converters (ATD), general-purpose timers (Programmable Interrupt Timer (PIT)) and two special-purpose timers (Real Time Interrupt (RTI) and Software Watchdog Timer (SWT)). The peripherals share a large number of general purpose input-output (GPIO) pins, all of which are bidirectional and may be configured to generate interrupts, which may be used to trigger wake-up from low-power modes. See Table 1-1 for a comparison of MAC7100 devices and the availability of peripheral modules on various devices.

Internal data paths between the CPU core, eDMA, memory and peripherals are all 32 bits wide, further improving performance for 32-bit applications. The MAC7111, MAC7116, MAC7131 and MAC7136 also offer a 16-bit wide external data bus with 22 address lines, allowing access of up to 4 MBytes of external address space. The inclusion of a programmable PLL module allows power consumption and performance to be adjusted to suit operational requirements. Both E-ICE and Nexus 2 interfaces are implemented to support development and debug tool chains.

MAC7100 devices include an on-chip multi-output voltage regulator, thus requiring only a single external 3.3V to 5V power supply. The maximum operating range of devices in the family covers a junction temperature of –40° C to 150° C and CPU clock frequencies up to 50 MHz. Packaging options range from 100-pin LQFP up to 208-pin MAP BGA.

1.2 Block Diagram

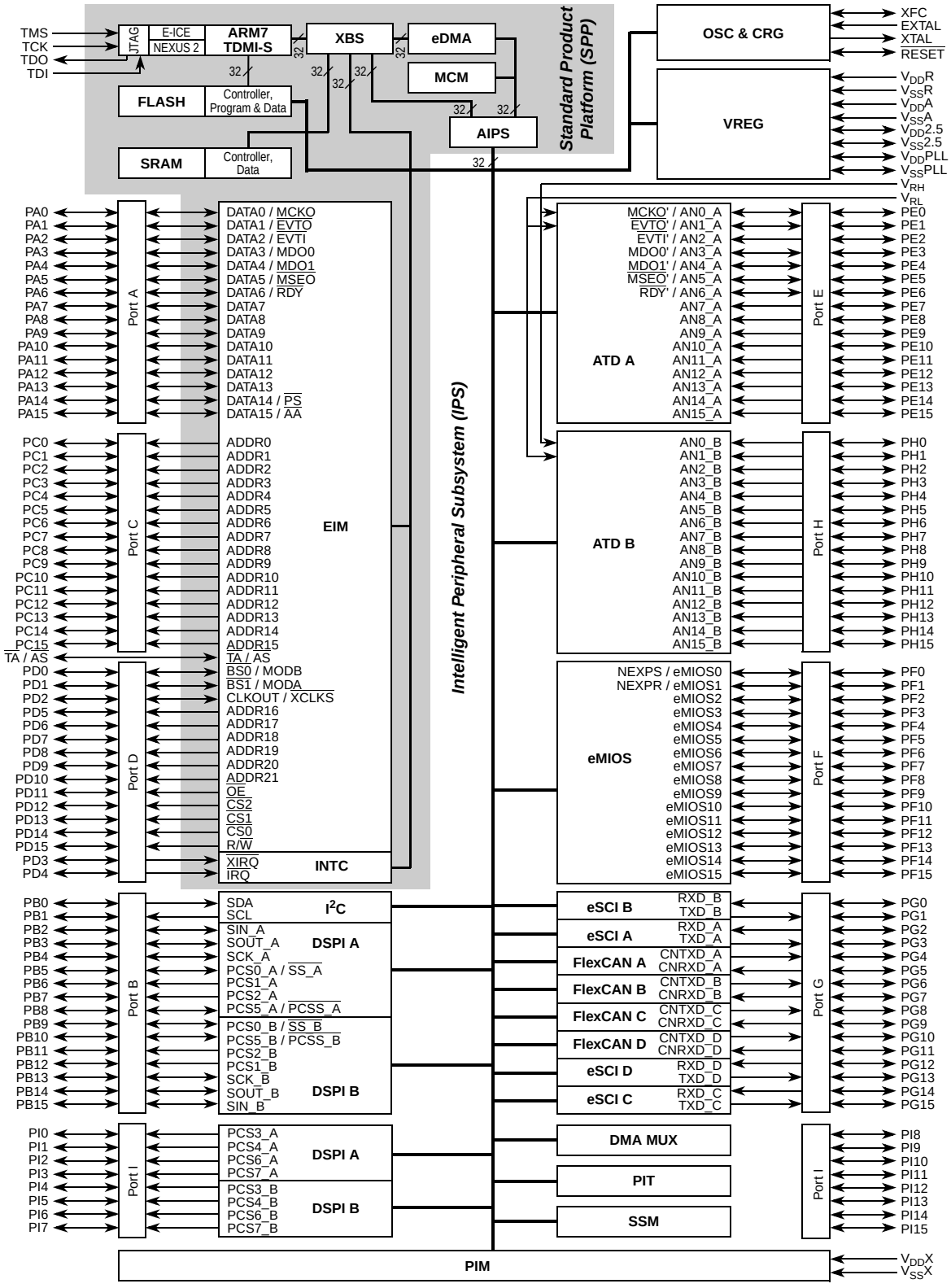


Figure 1-1. MAC7100 Family Block Diagram

The following table provides a comparison of members of the MAC7100 Family and the availability of peripheral modules on the various devices.

Table 1-1. MAC7100 Family Device Derivatives

Module Options		MAC7101	MAC7111	MAC7121	MAC7131	MAC7141	MAC7112	MAC7122	MAC7142	MAC7106	MAC7116	MAC7126	MAC7136
Program Flash		512 KBytes					256 KBytes			1 MByte			
Data Flash		32 KBytes											
SRAM		32 KBytes					16 KBytes			48 KBytes			
External Bus		—	Yes	—	Yes	—	—	—	—	—	Yes	—	Yes
ATD Modules [1]	A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	B	Yes	—	—	Yes	—	—	—	—	Yes	—	—	Yes
CAN Modules	A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	B	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	C	Yes	Yes	Yes	Yes	—	—	—	—	Yes	Yes	Yes	Yes
	D	Yes	Yes	Yes	Yes	—	—	—	—	Yes	Yes	Yes	Yes
eSCI Modules	A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	B	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	C	Yes	Yes	Yes	Yes	—	Yes	Yes	—	Yes	Yes	Yes	Yes
	D	Yes	Yes	Yes	Yes	Yes	—	—	—	Yes	Yes	Yes	Yes
DSPI Modules	A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes [2]
	B	Yes	Yes	Yes [3]	Yes	Yes	Yes	Yes [3]	Yes	Yes	Yes	Yes [3]	Yes [2]
I^2C Module		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
eMIOS Module		16 channels, 16-bit											
Timer Module		10 channels, 24-bit											
General-Purpose I/O Ports/Pins	A	10	16	10	16	4	16	10	4	10	16	10	16
	B	16	16	15	16	16	16	15	16	16	16	15	16
	C	12	16	1	16	—	16	1	—	12	16	1	16
	D	10 [4]	16 [4]	11 [4]	16 [4]	10 [4]	16	11	10	10	16	11	16
	E	16	16	16	16	16	16	16	16	16	16	16	16
	F	16	16	16	16	16	16	16	16	16	16	16	16
	G	16	16	16	16	10	16	16	10	16	16	16	16
	H	16	—	—	16	—	—	—	—	16	—	—	16
	I	—	—	—	—	—	—	—	—	—	—	—	16
Total (max.)		112 [4]	112 [4]	85 [4]	128 [4]	72 [4]	112	85	72	112	112	85	144
Package		144 LQFP	144 LQFP	112 LQFP	208 BGA	100 LQFP	144 LQFP	112 LQFP	100 LQFP	144 LQFP	144 LQFP	112 LQFP	208 BGA

1 16 channels, 8/10-bit, per module.

2 Four additional chip selects available.

3 PB11 / PCS2_B not available on non-L49P-mask devices; PB10 / PCS5_B / $\overline{\text{PCSS\_B}}$ not available on mask L47W devices.

4 Reduce these values by one for mask set L49P devices (PD2 is not available for general-purpose use).

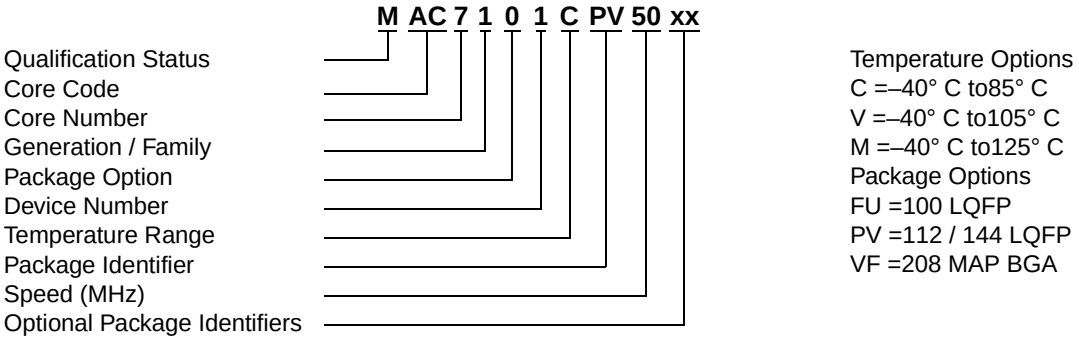


Figure 1-2. Order Part Number Example

1.3 Features

As shown in Figure 1-1, MAC7100 Family devices are organized into two major blocks: the standard product platform (SPP) and the intelligent peripheral subsystem (IPS). The SPP consists of the ARM7TDMI-S processor core and enhanced direct memory access (eDMA) controller connected to a high-performance 32-bit bus through a cross-bar bus switch (XBS) to the rest of the chip modules. The SPP also contains an interrupt controller (INTC), SRAM controller, Flash memory controller (CFM), external bus interface (EIM), peripheral bus bridge (AIPS), and miscellaneous control module (MCM).

The IPS consists of the voltage regulator (VREG), clock and reset generator (OSC & CRG), port integration module (PIM), DMA request multiplexer (DMAMux), analog-to-digital (ATD) converter(s), enhanced modular I/O subsystem (eMIOS), enhanced serial communications interface (eSCI) controllers, serial peripheral interface(s) (DSPI), FlexCAN controller(s), an inter-integrated circuit (I^2C) bus controller, programmable interrupt timer (PIT) and system services module (SSM).

The key features of each module are listed below:

- General MAC7100 Family features
  - — Up to 50MHz operating frequency.
  - — RISC core, eDMA and memory connected via high performance 32-bit bus.
  - — Separate 32-bit bus interface for slower system peripherals.
  - — External bus interface to support off-chip devices (7111/7116/7131/7136 only).
  - — Internal 5V to 2.5V Regulator.
- 32-bit ARM7TDMI-S RISC Core
  - — Supports 32-bit and 16-bit (THUMB) instruction sets for code size efficiency.
  - — 32 bit wide data path.
  - — Alternate general purpose registers.
  - — Byte (8-bit), halfword (16-bit), word (32-bit) data types supported.
- Enhanced Direct Memory Access (eDMA) Controller and Channel Multiplexer (DMAMux)
  - — Supports transfers between memory, external devices, peripherals (ATD, DSPI, eMIOS, eSCI and I^2C), and general-purpose I/O using a dual address transfer protocol.
  - — DMAMux allows assignment of any DMA request source to any eDMA channel.
  - — Programmable transfer control descriptors stored in local eDMA memory.
  - — Programmable source and destination address with configurable offset.

  - Programmable size, nested transfers via 32-bit major/16-bit minor loop counters.
  - Different final source and destination addresses allow circular queue operation.
  - Channel-to-channel linking and scatter/gather operation. [1]
  - Programmable priority levels for each channel, with channel preemption. [1]
  - Bandwidth control for each channel.
  - Independently programmable read/write sizes.
  - Periodic triggering of up to 8 channels.
- Memory options
  - Up to 1 MByte Program Flash EEPROM
    - 50 MHz single cycle non-sequential access for aligned halfword/word data.
    - State machine controlled program/erase operations; internal voltage generator.
    - Small Flash sector protection sizes.
    - Configurable flexible Flash protection fields.
    - Protection violation flag.
    - 10,000-cycle program / erase endurance
    - 15 year data retention.
  - Up to 32 KByte Data Flash EEPROM
    - 16-bit wide memory accessed via peripheral bus interface.
    - Relocatable to page zero to provide data Flash boot operation.
    - State machine controlled program/erase operations; internal voltage generator.
    - Up to 8 protected sectors in the data Flash.
    - 10,000-cycle program / erase endurance
    - 15 year data retention.
  - Up to 48 KByte RAM
    - Single cycle accesses to RAM for byte, halfword and word reads and writes.
- Interrupt Controller (INTC)
  - 64 vectored interrupt sources (44 peripheral, 17 DMA, 1 software watchdog timer, 2 external).
  - 16 programmable interrupt priorities for every source, even in low-power modes.
  - Multiple level interrupt nesting, with hardware support for first nesting level.
  - Normal and Fast interrupt support.
- General Purpose Input/Output
  - Up to 144 port pins shared with peripherals (for 208-pin package).
  - All ports are 16 bits wide, with pins bidirectional and independently controlled.
  - Port-wide and single-pin access methods for improved software performance.
  - Each pin configurable for drive type and strength, and internal pull-up/down.
  - Level or edge-sensitive interrupt available on all port pins.
- Analog-to-Digital Converter(s) (ATD)
  - One or two ATD modules.
  - 16 analog input channels per ATD module.
  - 10-bit resolution with ±2 counts accuracy.

1. Some eDMA features are not implemented on mask set L49P devices (refer to Chapter 12 for details).

  - 7 μS minimum conversion time.
  - Internal sample and hold circuitry.
  - Programmable input sample time for various source impedances.
  - Queued conversion sequences supported by eDMA controller.
  - Unused analog channels can be used as digital I/O.
  - External and on-chip sample triggers, including periodic triggering via the PIT.
  - Synchronized sampling on ATD modules using external or on-chip triggers.
- CAN 2.0 A, B software compatible modules (FlexCAN)
  - Up to four CAN modules.
  - Full implementation of the CAN 2.0 protocol specification.
  - Programmable bit rate up to 1 Mbps.
  - Up to 32 flexible message buffers of 0 to 8 bytes data length for each module.
  - All message buffers configurable for either Rx/Tx.
  - Unused message buffer space can be used as general purpose RAM.
  - Supports standard or extended messages.
  - Time stamp, based on a 16-bit free-running counter.
  - Maskable interrupts, including low-power mode wake up on bus activity.
  - Programmable I/O modes.
- Enhanced Modular I/O Subsystem (eMIOS)
  - 16 unified channels, each of which can be enabled for eDMA service.
  - Three 16-bit counter buses, with synchronization between timebases.
  - One global prescaler plus prescaler available on each channel.
  - Fourteen channel operating modes available:
    - General purpose input/output
    - Single action input capture
    - Single action output compare
    - Input pulse width measurement
    - Input period measurement
    - Double action output compare
    - Pulse/edge accumulation
    - Pulse/edge counting
    - Quadrature decode
    - Windowed programmable time accumulation modulus counter
    - Output pulse width and frequency modulation
    - Center aligned output pulse width modulation
    - Output pulse width modulation
  - Channels can be individually disabled to reduce power consumption.
- Serial Peripheral Interface (DSPI)
  - One or two DSPI modules.
  - Full duplex, synchronous transfers.
  - Master or slave operation.
  - Programmable master bit rates.

  - Programmable clock polarity and phase.
  - End-of-transmission interrupt flag.
  - Flexible baud rates available.
  - Programmable data frames from 4 bits to 16 bits.
  - Up to 8 chip selects support up to 256 external devices using external muxing.
  - Two DMA request lines per module for message frame support.
  - Separate transmit and receive FIFOs for improved system performance.
  - Queueing operation possible through use of the eDMA controllers channels.
  - Fast switching between SPI slave devices types via transfer attributes coupled with data.
  - General purpose I/O functionality on pins when not used for DSPI
- Enhanced Serial Communications Interface (eSCI)
  - Up to four eSCI modules.
  - Standard non-return-to-zero (NRZ) mark/space format.
  - Full-duplex operation.
  - Software selectable word length (8- or 9-bit words).
  - 10/11- or 13/14-bit break character formats available.
  - 13-bit programmable baud-rate modulus counter.
  - Separately enabled transmitter and receiver.
  - Separate receiver and transmitter CPU interrupt requests.
  - Programmable transmitter output polarity.
  - Two receiver wake-up methods.
  - Interrupt-driven operation with eight flags.
  - Receiver framing error detection.
  - Hardware parity checking.
  - 1/16-bit time noise reduction.
  - Two DMA request lines on per module for receive and transmit data.
  - Support for LIN bus protocol (version 1.2 and version 2.0).
    - Full LIN master node autonomous message frame handling.
    - Frame hardware significantly reduces interrupt overhead.
    - LIN message header generation.
    - Slave timeout detection.
    - Optional CRC message checking.
    - Detection and flagging of LIN errors.
- Inter-Integrated Circuit (I^2C) Bus module
  - Two wire bi-directional serial bus for on-board communications.
  - Compatibility with I^2C Bus standard.
  - Multi-master operation.
  - Software-programmable for one of 256 different serial clock frequencies.
  - Interrupt-driven byte-by-byte data transfer.
  - Arbitration-lost interrupt with automatic mode switching from master to slave.
  - Calling address identification interrupt.
  - Start and stop signal generation/detection.

  - Repeated START signal generation.
  - Acknowledge bit generation/detection.
  - Bus-busy detection.
  - Two DMA request lines to receive and transmit data
- Periodic Interrupt Timer (PIT) Module
  - Independent timeout periods for each of the ten 24-bit timers
  - One real-time interrupt (RTI) timer to wake up the CPU in wait mode or pseudo-stop mode
  - Eight timers that can be configured to generate DMA trigger pulses
  - Two timers that can be configured to generate ATD trigger pulses
  - Four timers that can be configured to generate interrupts instead of DMA triggers
- Miscellaneous Control Module (MCM)
  - Software watchdog timer with programmable, staged response.
  - Cross-bar switch remapping.
  - SPP module configuration status.
  - Access address information for faulted memory accesses.
- System Services Module (SSM)
  - System configuration and status
    - Memory sizes and status
    - Security status
    - Device mode
    - eDMA status
  - Wake-up Source ID
- Development support
  - Real-time instruction trace support via Nexus Class 2 Plus interface.
  - Selectable Nexus port position on 208-pin MAP BGA and 144-pin LQFP packaged devices.
  - ARM7 Embedded ICE debug interface.
  - JTAG Test Access Port (TAP) interface.
  - Debug mode access to CPU registers.
  - Real Time memory access.
  - Hardware Breakpoints.
- Clock generation
  - Selectable Standard Pierce or low-power Amplitude Loop Controlled (ALC) Pierce oscillator.
  - Phase-locked loop clock frequency multiplier.
  - Self-clocking mode available in absence of external clock.
  - Low power 0.5 to 16 MHz crystal oscillator reference clock.
  - Clock generation and monitor, reset control, and software watchdog timer.
- I/O lines with 5V input and drive capability.
- 5V ATD converter inputs and I/O lines with 5V input and drive capability.
- 2.5V logic supply.
- 208-pin MAP BGA, 144-pin LQFP, 112-pin LQFP and 100-pin LQFP package options.

1.4 Modes of Operation

Chip modes (Refer to Chapter 7, “Modes of Operation,” for more details)

- Normal Single-Chip Mode
  - All debug features available.
  - Boot from program Flash.
- Normal Expanded Mode
  - All debug features available.
  - Boot from off-chip device using the external bus interface.
  - Program and data Flash memory available.
- Secured Single-Chip Mode
  - No debug features available.
  - No external bus interface.
  - Boot from program Flash.
  - JTAG lockout recovery available.
- Secured Expanded Mode
  - All debug features available.
  - Boot from off-chip device using the external bus interface.
  - Program and data Flash memory not available.
- Data Flash Boot Mode
  - All debug features available.
  - Boot from data Flash.
- Secured Data Flash Boot Mode
  - No debug features available.
  - No external bus interface.
  - Boot from data Flash.
  - JTAG lockout recovery available.

Low power modes (available in all chip modes)

- Stop Mode (provides the lowest power consumption).
- Pseudo-Stop Mode.
- Doze Mode.

Chapter 2
Signal Description

MAC7100 Family device signals are divided into three primary groups: Clocks and Control, General Purpose/Peripheral I/O, and Power Supply. The following subsections contain descriptions of signal functionality in various operating modes, availability of signals in different packages and assignment of signals to pins in each package.

2.1 External Signal Description

2.1.1 Clocks and Control

2.1.1.1 EXTAL, XTAL — Oscillator

The crystal driver and external clock pins. On reset all the device clocks are derived from the EXTAL input frequency. XTAL is the crystal drive output.

2.1.1.2 XFC — PLL Loop Filter

The XFC pin allows the user to specify the external PLL loop filter components in order to modify the response rate and stability of the PLL. Refer to the *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC) for more details on this pin and for the calculation of the component values.

2.1.1.3 $\overline{\text{RESET}}$ — External Reset

The bidirectional, open drain, active low $\overline{\text{RESET}}$ signal acts as an input to initialize the MCU to a known start-up state, and an output when an internal MCU function causes a Reset.

2.1.1.4 TCK — Test Clock

The TCK pin is an input from the JTAG port. The signal provides the test clock input to synchronize the device test logic. TCK is independent of the processor clock.

2.1.1.5 TMS — Test Mode

The TMS pin is an input from the JTAG port. The input is used to initiate the test mode sequence of the TAP controller state machine. The signal is sampled on the rising edge of TCK.

2.1.1.6 TDI — Test Data In

The TDI pin is an input from the JTAG port. The signal is sampled on the rising edge of TCK.

2.1.1.7 TDO — Test Data Out

The TDO pin is an output to the JTAG port. It is a signal that is actively driven in the shift-IR and shift-DR controller states. The signal state changes on the falling edge of TCK.

2.1.1.8 $\overline{TA}$ / $\overline{AS}$ — Transfer Acknowledge / Address Strobe

This active-low synchronous input/output signal may be configured to indicate either the completion of a requested data transfer operation or the presence of a valid address on the external bus (present only on the MAC7111, MAC7116, MAC7131 and MAC7136). Note that the $\overline{AS}$ function is not available on mask set L49P devices.

2.1.1.9 TEST — Factory Test

This input only pin is reserved for Freescale factory testing. The pin must be tied to system ground in all applications.

2.1.1.10 Configuration and Optional Control Signals

In addition to the dedicated control signal pins described above, several of the general purpose / peripheral I/O pins described below also provide configuration or control functionality. Refer to Section 2.1.2.1.8 ($\overline{PS}$), Section 2.1.2.1.9 ($\overline{AA}$), Section 2.1.2.4.1 (MODA, MODB), Section 2.1.2.4.3 ($\overline{XIRQ}$) and Section 2.1.2.4.4 ($\overline{IRQ}$) for more information.

2.1.2 General Purpose / Peripheral I/O

MAC7100 Family devices have up to 128 pins (depending on package configuration) that can be used as general purpose or peripheral function I/O signals. Pins are grouped into 16-signal ports, with most pins capable of operating in two to four independent modes. Some pin modes are mutually exclusive while others are complementary.

When not in expanded mode (refer to Chapter 7, “Modes of Operation”), the default mode for all port pins is the general purpose input (GPI) function. As an input, a GPIO pin state can be read by software at any time, and either pull-up or pull-down resistors can be enabled or disabled for each pin. All port pins may also be configured as outputs, with the high or low state of a GPIO pin directly controlled by software via writing the appropriate port data register, and each pin can be independently configured for totem-pole or open-drain operation and to provide either high or reduced drive current. Refer to Chapter 18, “Port Integration Module (PIM),” for a more detailed description of the GPIO function.

In addition to the GPIO function, each pin can be independently configured to generate an interrupt and/or wake the system out of low-power mode (STOP or DOZE) when an external signal transition occurs. Refer to Section 7.3, “Power Consumption Considerations,” on page 7-90 and Section 18.7.2.4, “Using Port Interrupts,” on page 18-309 for more detailed information on the wake-up and interrupt functions.

In expanded mode (refer to Chapter 7, “Modes of Operation”), the GPIO mode for ports A, C and D (except for the PD3 and PD4) is overridden to provide the External Interface Module address and data bus function. Refer to Chapter 13, “External Interface Module (EIM),” for more information.

Depending on the state of the PF1/eMIOS0/NEXPS pin during $\overline{\text{RESET}}$ assertion, the GPIO mode for the lower six pins of either port A or E may be overridden to provide the Nexus debug port (refer to Appendix A, “Debug Interface”).

Most pins have an assigned peripheral input/output function that replaces the GPIO function when the associated peripheral module is enabled. A few pins also have an initialization function where the state of the signal is latched during $\overline{\text{RESET}}$ assertion and used to determine various operating modes. The following subsections describe in more detail the alternate functions associated with each port pin.

All signal functions described herein are not available on all devices. In packages smaller than 208 pins, all non-bonded out pins should be configured as an output after reset in order to avoid current draw from floating inputs. Table 2-1 below details which pins should be configured in this manner for each device.

Table 2-1. Pins Not Bonded Out By Device Derivative

Peripheral Function	MAC7101 MAC7106	MAC7111 MAC7116	MAC7112	MAC7121 MAC7126	MAC7122	MAC7131 MAC7136	MAC7141	MAC7142
External Bus Interface	$\overline{\text{TA}}$ / $\overline{\text{AS}}$ [1] PA[12:7] PC[15:12] PD[10:5]	—	—	$\overline{\text{TA}}$ / $\overline{\text{AS}}$ [1] PA[6:1] PC[14:0] PD[15:11]	PA[6:1] PC[14:0] PD[15:11]	—	$\overline{\text{TA}}$ / $\overline{\text{AS}}$ [1] PA[14:10,6:0] PC[15:0] PD[15:11,5]	PA[14:10,6:0] PC[15:0] PD[15:11,5]
DSPI Modules	—	—	—	PB11 [2]	PB11	—	—	—
eSCI Modules	—	—	—	—	—	—	PG[15:14]	PG[15:14]
CAN Modules	—	—	—	—	—	—	PG[11:8]	PG[11:8]
ATD Modules	—	PH[15:0]	—	PH[15:0]	—	—	PH[15:0]	—
PIM Module	PI[15:0] [3]	PI[15:0] [3]	—	PI[15:0] [3]	—	—	—	—

1 On mask set L49P devices, it is not possible to apply internal pull-up/down to $\overline{\text{TA}}$ or select the $\overline{\text{AS}}$ output to reduce power consumption.

2 On mask set L49P devices, PB10 is not bonded and must be configured as an output (PB11 may be used as required).

3 Mask set L38Y devices only.

2.1.2.1 Port A Signal Group

NOTE

If the Nexus interface is used in the primary position (PA[6:0]), the external bus size [1] must be 8 bits. Refer to Section 13.6.1.1, “8- and 16-Bit Port Sizing,” on page 13-191 for more information.

1. Only on devices that implement the EIM.

2.1.2.1.1 PA0 / DATA0 / MCKO — Port A I/O, external data bus [1] or Nexus

PA0 is multiplexed with the external data bus signal, DATA0, and the Nexus clock output, MCKO. PA[6:0] is the default position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG.

2.1.2.1.2 PA1 / DATA1 / $\overline{\text{EVTO}}$ — Port A I/O, external data bus [1] or Nexus

PA1 is multiplexed with the external data bus signal, DATA1, and the Nexus event out signal, $\overline{\text{EVTO}}$. PA[6:0] is the default position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG.

2.1.2.1.3 PA2 / DATA2 / $\overline{\text{EVTI}}$ — Port A I/O, external data bus [1] or Nexus

PA2 is multiplexed with the external data bus signal, DATA2, and the Nexus event in signal, $\overline{\text{EVTI}}$. PA[6:0] is the default position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG.

2.1.2.1.4 PA[4:3] / DATA[4:3] / MDO[1:0] — Port A I/O, external data bus [1] or Nexus

PA[4:3] are multiplexed with the external data bus signals, DATA[4:3], and the Nexus data out signals, MDO[1:0]. PA[6:0] is the default position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG.

2.1.2.1.5 PA5 / DATA5 / $\overline{\text{MSEO}}$ — Port A I/O, external data bus [1] or Nexus

PA5 multiplexed with the external data bus signal, DATA5, and the Nexus message start/end out signal, $\overline{\text{MSEO}}$. PA[6:0] is the default position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG.

2.1.2.1.6 PA6 / DATA6 / $\overline{\text{RDY}}$ — Port A I/O, external data bus [1] or Nexus

PA6 is multiplexed with the external data bus signal, DATA6, and the Nexus ready signal, $\overline{\text{RDY}}$. PA[6:0] is the default position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG.

2.1.2.1.7 PA[13:7] / DATA[13:7] — Port A I/O or external data bus [1]

PA[13:7] are multiplexed with the external data bus signals, DATA[13:7].

2.1.2.1.8 PA14 / DATA14 / $\overline{\text{PS}}$ — Port A I/O or external data bus [1]

PA14 is multiplexed with the external data bus signal, DATA14. At reset the level on the pin is read to determine width of the external data bus for expanded mode. [1] If the signal level is low an 8-bit data bus will be used, if the signal level is high a 16-bit data bus will be used.

2.1.2.1.9 PA15 / DATA15 / $\overline{\text{AA}}$ — Port A I/O or external data bus [1]

PA15 is multiplexed with the external data bus signal, DATA15. At reset the signal level on the pin is read to determine if the auto acknowledge feature is used for $\overline{\text{CS0}}$ in global chip select mode. [1] If the signal level on the pin is low, auto acknowledge is used; if the signal level is high, an external acknowledge must be supplied.

1. Only on devices that implement the EIM.

2.1.2.2 Port B Signal Group

2.1.2.2.1 PB0 / SDA — Port B I/O or I^2C bus

PB0 is multiplexed with the I^2C serial data signal, SDA.

2.1.2.2.2 PB1 / SCL — Port B I/O or I^2C bus

PB1 is multiplexed with the I^2C serial clock, SCL.

2.1.2.2.3 PB2 / SIN_A — Port B I/O or DSPI_A

PB2 is multiplexed with the DSPI A serial data in, SIN_A.

2.1.2.2.4 PB3 / SOUT_A — Port B I/O or DSPI_A

PB3 is multiplexed with the DSPI A serial data out, SOUT_A.

2.1.2.2.5 PB4 / SCK_A — Port B I/O or DSPI_A

PB4 is multiplexed with the DSPI A serial clock, SCK_A.

2.1.2.2.6 PB5 / PCS0_A / $\overline{\text{SS\_A}}$ — Port B I/O or DSPI_A

PB5 is multiplexed with the DSPI A chip select, PCS0_A, when in master mode or slave select, $\overline{\text{SS\_A}}$**,** when in slave mode.

2.1.2.2.7 PB[7:6] / PCS[2:1]_A — Port B I/O or DSPI_A

PB[7:6] are multiplexed with the DSPI A chip selects, PCS[2:1]_A, when in master mode.

2.1.2.2.8 PB8 / PCS5_A / $\overline{\text{PCSS\_A}}$ — Port B I/O or DSPI_A

PB8 is multiplexed with the DSPI A chip select, PCS5_A, when in master mode or as a peripheral chip select strobe, $\overline{\text{PCSS\_A}}$, to qualify the chip selects PCS[2:0]_A.

2.1.2.2.9 PB9 / PCS0_B / $\overline{\text{SS\_B}}$ — Port B I/O or DSPI_B

PB9 is multiplexed with the DSPI B chip select, PCS0_B, when in master mode or slave select, $\overline{\text{SS\_B}}$, when in slave mode.

2.1.2.2.10 PB10 / PCS5_B / $\overline{\text{PCSS\_B}}$ — Port B I/O or DSPI_B

PB10 is multiplexed with the DSPI B chip select, PCS5_B, when in master mode or as a peripheral chip select strobe, $\overline{\text{PCSS\_B}}$, to qualify the chip selects PCS[2:0]_B.

2.1.2.2.11 PB[12:11] / PCS[1:2]_B — Port B I/O or DSPI_B

PB[12:11] are multiplexed with the DSPI B chip selects, PCS[1:2]_B, when in master mode.

2.1.2.2.12 PB13 / SCK_B — Port B I/O or DSPI_B

PB13 is multiplexed with the DSPI B serial clock, SCK_B.

2.1.2.2.13 PB14 / SOUT_B — Port B I/O or DSPI_B

PB14 is multiplexed with the DSPI B serial data out, SOUT_B.

2.1.2.2.14 PB15 / SIN_B — Port B I/O or DSPI_B

PB15 is multiplexed with the DSPI B serial data in, SIN_B.

2.1.2.3 Port C Signal Group

2.1.2.3.1 PC[15:0] / ADDR[15:0] — Port C I/O or external address bus [1]

PC[15:0] are multiplexed with the external address bus signals, ADDR[15:0].

2.1.2.4 Port D Signal Group

2.1.2.4.1 PD[1:0] / $\overline{\text{BS}}$[1:0] / MOD[A:B] — Port D I/O or bus control, [1] and mode select

PD[1:0] are multiplexed with the external data bus byte select signals, $\overline{\text{BS}}$[1:0]. At reset the signal levels on these pins are read to determine the chip operating mode. Refer to Table 7-1 on page 7-87 for more information.

2.1.2.4.2 PD2 / CLKOUT / $\overline{\text{XCLKS}}$ — Port D I/O or Clock Out, and Oscillator Selection

PD2 is multiplexed with the clock out signal, which is used to output the system clock (f_{SYS}) for use with the external bus or to drive external synchronous devices. At reset the signal level on the pin is read to determine the mode of the Pierce oscillator. If the signal level is high, the oscillator operates in the loop controlled mode; if the signal level is low, the oscillator operates in the full swing or external clock mode.

NOTE

The PD2 function is not available on mask set L49P devices, as the corresponding pin position is used only for the CLKOUT / $\overline{\text{XCLKS}}$ functions.

2.1.2.4.3 PD3 / $\overline{\text{XIRQ}}$ — Port D I/O or interrupt input

PD3 is multiplexed with the interrupt request input, $\overline{\text{XIRQ}}$. Refer to Section 10.6, “Functional Description,” on page 10-116 for more information on interrupt signal behavior.

2.1.2.4.4 PD4 / $\overline{\text{IRQ}}$ — Port D I/O or interrupt input

PD4 is multiplexed with the interrupt request input, $\overline{\text{IRQ}}$. Refer to Section 10.6, “Functional Description,” on page 10-116 for more information on interrupt signal behavior.

1. Only on devices that implement the EIM.

2.1.2.4.5 PD[10:5] / ADDR[21:16] — Port D I/O or external address bus [1]

PD[10:5] are multiplexed with the external address bus, ADDR[21:16].

2.1.2.4.6 PD11 / $\overline{\text{OE}}$ — Port D I/O or bus control [1]

PD11 is multiplexed with the output enable, $\overline{\text{OE}}$.

2.1.2.4.7 PD[14:12] / $\overline{\text{CS}}$[0:2] — Port D I/O or bus control [1]

PD[14:12] are multiplexed with external bus chip select, $\overline{\text{CS}}$[0:2].

2.1.2.4.8 PD15 / R/$\overline{\text{W}}$ — Port D I/O or bus control [1]

PD15 is multiplexed with the external bus read/write, R/$\overline{\text{W}}$.

2.1.2.5 Port E Signal Group

2.1.2.5.1 PE0 / AN0_A / MCKO' — Port E I/O, ATD_A input or Nexus'

PE0 is multiplexed with the analog-to-digital converter input AN0_A and the Nexus clock output, MCKO'. PE[6:0] is the alternate position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG.

2.1.2.5.2 PE1 / AN1_A / $\overline{\text{EVTO}}$' — Port E I/O, ATD_A input or Nexus'

PE1 is multiplexed with the analog-to-digital converter input AN1_A and the Nexus event out signal, $\overline{\text{EVTO}}$'. PE[6:0] is the alternate position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG.

2.1.2.5.3 PE2 / AN2_A / $\overline{\text{EVTI}}$' — Port E I/O, ATD_A input or Nexus'

PE2 is multiplexed with the analog-to-digital converter input AN2_A and the Nexus event in signal, $\overline{\text{EVTI}}$'. PE[6:0] is the alternate position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG.

2.1.2.5.4 PE[4:3] / AN[4:3]_A / MDO[1:0]' — Port E I/O, ATD_A input or Nexus'

PE[4:3] is multiplexed with the analog-to-digital converter inputs AN[4:3]_A and the Nexus data out signals, MDO[1:0]'. PE[6:0] is the alternate position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG.

2.1.2.5.5 PE5 / AN5_A / $\overline{\text{MSEO}}$' — Port E I/O, ATD_A input or Nexus'

PE5 is multiplexed with the analog-to-digital converter input AN5_A and the Nexus message start/end out signal, $\overline{\text{MSEO}}$'. PE[6:0] is the alternate position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG.

1. Only on devices that implement the EIM.

2.1.2.5.6 PE6 / AN6_A / $\overline{\text{RDY}}$' — Port E I/O, ATD_A input or Nexus'

PE6 is multiplexed with the analog-to-digital converter input AN6_A and the Nexus ready signal, $\overline{\text{RDY}}$'. PE[6:0] is the alternate position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG.

2.1.2.5.7 PE[15:7] / AN[15:7]_A — Port E I/O or ATD_A input

PE[15:7] are multiplexed with the analog-to-digital converter inputs AN[15:7]_A.

2.1.2.6 Port F Signal Group

2.1.2.6.1 PF0 / eMIOS0 / NEXPS — Port F I/O or eMIOS I/O, and Nexus port selection

PF0 is multiplexed with the eMIOS unified channel 0 signal, eMIOS0. At reset the signal level on the pin is read to determine the position of the Nexus auxiliary port when Nexus debugging is enabled via JTAG. If the signal level is low the PA[6:0] position is used, if the signal level is high the PE[6:0] position is used.

2.1.2.6.2 PF1 / eMIOS1 / NEXPR — Port F I/O or eMIOS I/O, and Nexus present

PF1 is multiplexed with the eMIOS unified channel 1 signal, eMIOS1. At reset the signal level on the pin is read to determine if the Nexus Port can be enabled. If the signal level is low the Nexus interface is disabled, if the signal level is high the Nexus interface can be enabled based on the state of the JTAG $\overline{\text{EVTI}}$ signal (PA2 / DATA2 / $\overline{\text{EVTI}}$ or PE2 / AN2_A / $\overline{\text{EVTI}}$').

2.1.2.6.3 PF[15:2] / eMIOS[15:2] — Port F I/O or eMIOS I/O

PF[15:2] are multiplexed with the eMIOS unified channel signals, eMIOS[15:2].

NOTE

If the optional debug status port is enabled (see Section 26.4.1.6, “SSM Debug Status Port Control Register (DEBUGPORT),” on page 26-570), the PF[15:0] and eMIOS[15:0] signal functions are not available on the pins. The debug status port feature is not available on mask set L49P devices.

2.1.2.7 Port G Signal Group

2.1.2.7.1 PG0 / RXD_B — PORT G I/O or eSCI_B

PG0 is multiplexed with the eSCI controller B receive data signal, RXD_B.

2.1.2.7.2 PG1 / TXD_B — PORT G I/O or eSCI_B

PG1 is multiplexed with the eSCI controller B transmit data signal, TXD_B.

2.1.2.7.3 PG2 / RXD_A — PORT G I/O or eSCI_A

PG2 is multiplexed with the eSCI controller A receive data signal, RXD_A.

2.1.2.7.4 PG3 / TXD_A — PORT G I/O or eSCI_A

PG3 is multiplexed with the eSCI controller A transmit data signal, TXD_A.

2.1.2.7.5 PG4 / CNTX_A — PORT G I/O or FlexCAN_A

PG4 is multiplexed with the CAN controller A transmit signal, CNTX_A.

2.1.2.7.6 PG5 / CNRX_A — PORT G I/O or FlexCAN_A

PG5 is multiplexed with the CAN controller A receive signal, CNRX_A.

2.1.2.7.7 PG6 / CNTX_B — PORT G I/O or FlexCAN_B

PG6 is multiplexed with the CAN controller B transmit signal, CNTX_B.

2.1.2.7.8 PG7 / CNRX_B — PORT G I/O or FlexCAN_B

PG7 is multiplexed with the CAN controller B receive signal, CNTX_B.

2.1.2.7.9 PG8 / CNTX_C — PORT G I/O or FlexCAN_C

PG8 is multiplexed with the CAN controller C transmit signal, CNTX_C.

2.1.2.7.10 PG9 / CNRX_C — PORT G I/O or FlexCAN_C

PG9 is multiplexed with the CAN controller C receive signal, CNTX_C.

2.1.2.7.11 PG10 / CNTX_D — PORT G I/O or FlexCAN_D

PG10 is multiplexed with the CAN controller D transmit signal, CNTX_D.

2.1.2.7.12 PG11 / CNRX_D — PORT G I/O or FlexCAN_D

PG11 is multiplexed with the CAN controller D receive signal, CNTX_D.

2.1.2.7.13 PG12 / RXD_D — PORT G I/O or eSCI_D

PG12 is multiplexed with the eSCI controller D receive data signal, RXD_D.

2.1.2.7.14 PG13 / TXD_D — PORT G I/O or eSCI_D

PG13 is multiplexed with the eSCI controller D transmit data signal, TXD_D.

2.1.2.7.15 PG14 / RXD_C — PORT G I/O or eSCI_C

PG14 is multiplexed with the eSCI controller C receive data signal, RXD_C.

2.1.2.7.16 PG15 / TXD_C — PORT G I/O or eSCI_C

PG15 is multiplexed with the eSCI controller C transmit data signal, TXD_C.

2.1.2.8 Port H Signal Group

2.1.2.8.1 PH[15:0] / AN[15:0]_B — PORT H I/O or ATD_B input

PH[15:0] are multiplexed with the analog-to-digital converter inputs AN[15:0]_B.

2.1.2.9 Port I Signal Group

NOTE

Port I is available only on mask set L38Y devices in the 208-pin MAP BGA package (MAC7136).

2.1.2.9.1 PI[3:0] / PCS[7:6, 4:3]_A — Port I I/O or DSPI_A

PI[3:0] are multiplexed with the DSPI A chip selects, PCS[7:6, 4:3]_A, when in master mode.

2.1.2.9.2 PI[7:4] / PCS[7:6, 4:3]_B — Port I I/O or DSPI_B

PI[7:4] are multiplexed with the DSPI B chip selects, PCS[7:6, 4:3]_B, when in master mode.

2.1.2.9.3 PI[15:8] — Port I I/O

PH[15:8] are used only for general-purpose input/output.

2.2 Power Supply, Bypass and Reference

3.3 V or 5 V power is supplied to all MAC7100 Family devices via a set of V_{DD} and V_{SS} pins. Internal logic operates at 2.5 V, therefore an on-chip voltage regulator is used to generate the necessary internal voltage.

Because fast signal transitions place short-duration high current demands on the power supply, bypass capacitors with high-frequency characteristics should be used and placed as close to the MCU as possible. Bypass requirements depend on how heavily the MCU pins are loaded. Refer to Chapter 3, “Voltage Regulator Module (VREG),” for details on filter component selection and board layout.

NOTE

All V_{SS} pins must be connected to the same system ground.

2.2.1 $V_{DD}X$,$V_{SS}X$ — I/O Drivers Power and Ground

Power and ground inputs for I/O drivers. The nominal voltage for $V_{DD}X$ with respect to $V_{SS}X$ is 3.3 V or 5 V.

2.2.2 $V_{DD}R$, $V_{SS}R$ — Internal Voltage Regulator Supply

The $V_{DD}R$ and $V_{SS}R$ inputs provide the power input to the internal voltage regulator. The nominal voltage for $V_{DD}R$ with respect to $V_{SS}R$ is 3.3 V or 5 V. The internal voltage regulator is disabled if $V_{DD}R$ is tied to ground.

2.2.3 $V_{DD}A$, $V_{SS}A$ — Analog Reference Supply

The $V_{DD}A$ and $V_{SS}A$ inputs provide the quiet reference voltage for the internal voltage regulator and analog-to-digital converter(s) (ATD(s)). These pins are used to support separate external filter capacitors for the reference voltage, independent of the main I/O supply ($V_{DD}X$ and $V_{SS}X$) decoupling capacitors.

2.2.4 $V_{DD}2.5$, $V_{SS}2.5$ — Core Power Supply Bypass

$V_{DD}2.5$ and $V_{SS}2.5$ are connected to the primary output of the on-chip 2.5 V voltage regulator, which supplies the internal core logic. When the internal voltage regulator is enabled, no static load is allowed on $V_{DD}2.5$ and these pins are used only to provide external filter capacitors to improve regulator performance. If the internal voltage regulator is disabled, an external regulator must be used to power the core logic via these pins.

2.2.5 $V_{DD}PLL$, $V_{SS}PLL$ — PLL Power Supply Bypass

$V_{DD}PLL$ and $V_{SS}PLL$ are connected to the secondary output of the on-chip 2.5 V voltage regulator, which supplies the Pierce oscillator. When the internal voltage regulator is enabled, no static load is allowed on $V_{DD}PLL$ and these pins are used only to provide external filter capacitors to improve regulator performance. If the internal voltage regulator is disabled, an external regulator must be used to power the PLL via these pins.

2.2.6 V_{RH}, V_{RL} — ATD Reference Voltage

V_{RH} and V_{RL} are the reference voltage input pins for the resistor ladder digital-to-analog converter (DAC) block of the analog-to-digital ATD converter module(s).

2.3 Signal Properties Summary

Table 2-2 summarizes signal properties. Refer to the *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC) for availability of signals in various package options.

Table 2-2. Signal Properties Summary

Primary / GPIO Function [1]	Peripheral Function [1]	External Bus Function [1]	Debug Function [1]	Read on Reset	Internal Resistors		Power Supply	Description
					CTRL	Reset State		
EXTAL	—	—	—	—	N/A	None	V_{DD}PLL	Oscillator pins
XTAL	—	—	—	—	N/A	None	V_{DD}PLL	
XFC	—	—	—	—	N/A	None	V_{DD}PLL	PLL loop filter
$\overline{\text{RESET}}$	—	—	—	—	—	None	V_{DD}X	External Reset
TDI	—	—	—	—	PIM	Up [2]	V_{DD}X	JTAG test data in
TDO	—	—	—	—	PIM	None [2]	V_{DD}X	JTAG test data out
TCK	—	—	—	—	PIM	Down [2]	V_{DD}X	JTAG test clock
TMS	—	—	—	—	PIM	Up [2]	V_{DD}X	JTAG test mode
—	—	$\overline{\text{TA}}$ / $\overline{\text{AS}}$ [3]	—	—	PIM	Up [2]	V_{DD}A	External bus control
PA0	—	DATA0	MCKO	—	TDI	Disabled	V_{DD}X	Port A I/O, external data bus, primary Nexus port
PA1	—	DATA1	$\overline{\text{EVTO}}$	—	TDI	Disabled	V_{DD}X	
PA2	—	DATA2	$\overline{\text{EVTI}}$	—	TDI	Disabled	V_{DD}X	
PA3	—	DATA3	MDO0	—	TDI	Disabled	V_{DD}X	
PA4	—	DATA4	MDO1	—	TDI	Disabled	V_{DD}X	
PA5	—	DATA5	$\overline{\text{MSEO}}$	—	TDI	Disabled	V_{DD}X	
PA6	—	DATA6	$\overline{\text{RDY}}$	—	TDI	Disabled	V_{DD}X	
PA7	—	DATA7	—	—	—	Disabled	V_{DD}X	Port A I/O, external data bus
PA8	—	DATA8	—	—	—	Disabled	V_{DD}X	
PA9	—	DATA9	—	—	—	Disabled	V_{DD}X	
PA10	—	DATA10	—	—	—	Disabled	V_{DD}X	
PA11	—	DATA11	—	—	—	Disabled	V_{DD}X	
PA12	—	DATA12	—	—	—	Disabled	V_{DD}X	
PA13	—	DATA13	—	—	—	Disabled	V_{DD}X	
PA14	—	DATA14	—	$\overline{\text{PS}}$	—	Disabled	V_{DD}X	Port A I/O, ext. data bus, port size / $\overline{\text{CS0}}$ auto-ack
PA15	—	DATA15	—	$\overline{\text{AA}}$	—	Disabled	V_{DD}X	
PB0	SDA	—	—	—	—	Disabled	V_{DD}X	Port B I/O, I^2C bus
PB1	SCL	—	—	—	—	Disabled	V_{DD}X	
PB2	SIN_A	—	—	—	—	Disabled	V_{DD}X	Port B I/O, DSPI_A (serial data in and out, serial clock. chip select 0 or slave select, chip selects 1 and 2, chip select 5 or chip select strobe)
PB3	SOUT_A	—	—	—	—	Disabled	V_{DD}X	
PB4	SCK_A	—	—	—	—	Disabled	V_{DD}X	
PB5	PCS0_A / $\overline{\text{SS\_A}}$	—	—	—	—	Disabled	V_{DD}X	
PB6	PCS1_A	—	—	—	—	Disabled	V_{DD}X	
PB7	PCS2_A	—	—	—	—	Disabled	V_{DD}X	
PB8	PCS5_A / $\overline{\text{PCSS\_A}}$	—	—	—	—	Disabled	V_{DD}X	

Table 2-2. Signal Properties Summary (continued)

Primary / GPIO Function [1]	Peripheral Function [1]	External Bus Function [1]	Debug Function [1]	Read on Reset	Internal Resistors		Power Supply	Description
					CTRL	Reset State		
PB9	PCS0_B / SS_B	—	—	—	—	Disabled	$V_{DD}X$	Port B I/O, DSPI_B (chip select 0 or slave select, chip select 5 or chip select strobe, chip selects 2 and 1, serial clock, serial data out and in)
PB10	PCS5_B / PCSS_B	—	—	—	—	Disabled	$V_{DD}X$	
PB11	PCS2_B	—	—	—	—	Disabled	$V_{DD}X$	
PB12	PCS1_B	—	—	—	—	Disabled	$V_{DD}X$	
PB13	SCK_B	—	—	—	—	Disabled	$V_{DD}X$	
PB14	SOUT_B	—	—	—	—	Disabled	$V_{DD}X$	
PB15	SIN_B	—	—	—	—	Disabled	$V_{DD}X$	
PC0	—	ADDR0	—	—	—	Disabled	$V_{DD}X$	Port C I/O, external address bus
PC1	—	ADDR1	—	—	—	Disabled	$V_{DD}X$	
PC2	—	ADDR2	—	—	—	Disabled	$V_{DD}X$	
PC3	—	ADDR3	—	—	—	Disabled	$V_{DD}X$	
PC4	—	ADDR4	—	—	—	Disabled	$V_{DD}X$	
PC5	—	ADDR5	—	—	—	Disabled	$V_{DD}X$	
PC6	—	ADDR6	—	—	—	Disabled	$V_{DD}X$	
PC7	—	ADDR7	—	—	—	Disabled	$V_{DD}X$	
PC8	—	ADDR8	—	—	—	Disabled	$V_{DD}X$	
PC9	—	ADDR9	—	—	—	Disabled	$V_{DD}X$	
PC10	—	ADDR10	—	—	—	Disabled	$V_{DD}X$	Port C I/O, external address bus
PC11	—	ADDR11	—	—	—	Disabled	$V_{DD}X$	
PC12	—	ADDR12	—	—	—	Disabled	$V_{DD}X$	
PC13	—	ADDR13	—	—	—	Disabled	$V_{DD}X$	
PC14	—	ADDR14	—	—	—	Disabled	$V_{DD}X$	
PC15	—	ADDR15	—	—	—	Disabled	$V_{DD}X$	
PD0	—	$\overline{BS0}$	—	MODB	—	Down	$V_{DD}X$	Port D I/O, external bus byte select, mode select input
PD1	—	$\overline{BS1}$	—	MODA	—	Down	$V_{DD}X$	
PD2 [4]	CLKOUT	—	—	$\overline{XCLKS}$	—	Disabled	$V_{DD}X$	Port D I/O, clock, osc select
PD3	$\overline{XIRQ}$	—	—	—	—	Disabled	$V_{DD}X$	Port D I/O, external interrupt inputs
PD4	$\overline{IRQ}$	—	—	—	—	Disabled	$V_{DD}X$	
PD5	—	ADDR16	—	—	—	Disabled	$V_{DD}X$	Port D I/O, external address bus
PD6	—	ADDR17	—	—	—	Disabled	$V_{DD}X$	
PD7	—	ADDR18	—	—	—	Disabled	$V_{DD}X$	
PD8	—	ADDR19	—	—	—	Disabled	$V_{DD}X$	
PD9	—	ADDR20	—	—	—	Disabled	$V_{DD}X$	
PD10	—	ADDR21	—	—	—	Disabled	$V_{DD}X$	

Table 2-2. Signal Properties Summary (continued)

Primary / GPIO Function [1]	Peripheral Function [1]	External Bus Function [1]	Debug Function [1]	Read on Reset	Internal Resistors		Power Supply	Description
					CTRL	Reset State		
PD11	—	$\overline{OE}$	—	—	—	Disabled	$V_{DD}X$	Port D I/O, external bus control
PD12	—	$\overline{CS2}$	—	—		Disabled	$V_{DD}X$	
PD13	—	$\overline{CS1}$	—	—		Disabled	$V_{DD}X$	
PD14	—	$\overline{CS0}$	—	—	—	Disabled	$V_{DD}X$	
PD15	—	R/$\overline{W}$	—	—	—	Disabled	$V_{DD}X$	
PE0	AN0_A	—	MCKO'	—	TDI	None	$V_{DD}A$	Port E I/O, ATD_A analog input, alternate Nexus port
PE1	AN1_A	—	$\overline{EVTO}$'	—	TDI	None	$V_{DD}A$	
PE2	AN2_A	—	$\overline{EVTI}$'	—	TDI	None	$V_{DD}A$	
PE3	AN3_A	—	MDO0'	—	TDI	None	$V_{DD}A$	
PE4	AN4_A	—	MDO1'	—	TDI	None	$V_{DD}A$	
PE5	AN5_A	—	$\overline{MSE0}$'	—	TDI	None	$V_{DD}A$	
PE6	AN6_A	—	$\overline{RDY}$'	—	TDI	None	$V_{DD}A$	
PE7	AN7_A	—	—	—	—	None	$V_{DD}A$	Port E I/O, ATD_A analog input
PE8	AN8_A	—	—	—	—	None	$V_{DD}A$	
PE9	AN9_A	—	—	—	—	None	$V_{DD}A$	
PE10	AN10_A	—	—	—	—	None	$V_{DD}A$	
PE11	AN11_A	—	—	—	—	None	$V_{DD}A$	
PE12	AN12_A	—	—	—	—	None	$V_{DD}A$	
PE13	AN13_A	—	—	—	—	None	$V_{DD}A$	
PE14	AN14_A	—	—	—	—	None	$V_{DD}A$	
PE15	AN15_A	—	—	—	—	None	$V_{DD}A$	
PF0	eMIOS0	—	Debug Status	NEXPS	—	Disabled	$V_{DD}X$	Port F I/O, eMIOS I/O, Nexus port selection / port present, optional debug status port [5]
PF1	eMIOS1	—	Debug Status	NEXPR	—	Disabled	$V_{DD}X$	
PF2	eMIOS2	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF3	eMIOS3	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF4	eMIOS4	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF5	eMIOS5	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF6	eMIOS6	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF7	eMIOS7	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF8	eMIOS8	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF9	eMIOS9	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF10	eMIOS10	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF11	eMIOS11	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF12	eMIOS12	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF13	eMIOS13	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF14	eMIOS14	—	Debug Status	—	—	Disabled	$V_{DD}X$	
PF15	eMIOS15	—	Debug Status	—	—	Disabled	$V_{DD}X$	

Table 2-2. Signal Properties Summary (continued)

Primary / GPIO Function [1]	Peripheral Function [1]	External Bus Function [1]	Debug Function [1]	Read on Reset	Internal Resistors		Power Supply	Description
					CTRL	Reset State		
PG0	RXD_B	—	—	—	—	Disabled	$V_{DD}X$	Port G I/O, eSCI_B serial data
PG1	TXD_B	—	—	—	—	Disabled	$V_{DD}X$	
PG2	RXD_A	—	—	—	—	Disabled	$V_{DD}X$	Port G I/O, eSCI_A serial data
PG3	TXD_A	—	—	—	—	Disabled	$V_{DD}X$	
PG4	CNTX_A	—	—	—	—	Disabled	$V_{DD}X$	Port G I/O, CAN_A serial data
PG5	CNRX_A	—	—	—	—	Disabled	$V_{DD}X$	
PG6	CNTX_B	—	—	—	—	Disabled	$V_{DD}X$	Port G I/O, CAN_B serial data
PG7	CNRX_B	—	—	—	—	Disabled	$V_{DD}X$	
PG8	CNTX_C	—	—	—	—	Disabled	$V_{DD}X$	Port G I/O, CAN_C serial data
PG9	CNRX_C	—	—	—	—	Disabled	$V_{DD}X$	
PG10	CNTX_D	—	—	—	—	Disabled	$V_{DD}X$	Port G I/O, CAN_D serial data
PG11	CNRX_D	—	—	—	—	Disabled	$V_{DD}X$	
PG12	RXD_D	—	—	—	—	Disabled	$V_{DD}X$	Port G I/O, eSCI_D serial data
PG13	TXD_D	—	—	—	—	Disabled	$V_{DD}X$	
PG14	RXD_C	—	—	—	—	Disabled	$V_{DD}X$	Port G I/O, eSCI_C serial data
PG15	TXD_C	—	—	—	—	Disabled	$V_{DD}X$	
PH0	AN0_B	—	—	—	—	Disabled	$V_{DD}A$	Port H I/O, ATD_B analog input
PH1	AN1_B	—	—	—	—	Disabled	$V_{DD}A$	
PH2	AN2_B	—	—	—	—	Disabled	$V_{DD}A$	
PH3	AN3_B	—	—	—	—	Disabled	$V_{DD}A$	
PH4	AN4_B	—	—	—	—	Disabled	$V_{DD}A$	
PH5	AN5_B	—	—	—	—	Disabled	$V_{DD}A$	
PH6	AN6_B	—	—	—	—	Disabled	$V_{DD}A$	
PH7	AN7_B	—	—	—	—	Disabled	$V_{DD}A$	
PH8	AN8_B	—	—	—	—	Disabled	$V_{DD}A$	
PH9	AN9_B	—	—	—	—	Disabled	$V_{DD}A$	
PH10	AN10_B	—	—	—	—	Disabled	$V_{DD}A$	
PH11	AN11_B	—	—	—	—	Disabled	$V_{DD}A$	
PH12	AN12_B	—	—	—	—	Disabled	$V_{DD}A$	
PH13	AN13_B	—	—	—	—	Disabled	$V_{DD}A$	
PH14	AN14_B	—	—	—	—	Disabled	$V_{DD}A$	
PH15	AN15_B	—	—	—	—	Disabled	$V_{DD}A$	
PI0 [6]	PCS3_A	—	—	—	—	Disabled	$V_{DD}X$	Port I I/O, DSPI_A chip selects
PI1 [6]	PCS4_A	—	—	—	—	Disabled	$V_{DD}X$	
PI2 [6]	PCS6_A	—	—	—	—	Disabled	$V_{DD}X$	
PI3 [6]	PCS7_A	—	—	—	—	Disabled	$V_{DD}X$	
PI4 [6]	PCS3_B	—	—	—	—	Disabled	$V_{DD}X$	Port I I/O, DSPI_B chip selects
PI5 [6]	PCS4_B	—	—	—	—	Disabled	$V_{DD}X$	
PI6 [6]	PCS6_B	—	—	—	—	Disabled	$V_{DD}X$	
PI7 [6]	PCS7_B	—	—	—	—	Disabled	$V_{DD}X$	

Table 2-2. Signal Properties Summary (continued)

Primary / GPIO Function [1]	Peripheral Function [1]	External Bus Function [1]	Debug Function [1]	Read on Reset	Internal Resistors		Power Supply	Description
					CTRL	Reset State		
PI8 [6]	—	—	—	—	—	Disabled	$V_{DD}X$	Port I I/O
PI9 [6]	—	—	—	—	—	Disabled	$V_{DD}X$	
PI10 [6]	—	—	—	—	—	Disabled	$V_{DD}X$	
PI11 [6]	—	—	—	—	—	Disabled	$V_{DD}X$	
PI12 [6]	—	—	—	—	—	Disabled	$V_{DD}X$	
PI13 [6]	—	—	—	—	—	Disabled	$V_{DD}X$	
PI14 [6]	—	—	—	—	—	Disabled	$V_{DD}X$	
PI15 [6]	—	—	—	—	—	Disabled	$V_{DD}X$	

1 The MAC7100 family maximum GPIO and peripheral configurations are listed in these columns. Some family members do not implement an external bus interface or the full complement of ports, ATD, CAN, DSPI and eSCI peripherals. Refer to Table 1-1 on page 1-3 for availability of ports and peripheral functions on various devices.

2 Pull-up/down is not available on these pins on mask set L49P devices.

3 The $\overline{AS}$ function is not available on mask set L49P devices.

4 The PD2 function is not available on mask set L49P devices (CLKOUT may be controlled as described in Section 18.7.3, “PD2 / CLKOUT Configuration”).

5 The optional debug status port is not available on mask set L49P devices. Refer to Section 26.4.1.6, “SSM Debug Status Port Control Register (DEBUGPORT),” on page 26-570 for more information.

6 Port I functions are available only on mask set L38Y devices in the 208-pin MAP BGA package (MAC7136).

Table 2-3. Power Supply, Voltage Regulator and Reference Summary

Pin Name	Description
$V_{DD}X$	Power and ground input for I/O drivers
$V_{SS}X$	
$V_{DD}R$	Voltage regulator power input
$V_{SS}R$	
$V_{DD}A$	Voltage regulator and ATD quiet power input
$V_{SS}A$	
$V_{DD}2.5$	Voltage regulator bypass for core logic
$V_{SS}2.5$	
$V_{DD}PLL$	Voltage regulator bypass for PLL
$V_{SS}PLL$	
V_{RH}	ATD high reference input
V_{RL}	ATD low reference input
TEST	Reserved for testing, must be tied to system ground.

2.4 Packaging Options

The MAC7100 Family is available in 208-pin ball grid array (MAP BGA) package, 144-pin low profile quad flat package (LQFP), 112-pin LQFP, and 100-pin LQFP options. The family of devices offer pin-compatible packaged devices to assist with system development and accommodate expansion of the application. Refer to Table 1-1 on page 1-3 for a comparison of the peripheral sets and package options for each device. Refer to the *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC) for detailed package diagrams and pin assignments.

Chapter 3
Voltage Regulator Module (VREG)

3.1 Overview

The voltage regulator (VREG) module provides the internal voltage for the on-chip logic, which enables devices in the MAC7100 family to be supplied with a single 5V power supply source.

The VREG module is a dual output voltage regulator providing two separate 2.5V (typical) supplies differing in the amount of current that can be sourced. The regulator input voltage range is from 3.3V up to 5V (typical).

In low power modes of operation the voltage regulator output can be reduced in order to further assist with power saving. The VREG can also be shutdown by connecting the $V_{DD}R$ pin to $V_{SS}R$ (which must also be connected to $V_{SS}X$).

Figure 3-1 and Figure 3-2 show functional block diagrams of the voltage regulator for the L49P and later mask sets, respectively. The regulator core REG consists of two parallel subblocks, REG1 and REG2, providing two independent output voltages.

3.2 Features

The voltage regulator includes these distinctive features:

- Two parallel, linear voltage regulators
- Bandgap reference
- Power On Reset (POR)
- Low Voltage Reset (LVR)
- Low Voltage Detect (LVD) with Low Voltage Interrupt (LVI)
- Autonomous Periodic Interrupt (API) [1]
- High Temperature Detect (HTD)[1]

1. This feature is not implemented on mask set L49P devices.

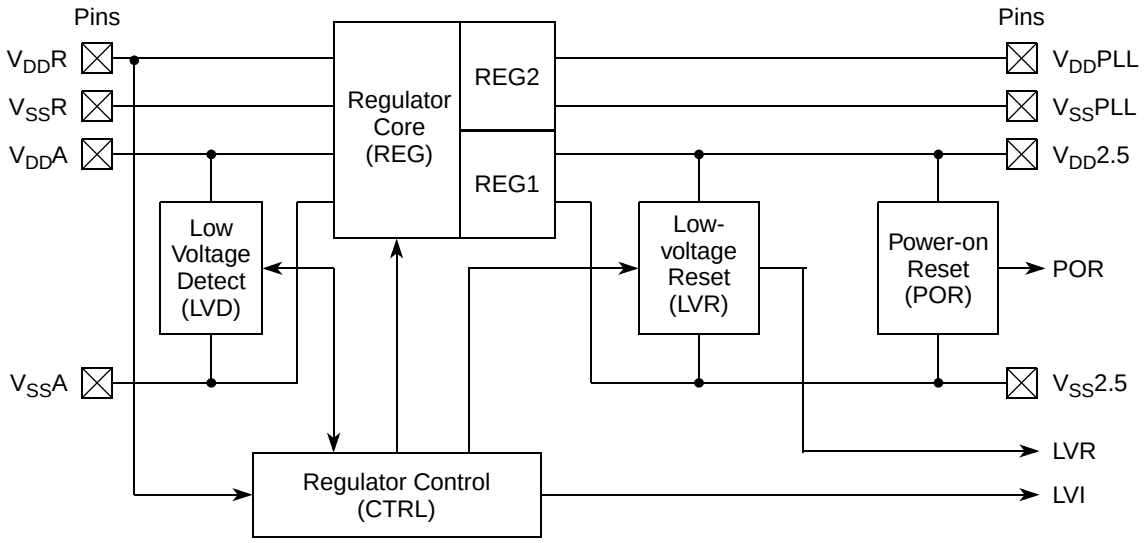


Figure 3-1. VREG Block Diagram for L49P Mask Devices

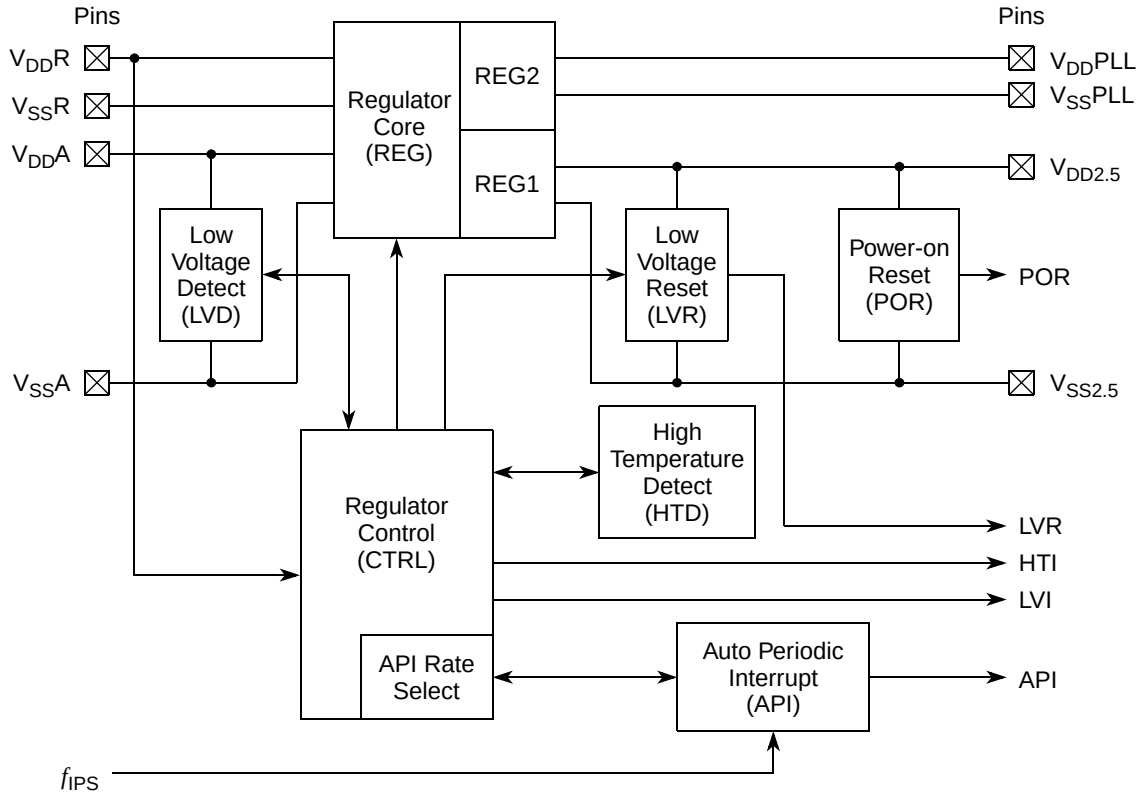


Figure 3-2. VREG Block Diagram for non-L49P Mask Devices

3.3 Modes of Operation

There are three modes in which the voltage regulator can operate:

- Full Performance Mode (FPM) (CPU is not in Stop Mode)

  The regulator is active, providing the nominal supply voltage of 2.5V with full current sourcing capability at both outputs. LVD (Low Voltage Detect), LVR (Low Voltage Reset), API [1] (Autonomous Periodic Interrupt), POR (Power-On Reset) and HTD[1] (High Temperature Detect) are available.

- Reduced Power Mode (RPM) (CPU is in Stop Mode)

  The purpose is to reduce power consumption of the device. The output voltage may degrade to a lower value than in Full Performance Mode, additionally the current sourcing capability is substantially reduced. Only the POR and API [1] are available; LVD, LVR, and HTD[1] are disabled.

- Shutdown Mode

  Controlled by the voltage level on the $V_{DD}R$ pin. This mode is characterized by minimum power consumption. The regulator outputs are in a high impedance state, only the POR and API [1] are available; LVD, LVR, and HTD[1] are disabled. This mode must be used to disable the chip internal regulator voltage regulator, i.e. to bypass the voltage regulator to use external supplies.

NOTE

Switching from FPM or RPM to shutdown mode and vice versa is not supported while the device is powered.

3.4 Signal Description

Since the voltage regulator provides the chip internal power supply voltages, most VREG signals are power supply signals connected to pads. Table 3-1 shows the signals of the voltage regulator associated with pins.

Table 3-1. VREG Signal Properties

Name	Function
$V_{DD}R$	Voltage Regulator power input (positive supply)
$V_{SS}R$	Voltage Regulator power input (ground)
$V_{DD}A$	Voltage Regulator quiet input (positive supply)
$V_{SS}A$	Voltage Regulator quiet input (ground)
$V_{DD}2.5$	Voltage Regulator primary output (positive supply)
$V_{SS}2.5$	Voltage Regulator primary output (ground)
$V_{DD}PLL$	Voltage Regulator secondary output (positive supply)
$V_{SS}PLL$	Voltage Regulator secondary output (ground)

Refer to Section 2.2, “Power Supply, Bypass and Reference,” on page 2-20 for the relationship of the VREG pins to the entire set of power supply pins utilized by MAC7100 Family devices.

1. This feature is not implemented on mask set L49P devices

3.4.1 $V_{DD}R$, $V_{SS}R$ — Regulator Power Input

Signal $V_{DD}R$ is the power input of the voltage regulator. All currents sourced into the regulator loads flow through this pin. A chip external decoupling capacitor (100nF...220nF, X7R ceramic) between $V_{DD}R$ and $V_{SS}R$ can smooth ripple on $V_{DD}R$. The regulator may be disabled (shutdown mode) by connecting $V_{DD}R$ to ground, in which case power must be supplied to $V_{DD}A$, $V_{DD}2.5$ and $V_{DD}PLL$ by external regulators.

3.4.2 $V_{DD}A$, $V_{SS}A$ — Regulator Reference Input

Signals $V_{DD}A/V_{SS}A$ are relatively quiet inputs used to supply the analog sections of the regulator, as well as the on-chip ATD module(s). Internal precision reference circuits are supplied from these signals. A chip external decoupling capacitor (100nF...220nF, X7R ceramic) between $V_{DD}A$ and $V_{SS}A$ can further improve the quality of this supply.

3.4.3 $V_{DD}2.5$, $V_{SS}2.5$ — Regulator Output 1 (Core Logic)

Signals $V_{DD}2.5/V_{SS}2.5$ are the primary outputs of the voltage regulator that provide the power supply for the core logic. These signals are connected to external pins to support decoupling capacitors (100nF...220nF, X7R ceramic) to improve the stability of the regulator output. In VREG shutdown mode an external voltage regulator must supply $V_{DD}2.5/V_{SS}2.5$.

3.4.4 $V_{DD}PLL$, $V_{SS}PLL$ — Regulator Output 2 (PLL)

Signals $V_{DD}PLL/V_{SS}PLL$ are the secondary outputs of the voltage regulator that provide the power supply for the PLL and Oscillator. These signals are connected to external pins to support decoupling capacitors (100nF...220nF, X7R ceramic) in order to improve the stability of the regulator output. In VREG shutdown mode an external voltage regulator must supply $V_{DD}PLL/V_{SS}PLL$.

3.5 Memory Map / Register Definition

Table 3-2 provides an overview of all registers used to control the VREG module.

Table 3-2. VREG Memory Map

VREG Offset	Register Description	Access
0x0000	VREG High Temperature Control Register (VREGHTCL) [1]	R/W
0x0001	VREG Control Register (VREGCTRL) [2]	R/W
0x0002	VREG Autonomous Periodic Interrupt Control Register (VREGAPICL) [3]	R/W
0x0003	VREG Autonomous Periodic Interrupt Trimming Register (VREGAPITR) [3]	R/W

1 On mask set L49P devices, this register is not implemented and VREGCTRL is at this offset

2 On mask set L49P devices, this register is at offset 0x0000.

3 On mask set L49P devices, this register is not implemented and the offset is reserved.

3.5.1 Register Descriptions

All registers utilize an 8-bit wide format. Even though MAC7100 Family devices implement a 32-bit peripheral interface, the implementation of the VREG module interface is only 16 bits wide, and 32-bit reads/writes to the VREG will produce unpredictable results. 8-bit accesses to any offset and 16-bit accesses to even offsets are supported.

3.5.1.1 VREG High Temperature Control Register (VREGHTCL)

The VREGHTCL register is used to configure the VREG temperature sense features.

	7	6	5	4	3	2	1	0
R	0	0	VSEL	VAE	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	VREG Base + 0x0000[1]							

1 For mask set L49P devices, this offset is used for the VREGCTRL register.

Figure 3-3. VREG High Temperature Control Register (VREGHTCL)

Table 3-3. VREGHTCL Field Descriptions

Bits	Name	Description
7–6	—	Reserved.
5	VSEL	Voltage access select. Selects the voltage that is available for monitoring via ATD A channel 0. Internal voltage monitoring must be enabled by the VAE bit. 0 Temperature proportional voltage V_{HT} can be monitored by ATD A channel 0. 1 Logic supply voltage $V_{DD}2.5$ can be monitored by ATD A channel 0.
4	VAE	Voltage access enable. If set the voltage selected by VSEL can be accessed internally via ATD A channel 0. 0 ATD A channel 0 connected to external pin. 1 ATD A channel 0 connected to voltage selected by VSEL.
3–0	—	Reserved.

3.5.1.2 VREG Control Register (VREGCTRL)

The VREGCTRL register is used to enable VREG low voltage monitoring features.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	LVDS	LVIE	LVIF
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	VREG Base + 0x0001 [1]							

1 For mask set L49P devices, this register is at offset 0x0000

Figure 3-4. VREG Control Register (VREGCTRL)

Table 3-4. VREGCTRL Field Descriptions

Bits	Name	Description
7–3	—	Reserved.
2	LVDS	Low voltage detect status. This read-only status bit reflects the input voltage. Writes have no effect. 0 Input voltage $V_{DD}A$ is above level V_{LVID} or RPM or Shutdown Mode. 1 Input voltage $V_{DD}A$ is below level V_{LVIA} and FPM.
1	LVIE	Low voltage interrupt enable. 0 Interrupt request is disabled. 1 Interrupt will be requested whenever LVIF is set.
0	LVIF	Low voltage interrupt flag. LVIF is set to 1 when LVDS status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (LVIE=1), LVIF causes an interrupt request. 0 No change in LVDS bit. 1 LVDS bit has changed. **Note:** LVIF is not cleared on entering the reduced power mode.

3.5.1.3 VREG Autonomous Periodic Interrupt Control Register (VREGAPICL)

NOTE

This register is not present on mask set L49P devices.

The VREGAPICL register is used to configure the autonomous periodic interrupt.

	7	6	5	4	3	2	1	0
R W	APICLK	APIR				APIFE	APIE	APIF
Reset	0	0	0	0	0	0	0	0
Reg Addr	VREG Base + 0x0002							

Figure 3-5. VREG Autonomous Periodic Interrupt Control Register (VREGAPICL)

Table 3-5. VREGCTRL Field Descriptions

Bits	Name	Description
7	APICLK	Autonomous periodic interrupt clock select. Selects the clock source for the API. Writable only if APIFE = 0. 0 Autonomous Periodic Interrupt clock used as source. 1 f_{IPS} used as source.
6–3	APIR[3:0]	Autonomous periodic interrupt rate. This field defines the timeout period of the API. See Table 3-6 below for selectable API periods. Writable only if APIFE = 0.
2	APIFE	Autonomous periodic interrupt feature enable. Enables the API feature and starts the API timer when set. 0 API is disabled. 1 API is enabled and timer starts running.

Table 3-5. VREGCTRL Field Descriptions (continued)

Bits	Name	Description
1	APIE	Autonomous periodic interrupt enable. 0 API interrupt request is disabled. 1 API interrupt will be requested whenever APIF is set.
0	APIF	Autonomous periodic interrupt flag. APIF is set when the API configured time has elapsed. This flag can only be cleared by writing a 1 to it. Clearing of the flag has precedence over setting. Writing a 0 has no effect. If APIE = 1, APIF = 1 generates an interrupt request. 0 API timeout has not yet occurred. 1 API timeout has occurred.

Table 3-6. VREG Selectable Autonomous Periodic Interrupt Periods

APIR[3:0]	Selected Period	
	APICLK = 0	APICLK = 1
0000	0.5 ms [1]	5 × f_{IPS} period
0001	1 ms [1]	10 × f_{IPS} period
0010	2 ms [1]	20 × f_{IPS} period
0011	2.5 ms [1]	25 × f_{IPS} period
0100	5 ms [1]	50 × f_{IPS} period
0101	7.5 ms [1]	75 × f_{IPS} period
0110	10 ms [1]	100 × f_{IPS} period
0111	20 ms [1]	200 × f_{IPS} period
1000	25 ms [1]	250 × f_{IPS} period
1001	50 ms [1]	500 × f_{IPS} period
1010	75 ms [1]	750 × f_{IPS} period
1011	100 ms [1]	1000 × f_{IPS} period
1100	200 ms [1]	2000 × f_{IPS} period
1101	250 ms [1]	2500 × f_{IPS} period
1110	500 ms [1]	5000 × f_{IPS} period
1111	750 ms [1]	7500 × f_{IPS} period

[1] When trimmed within specified accuracy. See electrical specification for details.

3.5.1.4 VREG Autonomous Periodic Interrupt Trimming Register (VREGAPITR)

NOTE

This register is not present on mask set L49P devices.

The VREGAPITR register allows trimming of the API timeout period.

	7	6	5	4	3	2	1	0
R	APITR						0	0
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	VREG Base + 0x0003							

Figure 3-6. VREG Autonomous Periodic Interrupt Trimming Register (VREGAPITR)

Table 3-7. VREGAPITR Field Descriptions

Bits	Name	Description
7-2	APITR[5:0]	API period trimming. See table below for trimming effects.
1–0	—	Reserved

Bit	Trimming Effect
APITR5	increases period by ~25.0%
APITR4	decreases period by ~12.5%
APITR3	decreases period by ~ 6.2%
APITR2	decreases period by ~ 3.1%
APITR1	decreases period by ~ 1.6%
APITR0	decreases period by ~ 0.8%

3.6 Functional Description

The VREG module is a dual voltage regulator as depicted in Figure 3-1 and Figure 3-2. The regulator functional elements are the regulator core (REG), a low voltage detect module (LVD), a control block (CTRL), a power-on reset module (POR), a low voltage reset module (LVR), and a high temperature sensor (HTD).

3.6.1 REG – Regulator Core

The VREG module has two parallel, independent regulation loops (REG1 and REG2) that differ only in the amount of current that can be delivered.

The regulator is a linear regulator with a bandgap reference when operated in full performance mode. It acts as a voltage clamp in reduced power mode. All load currents flow from the $V_{DD}R$ input to $V_{SS}2.5$ or $V_{SS}PLL$. The reference circuits are supplied by $V_{DD}A$ and $V_{SS}A$.

3.6.1.1 Full Performance Mode

In full performance mode the output voltage is compared with a reference voltage by an operational amplifier. The amplified input voltage difference drives the gate of an output transistor.

3.6.1.2 Reduced Power Mode

In reduced power mode the gate of the output transistor is connected directly to a reference voltage to reduce power consumption.

3.6.2 LVD – Low Voltage Detect

The LVD block shown in Figure 3-1 and Figure 3-2 generates the low voltage interrupt (LVI). LVD monitors the input voltage ($V_{DD}A$-$V_{SS}A$) and continuously updates the status flag LVDS. Interrupt flag LVIF is set whenever the status flag LVDS changes value. The LVD is available in Full Performance Mode and is inactive in reduced power mode or shutdown mode.

3.6.3 POR – Power-On Reset

This functional block monitors $V_{DD}2.5$. If $V_{DD}2.5$ is below V_{PORD}, POR is asserted, if $V_{DD}2.5$ is above V_{PORD}, the POR signal is driven low. Asserting POR forces the device into Reset, POR Deasserted will trigger the power-on sequence.

3.6.4 LVR – Low Voltage Reset

The LVR block monitors the primary output voltage of the regulator ($V_{DD}2.5$). If it drops below the assertion level (V_{LVRA}) a low voltage reset is asserted. When $V_{DD}2.5$ rises above the deassertion level (V_{LVRD}) the low voltage reset signal is negated. This function is available only in full performance mode.

3.6.5 CTRL – Regulator Control

This block contains the registers of the voltage regulator and further digital functionality needed to control the operating modes. CTRL also represents the interface to the digital core logic.

3.6.6 API – Autonomous Periodic Interrupt

The API block can generate periodic interrupts independent of the clock source of the device. The timer is enabled when the APIFE bit is set.

The API timer is either clocked by an adjustable internal RC oscillator or f_{IPS}. Timer operation will freeze when the device clock source is selected and f_{IPS} is turned off. See Chapter 4, “System Clocks Module (OSC and CRG),” for details. The clock source can be selected with APICLK bit. APICLK can only be written when APIFE is not set.

The period of the API timer is selected using the APIR[3:0] bits, which select the amount of time after which an interrupt should be generated. The first length of the first period after writing the APIR[3:0] value might differ if APIR[3:0] is changed while APIFE = 1. As soon as APIFE is set the timer starts running. When the configured time has elapsed, the APIF flag is set and an interrupt is signaled if the interrupt enable bit, APIE, is set. The timer is reset and continues counting after APIF is set.

The API trimming bits, APITR[5:0], may be used to calibrate the interrupt period. See Table 3-7 for the trimming effect of APITR.

NOTE

This feature is not present on mask set L49P devices.

The first period after enabling the counter by APIFE might be reduced.

The API internal RC-oscillator clock is not available if the VREG is in Shutdown Mode.

3.6.7 Resets

This section describes how the voltage regulator controls the reset of the device. The reset values of registers and signals are provided in Section 3.5, "Memory Map / Register Definition." Possible reset sources are listed in Table 3-8.

Table 3-8. VREG Reset Sources

Reset Source	Local Enable
Power-on Reset	always active
Low-voltage Reset	active only in full performance mode

3.6.7.1 Power-On Reset (POR)

During chip power-up, the digital core may not operate correctly if the supply voltage $V_{DD}2.5$ is below the POR deassertion level (V_{PORD}). Therefore the POR signal holds all other device modules in reset until $V_{DD}2.5$ exceeds V_{PORD}. The device begins the start-up sequence after POR is negated. The power-on reset is active in all VREG operating modes.

3.6.7.2 Low-Voltage Reset (LVR)

Refer to Section 3.6.4, "LVR – Low Voltage Reset," for more information.

3.6.8 Interrupts

The interrupt vectors requested by the VREG module are listed in Table 3-9. Refer to Chapter 6, "Exceptions," for specific vector addresses and interrupt priorities.

Table 3-9. VREG Interrupt Vectors

Interrupt Source	Local Enable
Low Voltage Interrupt (LVI)	LVIE=1 available only in full performance mode
Autonomous Periodic Interrupt (API)	APIE=1

3.6.8.1 LVI – Low Voltage Interrupt

In FPM the VREG module monitors the input voltage $V_{DD}A$. Whenever $V_{DD}A$ drops below level V_{LVIA} the status bit LVDS is set to 1. Vice versa, LVDS is reset to 0 when $V_{DD}A$ rises above level V_{LVID}. An

interrupt, indicated by flag LVIF=1, is triggered by any change of the status bit LVDS if interrupt enable bit LVIE=1.

NOTE

LVIF is not cleared on entering reduced power mode.

3.6.8.2 API – Autonomous Periodic Interrupt

When the configured timeout period of the API has elapsed, the APIF bit is set and an interrupt is signaled if the APIE bit is set.

NOTE

This feature is not present on mask set L49P devices.

3.7 Initialization / Application Information

3.7.1 Circuit Board Layout

The PCB must be carefully laid out to ensure proper operation of the voltage regulator. Table 3-10 and Table 3-11 below provides a description of the components used on the device power supplies and outlines recommended values to be used. Figure 3-7, Figure 3-8, Figure 3-9, Figure 3-10 and Figure 3-11 below provide recommended board layouts for the different package options for the MAC7100 family.

The following rules must be observed:

- The central point of the ground star should be the $V_{SS}R$ pin.
- Central power input should be the $V_{DD}A/V_{SS}A$ pins.
- Every supply pair must be decoupled by a ceramic capacitor connected as near as possible to the corresponding pins.
- Use low ohmic low inductance connections between $V_{SS}2.5$ and $V_{SS}R$.
- $V_{SS}PLL$ must be directly connected to $V_{SS}R$.
- Keep traces for $V_{SS}PLL$, EXTAL and XTAL as short as possible, and occupied board area for oscillator load capacitors, filter capacitors and quartz resonator as small as possible.
- Do not place other signals or supplies underneath the areas occupied by the oscillator load capacitors, quartz resonator and the connection area to the device.

Table 3-10. VREG Recommended Circuit Board Component Values – LQFP

Component	Purpose	Type	Value
C1	V_{DD}2.5 filter cap	ceramic X7R	100 .. 220nF
C2	V_{DD}2.5 filter cap	ceramic X7R	100 .. 220nF
C3	V_{DD}A filter cap	ceramic X7R	≥ 100nF
C4	V_{DD}R filter cap	X7R/tantalum	≥ 100nF
C5	V_{DD}PLL filter cap	ceramic X7R	100 .. 220nF
C6	V_{DD}X filter cap	X7R/tantalum	≥ 100nF
C7	OSC load cap	See *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC)	
C8	OSC load cap		
C9 / C_S	PLL loop filter cap	See *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC)	
C10 / C_P	PLL loop filter cap		
C11	V_{DD}X filter cap	X7R/tantalum	≥ 100nF
C12	V_{DD}X filter cap	X7R/tantalum	≥ 100nF
C13	V_{DD}X filter cap	X7R/tantalum	≥ 100nF
C14	V_{DD}X filter cap	X7R/tantalum	≥ 100nF
C15	V_{DD}X filter cap	X7R/tantalum	≥ 100nF
R1	PLL loop filter res	See *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC)	
R2 / R_B	OSC res		
R3 / R_S	OSC res		
Q1	Quartz		

Table 3-11. VREG Recommended Circuit Board Component Values – 208 MAP BGA

Component	Purpose	Type	Value
C1	V_{DD}A filter cap	ceramic X7R	100 .. 220nF
C2	V_{DD}2.5 filter cap	ceramic X7R	220nF
C3	V_{DD}X filter cap	ceramic X7R	100 .. 220nF
C4	V_{DD}X filter cap	ceramic X7R	100 .. 220nF
C5	V_{DD}R filter cap	ceramic X7R	220nF
C6	V_{DD}2.5 filter cap	ceramic X7R	220nF
C7	V_{DD}X filter cap	ceramic X7R	100 .. 220nF
C8	V_{DD}PLL filter cap	ceramic X7R	220nF
C9	OSC load cap	Contact crystal manufacturer	
C10	OSC load cap		
C11 / C_S	PLL loop filter cap	See *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC)	
C12 / C_P	PLL loop filter cap		
R1	PLL loop filter res		
Q1	Quartz	Resonator	—

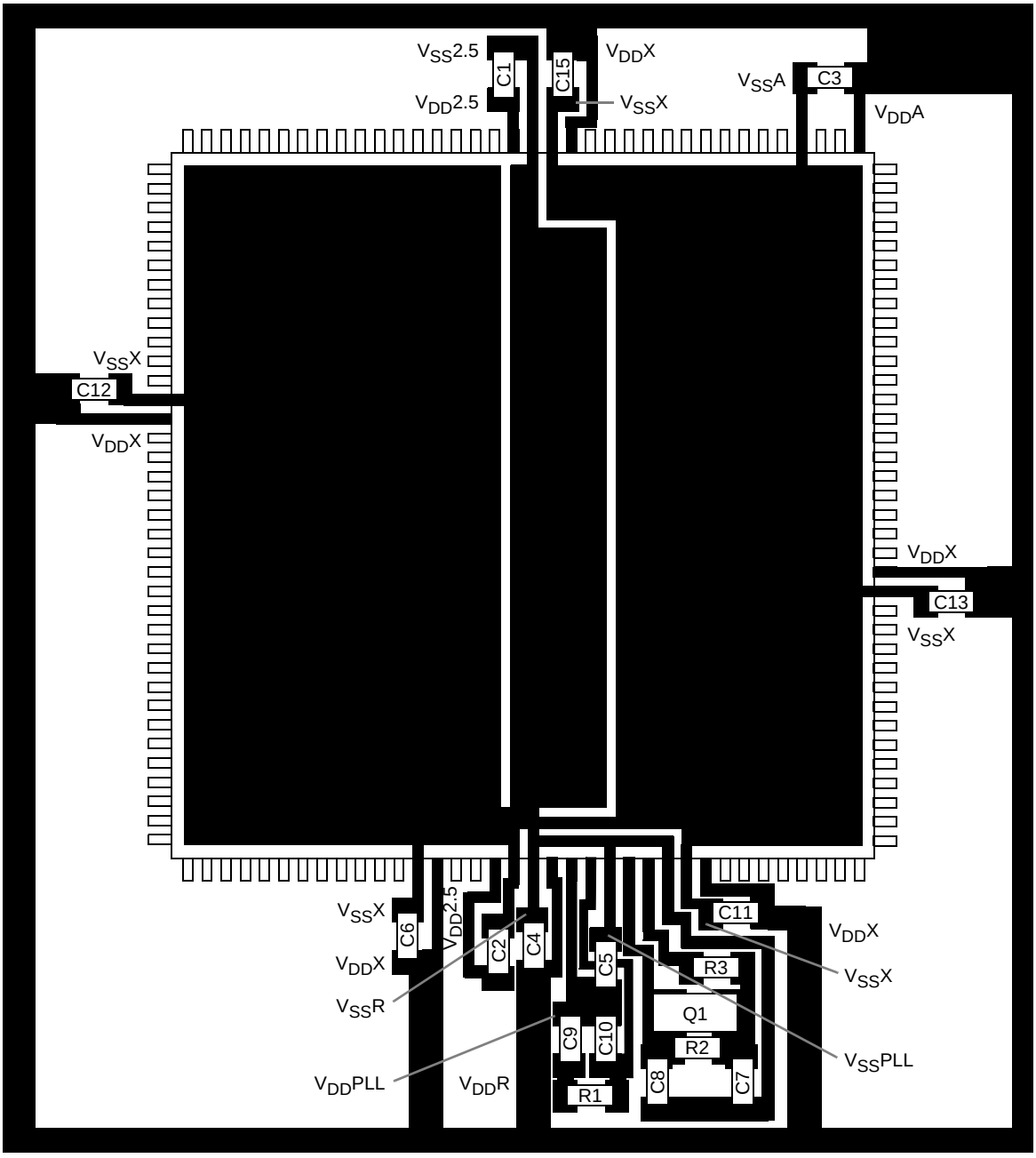


Figure 3-7. Recommended PCB Layout for 144 QFP

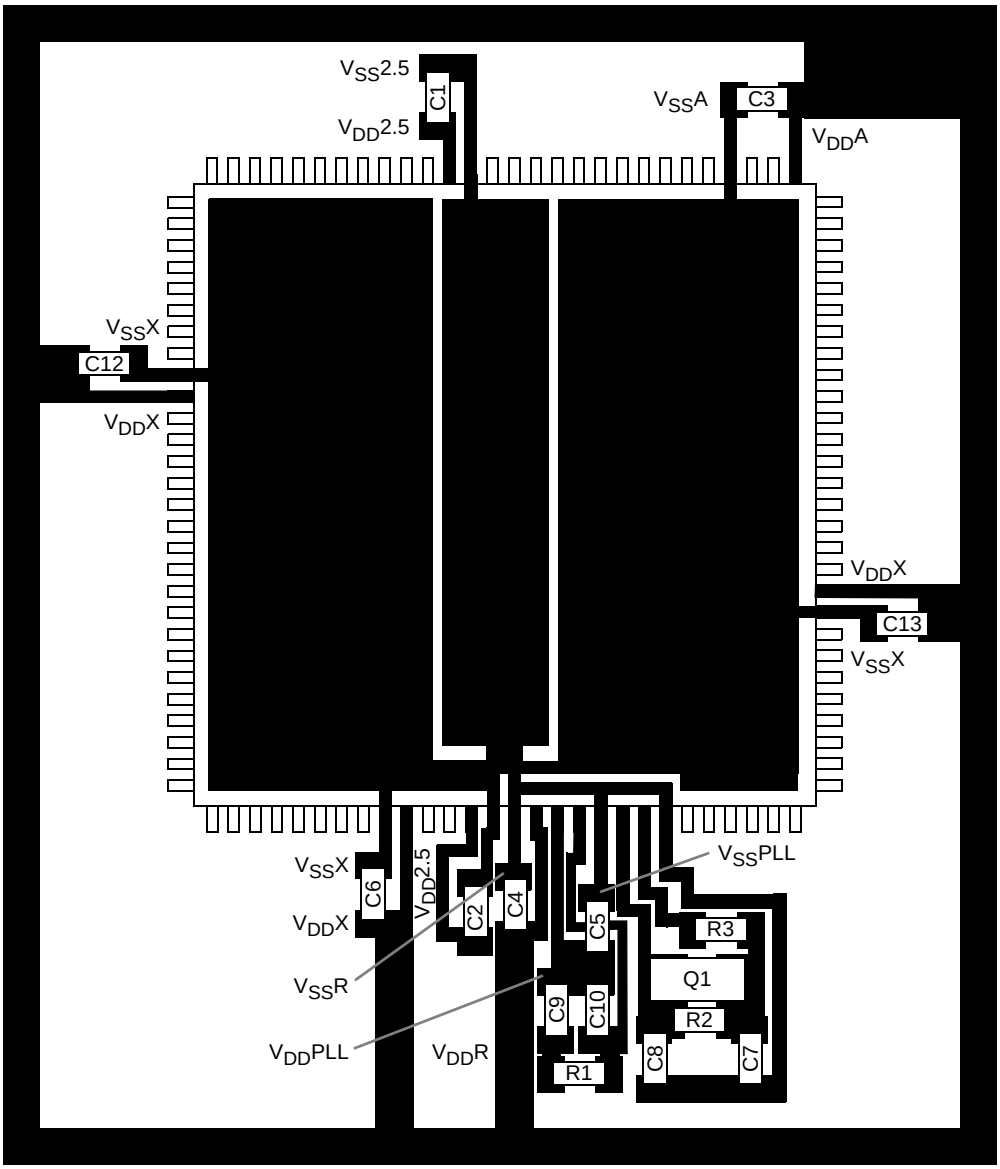


Figure 3-8. Recommended PCB Layout for 112 LQFP

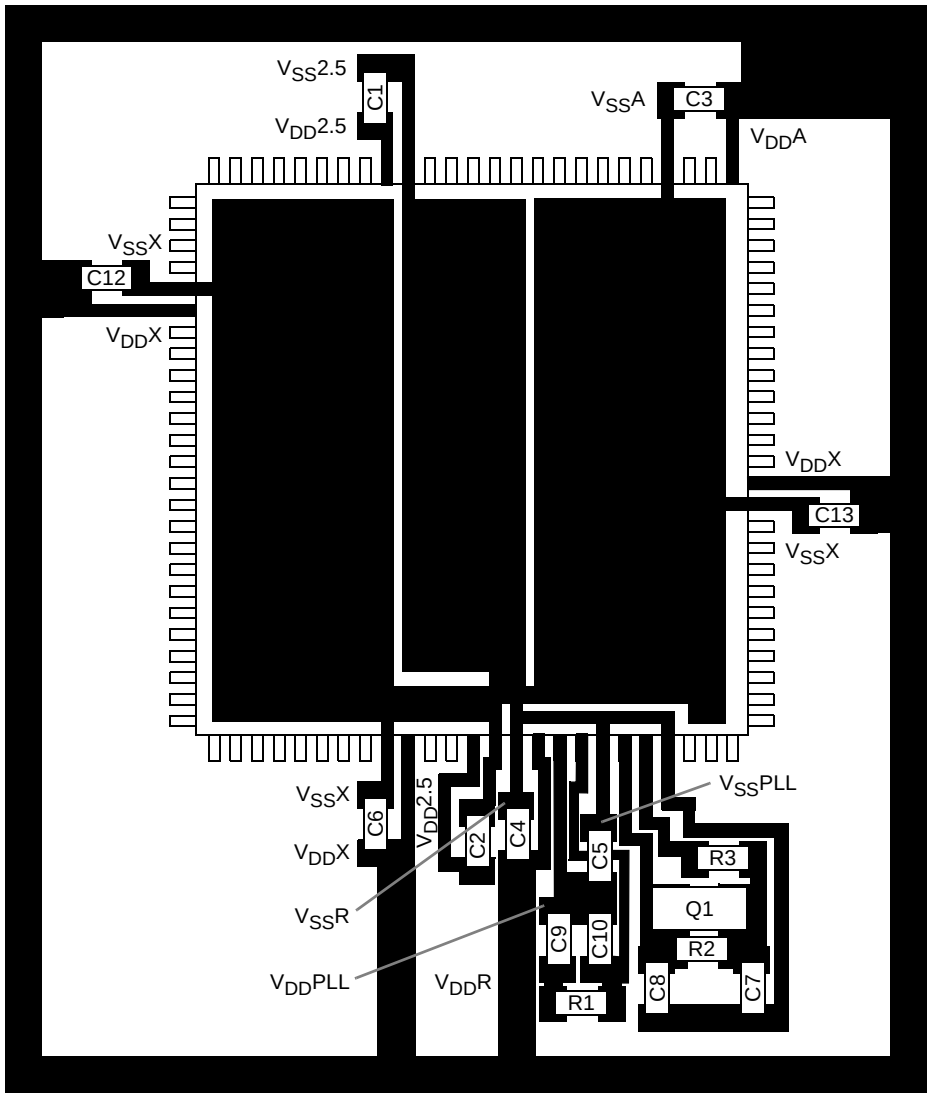


Figure 3-9. Recommended PCB Layout for 100 LQFP

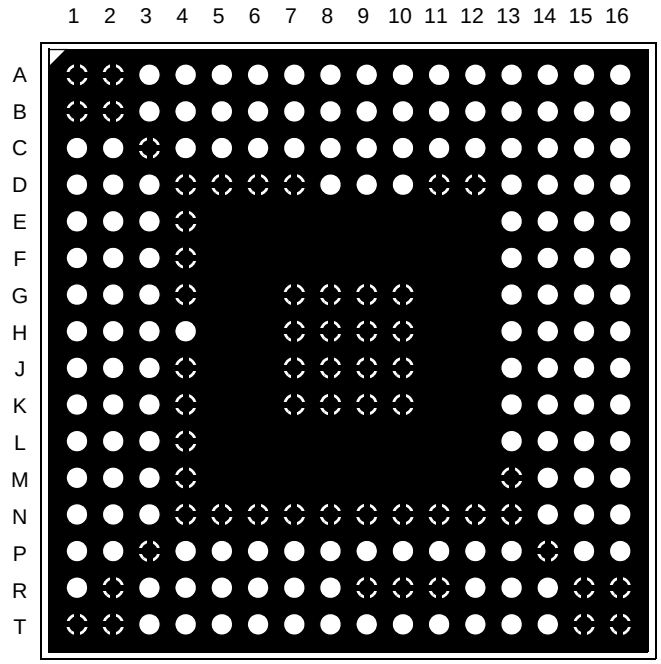


Figure 3-10. Recommended PCB Layout for 208 MAP BGA – Ground Plane

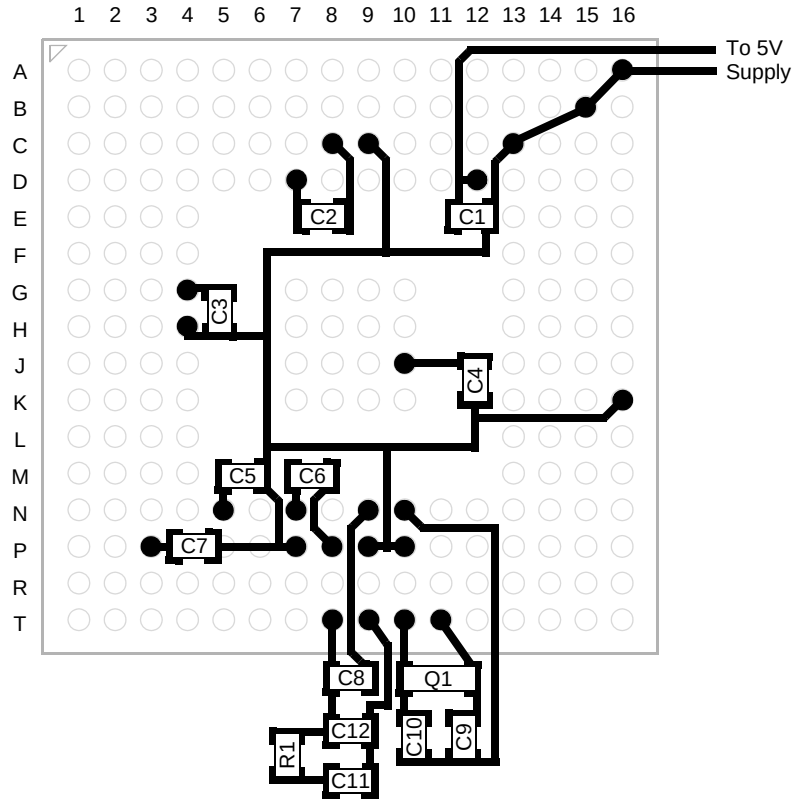


Figure 3-11. Recommended PCB Layout for 208 MAP BGA – Decoupling

Chapter 4
System Clocks Module (OSC and CRG)

4.1 Overview

The Clock and Reset Generator (CRG) module, which utilizes a Pierce oscillator (OSC), provides the internal clock signals for the core and all peripheral modules. Figure 4-1 shows the clock connections to all modules.

The system clock can be supplied to the device in several ways, enabling a range of system operating frequencies to be supported:

- From the oscillator, with a frequency set by an external crystal reference
- From the on-chip phase-locked loop, using the oscillator clock as a reference
- From the PLL in self-clocking mode

The clock generated by the oscillator or the phase-locked loop (PLL) provides the main system clock frequency, f_{SYS}. As shown in Figure 4-1, the system clock is used throughout the device to drive the core and the memories. The IPS peripherals and read access to the program Flash use a clock equal to half the system clock frequency, f_{IPS}.

NOTE

When using an oscillator-generated or a PLL synthesized clock, a clock reference of no more than 16 MHz must be applied to the external pins. This is not necessary when the PLL is in self-clocking mode.

The program Flash memory is supplied by both f_{SYS} and f_{IPS}. f_{SYS} is used for access to the Flash controller, but the interleaved Flash arrays operate at half the system frequency to allow enough time to access the array. Interleaved arrays enable the memory controller to achieve close to single-cycle access times at the full system frequency. Refer to Chapter 15, “Common Flash Module (CFM),” for more information.

The CAN modules may be configured to utilize either the peripheral clock (f_{IPS}) or the Oscillator clock (OSCCLK). This allows the user to select the CAN clock based on the required jitter performance. Refer to Chapter 23, “Controller Area Network Module (FlexCAN),” for more details.

The Periodic Interrupt Timer (PIT) and Software Watchdog Timer (SWT) can be configured to run from the Oscillator or the PLL generated clock. This allows these functions to continue to run during low power operating modes if required. Refer to Chapter 25, “Periodic Interrupt Timer Module (PIT),” and Chapter 11, “Miscellaneous Control Module (MCM),” for more information.

The frequency generated by the PLL, PLLCLK, is determined by the values of two of the CRG module control registers, REFDIV and SYNR. The frequency is calculated using the following equation.

$$PLLCLK = 2 \times OSCCLK \times \frac{SYN[5:0] + 1}{REFDV[3:0] + 1} \qquad \textit{Eqn. 4-1}$$

When the system frequency, f_{SYS}, is derived from the PLL, the value is calculated based on this equation:

$$f_{SYS} = PLLCLK \qquad \textit{Eqn. 4-2}$$

When the system frequency, f_{SYS}, is derived from the oscillator, the value is calculated based on this equation:

$$f_{SYS} = \text{OSCCLK} \qquad \textit{Eqn. 4-3}$$

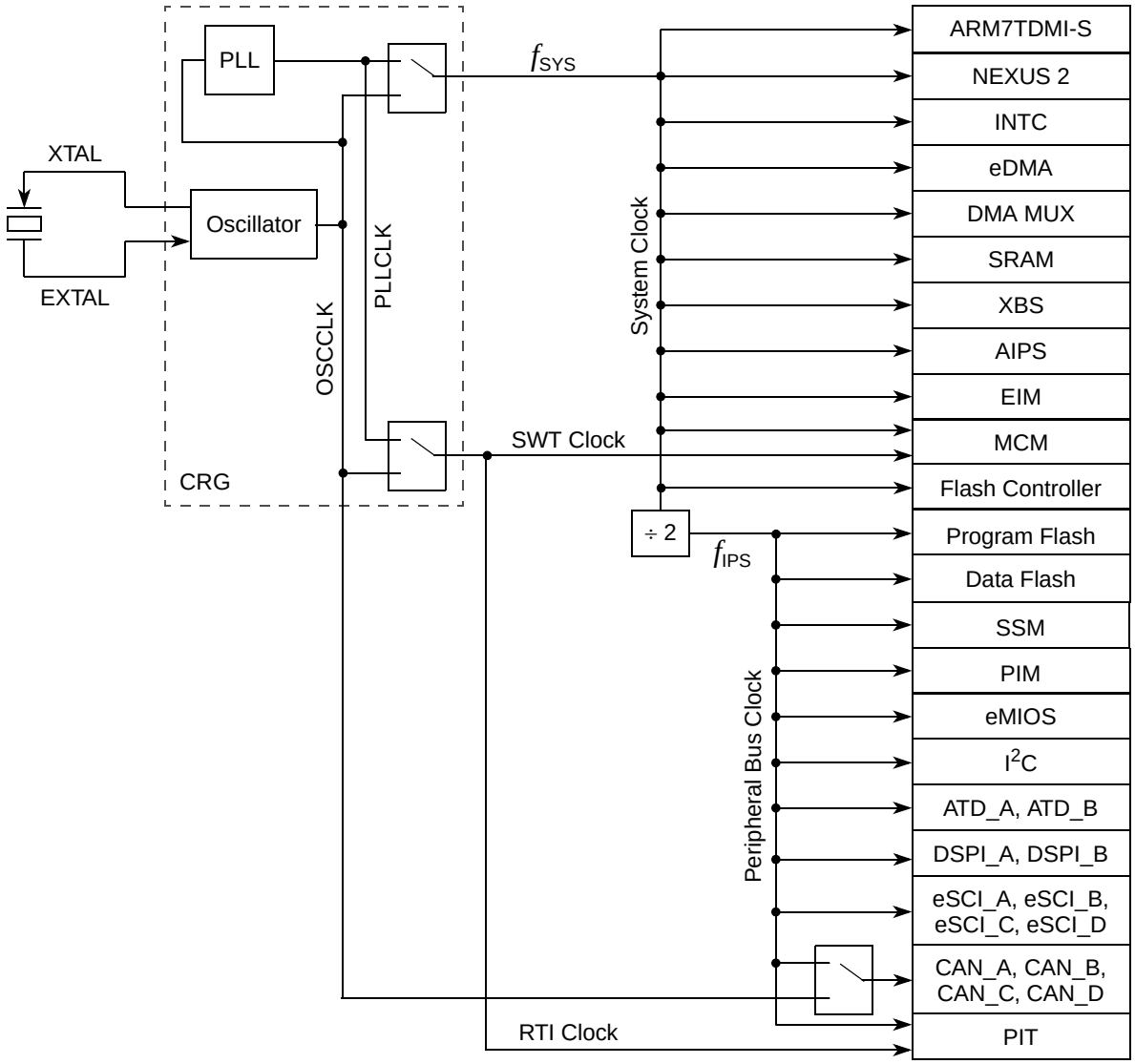


Figure 4-1. MAC7100 Family Clock Connections

Note that it is possible to configure the PLL to generate a system frequency higher than that supported by the design of the device. It is the responsibility for the user to insure that the device is operated within its specified limits at all times.

In order to ensure the presence of the clock, the device includes an on-chip hardware clock monitor connected to OSCCLK. The clock monitor can be configured to enable the PLL self-clocking mode or to generate a system reset if it is allowed to time out when no OSCCLK is present.

In addition to the clock monitor, the device also provides a clock quality checker which performs a more accurate check of the clock. The clock quality checker counts a predetermined number of clock edges

within a defined time window to insure that the clock is running. The checker can be invoked following specific events such as on wake-up or a clock monitor failure.

4.2 On-Chip Oscillator (OSC) Module

4.2.1 OSC Overview

MAC7100 Family devices feature an internal Pierce oscillator that can operate in standard Pierce and low power amplitude loop controlled modes with a minimum number of external components. The oscillator is designed for optimal startup margin with typical crystal oscillators. Selection of the oscillator type is performed at reset based on the signal level on the CLKOUT / $\overline{\text{XCLKS}}$ pin. After start-up or a power on reset, the quality of the oscillation is checked by the clock quality checker before the oscillator is connected to the internal system clocks. In the event that a stable oscillator output is not detected within a predefined time, the device will be switched to the internal self-clock mode. Figure 4-2 shows a block diagram of the Oscillator.

The oscillator power is supplied from a separate 2.5 V supply voltage (V_{DD}PLL/V_{SS}PLL) generated by the voltage regulator in order to minimize noise. The oscillator continues to run in doze and pseudo-stop low-power modes, but is halted in full-stop mode.

A square wave input can be supplied to the device through the oscillator by connecting an external clock source to the EXTAL pin with the oscillator operating in standard Pierce mode.

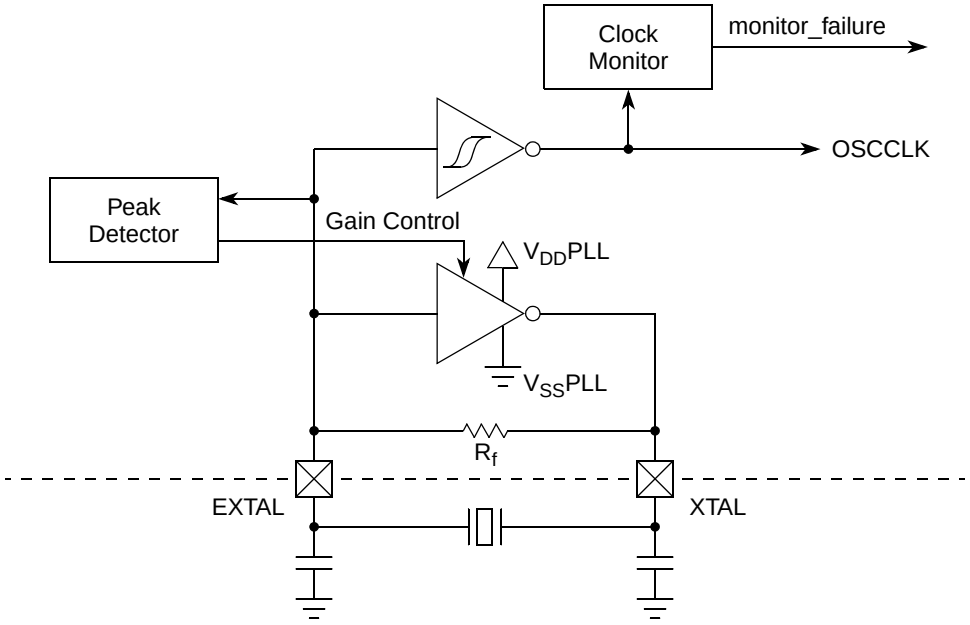


Figure 4-2. Oscillator Block Diagram

4.2.2 OSC Features

The OSC contains circuitry to dynamically control current gain in the output amplitude. This ensures a signal with low harmonic distortion, low power and good noise immunity.

- High noise immunity due to input hysteresis.
- Low RF emissions with peak to peak swing limited dynamically.
- Transconductance (gm) sized for optimum start-up margin for typical oscillators.
- Dynamic gain control eliminates the need for external current limiting resistor.
- Integrated resistor eliminates the need for external bias resistor.
- Low power consumption:
  - — Operates from 2.5 V (nominal) supply
  - — Amplitude control limits power consumption
- Clock monitor

4.2.3 OSC Modes of Operation

Two modes of operation are available for the OSC module:

- Loop controlled Pierce oscillator.
- External square wave mode using full swing Pierce without internal feedback resistor.

Device operating modes may also affect the oscillator module: it is placed in a static state when the device enters stop mode; the oscillator operates normally in run, doze and pseudo-stop modes.

4.2.4 OSC Signal Description

4.2.4.1 V_{DD}PLL, V_{SS}PLL

V_{DD}PLL, V_{SS}PLL are the power supply and ground input pins for the Pierce oscillator. The V_{DD}PLL and V_{SS}PLL pins allow the supply voltage to the PLL to use external filter capacitors independent of the main I/O supply (V_{DD}X and V_{SS}X).

4.2.4.2 EXTAL, XTAL

These pins provide the interface for either a crystal or a CMOS compatible clock which acts as a time-base reference for the oscillator. EXTAL is the external clock input or the input to the crystal oscillator amplifier. XTAL is the output of the crystal oscillator amplifier. In full-stop mode (PSTP=0) the EXTAL pin is pulled down by an internal resistor of 200k Ω (typical). Figure 4-3, Figure 4-4 and Figure 4-5 illustrate the configuration of EXTAL and XTAL in each mode of operation.

NOTE

Freescale recommends an evaluation of the application board layout and chosen resonator or crystal by the resonator or crystal supplier. Refer to Section 3.7.1, “Circuit Board Layout,” on page 3-39 for more details.

4.2.4.3 CLKOUT / $\overline{\text{XCLKS}}$

At reset the signal level on the CLKOUT / $\overline{\text{XCLKS}}$ pin is read to determine the mode of the Pierce oscillator. If the signal level is high the oscillator operates in the loop controlled mode, if the signal level is low the oscillator operates in the full swing or external clock mode.

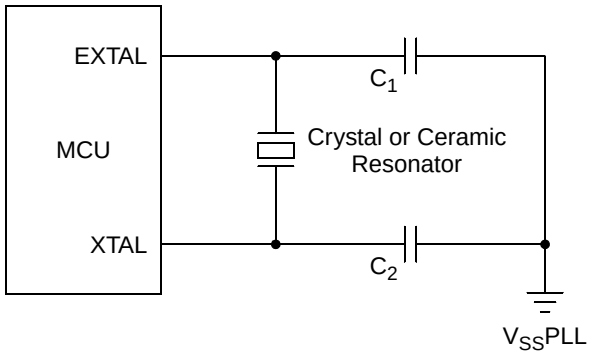
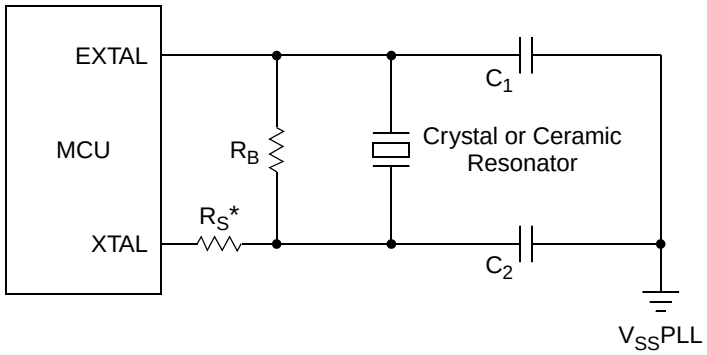


Figure 4-3. Loop Controlled Pierce Oscillator Connections

NOTE

Loop controlled circuit is not suited for overtone resonators and crystals.



* R_S can be zero (shorted) when used with higher frequency crystals. Refer to manufacturer's data.

Figure 4-4. Full Swing Pierce Oscillator Connections

NOTE

Full swing Pierce circuit is not suited for overtone resonators and crystals without a careful component selection

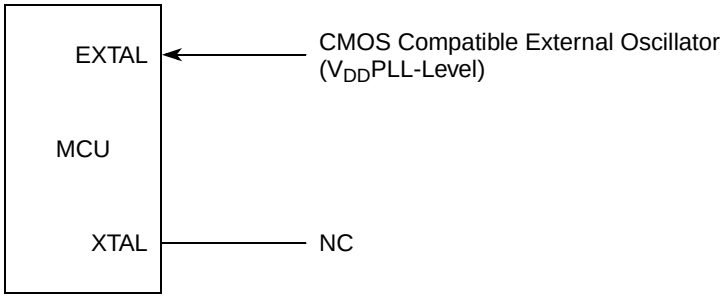


Figure 4-5. External Clock Connections

4.2.5 OSC Functional Description

The OSC module has control circuitry to maintain the crystal oscillator circuit voltage level to an optimal level as determined by the amount of hysteresis and the maximum oscillation range.

4.2.5.1 Gain control

A closed-loop control system is utilized to modulate the amplifier to keep the output waveform sinusoidal and to limit the oscillation amplitude. The output peak-to-peak voltage is kept above twice the maximum hysteresis level of the input buffer. Refer to *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC) for details.

4.2.5.2 Clock Monitor

The clock monitor is based on an internal RC time delay circuit such that it can operate without any device clocks. If no OSCCLK edges are detected within the RC time delay, the clock monitor indicates failure by enabling self-clock mode or generates a system reset depending on the state of CRGINT[SCME] bit (see Section 4.3.5.4, “CRG Interrupt Enable Register (CRGINT)”). If the clock monitor is disabled or the presence of clocks is detected no failure is indicated. The clock monitor function is enabled or disabled by the PLLCTL[CME] bit (see Section 4.3.5.6, “CRG PLL Control Register (PLLCTL)”).

4.3 Clock and Reset Generator (CRG) Module

4.3.1 CRG Overview

The CRG module uses the output of the OSC module to provide the clocks for the device, and controls reset operations and low power operating modes. The CRG module also provides information in a flag register to help identify the source of a reset event.

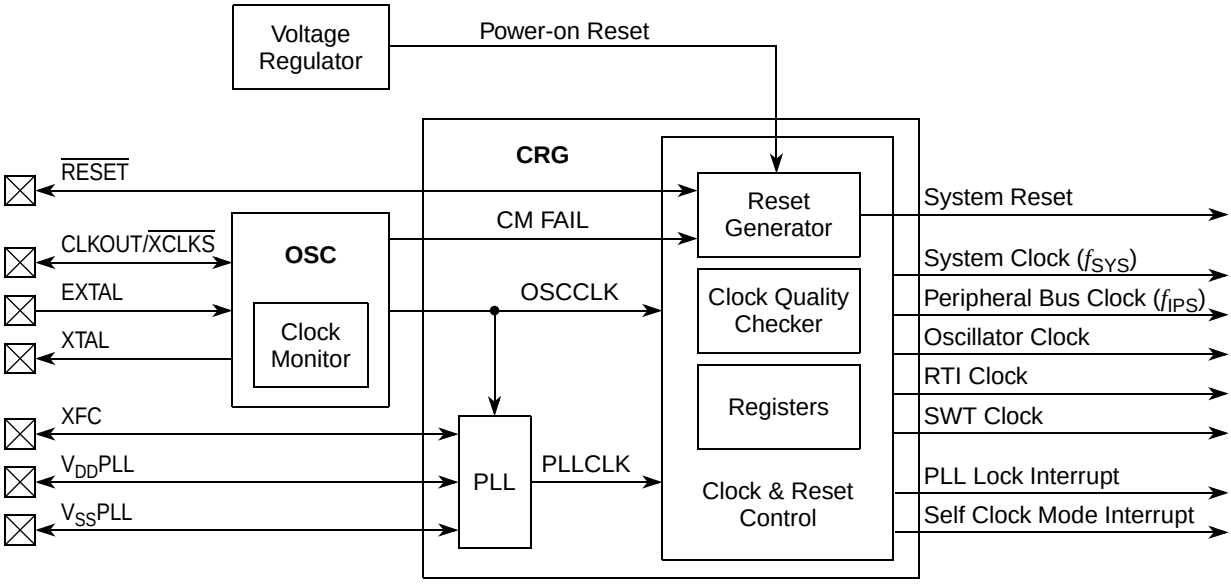


Figure 4-6. Clock and Reset Generator Block Diagram

The CRG is used to control entry into the low power operating modes of the device, as well as the features of the device which continue to operate during low power modes. Doze and stop mode are entered by writing to the Stop/Doze Control (SDMCTL) Register, while configuration of the Clock Select (CLKSEL) Register controls whether the PLL, Real Time Interrupt (RTI) and Software Watchdog Timer (SWT) clocks are disabled. By setting the [PSTP] bit, when stop mode is requested, the device will enter pseudo-stop mode, where the oscillator continues to run, allowing the RTI and SWT to continue operating.

The module provides a Clock Monitor which detects the presence of the oscillator clock. If the oscillator clock is not present within a defined time, determined by the Clock Monitor timeout period, the module will either generate a Reset, or initiate the PLLs self-clock mode. In self-clock mode the PLL will generate its own clock based on the minimum VCO frequency. This can be used to clock the device in order to continue some basic operation in the absence of an external clock.

The Clock Quality Checker (CQC) is included in the CRG and provides a more accurate check of the oscillator output clock. The CQC operates following events such as power on reset or wake-up from stop mode, and counts the number of clocks over a defined time window. Failure of the Clock Quality Checker can be used to initiate self-clocking mode or a clock monitor reset event.

4.3.2 CRG Features

The main features of this module are:

- Phase Locked Loop (PLL) frequency multiplier
  - — Reference divider
  - — Automatic bandwidth control mode for low-jitter operation
  - — Automatic frequency lock detector
  - — CPU interrupt on entry or exit from locked condition
  - — Self-clock mode in absence of reference clock
- System Clock Generator
  - — Clock quality check
  - — User selectable fast wake-up from stop in self-clock mode for power saving and immediate program execution [1]
  - — Clock switch for either oscillator or PLL based system clocks
  - — User selectable disabling of clocks during doze mode for reduced power consumption
- System reset generation from the following possible sources:
  - — Power on reset
  - — Low-voltage reset
  - — Software watchdog timer reset
  - — Clock monitor failure (with self-clock mode disabled) reset
  - — External pin reset

1. On mask set L49P devices, this feature is not implemented

4.3.3 CRG Modes of Operation

This subsection briefly describes all operating modes supported by the CRG.

- Run Mode — All functional parts of the CRG are running during normal run mode.
- Doze Mode — This mode allows to disable the system and peripheral clocks depending on the configuration of the individual bits in the CLKSEL register.
- Stop Mode — Depending on the setting of the PSTP bit, stop mode can be differentiated between full stop mode (PSTP=0) and pseudo-stop mode (PSTP=1).
  - — Full-Stop Mode — The oscillator is disabled and thus all system and peripheral clocks are stopped. The SWT and the RTI remain frozen.
  - — Pseudo-Stop Mode — The oscillator continues to run and most of the system and peripheral clocks are stopped. If the respective enable bits are set, the SWT and RTI will continue to run, else they remain frozen.
- Self-Clock Mode — Self-clock mode will be entered if the PLLCTL clock monitor enable (CME) and self-clock mode enable (SCME) bits are both set and the clock monitor in the oscillator block detects a loss of clock. As soon as self-clock mode is entered, the CRG starts to perform a clock quality check. Self-clock mode remains active until the clock quality check indicates that the required quality of the incoming clock signal is met (frequency and amplitude). Self-clock mode should be used for safety purposes only. It provides reduced functionality to the device in case where a loss of clock is causing severe system conditions.

4.3.4 CRG Signal Description

Table 4-1. CRG Signal Properties

Name	I/O Type	Function
V_{DD}PLL	Input	Operating Voltage
V_{SS}PLL	Input	Ground
XFC	Output/Input	External Loop Filter
$\overline{\text{RESET}}$	Output/Input	Reset Input/Output
CLKOUT / $\overline{\text{XCLKS}}$	Output/Input	Clock Output / OSC Select Input

4.3.4.1 XFC

A passive external loop filter must be placed on the XFC pin. The filter is a second-order, low-pass filter to eliminate the VCO input ripple. The value of the external filter network and the reference frequency determines the speed of the corrections and the stability of the PLL. Refer to *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC) for calculation of PLL loop filter (XFC) components. If PLL usage is not required, the XFC pin must be tied to V_{DD}PLL.

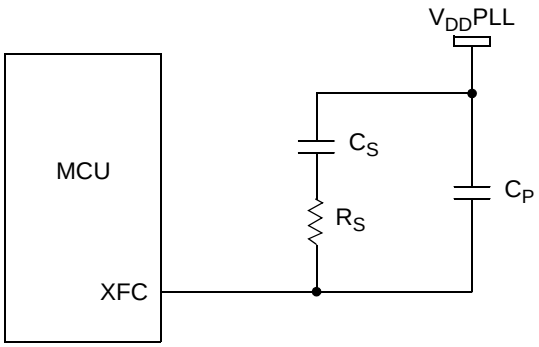


Figure 4-7. PLL Loop Filter Connections

4.3.4.2 $\overline{\text{RESET}}$

$\overline{\text{RESET}}$ is an active low bidirectional reset pin. As an input, it initializes the device asynchronously to a known start-up state. As an open-drain output, it indicates that a system reset (internal to device) has been triggered.

4.3.4.3 CLKOUT / $\overline{\text{XCLKS}}$

This pin is used to output the device system clock (f_{SYS}) for use with the external bus or to drive external synchronous devices. At reset the signal level on the pin is read to determine the mode of the Pierce oscillator. If the signal level is high the oscillator operates in the loop-controlled mode, if the signal level is low the oscillator operates in the full swing or external clock mode. Refer to Section 18.7.3, "PD2 / CLKOUT Configuration," for more information on utilizing this signal in various modes.

4.3.5 CRG Memory Map / Register Definition

Figure 4-2 gives an overview of all CRG registers.

Table 4-2. CRG Memory Map

CRG Offset [1]	Register Description	Access
0x0000	CRG Synthesizer Register (SYNR)	R/W
0x0001	CRG Reference Divider Register (REFDV)	R/W
0x0003	CRG Flags Register (CRGFLG)	R/W
0x0004	CRG Interrupt Enable Register (CRGINT)	R/W
0x0005	CRG Clock Select Register (CLKSEL)	R/W
0x0006	CRG PLL Control Register (PLLCTL)	R/W
0x0007	CRG Stop/Doze Control Register (SDMCTL)	R/W
0x0008	CRG BDM Control Register (BDMCTL)	R/W

[1] Register Address = CRG base address + offset, where the base address is defined in Chapter 8, "Device Memory Map."

4.3.5.1 CRG Synthesizer Register (SYNR)

The SYNR controls the multiplication factor of the PLL. The bits in this register can be written anytime except when the PLLSEL bit is set in the CLKSEL register.

If the PLL is on, the count in the loop divider register (SYNR) effectively multiplies the PLL clock (PLLCLK) from the reference frequency by 2 × (SYN + 1). PLLCLK will not be below the minimum VCO frequency (f_{SCM}).

$$PLLCLK = 2 \times OSCCLK \times \frac{SYN[5:0] + 1}{REFDV[3:0] + 1} \qquad \textit{Eqn. 4-4}$$

NOTE

If CLKSEL[PLLSEL] is set, f_{SYS} = PLLCLK. f_{SYS} must not exceed the maximum operating system frequency specified in *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC).

	7	6	5	4	3	2	1	0
R	0	0	SYN					
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	CRG Base + 0x0000							

Figure 4-8. CRG Synthesizer Register (SYNR)

NOTE

Writing to this register initializes the lock detector and track detector bits.

4.3.5.2 CRG Reference Divider Register (REFDV)

The REFDV register provides a finer granularity for the PLL multiplier steps. The bits in this register can be written anytime except when the CLKSEL[PLLSEL] bit is set. The count in the reference divider is used to divide the OSCCLK frequency by REFDV + 1.

	7	6	5	4	3	2	1	0
R	0	0	0	0	REFDV			
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	CRG Base + 0x0001							

Figure 4-9. CRG Reference Divider Register (REFDV)

NOTE

A write to this register initializes the lock detector and track detector bits.

4.3.5.3 CRG Flags Register (CRGFLG)

This register provides CRG status bits and flags.

	7	6	5	4	3	2	1	0
R	STPEF	PORF	LVRF	LOCKIF	LOCK	TRACK	SCMIF	SCM
W								
Reset	0	—[1]	—[2]	0	0	0	0	0
Reg Addr	CRG Base + 0x0003							

[1] PORF is set when a power on reset occurs. Unaffected by system reset.

[2] LVRF is set when a low voltage reset occurs. Unaffected by system reset.

Figure 4-10. CRG Flags Register (CRGFLG)

Table 4-3. CRGFLG Field Descriptions

Bits	Name	Description
7	STPEF [1]	Stop entry flag. STPEF is cleared when the SDMCTL[STOP] bit (see Section 4.3.5.7) is written. It is set when the system actually enters STOP mode. It is not set if the system receives a wake-up interrupt before the stop procedure completes. The flag is cleared by writing a 1. 0 System did not enter STOP mode after last STOP command 1 System entered STOP mode after last STOP command
6	PORF	Power-on reset flag. PORF is set to 1 when a power on reset occurs. This flag can be cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 Power-on reset has not occurred 1 Power-on reset has occurred
5	LVRG	Low-voltage reset flag. LVRF is set to 1 when a low-voltage reset occurs. This flag can be cleared by writing a 1 to this bit. Writing a 0 has no effect. This bit will not be valid if the Voltage Regulator is bypassed. 0 Low-voltage reset has not occurred 1 Low-voltage on reset has occurred
4	LOCKIF	PLL lock interrupt flag. LOCKIF is set to 1 when the LOCK status bit changes. This flag can be cleared by writing a 1 to this bit. Writing a 0 has no effect. If enabled (LOCKIE=1), LOCKIF causes an interrupt request. 0 No change in LOCK bit 1 LOCK bit has changed
3	LOCK	Lock status bit. LOCK reflects the current state of PLL lock condition. This bit is cleared in Self Clock Mode. Writes have no effect. 0 PLL VCO is not within the desired tolerance of the target frequency 1 PLL VCO is within the desired tolerance of the target frequency
2	TRACK	Track status bit. TRACK reflects the current state of the PLL track condition. This bit is cleared in Self-Clock Mode. Writes have no effect. 0 Acquisition mode status 1 Tracking mode status

Table 4-3. CRGFLG Field Descriptions (continued)

Bits	Name	Description
1	SCMIF	Self-clock mode interrupt flag. SCMIF is set to 1 when the SCM status bit changes. This flag can be cleared by writing a 1 to this bit. Writing a 0 has no effect. If enabled (SCMIE=1), SCMIF causes an interrupt request. 0 No change in SCM bit 1 SCM bit has changed
0	SCM	Self-clock mode status. SCM reflects the current clocking mode. Writes have no effect. 0 MCU is operating normally with OSCCLK available 1 MCU is operating in Self Clock Mode with OSCCLK in an unknown state. All clocks are derived from PLLCLK running at its minimum frequency f_{SCM}

[1] On mask set L49P devices, this bit is not implemented. It should be considered Reserved.

4.3.5.4 CRG Interrupt Enable Register (CRGINT)

This register enables CRG interrupt requests.

	7	6	5	4	3	2	1	0
R	0	0	0	LOCKIE	0	0	SCMIE	0
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	CRG Base + 0x0004							

Figure 4-11. CRG Interrupt Enable Register (CRGINT)

Table 4-4. CRGINT Field Descriptions

Bits	Name	Description
7–5	—	Reserved.
4	LOCKIE	Lock interrupt enable. 0 LOCK interrupt requests are disabled 1 Interrupt will be requested whenever LOCKIF is set
3–2	—	Reserved.
1	SCMIE	Self-clock mode interrupt enable. 0 SCM interrupt requests are disabled 1 Interrupt will be requested whenever SCMIF is set
0	—	Reserved.

4.3.5.5 CRG Clock Select Register (CLKSEL)

This register controls CRG clock selection. Refer to Figure 4-17 for more details.

	7	6	5	4	3	2	1	0
R	PLLSEL	PSTP	0	0	PLLDOZE	0	RTIDOZE	SWTDOZE
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	CRG Base + 0x0005							

Figure 4-12. CRG Clock Select Register (CLKSEL)

Table 4-5. CLKSEL Field Descriptions

Bits	Name	Description
7	PLLSEL	PLL select bit. Writing a one when LOCK=0 and AUTO=1, or TRACK=0 and AUTO=0 has no effect. This prevents the selection of an unstable PLLCLK as f_{SYS}. The PLLSEL bit is cleared when the MCU enters self-clock mode, stop mode or doze mode with the PLLDOZE bit set. 0 System clocks are derived from OSCCLK (f_{SYS} = OSCCLK) 1 System clocks are derived from PLLCLK (f_{SYS} = PLLCLK)
6	PSTP	Pseudo-stop. This bit controls the functionality of the oscillator during stop mode. 0 Oscillator is disabled in stop mode. 1 Oscillator continues to run in stop mode (pseudo-stop). **Note:** Pseudo-stop allows for faster stop recovery and reduces the mechanical stress and aging of the resonator when frequent stop conditions are asserted at the expense of a slightly increased power consumption.
5–4	—	Reserved.
3	PLLDOZE	PLL stops in doze mode. If PLLDOZE is set, the CRG will clear the PLLSEL bit before entering doze mode. The PLLON bit remains set during doze mode, but the PLL is powered down. Upon exiting doze mode, the PLLSEL bit must be set manually if PLL clock is required. While the PLLDOZE bit is set, the AUTO bit is set in order to allow the PLL to lock automatically on to the selected target frequency after exiting doze mode. 0 PLL keeps running in doze mode. 1 PLL stops in doze mode.
2	—	Reserved.
1	RTIDOZE	RTI stops in doze mode. 0 RTI keeps running in doze mode 1 RTI stops and initializes the RTI dividers when the part goes into doze mode
0	SWTDOZE	SWT stops in doze mode. In normal modes, this bit can be written once. In special modes, it can be written anytime. 0 SWT keeps running in doze mode 1 SWT stops and initializes the SWT dividers when the part goes into doze mode

4.3.5.6 CRG PLL Control Register (PLLCTL)

The PLLCTL register controls the PLL functionality.

	7	6	5	4	3	2	1	0
R W	CME	PLLON	AUTO	ACQ	FSTWKP	PRE	PWE	SCME
Reset	1	1	1	1	0	0	0	1
Reg Addr	CRG Base + 0x0006							

Figure 4-13. CRG PLL Control Register (PLLCTL)

Table 4-6. PLLCTL Field Descriptions

Bits	Name	Description
7	CME	Clock monitor enable. CME enables the clock monitor. Write anytime except when SCM = 1. 0 Clock monitor is disabled 1 Clock monitor is enabled. Slow or stopped clocks will cause a clock monitor reset sequence or self-clock mode. **Note:** In stop mode (PSTP=0) the clock monitor is disabled independently of the CME bit setting and loss of clock will not be detected. Operating with CME=0 will not detect any loss of clock. In cases of poor clock quality, this could cause unpredictable operation of the MCU.
6	PLLON	Phase lock loop on. PLLON turns on the PLL circuitry. In self-clock mode, the PLL is turned on, but the PLLON bit reads the last latched value. Write anytime except when PLLSEL = 1. 0 PLL is turned off 1 PLL is turned on. If AUTO bit is set, the PLL will lock automatically
5	AUTO	Automatic bandwidth control. AUTO selects either the high bandwidth (acquisition) mode or the low bandwidth (tracking) mode depending on how close to the desired frequency the VCO is running. Write anytime except when PLLDOZE=1, because PLLDOZE sets the AUTO bit to 1. 0 Automatic mode control is disabled and the PLL is under software control, using the ACQ bit 1 Automatic mode control is enabled and the ACQ bit has no effect
4	ACQ	Acquisition. Write anytime. If AUTO=1 this bit has no effect. 0 Low bandwidth filter is selected 1 High bandwidth filter is selected
3	FSTWKP [1]	Fast Wake-up. FSTWKP enables fast wake-up from full stop mode. If self-clock mode is disabled (SCME=0) this bit has no effect. 0 Fast wake-up from full stop mode is disabled. 1 Fast wake-up from full stop mode is enabled.
2	PRE	RTI enable during pseudo-stop mode. PRE enables the RTI clock during pseudo-stop mode. Write anytime. 0 RTI stops running during pseudo-stop mode. 1 RTI continues running during pseudo-stop mode.

Table 4-6. PLLCTL Field Descriptions (continued)

Bits	Name	Description
1	PWE	SWT enable during pseudo stop. PWE enables the SWT clock during pseudo-stop mode. Write anytime. 0 SWT stops running during pseudo-stop mode 1 SWT continues running during pseudo-stop mode
0	SCME	Self-clock mode enable. In normal modes, this bit can be written once. In special modes, it can be written at any time. SCME cannot be cleared while operating in self-clock mode (CRGFLG[SCM] bit is set). 0 Detection of crystal clock failure causes clock monitor reset (see Section 4.3.6.7.3, “Clock Monitor Reset”) 1 Detection of crystal clock failure forces the MCU into self-clock mode (see Section 4.3.6.10.3, “Run, Self-Clock Mode”)

[1] On mask set L49P devices, this bit is not implemented, and is reserved.

4.3.5.7 CRG Stop/Doze Control Register (SDMCTL)

This register controls how the MCU transitions between stop and doze modes.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DOZE	STOP
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	CRG Base + 0x0007							

Figure 4-14. CRG Stop/Doze Control Register (SDMCTL)

Table 4-7. SDMCTL Field Descriptions

Bits	Name	Description
7–2	—	Reserved.
1	DOZE	Doze control. Setting this bit starts a transition to doze mode. Refer to Figure 4-26. 0 Remain in run mode 1 Activate doze sequence
0	STOP	Stop control. Setting this bit starts a transition to stop mode. When a write access with the STOP bit set occurs, the CRG holds the bus until it is ready to turn off the clocks. Thus, after writing the STOP bit, no other accesses to peripherals are possible. Refer to Figure 4-27. 0 Remain in run mode 1 Activate stop sequence

4.3.5.8 CRG BDM Control Register (BDMCTL)

This register controls the SWT (Software Watchdog Timer) and RTI clocks in debug mode.

	7	6	5	4	3	2	1	0
R	0	RSBCK	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	CRG Base + 0x0008							

Figure 4-15. CRG BDM Control Register (BDMCTL)

Table 4-8. BDMCTL Field Description

Bits	Name	Description
7	—	Reserved.
6	RSBCK	SWT and RTI stop in debug mode. This bit can be written once. 0 Allows the SWT and RTI to keep running in debug mode 1 Stops the SWT and RTI counters whenever the part is in debug mode
5–0	—	Reserved.

4.3.6 CRG Functional Description

4.3.6.1 Phase Locked Loop (PLL)

The PLL is used to drive the MCU from a different time base than the incoming OSCCLK. For increased flexibility, OSCCLK can be divided by 1 to 16 to generate the reference frequency. The PLL can multiply this reference frequency by 2, 4, 6,... 126 or 128 based on the two control registers SYNR and REFDIV.

The frequency generated by the PLL, PLLCLK, is determined by the values of two of the CRG module control registers, REFDIV and SYNR. The frequency is calculated using the following equation:

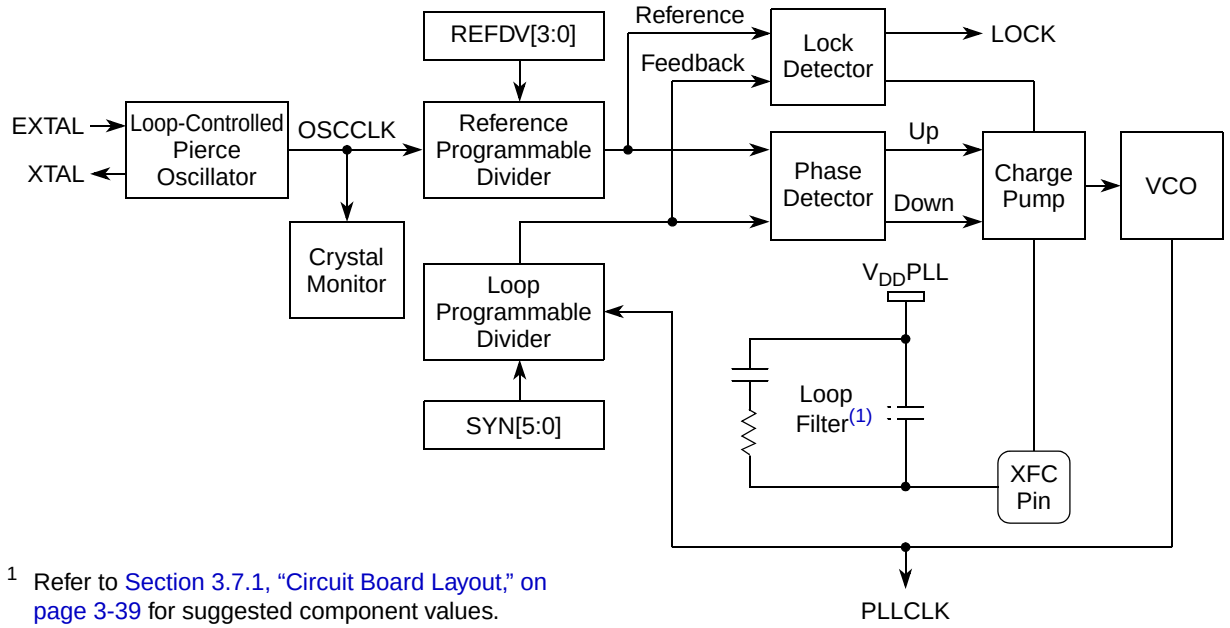
$$PLLCLK = 2 \times OSCCLK \times \frac{SYN[5:0] + 1}{REFDV[3:0] + 1} \qquad \textit{Eqn. 4-5}$$

NOTE

Although it is possible to choose the two parameters to set a very high clock frequency, do not exceed the frequency limit specified in the *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC). If the CLKSEL[PLLSEL] bit is set, f_{SYS} = PLLCLK.

The PLL is a frequency generator that operates in either acquisition mode or tracking mode, depending on the difference between the output frequency and the target frequency. The PLL can change between acquisition and tracking modes either automatically or manually.

The VCO has a minimum operating frequency, which corresponds to the self clock mode frequency f_{SCM}.



1 Refer to Section 3.7.1, “Circuit Board Layout,” on page 3-39 for suggested component values.

Figure 4-16. PLL Block Diagram

4.3.6.1.1 PLL Operation

The oscillator output clock signal (OSCCLK) is fed through the reference programmable divider and is divided in a range of 1 to 16 (REFDV + 1) to output the REFERENCE clock. The VCO output clock, (PLLCLK) is fed back through the programmable loop divider and is divided in a range of 2 to 128 in increments of [2 × (SYN + 1)] to output the FEEDBACK clock (see Figure 4-16).

The phase detector then compares the FEEDBACK clock with the REFERENCE clock. Correction pulses are generated based on the phase difference between the two signals. The loop filter then slightly alters the DC voltage on the external filter capacitor connected to XFC pin, based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, as described in the next subsection. The values of the external filter network and the reference frequency determine the speed of the corrections and the stability of the PLL.

4.3.6.1.2 Acquisition and Tracking Modes

The lock detector compares the frequencies of the FEEDBACK clock and the REFERENCE clock. Therefore, the speed of the lock detector is directly proportional to the final reference frequency. The circuit determines the mode of the PLL and the lock condition based on this comparison.

The PLL filter can be manually or automatically configured into one of two possible operating modes:

- Acquisition mode: The filter can make large frequency corrections to the VCO. This mode is used at PLL start-up or when the PLL has suffered a severe noise hit and the VCO frequency is far from the desired frequency. When in acquisition mode, the TRACK status bit is cleared in the CRGFLG register.
- Tracking mode: The filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct and the TRACK bit is set in the CRGFLG register.

The PLL can change the bandwidth or operational mode of the loop filter manually or automatically.

In automatic bandwidth control mode (PLLCTL[AUTO] = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode is also used to determine when the PLL clock (PLLCLK) is safe to use as the source for the system and peripheral clocks. If the PLL LOCK interrupt requests are enabled, the software can wait for an interrupt request and then check the CRGFLG[LOCK] bit. If CPU interrupts are disabled, software can poll the LOCK bit continuously (usually during PLL start-up) or at periodic intervals. In either case, only when the LOCK bit is set is the PLLCLK clock safe to use as the source for the system and peripheral clocks. If the PLL is selected as the source for the system and peripheral clocks and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application.

The following conditions apply when the PLL is in automatic bandwidth control mode (AUTO=1):

- The CRGFLG[TRACK] bit is a read-only indicator of the mode of the filter.
- The CRGFLG[TRACK] bit is set when the VCO frequency is within a certain tolerance, Δ_{trk}, and is clear when the VCO frequency is out of a certain tolerance, Δ_{unt}.
- The CRGFLG[LOCK] bit is a read-only indicator of the locked state of the PLL.
- The CRGFLG[LOCK] bit is set when the VCO frequency is within a certain tolerance, Δ_{Lock}, and is cleared when the VCO frequency is out of a certain tolerance, Δ_{unl}.
- CPU interrupts can occur if enabled (CRGINT[LOCKIE] = 1) when the lock condition changes, toggling the CRGFLG[LOCK] bit.

The PLL can also operate in manual mode (PLLCTL[AUTO] = 0). Manual mode is used by systems that do not require an indicator of the lock condition for proper operation. Such systems typically operate well below the maximum system frequency (f_{SYS}) and require fast start-up. The following conditions apply when in manual mode:

- PLLCTL[ACQ] is a writable control bit that controls the mode of the filter. Before turning on the PLL in manual mode, the ACQ bit should be set to configure the filter in acquisition mode.
- After turning on the PLL by setting the PLLCTL[PLLON] bit, software must wait a given time (t_{acq}) before entering tracking mode (ACQ = 0).
- After entering tracking mode, software must wait a given time (t_{al}) before selecting the PLLCLK as the source for system and peripheral clocks (by setting the CLKSEL[PLLSEL] bit).

4.3.6.2 System Clocks Generator

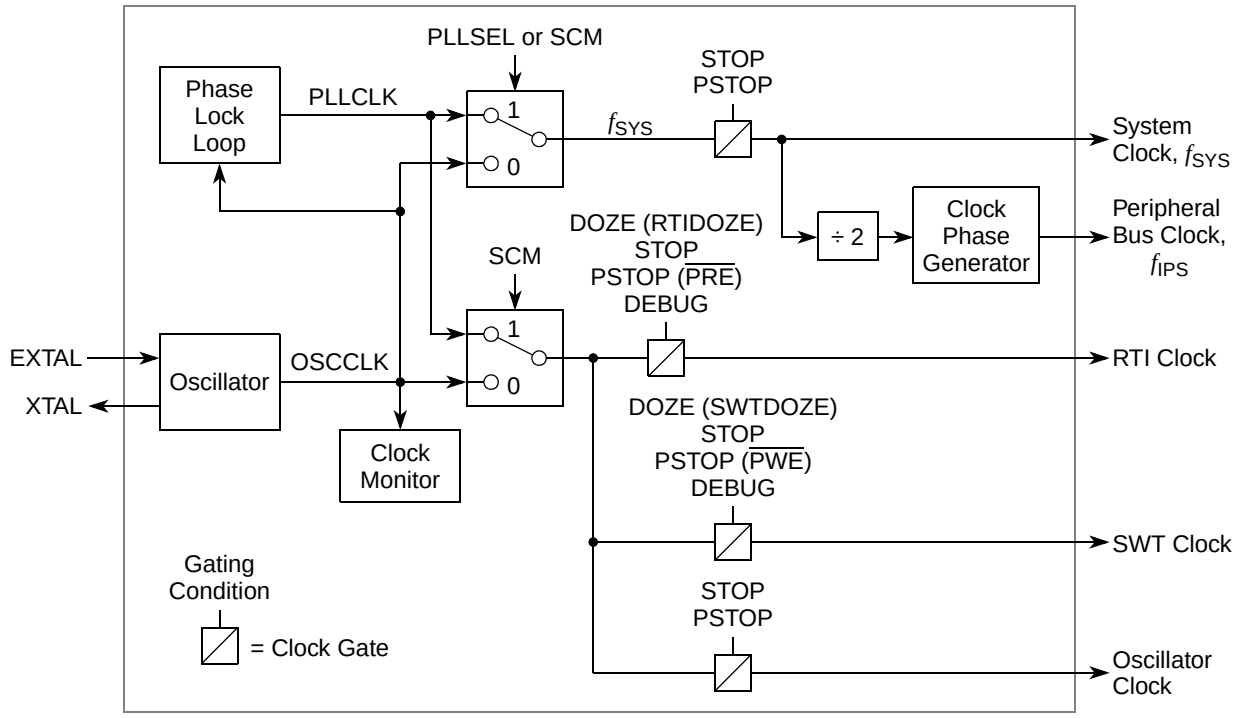


Figure 4-17. System Clocks Generator Block Diagram

The clock generator creates the clocks used in the MCU (see Figure 4-17). The gating condition placed on top of the individual clock gates indicates the dependencies of different modes (STOP, PSTOP, DOZE, DEBUG) and the setting of the respective configuration bits.

The CPU uses the system clock, f_{SYS}. The peripherals and memory modules use the peripheral bus clock, f_{IPS}. Some peripheral modules also use the oscillator clock (see Figure 4-1). If the MCU enters self-clock mode (see Section 4.3.6.10.3, “Run, Self-Clock Mode”) the oscillator clock source is switched to PLLCLK, f_{IPS}, running at its minimum frequency, f_{SCM}. The system clock is twice the peripheral bus clock, f_{IPS}, as shown in Figure 4-18. Note that a CPU cycle corresponds to one peripheral bus clock, f_{IPS}.

PLL clock mode is selected with the PLLSEL bit in the CLKSELregister. When selected, the PLL output clock drives f_{SYS} for the main system, including the CPU and peripherals. The PLL cannot be turned off by clearing the PLLCTL[PLLON] bit if the PLL clock is selected. When CLKSEL[PLLSEL] is changed, it takes a maximum of 4 OSCCLK plus 4 PLLCLK cycles to make the transition. During the transition, all clocks freeze and CPU activity ceases.

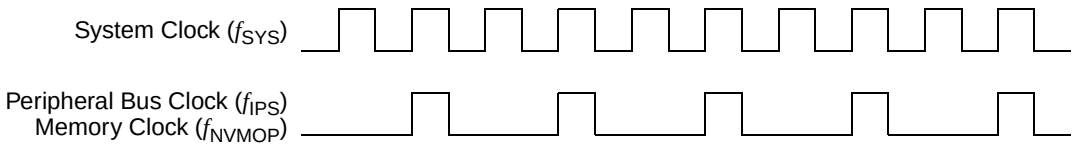


Figure 4-18. System Clock and Peripheral Bus Clock Relationship

4.3.6.3 Clock Monitor (CM)

If no OSCCLK edges are detected within a certain time, the clock monitor within the oscillator block generates a clock monitor fail event. The CRG then asserts self-clock mode or generates a system reset depending on the state of the PLLCTL[SCME] bit. If the clock monitor is disabled or the presence of clocks is detected, no failure is indicated by the oscillator block. The clock monitor function is enabled/disabled by the PLLCTL[CME] bit.

4.3.6.4 Clock Quality Checker

The clock monitor performs a coarse check on the incoming clock signal. The clock quality checker provides a more accurate check in addition to the clock monitor.

A clock quality check is triggered by any of the following events:

- Power-on reset (POR)
- Low-voltage reset (LVR)
- Wake-up from full-stop mode (exit full-stop)
- Clock monitor fail indication (CM FAIL)

A time window of 50,000 VCO clock cycles (which are generated by the PLL when running at minimum frequency f_{SCM}) is called *check window.* A number greater than or equal to 4096 rising OSCCLK edges within a check window is called osc ok. Note that osc ok immediately terminates the current check window. See Figure 4-19 as an example.

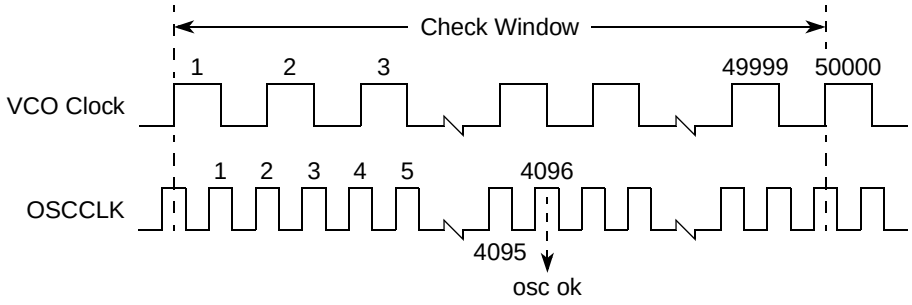


Figure 4-19. Check Window Example

The sequence for clock quality check is shown in Figure 4-20 and Figure 4-21.

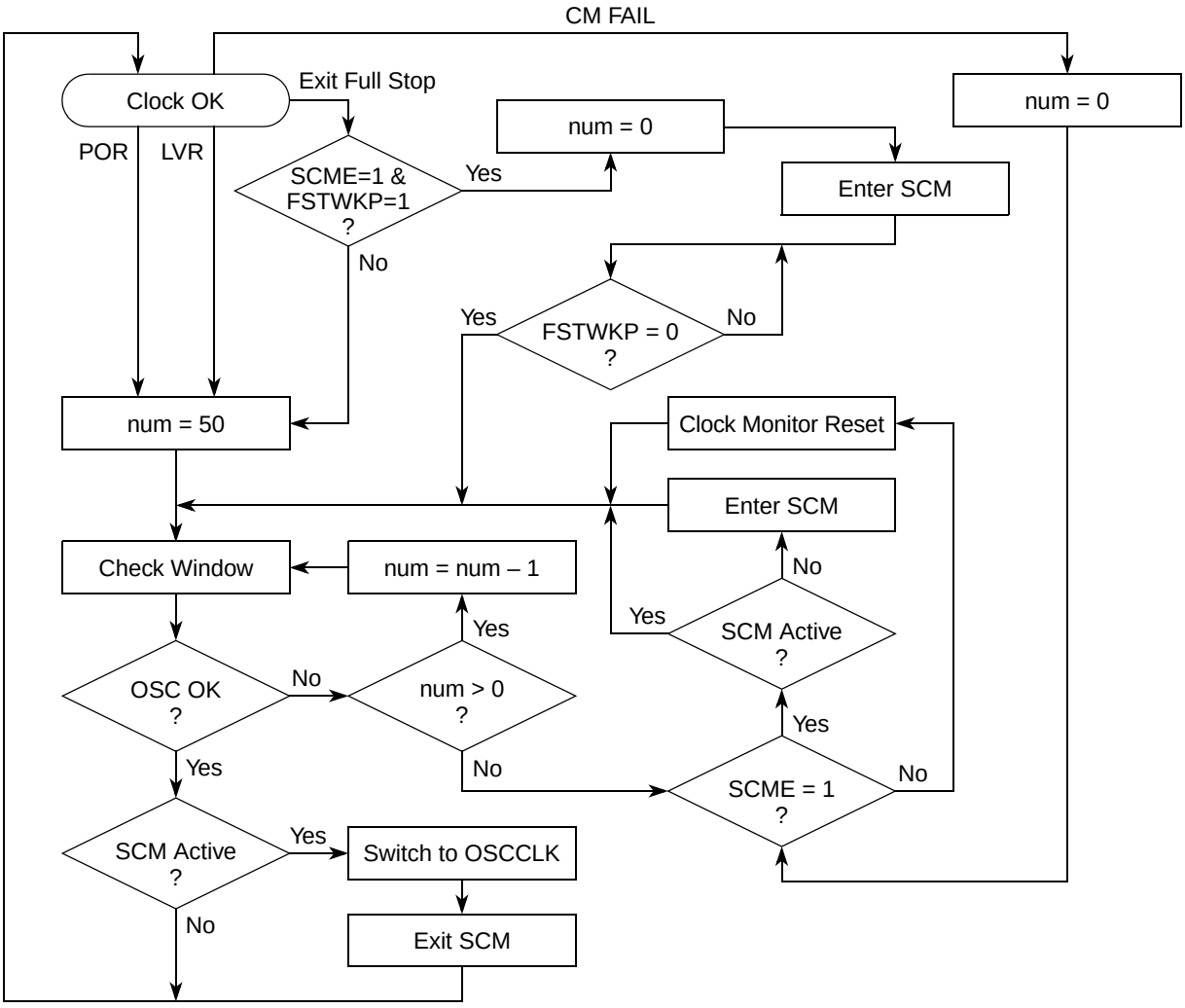


Figure 4-20. Sequence for Clock Quality Check on Non-Mask Set L49P Devices

NOTE

Note that in parallel to additional actions caused by self-clock mode or clock monitor reset handling (setting the SCME bit), the clock quality checker continues to check the OSCCLK signal.

The clock quality checker enables the PLL and the VREG anytime a clock check has to be performed. An ongoing clock quality check could also cause a running PLL (f_{SCM}) and an active VREG during pseudo-stop mode or doze mode

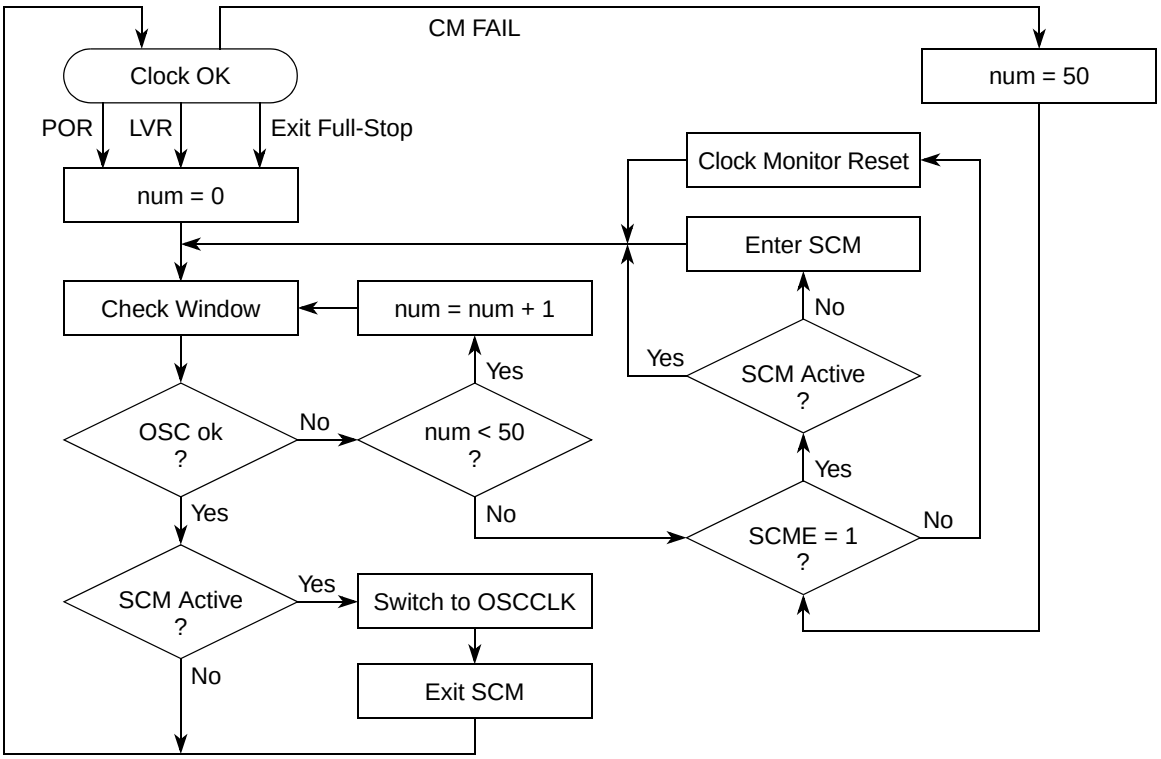


Figure 4-21. Sequence for Clock Quality Check on Mask Set L49P Devices

4.3.6.5 Software Watchdog Timer (SWT)

The SWT (free running watchdog timer) allows the user to check that a program is running and sequencing properly. The watchdog timer is contained in the MCM module (refer to Chapter 11, “Miscellaneous Control Module (MCM)”). However, the CRG module provides the clock for the watchdog timer. If the PLLCTL[PWE] bit is set, the SWT will continue to run in pseudo-stop mode.

4.3.6.6 Real Time Interrupt (RTI)

The RTI (real time interrupt) counter is contained in the PIT module (refer to Chapter 25, “Periodic Interrupt Timer Module (PIT)”). However, the CRG module provides a special RTI clock. This is a gated OSCCLK. If the PLLCTL[PRE] bit is set, the RTI will continue to run in pseudo-stop mode.

4.3.6.7 Resets

This section describes how to reset the CRG and how the CRG controls the reset of the MCU, including all special reset requirements. Since the reset generator for the MCU is part of the CRG, this section also describes all automatic actions that occur during or as a result of individual reset conditions. The reset values of registers and signals are provided in Section 4.3.5, “CRG Memory Map / Register Definition.” All reset sources are listed in Table 4-9. Refer to Chapter 6, “Exceptions,” for related vector addresses and priorities.

Table 4-9. CRG Reset Source Summary

Reset Source	Local Enable
Power on Reset	None
External Reset	None
Clock Monitor Reset	PLLCTL (CME=1, SCME=0)
SWT Watchdog Reset	None

4.3.6.7.1 Reset Operation Description

The reset sequence is initiated by any of the following events:

- Low level is detected at the $\overline{\text{RESET}}$ pin
- Power on is detected
- Software watchdog timeout
- Clock monitor failure is detected and self-clock mode was disabled (PLLCTL[SCME]=0)

On detection of any reset event, an internal circuit drives the $\overline{\text{RESET}}$ pin low for 256 f_{SYS} cycles (see Figure 4-22). Since entry into reset is asynchronous, it does not require a running f_{SYS}. However, the internal reset circuit of the CRG cannot sequence out of the current reset condition without a running f_{SYS}. The number of 256 f_{SYS} cycles might be increased by n=3 to 6 additional f_{SYS} cycles depending on the internal synchronization latency. After 256 + n f_{SYS} cycles the $\overline{\text{RESET}}$ pin is released. The reset generator of the CRG waits for an additional 64 f_{SYS} cycles and then samples the $\overline{\text{RESET}}$ pin to determine the originating source. Figure 4-10 shows which vector will be fetched.

Table 4-10. CRG Reset Vector Selection

sampled $\overline{\text{RESET}}$ pin (64 cycles after release)	Clock Monitor Reset pending	SWT Reset pending	Vector fetch
1	0	0	POR / LVR / External Reset
1	1	X	Clock Monitor Reset
1	0	1	SWT Reset
0	X	X	POR / LVR / External Reset with rise of $\overline{\text{RESET}}$ pin

NOTE

External circuitry connected to the $\overline{\text{RESET}}$ pin should not include a large capacitance that would interfere with the ability of this signal to rise to a valid logic one within 64 f_{SYS} cycles after the low drive is released.

The internal reset of the MCU remains asserted while the reset generator completes the 320 f_{SYS} long reset sequence. The reset generator circuitry always makes sure the internal reset is negated synchronously after completion of the 320 f_{SYS} cycles. When the $\overline{\text{RESET}}$ pin is externally driven low for more than these 320 f_{SYS} cycles (external reset), the internal reset also remains asserted.

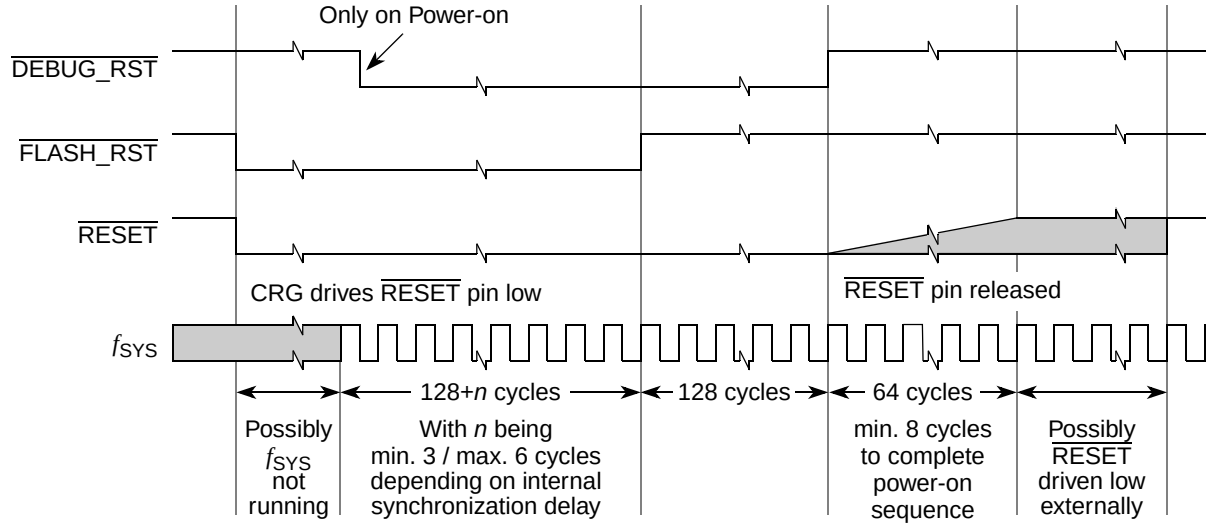


Figure 4-22. RESET Timing

At least 128 cycles before the internal reset is negated, the Flash reset is negated. This provides the CFM controller with ample time to execute a self-test sequence. Like the internal reset, the Flash reset will also be negated synchronously. The debug module reset is asserted together with the internal reset, but will not be extended by the external RESET pin. In addition, the debug module reset will be asserted only if the reset sequence was initiated by a power-on or low-voltage indication.

This enables the user to activate the debugger in two ways: either by holding the reset input asserted while the debugger is being activated or by setting up the debugger after a power-on reset, then applying an additional reset that will not affect the debug logic (i.e. not a power-on reset).

Note that the power-on reset sequence is not complete until at least 8 cycles after the CRG releases RESET. If another external reset occurs before the 8 cycles have expired, the new reset will also be treated as a power-on reset, so DEBUG_RESET will be asserted again.

4.3.6.7.2 JTAG Reset [1]

DEBUG_RESET can also be asserted via a JTAG command. This causes a much shorter reset sequence solely for the DEBUG_RESET; other parts of the device are not affected. Note that the JTAG reset sequence may not be combined with any other reset sequence.

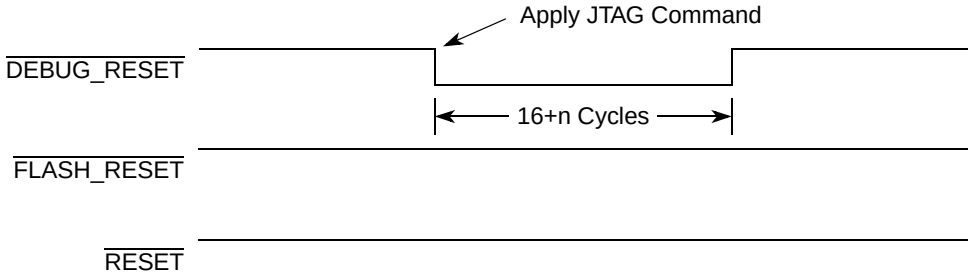


Figure 4-23. RESET Timing controlled by JTAG

1. This feature is not implemented on mask set L49P devices.

4.3.6.7.3 Clock Monitor Reset

The CRG generates a Clock Monitor Reset when all of the following conditions are true:

- Clock monitor is enabled (PLLCTL[CME]=1)
- Loss of clock is detected
- Self-Clock Mode is disabled (PLLCTL[SCME]=0)

The reset event asynchronously forces the configuration registers to their default settings (see Section 4.3.5, “CRG Memory Map / Register Definition”). The PLLCTL[CME] and PLLCTL[SCME] bits are set (which doesn’t change the state of the CME bit, because it was already set). As a consequence, the CRG immediately enters self-clock mode and starts its internal reset sequence. In parallel, the clock quality check starts. As soon as the clock quality check indicates a valid oscillator clock, the CRG switches to OSCCLK and exits self-clock mode. Since the clock quality checker is running in parallel with the reset generator, the CRG may exit self-clock mode while still completing the internal reset sequence. When the reset sequence is finished, the CRG checks the internally latched state of the clock monitor fail circuit. If a clock monitor fail is indicated, processing begins by fetching the clock monitor reset vector.

4.3.6.8 Software Watchdog Timer (SWT) Reset

The CRG will generate a reset if the SWT reset is enabled in the MCM (refer to Section 11.3.1.8, “MCM Software Watchdog Timer Control Register (MSWTCR),” on page 11-129,” for configuration information).

4.3.6.8.1 Power On Reset

The on-chip voltage regulator detects when $V_{DD}2.5$ to the MCU has reached a certain level and asserts a power on reset. As soon as a power on reset is triggered, the CRG performs a quality check on the incoming clock signal. As soon as clock quality check indicates a valid oscillator clock signal, the reset sequence starts using the oscillator clock. If after 50 check windows the clock quality check indicates a non-valid oscillator clock, the reset sequence starts using self-clock mode.

Figure 4-24 and Figure 4-25 show the power-up sequence for cases when the $\overline{\text{RESET}}$ pin is tied to $V_{DD}2.5$ and when the $\overline{\text{RESET}}$ pin is held low.

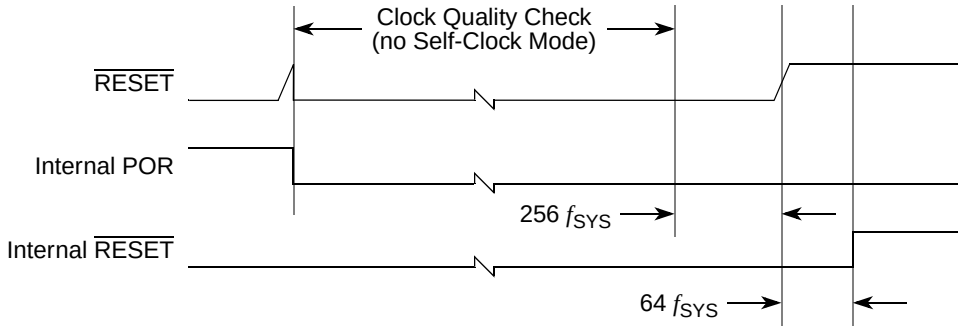


Figure 4-24. Power-Up Sequence — $\overline{\text{RESET}}$ Pin Tied to $V_{DD}2.5$ via a Pull-up Resistor

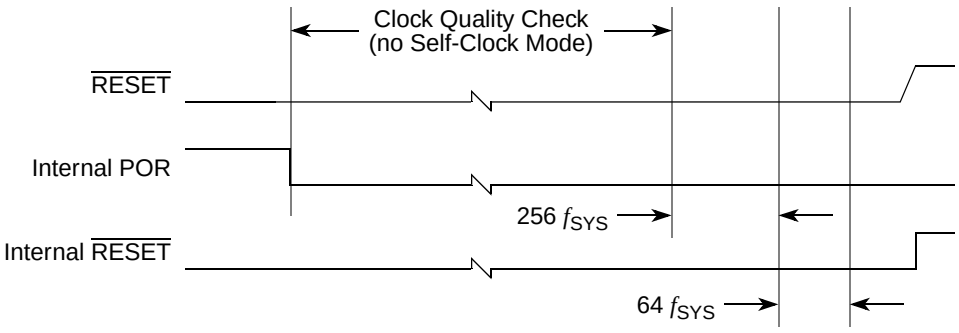


Figure 4-25. Power-Up Sequence — RESET Pin Driven Low Externally

4.3.6.9 Interrupts

The interrupt requests generated by the CRG are listed in Table 4-11. Refer to Table 6-2 on page 6-85 for related vector addresses and priorities.

Table 4-11. CRG Interrupt Sources

Interrupt Source	CCR Mask	Local Enable
LOCK interrupt	I bit	CRGINT[LOCKIE]
SCM interrupt	I bit	CRGINT[SCMIE]

4.3.6.9.1 PLL Lock Interrupt

The CRG generates a PLL lock interrupt when the lock condition of the PLL has changed, either from a locked state to an unlocked state or vice versa. Lock interrupts are locally disabled by clearing the CRGINT[LOCKIE] bit. The PLL lock interrupt flag (CRGFLG[LOCKIF]) is set when the lock condition has changed and is cleared by writing a 1 to the LOCKIF bit.

4.3.6.9.2 Self Clock Mode Interrupt

The CRG generates a self-clock mode interrupt when the SCM condition of the system has changed (on self-clock mode entry or exit). SCM conditions can only change if the self-clock mode enable bit (PLLCTL[SCME]) is set. SCM conditions are caused by failing clock quality check after power-on reset (POR), low-voltage reset (LVR), recovery from full stop mode (CLKSEL[PSTP]=0) or clock monitor failure. Refer to Section 4.3.6.4, “Clock Quality Checker,” for details. If the clock monitor is enabled (PLLCTL[CME]=1) a loss of external clock will also cause a SCM condition (PLLCTL[SCME]=1).

SCM interrupts are disabled locally by clearing the CRGINT[SCMIE] bit. The SCM interrupt flag (CRGFLG[SCMIF]) is set when the SCM condition has changed, and is cleared by setting the SCMIF bit.

4.3.6.10 CRG Operating Mode Details

While most modules on MAC7100 family devices operate in one of four modes (normal, doze, pseudo-stop and stop), the CRG has two distinct normal modes, determined by the state of the external time base signal used by the oscillator and PLL to generate internal clocks.

4.3.6.10.1 Run Modes

These are the normal modes, where all components of the system are clocked.

4.3.6.10.2 Run, Normal Clock Mode

The normal clock mode is the expected mode of operation, and is the assumed state for most discussions in the document. In this mode, the external time base reference is stable and drives the oscillator and PLL as summarized in Section 4.2.1, “OSC Overview,” and Section 4.3.1, “CRG Overview.”

4.3.6.10.3 Run, Self-Clock Mode

If the external clock frequency is not available due to a failure or a long crystal start-up time, the peripheral bus clock (f_{IPS}) and the system clock (f_{SYS}) are derived from the VCO running at the minimum operating frequency (f_{SCM}); this mode of operation is called self-clock mode. This requires PLLCTL[CME]=1 and PLLCTL[SCME]=1. If the MCU was clocked by the PLL clock prior to entering self-clock mode, the CLKSEL[PLLSEL] bit will be cleared. If the external clock signal has stabilized again, the CRG will automatically select OSCCLK to be the system clock and return to normal mode. Refer to Section 4.3.6.4, “Clock Quality Checker,” for more information on entering and leaving self-clock mode.

NOTE

In order to detect a potential clock loss, the PLLCTL[CME] bit should always be set. If the PLLCTL[CME] bit is clear and the MCU is configured to use the PLL clock, a loss of external clock (OSCCLK) will not be detected and will cause the system clock to drift towards the VCO minimum frequency, f_{SCM}. As soon as the external clock is available again, the system clock ramps up to its PLL target frequency. If the MCU is running on an external clock, any loss of clock will cause the system to go static.

4.3.6.10.4 Doze Mode

Setting the SDMCTL[DOZE] bit puts the system in a low power consumption stand-by mode, further controlled by the CLKSEL register settings. This provides enhanced granularity in reducing the level of power consumption. Table 4-12 lists the individual configuration bits and the parts of the MCU that are affected in doze mode if the corresponding CLKSEL bit is set.

Table 4-12. MCU Configuration During Doze Mode

Clock Source	PLLDOZE	RTIDOZE	SWTDOZE
PLL	stopped	—	—
RTI Clock	—	stopped	—
SWT Clock	—	—	stopped

The core requests the CRG to switch into doze mode by writing to the SDMCTL[DOZE] bit. The CRG then checks whether the CLKSEL[PLLDOZE] bit is asserted (see Figure 4-26). Depending on the configuration, the CRG switches the system clock to OSCCLK by clearing the CLKSEL[PLLSEL] bit and disables the PLL.

There are five different scenarios for the CRG to restart the MCU from doze mode:

- External Reset
- Clock Monitor Reset
- SWT Reset
- Self Clock Mode Interrupt
- Wake-up Interrupt (e.g. RTI)

If the MCU receives an external reset while in doze mode, the CRG asynchronously restores all configuration bits in the register space to their default settings and starts the reset generator. After completing the reset sequence, processing begins by fetching the normal reset vector. Doze mode is exited and the MCU is in run mode again.

If the clock monitor is enabled (PLLCTL[CME]=1) the MCU is able to exit doze mode when loss of oscillator/external clock is detected by a clock monitor fail. If the PLLCTL[SCME] bit is not set, the CRG generates a clock monitor fail reset (CMRESET). The CRG behavior for CMRESET is the same as the response to an external reset, but a different reset vector is fetched after completion of the reset sequence. If the SCME bit is set, the CRG generates an SCM interrupt if enabled (CRGINT[SCMIE]=1). After generating the interrupt, the CRG enters self-clock mode and starts the clock quality checker (see Section 4.3.6.4, “Clock Quality Checker”). Then the MCU continues with normal operation. If the SCM interrupt is blocked by SCMIE=0, the CRGFLG[SCMIF] bit will be set and clock quality checks will be performed, but the MCU will not wake-up from doze mode.

If any other interrupt source (e.g. RTI) triggers the exit from doze mode, the MCU immediately continues with normal operation. If the PLL has been powered down during doze mode, the CLKSEL[PLLSEL] bit is cleared and the MCU runs on OSCCLK after leaving doze mode. The software must manually set the PLLSEL bit again in order to switch system and peripheral clocks to the PLLCLK.

If doze mode is entered from self-clock mode, the CRG will continue to check the clock quality until the clock check is successful. The PLL and voltage regulator will remain enabled. Table 4-13 summarizes the outcome of a clock loss while in doze mode.

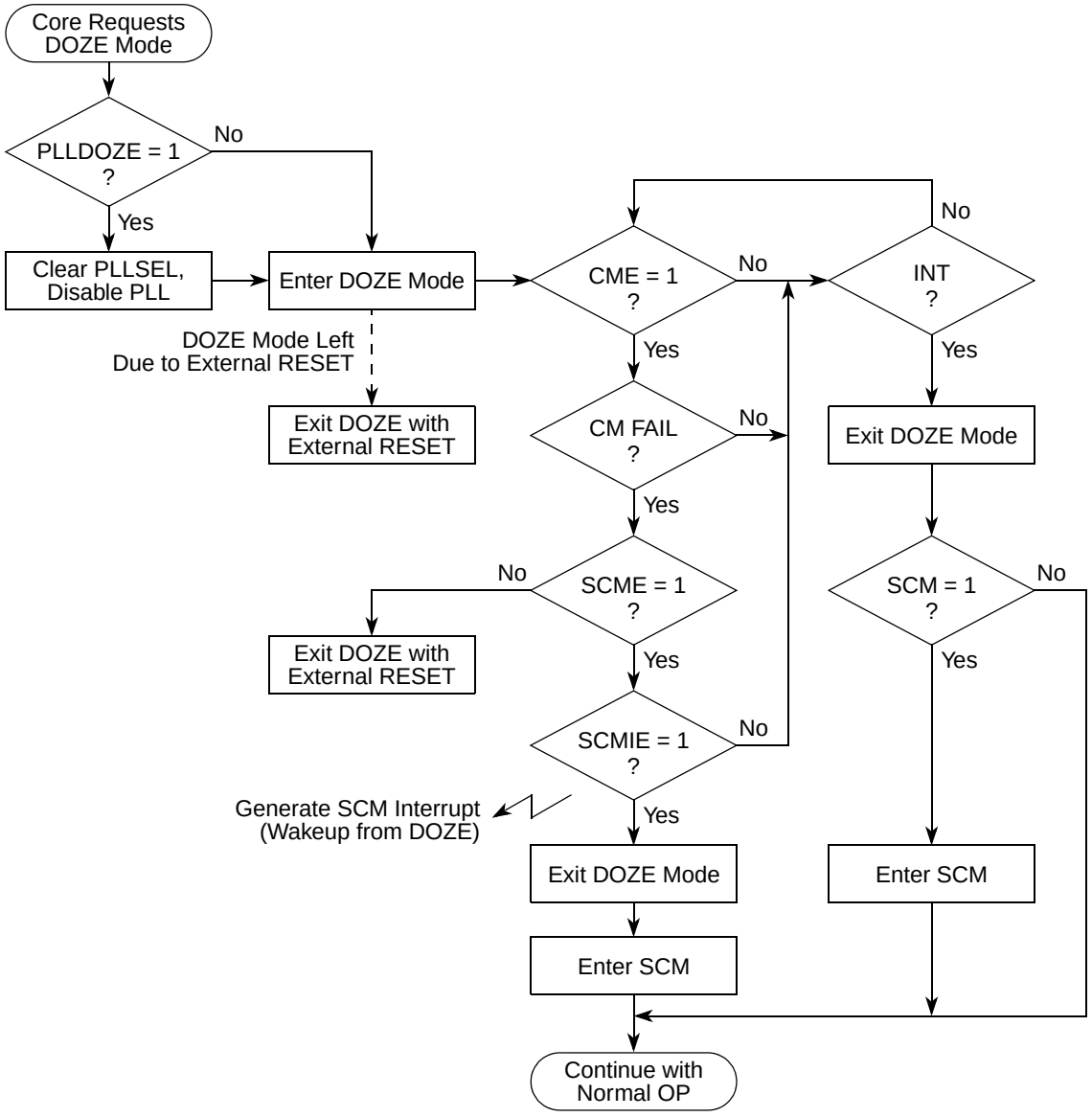


Figure 4-26. Doze Mode Entry/Exit Sequence

Table 4-13. Outcome of Clock Loss in Doze Mode

CME	SCME	SCMIE	CRG Actions
0	X	X	Clock failure → • No action, clock loss not detected.
1	0	X	Clock failure → • CRG performs clock monitor reset immediately
1	1	0	Clock failure → Scenario 1: OSCCLK **recovers** prior to exiting doze mode • MCU remains in doze mode • VREG enabled • PLL enabled • SCM activated • Start clock quality check • Set CRGFLG[SCMIF] interrupt flag Some time later OSCCLK recovers • CM no longer indicates a failure • 4096 OSCCLK cycles later clock quality check indicates clock is stable • SCM deactivated • PLL disabled depending on PLLDOZE • VREG remains enabled *(never gets disabled in doze mode)* • MCU remains in doze mode Some time later either a wakeup interrupt occurs (no SCM interrupt) • Exit doze mode using OSCCLK as system clock (f_{SYS}) • Continue normal operation or an External Reset is applied • Exit doze mode using OSCCLK as system clock (f_{SYS}) • Start reset sequence Scenario 2: OSCCLK **does not recover** prior to exiting doze mode • MCU remains in doze mode • VREG enabled • PLL enabled • SCM activated • Start clock quality check • Set CRGFLG[SCMIF] interrupt flag • Continue performing clock quality checks while in doze mode Some time later either a wakeup interrupt occurs (no SCM interrupt) • Exit doze mode in SCM using PLL clock (f_{SCM}) as system clock (f_{SYS}) • Continue to perform additional clock quality checks until OSCCLK is stable again or an external $\overline{\text{RESET}}$ is applied • Exit doze mode in SCM using PLL clock (f_{SCM}) as system clock (f_{SYS}) • Start reset sequence • Continue to perform additional clock quality checks until OSCCLK is stable again
1	1	1	Clock failure → • VREG enabled • PLL enabled • SCM activated • Start clock quality check • CRGFLG[SCMIF] set SCMIF generates self-clock mode wakeup interrupt • Exit doze mode in SCM using PLL clock (f_{SCM}) as system clock (f_{SYS}) • Continue to perform additional clock quality checks until OSCCLK is stable again

4.3.6.10.5 Stop Mode

All clocks are stopped in stop mode, depending of the setting of the PLLCTL[PWE], PLLCTL[PRE] and CLKSEL[PSTP] bits (PSTP = pseudo-stop mode, PWE = pseudo-stop watchdog enable, PRE = pseudo-stop RTI enable). The oscillator is disabled in stop mode unless the CLKSEL[PSTP] bit is set. All counters and dividers remain frozen, but do not initialize. If the PLLCTL[PRE] or PLLCTL[PWE] bits are set, the RTI or SWT continues to run in pseudo-stop mode. In addition to disabling system and peripheral clocks, the CRG requests other functional units of the MCU (e.g. voltage regulator) to enter their individual power saving modes (if available). This is the main difference between pseudo-stop mode and doze mode.

By writing the SDMCTL[STOP] bit, the core requests the CRG to switch the MCU into stop mode. If the CLKSEL[PLLSEL] bit is still set when entering stop mode, the CRG will switch the system and peripheral clocks to OSCCLK by clearing the CLKSEL[PLLSEL] bit. Then the CRG disables the PLL, disables the peripheral clock and finally disables the remaining system clocks. As soon as all clocks are switched off, stop mode is active.

If pseudo-stop mode is entered (CLKSEL[PSTP]=1) from self-clock mode, the CRG will continue to check the clock quality until clock check is successful. The PLL and the voltage regulator will remain enabled. If full-stop mode (PSTP=0) is entered from self-clock mode, an ongoing clock quality check will be stopped. A complete timeout window check will be started when stop mode is exited again.

Wake up from stop mode also depends on the setting of the CLKSEL[PSTP] bit.

When writing the SDMCTL[STOP] bit, the CRG will hold the bus until it is ready to turn off the clocks. After writing the STOP bit, no other accesses to peripherals are possible.

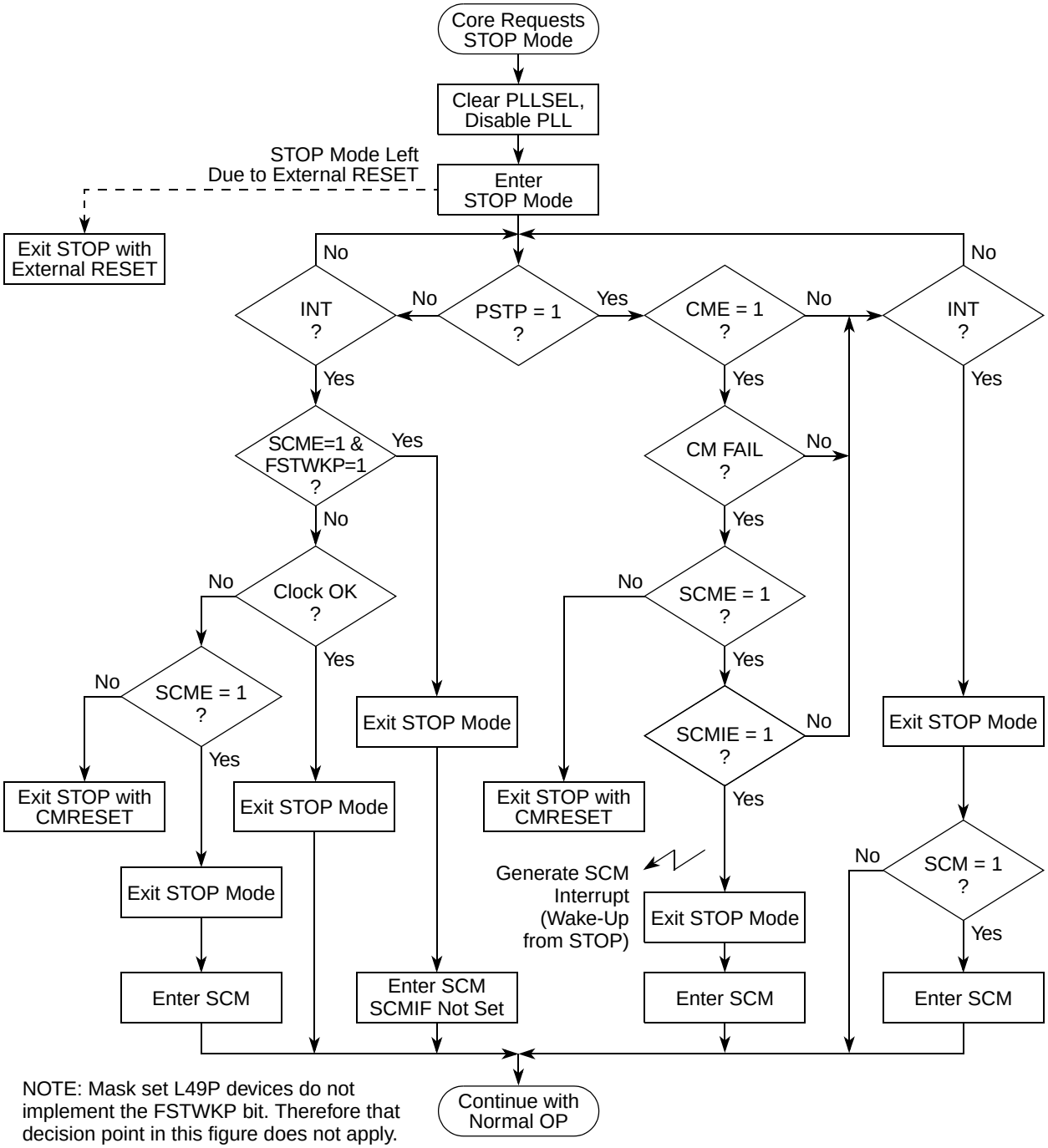


Figure 4-27. Stop Mode Entry/Exit Sequence

4.3.6.10.6 Wake-up from Pseudo-Stop

Wake-up from pseudo-stop (CLKSEL[PSTP]=1) is the same as wake-up from doze mode. There are also three different scenarios for the CRG to restart the MCU from pseudo-stop mode:

- External Reset
- Clock Monitor Fail
- Wake-up Interrupt

If the MCU gets an external reset during pseudo-stop mode, the CRG asynchronously restores all configuration bits in the register space to their default settings and starts the reset generator. After completing the reset sequence processing begins by fetching the normal reset vector. Pseudo-stop mode is exited and the MCU is in run mode again.

If the clock monitor is enabled (PLLCTL[CME]=1) the MCU is able to exit pseudo-stop mode when loss of oscillator/external clock is detected by a clock monitor fail. If the SCME bit is not asserted, the CRG generates a clock monitor fail reset (CMRESET). The CRG behavior for CMRESET is the same compared to external reset, but another reset vector is fetched after completion of the reset sequence. If the SCME bit is asserted, the CRG generates an SCM interrupt if enabled (CRGINT[SCMIE]=1). After generating the interrupt the CRG enters Self-Clock Mode and starts the clock quality checker (see Section 4.3.6.4, “Clock Quality Checker”). Then the MCU continues with normal operation. If the SCM interrupt is blocked by SCMIE=0, the CRGFLG[SCMIF] flag will be asserted, but the CRG will not wake-up from pseudo-stop mode.

If any other interrupt source (e.g. RTI) triggers exit from pseudo-stop mode, the MCU immediately continues with normal operation. Because the PLL has been powered-down during stop mode, the CLKSEL[PLLSEL] bit is cleared and the MCU runs on OSCCLK after leaving stop mode. The software must set the PLLSEL bit again, in order to switch system and peripheral clocks to the PLLCLK.

Table 4-14 summarizes the outcome of a clock loss while in pseudo-stop mode.

Table 4-14. Outcome of Clock Loss in Pseudo-Stop Mode

CME	SCME	SCMIE	CRG Actions
0	X	X	Clock failure → • No action, clock loss not detected
1	0	X	Clock failure → • CRG performs clock monitor reset immediately
1	1	0	Clock monitor failure → Scenario 1: OSCCLK **recovers** prior to exiting pseudo-stop mode • MCU remains in pseudo-stop mode • VREG enabled • PLL enabled • SCM activated • Start clock quality check • Set CRGFLG[SCMIF] interrupt flag Some time later OSCCLK recovers. • CM no longer indicates a failure • 4096 OSCCLK cycles later clock quality check indicates clock is stable • SCM deactivated • PLL disabled • VREG disabled • MCU remains in pseudo-stop mode Some time later either a wakeup interrupt occurs (no SCM interrupt). • Exit pseudo-stop mode using OSCCLK as system clock (f_{SYS}) • Continue normal operation or an External Reset is applied. • Exit pseudo-stop mode using OSCCLK as system clock (f_{SYS}) • Start reset sequence Scenario 2: OSCCLK **does not recover** prior to exiting pseudo-stop mode. • MCU remains in pseudo-stop mode • VREG enabled • PLL enabled • SCM activated • Start clock quality check • Set CRGFLG[SCMIF] interrupt flag • Continue performing clock quality checks while in pseudo-stop mode Some time later either a wakeup interrupt occurs (no SCM interrupt) • Exit pseudo-stop mode in SCM using PLL clock (f_{SCM}) as system clock (f_{SYS}) • Continue to perform additional clock quality checks until OSCCLK is stable again or an external $\overline{\text{RESET}}$ is applied. • Exit pseudo-stop mode in SCM using PLL clock (f_{SCM}) as system clock (f_{SYS}) • Start reset sequence • Continue to perform additional clock quality checks until OSCCLK is stable again
1	1	1	Clock failure → • VREG enabled • PLL enabled • SCM activated • Start clock quality check • CRGFLG[SCMIF] set SCMIF generates self-clock mode wakeup interrupt. • Exit pseudo-stop mode in SCM using PLL clock (f_{SCM}) as system clock (f_{SYS}) • Continue to perform a additional clock quality checks until OSCCLK is stable again

4.3.6.10.7 Wake-up from Full-Stop

The MCU requires an external interrupt or an external reset in order to wake-up from stop mode (CLKSEL[PSTP]=0).

If the MCU receives an external reset during full stop mode, the CRG asynchronously restores all configuration bits in the register space to their default settings and will perform a maximum of 50 clock check windows (see Section 4.3.6.4, “Clock Quality Checker”). After completing the clock quality check, the CRG starts the reset generator. After completing the reset sequence, processing begins by fetching the normal reset vector. Full-stop mode is exited and the MCU returns to run mode.

If the MCU receives interrupt request to initiate a wake-up and the fast wake-up feature [1] is disabled (FSTWKP=0 or SCME=0), the CRG will also perform a maximum of 50 clock check windows (see Section 4.3.6.4, “Clock Quality Checker”). If the clock quality check is successful, the CRG will release all system and peripheral clocks and will continue with normal operation. If all clock checks within the timeout window are failing, the CRG will switch to self-clock mode or generate a clock monitor reset (CMRESET), depending on the setting of the PLLCTL[SCME] bit.

When waking up from full stop mode by an interrupt with the fast wake-up feature [1] enabled (FSTWKP=1 and SCME=1), the system will immediately resume operation in Self-Clock Mode (see Section 4.3.6.4, “Clock Quality Checker”). The SCMIF flag will not be set. The system will remain in Self-Clock Mode with oscillator and clock monitor disabled until FSTWKP bit is cleared. Clearing of FSTWKP starts the oscillator, the clock monitor and the clock quality check. If the clock quality check is successful, the CRG will switch all system clocks to oscillator clock. The SCMIF flag will be set. See application examples in Figure 4-28 and Figure 4-29.

Because the PLL has been powered-down during stop mode, the CLKSEL[PLLSEL] bit is cleared and the MCU runs on OSCCLK after leaving stop mode. The software must manually set the PLLSEL bit again in order to switch system and peripheral clocks to the PLLCLK.

NOTE

In full-stop mode, the clock monitor is disabled and any loss of clock will not be detected.

1. The fast wake-up feature is not implemented on mask set L49P devices.

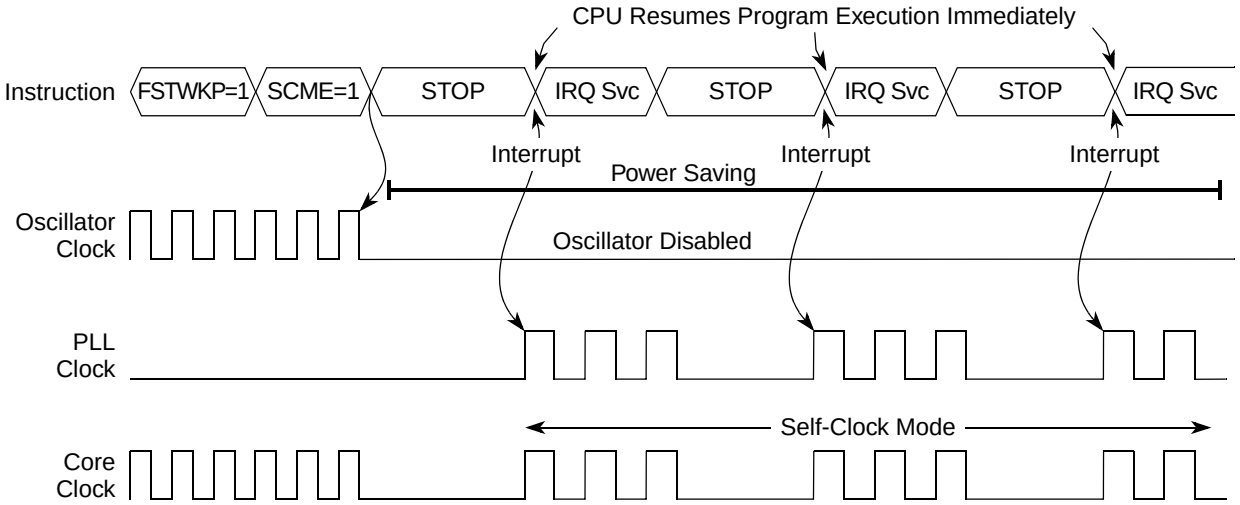


Figure 4-28. Fast Wake-up from Full Stop mode: Example 1

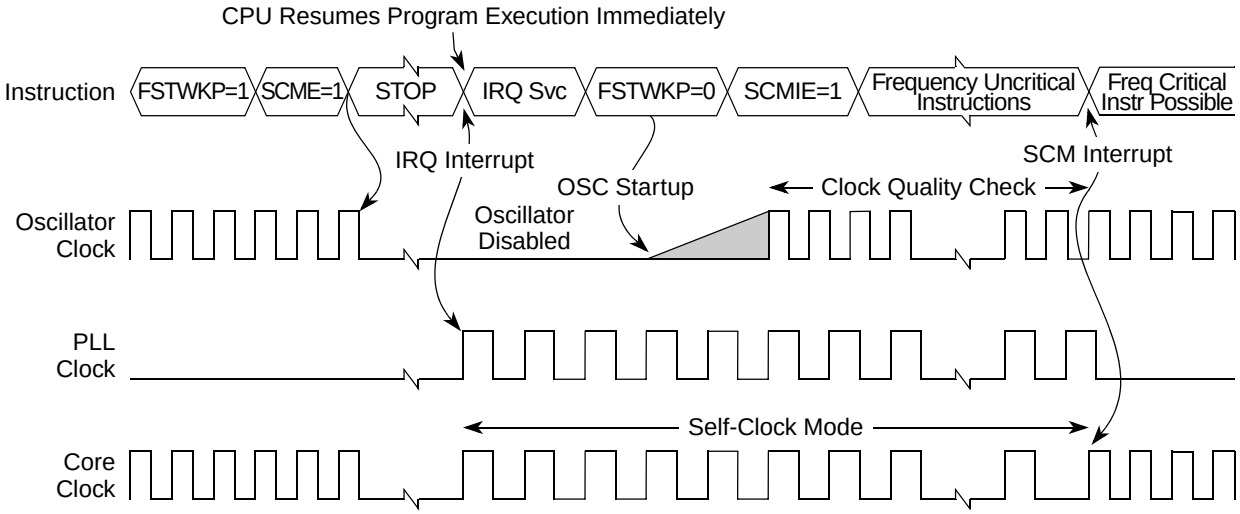


Figure 4-29. Fast Wake-up from Full Stop mode: Example 2

4.4 System Clocks Summary

Table 4-15. CRG Modes Active Device Clocks Summary

Mode	Oscillator Clock	PLL Clock	RTI Clock	Software Watchdog Timer Clock	System Clock (f_{SYS}), Peripheral Bus Clock (f_{IPS})	Enable or Disable Control Bits
Run	ON	ON	ON disable on debug	ON disable on debug	OSCCLK or PLLCLK	CLKSEL[PLLSEL] BDMCTL[RSBCK]
Doze	ON	ON disable	ON disable disable on debug	ON disable disable on debug	OSCCLK or PLLCLK	CLKSEL[PLLSEL] CLKSEL[PLLDOZE] CLKSEL[RTIDOZE] CLKSEL[SWTDOZE] BDMCTL[RSBCK]
Pseudo-stop	ON	OFF	OFF enable disable on debug	OFF enable disable on debug	OFF	PLLCTL[PRE] PLLCTL[PWE] BDMCTL[RSBCK]
Stop	OFF	OFF	OFF	OFF	OFF	

Table 4-16. CRG Modes Entry Sequences

Mode	Sequence
Stop	• SDMCTL[STOP] bit is set • CRG requests system stop • All peripherals acknowledge the stop signal • CRG asserts internal *crg_stop_mode* signal, turns off clocks
Pseudo-stop	• SDMCTL[STOP] bit is set while the CLKSEL[PSTP] bit is set • RTI and SWT clocks can be enabled and will then continue to run
Doze	• SDMCTL[DOZE] bit is set • CRG indicates doze mode • The peripheral clocks are turned off according to their DOZE bits • RTI and SWT clocks are turned off according to their DOZE bits

Chapter 5
Resets

This section is an overview of how reset operations affect the behavior of the modules detailed within this manual. Each module description section that follows has detailed information on the affect of reset on specific modules. Consult the additional documentation referenced in Chapter 9, “ARM7TDMI-S™ Processor Core,” for information on how resets affect the CPU core.

5.1 Effects of Reset

When a reset occurs, MCU registers and control bits are changed to known start-up states. Refer to the respective module sections for register Reset states. Following a reset all of the peripheral modules are disabled, with the exception of the External Interface Module (EIM) when in expanded unsecured mode. All modules must be enabled before they can be used.

5.1.1 I/O pins

Refer to Chapter 2, “Signal Description,” and Chapter 18, “Port Integration Module (PIM),” for mode dependent pin configuration out of reset.

5.1.2 Memory

Refer to Chapter 8, “Device Memory Map,” for locations of the memories depending on the operating mode after reset. The SRAM array is not automatically initialized out of reset.

5.2 Keyboard Wake-up on Port Pins

Ports A, B, C, D, E, F, G and H provide keyboard wake-up on all pins to enable the device to be returned from low power operating modes. Any of these pins can also be used to generate an interrupt to the core via the interrupt controller, using the PIM interrupt vector (refer to Table 6-2 on page 6-85).

Chapter 6
Exceptions

Consult the exception section of the *ARM Architectural Reference Manual* or the *ARM7TDMI-S Technical Reference Manual* for additional information on interrupts and exceptions.

6.1 Exception Vector Assignments

Table 6-1 and Table 6-2 lists exception and interrupt sources and their vectors in default order of priority.

Table 6-1. ARM7 Exception Table

Vector Address	Interrupt Source	Interrupt Type
0x0000 0000	RESET	RESET
0x0000 0004	UNDEFINED INSTRUCTION	Undef' Instr'
0x0000 0008	SWI	S/W Int
0x0000 000C	ABORT (Prefetch)	ABORT
0x0000 0010	ABORT (Data)	ABORT
0x0000 0018	IRQ	Normal Interrupt
0x0000 001C	FIQ	Fast Interrupt

Table 6-2. MAC7100 Family Interrupt Vector Assignments

Vector Number	Priority	Interrupt Source	Refer To
0x0000	64 (lowest)	eDMA0	Page 12-153
0x0001	63	eDMA1	Page 12-153
0x0002	62	eDMA2	Page 12-153
0x0003	61	eDMA3	Page 12-153
0x0004	60	eDMA4	Page 12-153
0x0005	59	eDMA5	Page 12-153
0x0006	58	eDMA6	Page 12-153
0x0007	57	eDMA7	Page 12-153
0x0008	56	eDMA8	Page 12-153
0x0009	55	eDMA9	Page 12-153
0x000A	54	eDMA10	Page 12-153
0x000B	53	eDMA11	Page 12-153
0x000C	52	eDMA12	Page 12-153
0x000D	51	eDMA13	Page 12-153
0x000E	50	eDMA14	Page 12-153
0x000F	49	eDMA15	Page 12-153
0x0010	48	eDMA Error	Page 12-154
0x0011	47	MCM SWT	Page 11-132
0x0012	46	CRG	Page 4-55
0x0013	45	PIT1	Page 25-553
0x0014	44	PIT2	Page 25-553
0x0015	43	PIT3	Page 25-553
0x0016	42	PIT4 / RTI [1]	Page 25-553
0x0017	41	VREG [2]	Page 3-33

Table 6-2. MAC7100 Family Interrupt Vector Assignments (continued)

Vector Number	Priority	Interrupt Source	Refer To
0x0018	40	CAN_A MB[31:15, 13:0]	Page 23-511
0x0019	39	CAN_A MB14	Page 23-511
0x001A	38	CAN_A Bus-off, Error / Wake-Up [3]	Page 23-507
0x001B	37	CAN_B MB[31:15, 13:0]	Page 23-511
0x001C	36	CAN_B MB14	Page 23-511
0x001D	35	CAN_B Bus-off, Error / Wake-Up [3]	Page 23-507
0x001E	34	CAN_C MB[31:15, 13:0]	Page 23-511
0x001F	33	CAN_C MB14	Page 23-511
0x0020	32	CAN_C Bus-off, Error / Wake-Up [3]	Page 23-507
0x0021	31	CAN_D MB[31:15, 13:0]	Page 23-511
0x0022	30	CAN_D MB14	Page 23-511
0x0023	29	CAN_D Bus-off, Error / Wake-Up [3]	Page 23-507
0x0024	28	I^2C	Page 24-534
0x0025	27	DSPI_A	Page 22-462
0x0026	26	DSPI_B	Page 22-462
0x0027	25	eSCI_A	Page 21-440
0x0028	24	eSCI_B	Page 21-440
0x0029	23	eSCI_C	Page 21-440
0x002A	22	eSCI_D	Page 21-440
0x002B	21	eMIOS0	Pages 20-355 and 20-363
0x002C	20	eMIOS1	Pages 20-355 and 20-363
0x002D	19	eMIOS2	Pages 20-355 and 20-363
0x002E	18	eMIOS3	Pages 20-355 and 20-363
0x002F	17	eMIOS4	Pages 20-355 and 20-363
0x0030	16	eMIOS5	Pages 20-355 and 20-363
0x0031	15	eMIOS6	Pages 20-355 and 20-363
0x0032	14	eMIOS7	Pages 20-355 and 20-363
0x0033	13	eMIOS8	Pages 20-355 and 20-363
0x0034	12	eMIOS9	Pages 20-355 and 20-363
0x0035	11	eMIOS10	Pages 20-355 and 20-363
0x0036	10	eMIOS11	Pages 20-355 and 20-363
0x0037	9	eMIOS12	Pages 20-355 and 20-363
0x0038	8	eMIOS13	Pages 20-355 and 20-363
0x0039	7	eMIOS14	Pages 20-355 and 20-363
0x003A	6	eMIOS15	Pages 20-355 and 20-363
0x003B	5	ATD_A, ATD_B	Page 19-326
0x003C	4	CFM	Page 15-223
0x003D	3	PIM	Pages 18-286, 18-287 and 18-289
0x003E	2	$\overline{\text{IRQ}}$	Pages 10-106 and 10-115
0x003F	1 (highest)	$\overline{\text{XIRQ}}$	Pages 10-106 and 10-115

1 For mask set L49P devices, the RTI wake-up request does not generate an interrupt via the INTC. For later mask set devices, the RTI wake-up shares the PIT Timer 1 interrupt vector.

2 Low-voltage interrupt only; note that the API does not produce an interrupt, but rather is used only to generate system wake-up events.

3 For mask set L49P devices, the FlexCAN wake-up requests do not generate interrupts via the INTC. For later mask set devices, they share the Bus-off/Error vector for each FlexCAN module.

Chapter 7
Modes of Operation

Devices in the MAC7100 family operate in several different modes, depending on the particular application and stage of development. In general, there are 6 different modes available for MAC7100 Family devices, which are determined by the MODA and MODB pins as well as the security state of the on-chip program Flash memory. The selection of a particular mode affects the following device characteristics:

- The memory map for the device. A detailed description of the memory map of the MCU in each chip mode can be found in Chapter 8, “Device Memory Map.”
- Which debug features are enabled or disabled
- Which security features are enabled or disabled

In addition to the six chip configuration modes, there are three low-power modes, stop, pseudo-stop and doze, which may be used in any chip configuration. The low-power modes are discussed further in Section 7.3, “Power Consumption Considerations.”

7.1 Chip Hardware Configuration Summary

During reset it is possible to configure several options of the system, including:

- MCU mode
- Oscillator type
- External bus interface attributes
- Nexus port attributes

Note that all hardware configuration is done during a reset operation. The values of the configuration pins described below are latched on the rising edge of the $\overline{\text{RESET}}$ signal, and they must be held stable throughout the assertion of $\overline{\text{RESET}}$. Reconfiguration of the characteristics listed above may be done only by resetting the device.

7.1.1 MCU Mode Selection

The chip operating mode is determined by the states of the MODA and MODB pins (PD[1:0]) at reset and the security status of the program Flash as shown in Table 7-1.

Table 7-1. MCU Mode Selection

Program Flash Secured?	MODA (PD1)	MODB (PD0)	Mode
No	0	0	Normal Single-Chip Mode
No	0	1	Normal Expanded Mode
No	1	0	Normal Data Flash Boot Mode
No	1	1	Reserved for future use
Yes	0	0	Secured Single-Chip Mode
Yes	0	1	Secured Expanded Mode
Yes	1	0	Secured Data Flash Boot Mode
Yes	1	1	Reserved for future use

7.1.1.1 Expanded Modes

In Expanded Mode, the MCU boots from an external source by accessing external memory and hardware across the external address and data bus. In this mode the program and data Flash arrays are also available if the program Flash is unsecured. If the device is secured, then neither the program or data Flash arrays may be accessed, except for the accesses necessary to unsecure the device. Refer to Section 8.1.3, “Normal Expanded Mode,” and Section 8.1.4, “Secured Expanded Mode,” for more details.

7.1.1.2 Single-Chip Modes

In Single Chip Mode, the system boots from the program Flash. If the device is secured, the external bus interface is unavailable. However, if the device is unsecured, the system may be reconfigured by software to enable the external bus. Refer to Section 8.1.1, “Normal Single-Chip Mode,” and Section 8.1.2, “Secured Single-Chip Mode,” for more details.

7.1.1.3 Data Flash Boot Mode

In data Flash boot mode, the memory map of the system is modified to mirror the location of the data Flash from its default reset to location $0000 0000. This remapping enables the device to boot from the data Flash. Refer to Section 8.1.5, “Normal/Secured Data Flash Boot Mode,” for more details.

7.1.2 Oscillator Type Selection

At reset the type of oscillator is selected using the $\overline{\text{XCLKS}}$ pin. Refer to Section 4.2, “On-Chip Oscillator (OSC) Module,” on page 4-47 for more details.

Table 7-2. Clock Selection based on $\overline{\text{XCLKS}}$

$\overline{\text{XCLKS}}$	Description
Negated (high)	Loop controlled, low power Pierce Oscillator
Asserted (low)	Full swing Pierce Oscillator or External Clock

7.1.3 External Bus Interface Configuration

If the device and chip mode support an external bus, then attributes of the global chip select may be configured via the PA14 and PA15 pins. Refer to Section 13.6.1.2, “Global Chip Select,” on page 13-191 for more information.

7.1.4 Nexus Port Configuration

If use of the Nexus port is required for debugging, it may be connected to either the Nexus primary port (PA[6:0]) or the secondary port (PE[6:0]). Please refer to Appendix A, “Debug Interface,”for more information on configuring the Nexus Port.

7.2 Security

MAC7100 Family devices implement a security feature that prevents the unauthorized read and write of the memory contents. This feature allows:

- Protection of the contents of program Flash,
- Protection of the contents of data Flash,
- Operation in single-chip mode,
- Operation from external memory with internal program Flash and data Flash disabled.

Programmers developing code for MAC7100 Family devices must be aware that part of the security mechanism is the responsibility of the user's code. An extreme example of defeating the security mechanism would be user code that dumps the contents of the internal Flash to external memory or an I/O port. However, the user may also wish to put a back door in the application code. For example, in order to update parameters stored in Flash, a routine might be developed to allow the download of a security key through the eSCI, allowing access to a programming routine.

7.2.1 Securing the Microcontroller

Once the user has written the contents of the program Flash and data Flash (if desired), the device can be secured by programming the security bits located in the Flash module CFM Security Register (CFMSR, refer to Figure 15-8 on page 15-217). These non-volatile bits will keep the part secured through resets and power down. When security is enabled both the program and data Flash memories are secured, it is not possible to configure security independently for these two memory types.

The Flash Memory Security Bits of the CFMSR contain the setup for the controlling access to the Flash memories and resides in an area of the Flash array.

For further information on the security of the program Flash or data Flash, consult Chapter 15, "Common Flash Module (CFM)."

7.2.2 Operation of the Secured Microcontroller

7.2.2.1 Secured Single-Chip Mode

Operating in single-chip mode with the non-volatile memory secured is the most common usage of a secured part. To software, the MCU appears to be identical in operation as if it were not secured. The only exception to this is that access to the device via the debug port is blocked in order to prevent access to the contents of the memory. Refer to Section 8.1.2, "Secured Single-Chip Mode," on page 8-95 and Section 15.4.2, "Flash Security Operation," on page 15-247 for more information.

7.2.2.2 Secured Expanded Mode

It is possible to access external memory space with a secured microcontroller. This is accomplished by resetting directly into expanded mode. With the MCU in secure mode the internal program Flash and data Flash are disabled. Refer to Section 8.1.4, "Secured Expanded Mode," on page 8-97 and Section 15.4.2, "Flash Security Operation," on page 15-247 for more information.

7.2.3 Unsecuring the Microcontroller

In order to unsecure a previously secured microcontroller, the internal program and data Flash must be erased. This is done through an external program in expanded mode or with a JTAG instruction.

Once the program Flash and the data Flash have been erased, the part can be reset into special single chip mode. This invokes the memory controller to verify that the internal program and data Flash have been erased. On completion of this verification, the user can erase and program the Flash Memory Security Bits to return the device to the unsecured state. This is generally done through the debug port, but the user could alternatively change to expanded mode (by writing the mode bits through the debug interface) and jumping to an external program (through debugger commands). Note that if the part goes through a reset condition before the security bits are reprogrammed to the unsecure state, the part will return from reset in secured mode. Refer to Section 15.4.2, “Flash Security Operation,” on page 15-247 for more information.

There are two methods to unsecure a device that has been secured:

1. By using a backdoor access key, and
2. by using a lockout recovery procedure.

7.2.3.1 Backdoor Access Key

By programming a user-defined backdoor access key into the Flash array before the device is secured, the user may unsecure the device at any time by writing the same backdoor access key to registers in the CFM. Once this operation is performed successfully, the device is in a “pseudo-secured” state in which the device is unsecured, but a reset will force the device back into the secured state. The device may be fully unsecured by programming the Flash security byte before resetting the device.

7.2.3.2 Lockout Recovery Procedure

If the backdoor access key is not programmed or has been lost, an alternative method exists for unsecuring the device. In this method, called lockout recovery, the internal program and data Flash arrays are first erased. This procedure can be performed either by writing to CFM registers, or via the JTAG interface of the device. Refer to Section 15.4.2, “Flash Security Operation,” on page 15-247 for more information.

7.3 Power Consumption Considerations

MAC7100 family devices feature four operating modes, three of which support various levels of reduced power consumption:

- Run mode, where all modules are clocked at the selected operating frequency.
- Doze mode, where individual peripheral modules may be configured to remain operational or enter a static state in order to tailor the performance-to-power consumption profile of the system.
- Stop mode, where the device is in the fully static, lowest power mode.
- Pseudo-stop mode, where only the RTI and/or SWT timers continue to run.

Consult the appropriate module sections for information on the behavior of each peripheral module in stop, pseudo-stop, and doze mode. For overall information on system low power modes, refer to Section 4.3.3, “CRG Modes of Operation,” on page 4-52.

7.3.1 Run Mode

Although this is not a low power mode, it is possible to reduce power consumption when operating in run mode. When the device is reset, all peripheral modules are placed in a static, disabled mode. In order to minimize power consumption in run mode, only those modules required for the application should be enabled, leaving those that are not required disabled and thus reducing power consumption.

7.3.2 Doze Mode

In doze mode the device can be configured to selectively halt the operation of individual peripherals. In this mode it is possible to maintain operation of the eDMA. For further power consumption savings, the RTI and SWT can be disabled. This mode is entered by setting the DOZE bit of the SDMCTL register in the CRG module. Wake up from this mode can be initiated via a reset, an RTI (if enabled), and SWT timeout (if enabled), a self-clock mode interrupt, a peripheral interrupt, an external interrupt or the core writing to the SDMCTL register.

7.3.3 Stop Mode

In stop mode all of the MCU peripherals and the core are shut down, all clocks are stopped and the oscillator is stopped. This puts the device into a fully static mode and offers the lowest power consumption. Stop mode is entered by setting the STOP bit of the SDMCTL register in the CRG module with the PSTP bit of the CLKSEL register clear. Wake up from this mode can be initiated via an external reset or external interrupts. On wake-up from stop mode execution is delayed until the oscillator has stabilized and the PLL has locked to the required frequency.

To enter and exit stop mode, the MCM wake-up control register (MWCR) ENBWCR bit must be set. Refer to Section 11.3.1.7, “MCM Wake-up Control Register (MWCR),” on page 11-128 for more information.

7.3.4 Pseudo-Stop Mode

In pseudo-stop mode the oscillator is continues to run and the Real Time Interrupt (RTI) or Software Watchdog Timer (SWT) may be enabled to remain active. It is also possible to allow the CAN modules to continue to monitor the activity on the bus. The main system clock (f_{SYS}) is stopped, which disables other peripherals and stops the core, and the voltage regulator power consumption is reduced. This mode consumes more current than the full stop mode, but the wake up time is significantly shorter as there is no need to delay for an oscillator restart. Pseudo-stop mode is entered by setting the STOP bit of the SDMCTL register in the CRG module with the PSTP bit of the CLKSEL register set. Wake up from this mode can be initiated via a reset, RTI, SWT, external interrupt, self-clock mode interrupt, or CAN bus activity interrupt.

7.4 Mode and Configuration Identification

Software can determine various characteristics of a device by reading registers contained in two peripheral modules. The Miscellaneous Control Module (MCM) contains several registers to allow identification of the device type and module configuration:

- MCM Processor Core Type Register (PCT)
- MCM Device Revision Register (REV)
- MCM XBS Master Configuration Register (AMC)
- MCM XBS Slave Configuration Register (ASC)
- MCM IPS On-Platform Module Configuration Register (IOPMC)

The chip mode and memory configuration can be determined via the System Services Module (SSM):

- SSM Current System Status Register (STATUS)
- SSM System Memory Configuration Register (MEMCONFIG)

Chapter 8
Device Memory Map

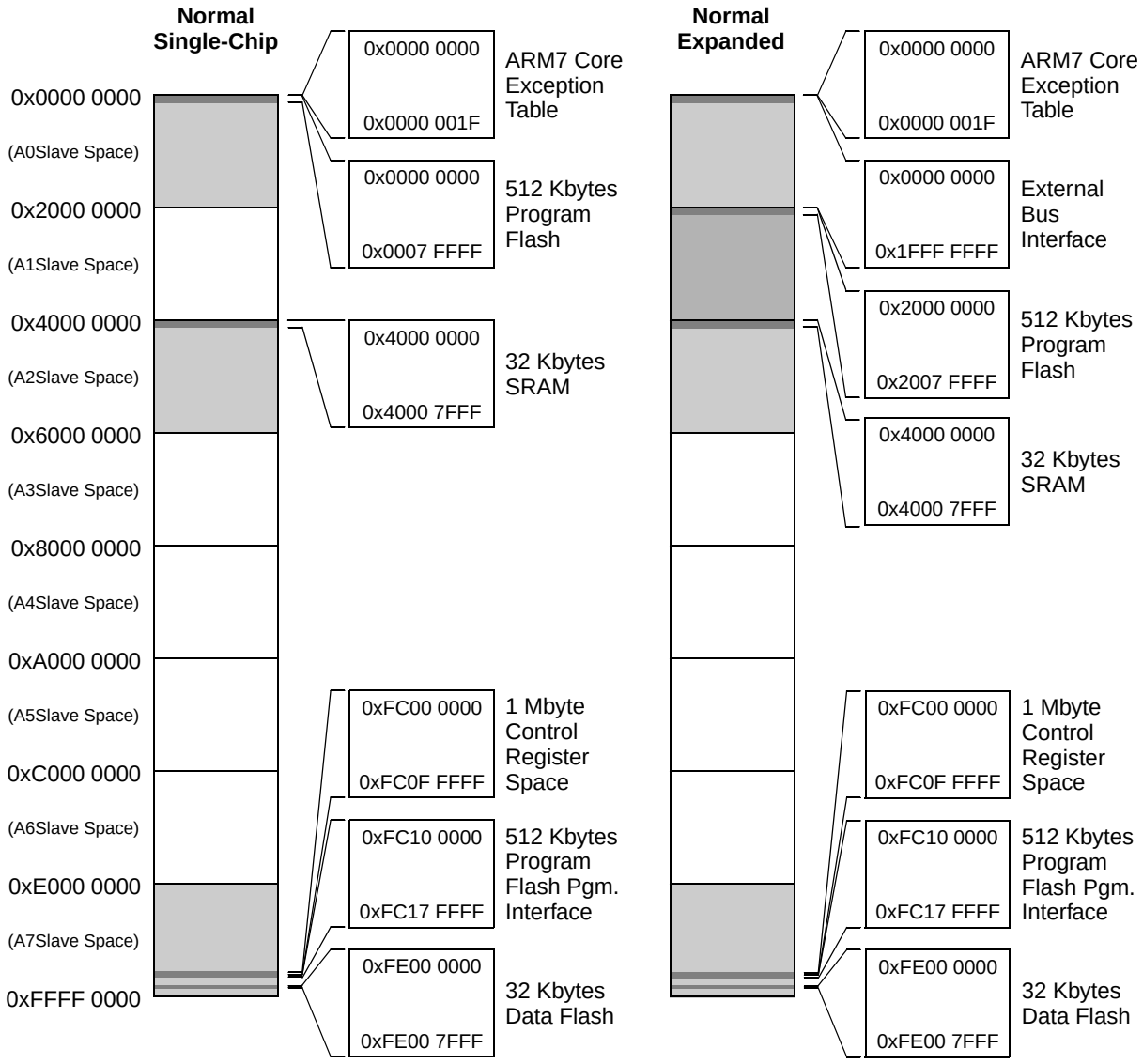
This section describes the memory map of MAC7100 Family devices after reset into various chip modes. For more details on selecting chip modes, refer to Chapter 7, "Modes of Operation." The memory map of the device immediately following reset is based on the mode of operation that has been entered, as summarized in Table 8-1 and detailed in Table 8-2 through Table 8-6. After reset, it is possible to relocate many of the blocks (such as the program Flash, SRAM and external bus interface) in the memory map, as described in Section 11.3.1.11, "MCM XBS Address Map Register (AAMR)," on page 11-132, although all mapping options are not available in all modes, as described later in this chapter. Note that the ARM7 core exception table is fixed at 0x0000 0000 – 0x0000 001F regardless of the chip mode and address map configuration. The peripheral control register space is fixed at $FC00 0000, and may not be remapped.

Table 8-1. Available Address Map Configurations

	Mode / AAMR Value	Base Address Mapping			
		0xE000 0000	0x4000 0000	0x2000 0000	0x0000 0000
Reset Configurations	Normal Single-Chip / 0xF000_0B98	Peripheral Bus	SRAM	External Bus	Program Flash
	Secured Single-Chip / 0xF000_0B08	Peripheral Bus	SRAM	—	Program Flash
	Normal Expanded / 0xF000_0B89	Peripheral Bus	SRAM	Program Flash	External Bus
	Secured Expanded / 0xF000_0B09	Peripheral Bus	SRAM	—	External Bus
	Normal Data Flash Boot / 0xF000_0B80	Peripheral Bus	SRAM	Program Flash	—
	Secured Data Flash Boot / 0xF000_0B80	Peripheral Bus	SRAM	Program Flash	—
Custom Configurations	0xF000_0B00	Peripheral Bus	SRAM	—	—
	0xF000_0B0B	Peripheral Bus	SRAM	—	SRAM
	0xF000_0B8B	Peripheral Bus	SRAM	Program Flash	SRAM
	0xF000_0B90	Peripheral Bus	SRAM	External Bus	—
	0xF000_0B99	Peripheral Bus	SRAM	External Bus	External Bus
	0xF000_0B9B	Peripheral Bus	SRAM	External Bus	SRAM
	0xF000_0BB0	Peripheral Bus	SRAM	SRAM	—
	0xF000_0BB8	Peripheral Bus	SRAM	SRAM	Program Flash
	0xF000_0BB9	Peripheral Bus	SRAM	SRAM	External Bus
	0xF000_0BBB	Peripheral Bus	SRAM	SRAM	SRAM

8.1 Memory Map Details

Figure 8-1 shows examples of the memory map configuration for MAC71*x*1 devices (512 Kbytes program Flash, 32 Kbytes SRAM) immediately after reset in two operating modes. The subsections below detail the reset memory map and available options for each chip operating mode.



Immediately after reset the map is:

0x0000 0000 – 0x0007 FFFF 512 K Flash
0x4000 0000 – 0x4000 7FFF 32 K RAM
0xFC00 0000 – 0xFC03 FFFF Register Space
0xFE00 0000 – 0xFE00 7FFF 32 K Data Flash

0x0000 0000 – 0x1FFF FFFF External Interface
0x2000 0000 – 0x2007 FFFF 512 K Flash
0x4000 0000 – 0x4000 7FFF 32 K RAM
0xFC00 0000 – 0xFC03 FFFF Register Space
0xFE00 0000 – 0xFE00 7FFF 32 K Data Flash

Figure 8-1. MAC71*x*1 Family Memory Map Examples

8.1.1 Normal Single-Chip Mode

Table 8-2 shows the memory map of MAC7100 Family devices in normal single-chip mode after reset. This mode is typically used to debug and develop application code where a software bootloader is not used. Booting is performed from the program Flash. In this mode, the following resources are available:

- Program and data Flash
- External bus interface (must be enabled via software)
- Full debug functionality

Table 8-2. Reset Memory Map in Normal Single-Chip Mode

Address	Module	Size (Bytes)
0x0000 0000 – 0x000*n* FFFF	Program Flash (*n* = 3, 7 or F for 256 K, 512 K and 1 M devices, respectively)	256 K, 512 K or 1 M
0x00*nn* 0000 – 0x1FFF FFFF	Reserved (*nn* = 04, 08 or 10 per Flash size)	~511 M
0x2000 0000 – 0x3FFF FFFF	External Bus Interface [1] (disabled at reset, must be enabled via software)	512 M
0x4000 0000 – 0x4000 *n*FFF	SRAM [2] (*n* = 3, 7 or C for 16 K, 32 K and 48 K devices, respectively)	16 K, 32 K or 48 K
0x4000 *n*000 – 0xFBFF FFFF	Reserved [2] (*n* = 4, 8 or D per Flash size)	~3008 M
0xFC00 0000 – 0xFFFF FFFF	Registers, programming interface and Data Flash (see Table 8-10 for details)	64 M

1 MAC7111, MAC7116, MAC7131 and MAC7136 only. This area must be treated as reserved for all other devices.

2 The SRAM is mirrored across the address range 0x4000 0000 – 0x4007 FFFF on modulo block size boundaries. Therefore, the first word in a 32 K SRAM may be read from or written to using addresses 0x4000 0000, 0x4000 8000, 0x4001 0000, etc.

After reset, the AAMR may be programmed with any value shown in Table 8-1 to reconfigure the memory map as required.

8.1.2 Secured Single-Chip Mode

Table 8-3 shows the memory map of MAC7100 Family devices in secured single-chip mode after reset. This mode is typically used to execute code in a final application, where a software bootloader is not required. The program Flash is used for boot code. Resource availability differs from normal single-chip mode as follows:

- Program and data Flash are available
- External bus interface is not available
- Debug functionality is limited to Flash lockout recovery

Table 8-3. Reset Memory Map in Secured Single-Chip Mode

Address	Module	Size (Bytes)
0x0000 0000 – 0x000*n* FFFF	Program Flash (*n* = 3, 7 or F for 256 K, 512 K and 1 M devices, respectively)	256 K, 512 K or 1 M
0x00*nn* 0000 – 0x3FFF FFFF	Reserved (*nn* = 04, 08 or 10 per Flash size)	~1023 M
0x4000 0000 – 0x4000 *n*FFF	SRAM [1] (*n* = 3, 7 or C for 16 K, 32 K and 48 K devices, respectively)	16 K, 32 K or 48 K
0x4000 *n*000 – 0xFBFF FFFF	Reserved [1] (*n* = 4, 8 or D per Flash size)	~3008 M
0xFC00 0000 – 0xFFFF FFFF	Registers, programming interface and Data Flash (see Table 8-10 for details)	64 M

1 The SRAM is mirrored across the address range 0x4000 0000 – 0x4007 FFFF on modulo block size boundaries. Therefore, the first word in a 32 K SRAM may be read from or written to using addresses 0x4000 0000, 0x4000 8000, 0x4001 0000, etc.

After reset, reprogramming of the AAMR to reconfigure the memory map is limited to the values shown in Table 8-4.

Table 8-4. Available Address Map Configurations in Secured Single-Chip Mode

	AAMR Value	Base Address Mapping			
		0xE000 0000	0x4000 0000	0x2000 0000	0x0000 0000
Reset	0xF000_0B08	Peripheral Bus	SRAM	—	Program Flash
Custom Configurations	0xF000_0B80	Peripheral Bus	SRAM	Program Flash	—
	0xF000_0B00	Peripheral Bus	SRAM	—	—
	0xF000_0B0B	Peripheral Bus	SRAM	—	SRAM
	0xF000_0BB0	Peripheral Bus	SRAM	SRAM	—
	0xF000_0B8B	Peripheral Bus	SRAM	Program Flash	SRAM
	0xF000_0BB8	Peripheral Bus	SRAM	SRAM	Program Flash
	0xF000_0BBB	Peripheral Bus	SRAM	SRAM	SRAM

8.1.3 Normal Expanded Mode

Table 8-5 shows the memory map of MAC7100 Family devices in normal expanded mode after reset (available on the MAC7111, MAC7116, MAC7131 and MAC7136 only). This mode is typically used to debug and develop application code using external memory. The external bus is used for boot code, and all functionality is available. The advantage of debugging code from an external RAM, versus the internal program Flash, is that software breakpoints may be inserted without the burden of re-programming the program Flash. In this mode, the following resources are available:

- Program and data Flash
- External bus interface
- Full debug functionality

Table 8-5. Reset Memory Map in Normal Expanded Mode

Address	Module	Size (Bytes)
0x0000 0000 – 0x1FFF FFFF	External Bus Interface [1]	512 M
0x2000 0000 – 0x200*n* FFFF	Program Flash (*n* = 3, 7 or F for 256 K, 512 K and 1 M devices, respectively)	256 K, 512 K or 1 M
0x00*nn* 0000 – 0x1FFF FFFF	Reserved (*nn* = 04, 08 or 10 per Flash size)	~511 M
0x4000 0000 – 0x4000 *n*FFF	SRAM [2] (*n* = 3, 7 or C for 16 K, 32 K and 48 K devices, respectively)	16 K, 32 K or 48 K
0x4000 *n*000 – 0xFBFF FFFF	Reserved [2] (*n* = 4, 8 or D per Flash size)	~3008 M
0xFC00 0000 – 0xFFFF FFFF	Registers, programming interface and Data Flash (see Table 8-10 for details)	64 M

1 MAC7111, MAC7116, MAC7131 and MAC7136 only. This area must be treated as reserved for all other devices.

2 The SRAM is mirrored across the address range 0x4000 0000 – 0x4007 FFFF on modulo block size boundaries. Therefore, the first word in a 32 K SRAM may be read from or written to using addresses 0x4000 0000, 0x4000 8000, 0x4001 0000, etc.

After reset, the AAMR may be programmed with any value shown in Table 8-1 to reconfigure the memory map as required.

8.1.4 Secured Expanded Mode

Table 8-6 shows the memory map of MAC7100 Family devices in secured expanded mode after reset (available on the MAC7111, MAC7116, MAC7131 and MAC7136 only). This mode is typically used to execute application code in the final application, where on-chip Flash is not required. Alternatively, it is also used to unsecure secured devices. Boot code is executed via the external bus, with limited device functionality. Resource availability differs from normal expanded mode as follows:

- Program and data Flash are not available
- External bus interface is available
- Debug functionality is limited to Flash lockout recovery

Table 8-6. Reset Memory Map in Secured Expanded Mode

Address	Module	Size (Bytes)
0x0000 0000 – 0x3FFF FFFF	External Bus Interface [1]	512 M
0x4000 0000 – 0x4000 *n*FFF	SRAM [2] (*n* = 3, 7 or C for 16 K, 32 K and 48 K devices, respectively)	16 K, 32 K or 48 K
0x4000 *n*000 – 0xFBFF FFFF	Reserved [2] (*n* = 4, 8 or D per Flash size)	~3008 M
0xFC00 0000 – 0xFFFF FFFF	Registers, programming interface and Data Flash (see Table 8-10 for details)	64 M

1 MAC7111, MAC7116, MAC7131 and MAC7136 only. This area must be treated as reserved for all other devices.

2 The SRAM is mirrored across the address range 0x4000 0000 – 0x4007 FFFF on modulo block size boundaries. Therefore, the first word in a 32 K SRAM may be read from or written to using addresses 0x4000 0000, 0x4000 8000, 0x4001 0000, etc.

After reset, reprogramming of the AAMR to reconfigure the memory map is limited to the values shown in Table 8-7.

Table 8-7. Available Address Map Configurations in Secured Expanded Mode

	AAMR Value	Base Address Mapping			
		0xE000 0000	0x4000 0000	0x2000 0000	0x0000 0000
Reset	0xF000_0B09	Peripheral Bus	SRAM	—	External Bus
Custom Configurations	0xF000_0B00	Peripheral Bus	SRAM	—	—
	0xF000_0B0B	Peripheral Bus	SRAM	—	SRAM
	0xF000_0B90	Peripheral Bus	SRAM	External Bus	—
	0xF000_0B99	Peripheral Bus	SRAM	External Bus	External Bus
	0xF000_0B9B	Peripheral Bus	SRAM	External Bus	SRAM
	0xF000_0BB0	Peripheral Bus	SRAM	SRAM	—
	0xF000_0BB9	Peripheral Bus	SRAM	SRAM	External Bus
	0xF000_0BBB	Peripheral Bus	SRAM	SRAM	SRAM

8.1.5 Normal/Secured Data Flash Boot Mode

Table 8-8 shows the memory map of MAC7100 Family devices in both normal and secured data Flash boot mode after reset. This mode is typically used to debug and develop application code, where a software bootloader is required. Typically the primary boot loader, executed from the data Flash, will relocate the SRAM to 0x0000 0000 (in order to support dynamic exception service routines), load a secondary boot

loader into SRAM, and pass control to that program. The secondary boot loader will then erase the Flash, download a new code image via the appropriate peripheral interface, and reprogram the Flash. Resource availability is as follows:

- Program and data Flash are available
- External bus interface is not available
- Full debug functionality if not secured, limited to Flash lockout recovery if secured

Table 8-8. Reset Memory Map in Normal/Secured Data Flash Boot Mode

Address	Module	Size (Bytes)
0x0000 0000 – 0x0000 7FFF	Data Flash (mirrored at 0xFE00 0000)	32 K
0x0008 0000 – 0x1FFF FFFF	Reserved	~511 M
0x2000 0000 – 0x200*n* FFFF	Program Flash (*n* = 3, 7 or F for 256 K, 512 K and 1 M devices, respectively)	256 K, 512 K or 1 M
0x20*nn* 0000 – 0x3FFF FFFF	Reserved (*nn* = 04, 08 or 10 per Flash size)	~511 M
0x4000 0000 – 0x4000 *n*FFF	SRAM [1] (*n* = 3, 7 or C for 16 K, 32 K and 48 K devices, respectively)	16 K, 32 K or 48 K
0x4000 *n*000 – 0xFBFF FFFF	Reserved [1] (*n* = 4, 8 or D per Flash size)	~3008 M
0xFC00 0000 – 0xFFFF FFFF	Registers, programming interface and Data Flash (see Table 8-10 for details)	64 M

1 The SRAM is mirrored across the address range 0x4000 0000 – 0x4007 FFFF on modulo block size boundaries. Therefore, the first word in a 32 K SRAM may be read from or written to using addresses 0x4000 0000, 0x4000 8000, 0x4001 0000, etc.

After reset, reprogramming of the AAMR to reconfigure the memory map is limited to the values shown in Table 8-9.

Table 8-9. Available Address Map Configurations in Data Flash Boot Modes

	AAMR Value	Base Address Mapping			
		0xE000 0000	0x4000 0000	0x2000 0000	0x0000 0000
Reset	0xF000_0B80	Peripheral Bus	SRAM	Program Flash	—
Custom Configurations	0xF000_0B00	Peripheral Bus	SRAM	—	—
	0xF000_0B08	Peripheral Bus	SRAM	—	Program Flash
	0xF000_0B0B	Peripheral Bus	SRAM	—	SRAM
	0xF000_0B8B	Peripheral Bus	SRAM	Program Flash	SRAM
	0xF000_0BB0	Peripheral Bus	SRAM	SRAM	—
	0xF000_0BB8	Peripheral Bus	SRAM	SRAM	Program Flash
	0xF000_0BBB	Peripheral Bus	SRAM	SRAM	SRAM

8.1.6 Peripheral Bus Memory Map

Control registers for peripheral modules use the same mapping in all modes, as shown in Table 8-10.

Table 8-10. Peripheral Bus Memory Map

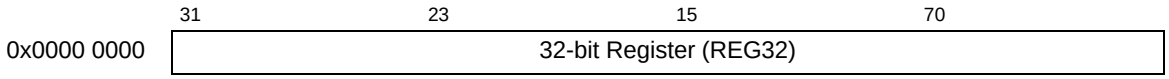
Address	Module	Size (Bytes)
0xFC00 0000 – 0xFC00 3FFF	AMBA to IP Bus Bridge (AIPS) Configuration Registers	16 K
0xFC00 4000 – 0xFC00 7FFF	Crossbar Bus Switch (XBS) Configuration Registers	16 K
0xFC00 8000 – 0xFC00 BFFF	External Interface Module (EIM) Configuration Registers	16 K
0xFC00 C000 – 0xFC03 FFFF	Reserved	208 K
0xFC04 0000 – 0xFC04 3FFF	Miscellaneous Control Module (MCM)	16 K
0xFC04 4000 – 0xFC04 7FFF	Enhanced Direct Memory Access (eDMA) Controller	16 K
0xFC04 8000 – 0xFC04 BFFF	Interrupt Controller (INTC)	16 K
0xFC04 C000 – 0xFC07 FFFF	Reserved	212 K
0xFC08 0000 – 0xFC08 3FFF	System Services Module (SSM)	16 K
0xFC08 4000 – 0xFC08 7FFF	Direct Memory Access Controller Multiplexer (DMA Mux)	16 K
0xFC08 8000 – 0xFC08 BFFF	Clock and Reset Generator (CRG)	16 K
0xFC08 C000 – 0xFC08 FFFF	Programmable Interval Timer (PIT)	16 K
0xFC09 0000 – 0xFC09 3FFF	Voltage Regulator (VREG)	16 K
0xFC09 4000 – 0xFC09 7FFF	CAN controller A (FlexCAN_A)	16 K
0xFC09 8000 – 0xFC09 BFFF	CAN controller B (FlexCAN_B)	16 K
0xFC09 C000 – 0xFC09 FFFF	CAN controller C (FlexCAN_C) [1]	16 K
0xFC0A 0000 – 0xFC0A 3FFF	CAN controller D (FlexCAN_D) [1]	16 K
0xFC0A 4000 – 0xFC0A BFFF	Reserved	32 K
0xFC0A C000 – 0xFC0A FFFF	Inter-IC bus (I^2C)	16 K
0xFC0B 0000 – 0xFC0B 3FFF	Reserved	16 K
0xFC0B 4000 – 0xFC0B 7FFF	Serial Peripheral Interface A (DSPI_A)	16 K
0xFC0B 8000 – 0xFC0B BFFF	Serial Peripheral Interface B (DSPI_B) [2]	16 K
0xFC0B C000 – 0xFC0C 3FFF	Reserved	32 K
0xFC0C 4000 – 0xFC0C 7FFF	Enhanced Serial Communication Interface A (eSCI_A)	16 K
0xFC0C 8000 – 0xFC0C BFFF	Enhanced Serial Communication Interface B (eSCI_B)	16 K
0xFC0C C000 – 0xFC0C FFFF	Enhanced Serial Communication Interface C (eSCI_C) [3]	16 K
0xFC0D 0000 – 0xFC0D 3FFF	Enhanced Serial Communication Interface D (eSCI_D) [3]	16 K
0xFC0D 4000 – 0xFC0D BFFF	Reserved	32 K
0xFC0D C000 – 0xFC0D FFFF	Enhanced Modular I/O Subsystem (eMIOS)	16 K
0xFC0E 0000 – 0xFC0E 3FFF	Analog-to-Digital Converter A (ATD_A)	16 K
0xFC0E 4000 – 0xFC0E 7FFF	Analog-to-Digital Converter B (ATD_B) [4]	16 K
0xFC0E 8000 – 0xFC0E BFFF	Port Integration Module (PIM)	16 K
0xFC0E C000 – 0xFC0F 0000	Reserved	16 K
0xFC0F 0000 – 0xFC0F 3FFF	Common Flash Module (CFM)	16 K
0xFC0F 4000 – 0xFC0F FFFF	Reserved	48 K
0xFC10 0000 – 0xFC1*n* FFFF	Program Flash EEPROM array (programming interface, *n* = 3, 7 or F per Flash size)	256 K, 512 K or 1 M
0xFC*nn* 0000 – 0xFDFF FFFF	Reserved (*nn* = 14, 18 or 20 per Flash size)	~31 M
0xFE00 0000 – 0xFE00 7FFF	Data Flash EEPROM array	32 K
0xFE00 8000 – 0xFFFF FFFF	Reserved	~32 M

1 For MAC7100 Family devices that do not implement CAN C or D, these memory map areas must be treated as reserved.
2 For MAC7100 Family devices that do not implement DSPI B, this memory map area must be treated as reserved.
3 For MAC7100 Family devices that do not implement eSCI C or D, these memory map areas must be treated as reserved.
4 For MAC7100 Family devices that do not implement ATD B, this memory map area must be treated as reserved.

8.2 Accessing Registers

All register descriptions in this manual use bit 31 to represent the most significant bit and bit 0 to represent the least significant bit. All memory-mapped registers may be accessed using properly aligned 8-, 16- or 32-bit accesses. Thus, all addresses may be accessed using 8-bit accesses, all even address may be accessed using 16-bit accesses, and every fourth address may be accessed using 32-bit accesses. Bus aborts may be enabled for all illegal register accesses, as described in Section 26.4.1.4, “SSM Error Configuration Register (ERROR),” on page 26-568. On MAC7100 family devices, which implement big-endian mode, this results in the following register mapping:

8.2.1 32-Bit Register Accesses

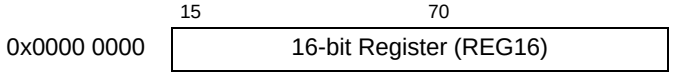


A read to the above register will result in the following values:

8-bit read from 0x0000 0000 returns REG32[31:24]
8-bit read from 0x0000 0001 returns REG32[23:16]
8-bit read from 0x0000 0002 returns REG32[15:8]
8-bit read from 0x0000 0003 returns REG32[7:0]
16-bit read from 0x0000 0000 returns REG32[31:16]
16-bit read from 0x0000 0002 returns REG32[15:0]
32-bit read from 0x0000 0000 returns REG32[31:0]

All other accesses (e.g., a 16-bit read from 0x0000 0001) are not allowed.

8.2.2 16-Bit Register Accesses

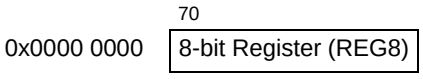


A read to the above register will result in the following values:

8-bit read from 0x0000 0000 returns REG16[15:8]
8-bit read from 0x0000 0001 returns REG16[7:0]
16-bit read from 0x0000 0000 returns REG16[15:0]
32-bit read from 0x0000 0000 returns REG16[15:0] + the next 16 bits in the memory map

All other accesses (e.g., a 16-bit read from 0x0000 0001) are not allowed.

8.2.3 8-Bit Register Accesses



A read to the above register will result in the following values:

8-bit read from 0x0000 0000 returns REG8[7:0]
16-bit read from 0x0000 0000 returns REG8[7:0] + the next 8 bits in the memory map
32-bit read from 0x0000 0000 returns REG8[7:0] + the next 24 bits in the memory map

All other accesses (e.g., a 16-bit read from 0x0000 0001) are not allowed.

Chapter 9
ARM7TDMI-S™ Processor Core

9.1 Overview

The MAC7100 Family is implemented with a licensed ARM7 processor core. This is a 32-bit RISC core with a three stage pipeline offering high instruction throughput. The core used across the family is the ARM7TDMI-S which supports both 32-bit and 16-bit (THUMB) instruction sets to allow code density optimization. No architectural modifications have been made to this core during implementation, enabling the MAC7100 Family to remain compliant to the ARM ISA V4T and existing tool chains. The core is configured to support big endian memory systems.

For detailed information about the ARM7 core please consult the following documents:

- *ARM Architecture Reference Manual* (ARM DDI-0100)
- *ARM7TDMI-S (Rev 4) Technical Reference Manual* (ARM DDI 0234A)

Note that mask sets L49P and L47W implement revision 4p2.04 of the core, while mask sets L61W and L38Y implement revision 4p3.04 of the core (see Table D-1 on page D-635 for mask set to part number correspondence). Refer to *ARM7TDMI-S Errata List* (FR002-PRDC-002719 3.0) from ARM for detailed information.

Chapter 10
Interrupt Controller Module (INTC)

10.1 Overview

MAC7100 Family devices implement one Interrupt Controller (INTC) Module which supports 64 interrupt requests. The interrupt controller is accessed via the peripheral bus for both configuration and for fetching the interrupt vector.

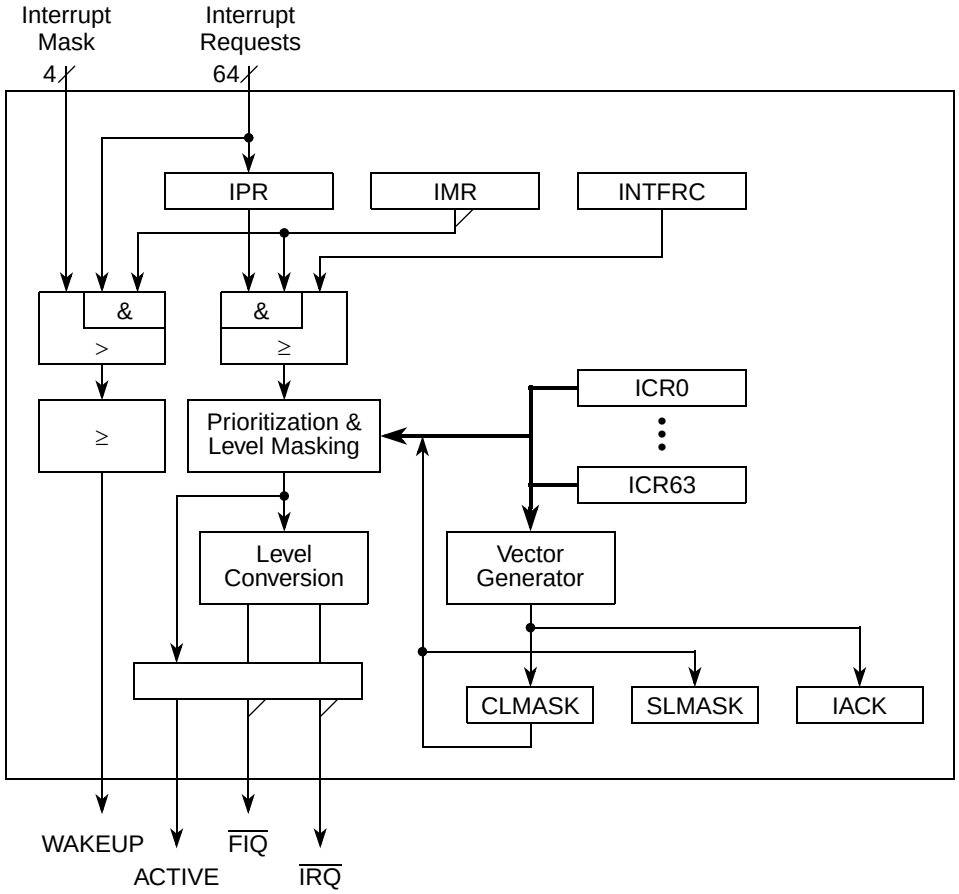


Figure 10-1. Interrupt Controller Block Diagram

The INTC is a highly-programmable controller, collecting interrupt requests, mapping the requests into 16 priority levels, and then signalling the ARM7 core when a properly enabled, non-masked request is active. In response to the service routine's memory mapped interrupt acknowledge read cycle, the INTC returns a unique vector for each interrupt request and automatically manages masking of lower level requests. As any interrupt source can be assigned to any of the sixteen priority levels, and the INTC enables the definition of which priority levels are assigned to fast or normal interrupts, each interrupt source can be selected to generate either a fast or normal interrupt to the core by its assigned level. The mapping of peripheral interrupt sources to interrupt vectors for MAC7100 Family devices is detailed in Table 6-2 on page 6-85.

10.2 Features

The INTC module supports the following features:

- 64 interrupt sources, organized as 16 programmable levels, with an arbitrary number of sources supported on any given level
- Each of the 64 sources has an interrupt control register (ICR*n*) to define the software-assigned levels
- Unique vector number for each interrupt source
- Optional hardware support for automatic level masking
- Ability to mask any individual interrupt source, plus global mask-all capability
- Support for service routine interrupt acknowledge (IACK) read cycles
- Combinatorial path to provide “wake-up” signal from low-power sleep modes
- Memory-mapped device connecting to the peripheral bus (IPS) interface

Figure 10-1 presents a simplified block diagram for the ARM7 interrupt controller.

10.3 Review of ARM7™ Interrupt Architecture

Before continuing with the specifics of the interrupt controller, a brief review of the interrupt architecture of the ARM7 core family is appropriate.

ARM7 cores support two direct interrupt request signals: an FIQ (fast interrupt request) and an IRQ (normal interrupt request). These two inputs are prioritized by the hardware with the FIQ being higher priority than the IRQ. Like most processor cores, interrupts are sampled once per instruction. If the FIQ input signal is asserted and enabled in the processor’s status register (CPSR), then the core suspends normal execution and initiates processing of a fast interrupt exception.

During exception processing, the contents of two key registers are copied into “shadow” or banked registers. Specifically, the current program counter (R15) is copied into the banked copy of the R14 link register named R14_fiq and the current processor status register (CPSR) value is copied into the saved processor status register (SPSR_fiq). Finally, a unique fast interrupt stack pointer, contained in R13_fiq, is activated and the core completes the exception processing by setting bits in the CPSR to disable both FIQ and IRQ interrupts, and then fetching the instruction at address 0x0000_001C. The processor continues execution in the FIQ mode, as indicated by the low-order 5 bits (the mode field) of the CPSR. While executing in this mode, banked copies of five additional general purpose registers (R[8,9,10,11,12]_fiq) are also available. The shadow copies of these registers are intended to minimize the need to save/restore the machine register state in memory using the fast interrupt stack pointer.

IRQ processing is similar with the following differences: in IRQ mode, shadow copies of only three registers are active: R13_irq (the interrupt request stack pointer), R14_irq (the PC of the interrupted instruction) and the SPSR_irq (the saved processor status register). The CPSR is configured to disable only IRQ interrupts, and the processor completes exception processing by fetching the instruction at address 0x0000_0018.

For all ARM7 cores, the processing of any exception forces exit from Thumb mode (if enabled), and the address loaded into the R14_{fiq,irq} link register is actually the program counter of the interrupted

instruction plus 4. Thus, this instruction address must be adjusted before returning to the interrupted instruction at the completion of the service routine.

A summary of the operations associated with interrupt exception processing is shown in Table 10-1:

Table 10-1. ARM7 Interrupt Exception Summary

Register	FIQ	IRQ
Interrupt Sample	Detect FIQ asserted	Detect FIQ negated, IRQ asserted
Link Register	R14_fiq = nextInst + 4	R14_irq = nextInst + 4
Saved Status Register	SPSR_fiq = CPSR	SPSR_irq = CPSR
Current Processor Status Register	CPSR[M] = `5'b10001`, CPSR[T] = `1'b0`, CPSR[F] = `1'b1`, CPSR[I] = `1'b1`	CPSR[M] = `5'b10010`, CPSR[T] = `1'b0`, CPSR[F] = unaffected, CPSR[I] = `1'b1`
Stack Pointer	Activate R13_fiq	Activate R13_irq
Other Banked Registers	Activate R[8-12]_fiq	None
Program Counter	PC = `0x0000_001C`	PC = `0x0000_0018`

10.4 Signal Description

10.4.1 $\overline{\text{XIRQ}}$

$\overline{\text{XIRQ}}$ provides an external interrupt input signal with the highest priority, as it is assigned to interrupt request 63. In order to be utilized as an interrupt pin, PD3 / $\overline{\text{XIRQ}}$ must be configured for peripheral mode via the Port Integration Module (PIM). Refer to Section 18.6.2, “Peripheral Mode,” on page 18-296 for more information.

10.4.2 $\overline{\text{IRQ}}$

$\overline{\text{IRQ}}$ provides an external interrupt input signal with priority higher than any on-chip peripheral but lower than $\overline{\text{XIRQ}}$, as it is assigned to interrupt request 62. In order to be utilized as an interrupt input signal, PD4 / $\overline{\text{IRQ}}$ must be configured for peripheral mode via the PIM. Refer to Section 18.6.2, “Peripheral Mode,” on page 18-296 for more information.

10.5 Memory Map / Register Definition

The register programming model for the INTC is memory-mapped to a 256-byte space within the addresses serviced by the AIPS bus bridge controller. In the following discussion, there are a number of program-visible registers greater than 32 bits in size. For these control fields, the physical register is partitioned into two 32-bit values: a register “high” (the upper word) and a register “low” (the lower word). The nomenclature <reg_name>H and <reg_name>L is used to reference these values.

On-chip interconnection of peripheral module interrupt request signals to the INTC module define the control register, pending flag, and mask bit assignment for each interrupt source. Table 10-2 below shows the assignment of peripheral interrupts to the INTC internal logic.

Table 10-2. INTC Interrupt Source-to-ICR*n* Assignments

Interrupt Source	Assigned ICR*n*	Interrupt Source	Assigned ICR*n*	Interrupt Source	Assigned ICR*n*	Interrupt Source	Assigned ICR*n*
eDMA0	0	eDMA Error	16	CAN_C Err./WU [2]	32	eMIOS5	48
eDMA1	1	MCM SWT	17	CAN_D MB	33	eMIOS6	49
eDMA2	2	CRG	18	CAN_D MB14	34	eMIOS7	50
eDMA3	3	PIT1	19	CAN_D Err./WU [2]	35	eMIOS8	51
eDMA4	4	PIT2	20	I^2C	36	eMIOS9	52
eDMA5	5	PIT3	21	DSPI_A	37	eMIOS10	53
eDMA6	6	PIT4 / RTI [1]	22	DSPI_B	38	eMIOS11	54
eDMA7	7	VREG	23	eSCI_A	39	eMIOS12	55
eDMA8	8	CAN_A MB	24	eSCI_B	40	eMIOS13	56
eDMA9	9	CAN_A MB14	25	eSCI_C	41	eMIOS14	57
eDMA10	10	CAN_A Err./WU [2]	26	eSCI_D	42	eMIOS15	58
eDMA11	11	CAN_B MB	27	eMIOS0	43	ATD_A, ATD_B	59
eDMA12	12	CAN_B MB14	28	eMIOS1	44	CFM	60
eDMA13	13	CAN_B Err./WU [2]	29	eMIOS2	45	PIM	61
eDMA14	14	CAN_C MB	30	eMIOS3	46	$\overline{\text{IRQ}}$	62
eDMA15	15	CAN_C MB14	31	eMIOS4	47	$\overline{\text{XIRQ}}$	63

1 On mask set L49P devices, the RTI wake-up request is not handled by the INTC (refer to Section 25.5.4.2.1).

2 On mask set L49P devices, the CAN wake-up requests are not handled by the INTC (refer to Section 23.6.8.4).

The INTC registers and their locations are defined in Table 10-3. Attempted accesses to the reserved locations are terminated with an error.

Table 10-3. INTC Memory Map

INTC Offset	Register Description			
0x0000	INTC Interrupt Pending Register High (IPRH)			
0x0004	INTC Interrupt Pending Register Low (IPRL)			
0x0008	INTC Interrupt Mask Register High (IMRH)			
0x000C	INTC Interrupt Mask Register Low (IMRL)			
0x0010	INTC Force Interrupt Register High (INTFRCH)			
0x0014	INTC Force Interrupt Register Low (INTFRCL)			
0x0018	Reserved			INTC Interrupt Configuration Register (ICONFIG)
0x001C	INTC Set Interrupt Mask Register (SIMR)	INTC Clear Interrupt Mask Register (CIMR)	INTC Current Level Mask Register (CLMASK)	INTC Saved Level Mask Register (SLMASK)

Table 10-3. INTC Memory Map (continued)

INTC Offset	Register Description			
0x0020–0x003F	Reserved			
0x0040	INTC Interrupt Control Register 0 (ICR1)	INTC Interrupt Control Register 1 (ICR1)	INTC Interrupt Control Register 2 (ICR2)	INTC Interrupt Control Register 3 (ICR3)
0x0044	INTC Interrupt Control Register 4 (ICR4)	INTC Interrupt Control Register 5 (ICR5)	INTC Interrupt Control Register 6 (ICR6)	INTC Interrupt Control Register 7 (ICR7)
0x0048	INTC Interrupt Control Register 8 (ICR8)	INTC Interrupt Control Register 9 (ICR9)	INTC Interrupt Control Register 10 (ICR10)	INTC Interrupt Control Register 11 (ICR11)
0x004C	INTC Interrupt Control Register 12 (ICR12)	INTC Interrupt Control Register 13 (ICR13)	INTC Interrupt Control Register 14 (ICR14)	INTC Interrupt Control Register 15 (ICR15)
0x0050	INTC Interrupt Control Register 16 (ICR16)	INTC Interrupt Control Register 17 (ICR17)	INTC Interrupt Control Register 18 (ICR18)	INTC Interrupt Control Register 19 (ICR19)
0x0054	INTC Interrupt Control Register 20 (ICR20)	INTC Interrupt Control Register 21 (ICR21)	INTC Interrupt Control Register 22 (ICR22)	INTC Interrupt Control Register 23 (ICR23)
0x0058	INTC Interrupt Control Register 24 (ICR24)	INTC Interrupt Control Register 25 (ICR25)	INTC Interrupt Control Register 26 (ICR26)	INTC Interrupt Control Register 27 (ICR27)
0x005C	INTC Interrupt Control Register 28 (ICR28)	INTC Interrupt Control Register 29 (ICR29)	INTC Interrupt Control Register 30 (ICR30)	INTC Interrupt Control Register 31 (ICR31)
0x0060	INTC Interrupt Control Register 32 (ICR32)	INTC Interrupt Control Register 33 (ICR33)	INTC Interrupt Control Register 34 (ICR34)	INTC Interrupt Control Register 35 (ICR35)
0x0064	INTC Interrupt Control Register 36 (ICR36)	INTC Interrupt Control Register 37 (ICR37)	INTC Interrupt Control Register 38 (ICR38)	INTC Interrupt Control Register 39 (ICR39)
0x0068	INTC Interrupt Control Register 40 (ICR40)	INTC Interrupt Control Register 41 (ICR41)	INTC Interrupt Control Register 42 (ICR42)	INTC Interrupt Control Register 43 (ICR43)
0x006C	INTC Interrupt Control Register 44 (ICR44)	INTC Interrupt Control Register 45 (ICR45)	INTC Interrupt Control Register 46 (ICR46)	INTC Interrupt Control Register 47 (ICR47)
0x0070	INTC Interrupt Control Register 48 (ICR48)	INTC Interrupt Control Register 49 (ICR49)	INTC Interrupt Control Register 50 (ICR50)	INTC Interrupt Control Register 51 (ICR51)
0x0074	INTC Interrupt Control Register 52 (ICR52)	INTC Interrupt Control Register 53 (ICR53)	INTC Interrupt Control Register 54 (ICR54)	INTC Interrupt Control Register 55 (ICR55)
0x0078	INTC Interrupt Control Register 56 (ICR56)	INTC Interrupt Control Register 57 (ICR57)	INTC Interrupt Control Register 58 (ICR58)	INTC Interrupt Control Register 59 (ICR59)
0x007C	INTC Interrupt Control Register 60 (ICR60)	INTC Interrupt Control Register 61 (ICR61)	INTC Interrupt Control Register 62 (ICR62)	INTC Interrupt Control Register 63 (ICR63)
0x0080–0x00EB	Reserved			
0x00EC	INTC IRQ Acknowledge Register (IRQIACK)	Reserved		
0x00F0	INTC FIQ Acknowledge Register (FIQIACK)	Reserved		
0x00F4–0x00FF	Reserved			

10.5.1 Register Descriptions

10.5.1.1 INTC Interrupt Pending Register (IPRH, IPRL)

The IPRH and IPRL registers are each 32 bits in size, and provide a bit map for each interrupt request to indicate if there is an active request (1 = active request, 0 = no request) for the given source. The state of the Interrupt Mask Register does not affect the IPR. The IPR is cleared by reset and updated each platform clock cycle. The IPR is a read-only register, so any attempted write to this register is terminated with an error.

Each bit of the IPR*n* is mapped to the corresponding input signal ipi_int[n], i.e., IPR[63:0] is mapped to ipi_int[63:0].

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IPR63	IPR62	IPR61	IPR60	IPR59	IPR58	IPR57	IPR56	IPR55	IPR54	IPR53	IPR52	IPR51	IPR50	IPR49	IPR48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	INTC Base + 0x0000															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IPR47	IPR46	IPR45	IPR44	IPR43	IPR42	IPR41	IPR40	IPR39	IPR38	IPR37	IPR36	IPR35	IPR34	IPR33	IPR32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	INTC Base + 0x0000															

Figure 10-2. INTC Interrupt Pending Register High (IPRH)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IPR31	IPR30	IPR29	IPR28	IPR27	IPR26	IPR25	IPR24	IPR23	IPR22	IPR21	IPR20	IPR19	IPR18	IPR17	IPR16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	INTC Base + 0x0004															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IPR15	IPR14	IPR13	IPR12	IPR11	IPR10	IPR9	IPR8	IPR7	IPR6	IPR5	IPR4	IPR3	IPR2	IPR1	IPR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	INTC Base + 0x0004															

Figure 10-3. INTC Interrupt Pending Register Low (IPRL)

Table 10-4. IPRH/IPRL Field Descriptions

Bits	Name	Description
31–0	IPR*n*	Interrupt pending register *n.* 0 The interrupt request n is negated. 1 The interrupt request n is asserted.

10.5.1.2 INTC Interrupt Mask Register (IMRH, IMRL)

The IMRH and IMRL registers are each 32 bits in size, and provide a bit map for each interrupt to allow the request to be disabled or “masked” (1 = disable the request, 0 = enable the request). The IMR is set to all ones by reset, disabling all interrupt requests. The IMR can be read and written directly, or individual mask flags can be set or cleared by accesses through the SIMR (Set Interrupt Mask) or CIMR (Clear Interrupt Mask).

Each bit of the IMR{H, L} is associated with the corresponding bit of the IPR*n*.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IMR63	IMR62	IMR61	IMR60	IMR59	IMR58	IMR57	IMR56	IMR55	IMR54	IMR53	IMR52	IMR51	IMR50	IMR49	IMR48
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reg Addr	INTC Base + 0x0008															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR47	IMR46	IMR45	IMR44	IMR43	IMR42	IMR41	IMR40	IMR39	IMR38	IMR37	IMR36	IMR35	IMR34	IMR33	IMR32
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reg Addr	INTC Base + 0x0008															

Figure 10-4. INTC Interrupt Mask Register High (IMRH)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IMR31	IMR30	IMR29	IMR28	IMR27	IMR26	IMR25	IMR24	IMR23	IMR22	IMR21	IMR20	IMR19	IMR18	IMR17	IMR16
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reg Addr	INTC Base + 0x000C															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR15	IMR14	IMR13	IMR12	IMR11	IMR10	IMR9	IMR8	IMR7	IMR6	IMR5	IMR4	IMR3	IMR2	IMR1	IMR0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reg Addr	INTC Base + 0x000C															

Figure 10-5. INTC Interrupt Mask Register Low (IMRL)

Table 10-5. IMRH/IMRL Field Descriptions

Bits	Name	Description
31–0	IMR*n*	Interrupt mask register *n.* 0 The interrupt request n is enabled. 1 The interrupt request n is disabled, i.e., masked.

10.5.1.3 INTC Force Interrupt Register (INTFRCH, INTFRCL)

The INTFRCH and INTFRCL registers are 32 bits wide, and provide a mechanism to allow software generation of interrupts for each source for functional or debug purposes. System design may reserve sources to allow software to schedule interrupts by setting one or more bits in the appropriate INTFRC register. In some cases, the handling of an interrupt request may cause critical processing by the service routine along with the scheduling (using the INTFRC register) of a lower-priority interrupt request to be processed later for less-critical tasks. The assertion of an interrupt request via INTFRC is not affected by IMRH/L registers.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IFRC63	IFRC62	IFRC61	IFRC60	IFRC59	IFRC58	IFRC57	IFRC56	IFRC55	IFRC54	IFRC53	IFRC52	IFRC51	IFRC50	IFRC49	IFRC48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	INTC Base + 0x0010															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IFRC47	IFRC46	IFRC45	IFRC44	IFRC43	IFRC42	IFRC41	IFRC40	IFRC39	IFRC38	IFRC37	IFRC36	IFRC35	IFRC34	IFRC33	IFRC32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	INTC Base + 0x0010															

Figure 10-6. INTC Force Interrupt Register High (INTFRCH)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IFRC31	IFRC30	IFRC29	IFRC28	IFRC27	IFRC26	IFRC25	IFRC24	IFRC23	IFRC22	IFRC21	IFRC20	IFRC19	IFRC18	IFRC17	IFRC16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	INTC Base + 0x0014															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IFRC15	IFRC14	IFRC13	IFRC12	IFRC11	IFRC10	IFRC9	IFRC8	IFRC7	IFRC6	IFRC5	IFRC4	IFRC3	IFRC2	IFRC1	IFRC0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	INTC Base + 0x0014															

Figure 10-7. INTC Force Interrupt Register Low (INTFRCL)

Table 10-6. INTFRCH/INTFRCL Field Descriptions

Bits	Name	Description
31–0	IFRC*n*	Interrupt force register *n.* 0 The forced interrupt request *n* is disabled. 1 A forced interrupt request *n* is enabled.

10.5.1.4 INTC Module Configuration Register (ICONFIG)

The ICONFIG defines the operating configuration for the INTC module.

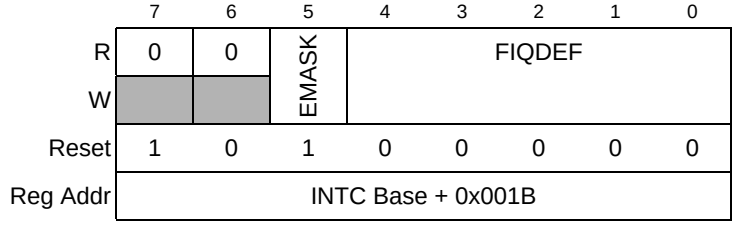


Figure 10-8. INTC Interrupt Configuration Register (ICONFIG)

Table 10-7. ICONFIG Field Descriptions

Bits	Name	Description
7:6	—	Reserved.
5	EMASK	Enable hardware level masking. If set, the INTC automatically loads the level of an interrupt request into the CLMASK (current level mask) when an acknowledge is performed. At the exact same cycle, the value of the current interrupt level mask is saved in the SLMASK (saved level mask) register. This feature can be used to support software-managed nested interrupts. The value of the SLMASK register should be read from the INTC and saved in the interrupt stack frame in memory, and restored near the service routine's exit. If cleared, the INTC does not perform any automatic masking of interrupt levels.
4–0	FIQDEF[4:0]	FIQ interrupt level definition. This 5-bit field defines the mapping of the 16 interrupt levels into the FIQ output signal. The field is defined as: 0x00 Levels 0–15 are mapped as FIQs 0x01 Levels 1–15 are mapped as FIQs 0x02 Levels 2–15 are mapped as FIQs 0x03 Levels 3–15 are mapped as FIQs 0x04 Levels 4–15 are mapped as FIQs 0x05 Levels 5–15 are mapped as FIQs 0x06 Levels 6–15 are mapped as FIQs 0x07 Levels 7–15 are mapped as FIQs 0x08 Levels 8–15 are mapped as FIQs 0x09 Levels 9–15 are mapped as FIQs 0x0A Levels 10–15 are mapped as FIQs 0x0B Levels 11–15 are mapped as FIQs 0x0C Levels 12–15 are mapped as FIQs 0x0D Levels 13–15 are mapped as FIQs 0x0E Levels 14–15 are mapped as FIQs 0x0F Level 15 is mapped as FIQ 0x10–0x1F No levels are mapped as FIQs; all are IRQs

10.5.1.5 INTC Set Interrupt Mask Register (SIMR)

The SIMR provides a simple memory-mapped mechanism to set a given bit in the IMR{H, L} registers to disable (“mask”) a given interrupt request. The data value on a register write causes the corresponding bit in the IMR{H, L} register to be set. A data value greater than 63 provides a global set function, forcing the entire contents of IMR{H, L} to be asserted, masking all interrupts. Reads of this register return all zeroes.

This register is provided so interrupt service routines can easily mask the given interrupt request without the need to perform a read-modify-write sequence on the IMR{H, L}.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W		SIMR						
Reset	—	0	0	0	0	0	0	0
Reg Addr	INTC Base + 0x001C							

Figure 10-9. INTC Set Interrupt Mask Register (SIMR)

Table 10-8. SIMR Field Descriptions

Bits	Name	Description
7	—	Reserved.
6–0	SIMR[6:0]	Set interrupt mask 0x00–0x3F Set the corresponding bit in IMR{H, L}, masking the interrupt request 0x40–0x7F Set all bits in IMR{H, L}, masking all interrupt requests

10.5.1.6 INTC Clear Interrupt Mask Register (CIMR)

The CIMR provides a simple memory-mapped mechanism to clear a given bit in the IMR{H,L} registers to enable a given interrupt request. The data value on a register write causes the corresponding bit in the IMR{H,L} register to be cleared. A data value greater than 63 provides a global clear function, forcing the entire contents of IMR{H,L} to be cleared, enabling all interrupts. Reads of this register return all zeroes.

This register is provided so interrupt service routines can easily enable the given interrupt request *without* the need to perform a read-modify-write sequence on the IMR{H,L}.

In the event of a simultaneous write to both the CIMR and SIMR, the SIMR has priority and the resulting function would be a *set* of the interrupt mask register.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W		CIMR						
Reset	—	0	0	0	0	0	0	0
Reg Addr	INTC Base + 0x001D							

Figure 10-10. INTC Clear Interrupt Mask Register (CIMR)

Table 10-9. CIMR Field Descriptions

Bits	Name	Description
7	—	Reserved.
6–0	CIMR[6:0]	Clear interrupt mask. 0x00–0x3F Clear the corresponding bit in IMR{H,L}, enabling the interrupt request 0x40–0x7F Clear all bits in IMR{H,L}, enabling all interrupt requests

10.5.1.7 INTC Current Level Mask Register (CLMASK)

The CLMASK register is provided so the INTC can automatically manage masking of interrupt requests based on the programmed priority level. If enabled by ICONFIG[EMASK] being set, an interrupt acknowledge read cycle returns a vector number identifying the physical request source, and the CLMASK register is loaded with the level number associated with the request. Once the CLMASK register is updated, then all interrupt requests with level numbers equal to or less than this value are masked by the controller and are not allowed to cause the assertion of the interrupt signal to the processor core. As the CLMASK register is updated during the IACK cycle read, the former value is saved in the SLMASK register. Typically, once a level-n interrupt request is handled, the service routine restores the saved level mask value into the current level mask register to re-enable the lower priority requests. In addition, an interrupt service routine can explicitly load this register with a lower priority value to query for any pending interrupts via software interrupt acknowledge cycles. This topic is covered in more detail in Section 10.7.2, “Interrupt Service Routines.”

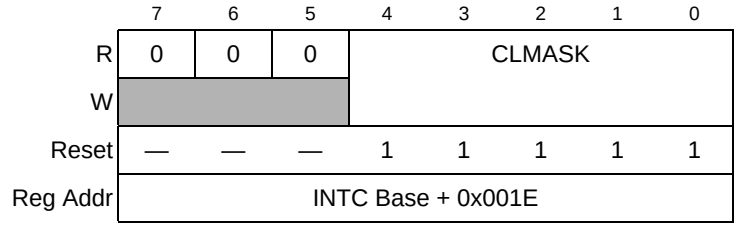


Figure 10-11. INTC Current Level Mask Register (CLMASK)

Table 10-10. CLMASK Field Descriptions

Bits	Name	Description
7–5	—	Reserved.

Table 10-10. CLMASK Field Descriptions (continued)

Bits	Name	Description
4–0	CLMASK[4:0]	Current level mask. This 5-bit field is treated as a signed integer within the range [–1, 0, 1, ..., 15]. The value defines the level mask, where only interrupt levels greater than the current value are processed by the controller. 0x1F Level 00–15 requests are processed 0x00 Level 01–15 requests are processed 0x01 Level 02–15 requests are processed 0x02 Level 03–15 requests are processed 0x03 Level 04–15 requests are processed 0x04 Level 05–15 requests are processed 0x05 Level 06–15 requests are processed 0x06 Level 07–15 requests are processed 0x07 Level 08–15 requests are processed 0x08 Level 09–15 requests are processed 0x09 Level 10–15 requests are processed 0x0A Level 11–15 requests are processed 0x0B Level 12–15 requests are processed 0x0C Level 13–15 requests are processed 0x0D Level 14–15 requests are processed 0x0E Level 15 requests are processed 0x0F–0x1E All requests are masked

10.5.1.8 INTC Saved Level Mask Register (SLMASK)

The SLMASK is provided so the INTC can automatically manage masking of interrupt requests based on the programmed priority level. If enabled by ICONFIG[EMASK] being set, an interrupt acknowledge read cycle returns a vector number identifying the physical request source, and the CLMASK register is loaded with the level number associated with the request and the current contents of the CLMASK register are loaded into the SLMASK register. Typically, once a level-n interrupt request is handled, the service routine restores the saved level mask value into the current level mask register to re-enable the lower priority requests.

Figure 10-12. INTC Saved Level Mask Register (SLMASK)

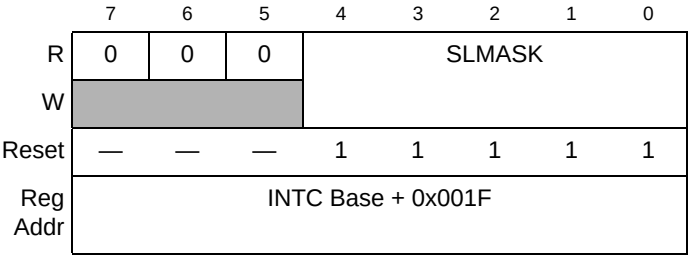


Table 10-11. SLMASK Field Descriptions

Bits	Name	Description
7–5	—	Reserved.
4–0	SLMASK[4:0]	Saved level mask. This 5-bit field is treated as a signed integer within the range [–1, 0, 1, ..., 15]. The value defines the saved level mask. See the CLMASK field definition for more information on the specific values.

10.5.1.9 INTC Interrupt Control Registers (ICR*n*)

The ICR*n* contains the software-defined interrupt level for each interrupt request. Each ICR*n* contains a 4-bit interrupt level [0-15]. These registers are cleared by reset and should be programmed with the appropriate levels before interrupts are enabled.

When multiple interrupt requests are programmed to the same level number, they are processed in a descending request number order. As an example, if requests 63, 62, 2, 1 are programmed to a common level, request 63 is processed first, then request 62, then request 2 and finally request 1.

This definition allows software maximum flexibility in grouping interrupt request sources within any given priority level.

	7	6	5	4	3	2	1	0
R	0	0	0	0	LEVEL			
W								
Reset	—	—	—	—	0	0	0	0
Reg Addr	INTC Base + 0x0040 + *n*							

Figure 10-13. INTC Interrupt Control Registers (ICR*n*)

Bits	Name	Description
3–0	LEVEL[3:0]	Interrupt request level. This 4-bit field maps the given interrupt request to one of 16 levels, where 0x0 is the lowest priority level and 0xF is the highest. If interrupt masking is enabled (ICONFIG[EMASK] = 1), the acknowledgment of a level-n request forces the controller to automatically mask all interrupt requests of level-n and lower.

Table 10-12. ICR*n* Field Descriptions

10.5.1.10 INTC IRQ Acknowledge Register (IRQIACK)

The IRQIACK is a read-only resource containing the vector number of the interrupt request currently being processed. It is typically read early in an IRQ interrupt service routine. There is a fixed association between the vector number returned in the IRQIACK register and the physical interrupt request input:

$$\text{vector number} = 64 + \text{ipi\_int[x]} \qquad \textit{Eqn. 10-1}$$

If there is no pending IRQ interrupt request when the read of the IRQIACK is performed, the interrupt controller returns a value of 63 signalling a spurious interrupt. This is also the reset value of the register. Any attempted write to this register generates an error termination.

	7	6	5	4	3	2	1	0
R	IAVECT							
W								
Reset	0	0	1	1	1	1	1	1
Reg Addr	INTC Base + 0x00EC							

Figure 10-14. INTC IRQ Acknowledge Register (IRQIACK)

Table 10-13. IRQIACK Field Descriptions

Bits	Name	Description
7–0	IAVECT[7:0]	Interrupt vector number. This 8-bit field provides the vector number for the interrupt request currently being acknowledged. The vector number is derived from the physical interrupt request signal as: vector_number = 64 + ipi_int[x] If there is no pending IRQ request when the IRQIACK is read, a spurious interrupt vector number (63) is returned.

10.5.1.11 INTC FIQ Acknowledge Register (FIQIACK)

The FIQIACK register is a read-only resource containing the vector number of the interrupt request currently being processed. It is typically read early in an FIQ interrupt service routine. There is a fixed association between the vector number returned in the FIQIACK register and the physical interrupt request input signal, namely:

$$\text{vector number} = 64 + \text{ipi\_int[x]} \qquad \textit{Eqn. 10-2}$$

If there is no pending FIQ interrupt request when the read of the FIQIACK is performed, the interrupt controller returns a value of 63 signalling a spurious interrupt. This is also the reset value of the register. Any attempted write to this register generates an error termination.

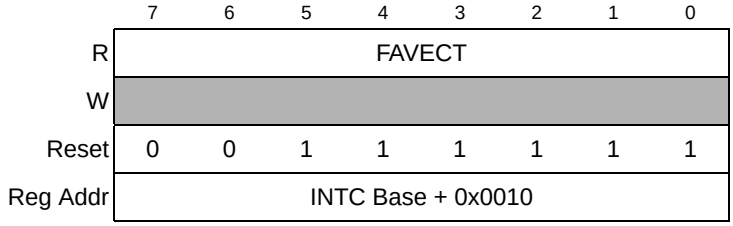


Figure 10-15. INTC FIQ Acknowledge Register (FIQIACK)

Table 10-14. FIQIACK Field Descriptions

Bits	Name	Description
7–0	FAVECT[7:0]	Interrupt vector number. This 8-bit field provides the vector number for the interrupt request currently being acknowledged. The vector number is derived from the physical interrupt request signal as: vector number = 64 + ipi_int[x] If there is no pending FIQ request when the FIQIACK is read, a spurious interrupt vector number (63) is returned.

10.6 Functional Description

To support the interrupt architecture of the ARM7 core programming model, the combined 64 interrupt sources are organized as 16 levels, with an arbitrary number of requests programmed to each level. Consider the priority structure within a single interrupt level (from highest to lowest priority):

level i: ipi_int[a] programmed as level i (highest priority)
ipi_int[b] programmed as level i
ipi_int[c] programmed as level i
ipi_int[d] programmed as level i (lowest priority)

where the bit numbers [a, b, c, d] are defined such that a > b > c > d. In this example, 4 programmable interrupt sources are mapped into a single interrupt level.

The operation of the interrupt controller can be broadly partitioned into three activities:

- Recognition
- Prioritization
- Vector generation during IACK

Recall the INTC is designed to provide a unique vector number for each interrupt request. This allows the operating system kernel to create a vector table of addresses defining the starting location for each interrupt service routine. Throughout this discussion, it is assumed that the vector table contains 32-bit addresses and uses the interrupt vector number as an index into this table so that execution in the appropriate service routine can begin as quickly as possible.

Refer to the INTC block diagram, Figure 10-1, for the subsequent discussion.

10.6.1 Interrupt Recognition

The interrupt controller continuously examines the request sources (the IPR register) and the interrupt mask register (IMR) to determine if there are active requests. This is the recognition phase. The Interrupt Force Register (INTFRC) also factors into the generation on an active request. An active request (assuming the hardware masking is enabled) is defined by the following boolean equation:

$$\text{active\_request}[n] = (\text{IPR}[n] \;\&\; \sim\text{IMR}[n] \mid \text{INTFRC}[n]) \;\&\; (\text{ICR}[n] > \text{CLMASK}) \qquad \textit{Eqn. 10-3}$$

10.6.2 Interrupt Prioritization and Level Masking

As an active request is detected, it is first translated into the programmed interrupt level. Next, the appropriate level masking is performed, if this feature is enabled. Recall the level of the active request must be greater than the current mask level before it is signaled to the processor. The resulting 16-bit unmasked decoded priority level (intc_active_level[15:0]) is then driven out of the interrupt controller. The decoded priority levels from all the interrupt controllers are logically summed together and the highest priority interrupt level is then encoded into the 2-bit FIQ/IRQ signals that are sent to the processor core during this prioritization phase. The mapping of the interrupt levels into the 2-bit FIQ/IRQ signals (the level conversion) is controlled by the ICONFIG[FIQDEF] field.

10.6.3 Vector Generation During IACK

Once the core has sampled for pending interrupts and completed interrupt exception processing, it begins execution of the interrupt service routine (ISR) and typically generates a byte-sized operand read from the controller known as an interrupt acknowledge cycle. The type of interrupt request being acknowledged (FIQ or IRQ) determines the access address. The IACK transfer is a memory-mapped byte read via the

AIPS controller of the FIQIACK or IRQIACK register, and routed to the interrupt controller. Next, the INTC determines the highest unmasked level for the type of interrupt being acknowledged, and generates an 8-bit interrupt vector for that request to complete the cycle. The 8-bit interrupt vector is formed using the following algorithm:

$$vector\_number = 64 + IPR[n] \qquad \textit{Eqn. 10-4}$$

where the bit position [n] within the IPR[63:0] source directly determines the vector number. Vector numbers 0 – 63 are reserved for the processor and its internal exceptions. Thus, the following mapping of bit positions to vector numbers applies for the INTC:

```
if ipi_int[0] is active and acknowledged,then vector_number = 64
if ipi_int[1] is active and acknowledged,then vector_number = 65
if ipi_int[2] is active and acknowledged,then vector_number = 66
                                .
                                .
                                .
if ipi_int[63] is active and acknowledged,then vector_number = 127
```

The net effect is a fixed mapping between the bit position within the source requests to the actual interrupt vector number. The mapping of peripheral interrupts to interrupt vectors for MAC7100 Family devices is detailed in Table 6-2 on page 6-85.

If there is no active unmasked interrupt source at the time of the IACK, a special “spurious interrupt” vector (vector_number = 63) is returned and it is the responsibility of the service routine to handle this error situation.

Note this protocol implies the interrupting peripheral is not accessed during the acknowledge cycle since the interrupt controller completely services the acknowledge. This means the interrupt source must be explicitly disabled in the interrupt service routine. This design provides unique vector capability for all interrupt requests, regardless of the “complexity” of the peripheral device.

In most applications, it is expected that the hardware masking of interrupt levels is enabled. In this mode of operation, the IACK read cycle also causes the current interrupt level mask to be saved in the SLMASK register, and the new level being acknowledged loaded into the CLMASK register. This operation then automatically masks the new level (and all lower levels) while in the service routine. Generally, as the service routine completes execution and the initiating request source has been negated, the saved mask level is restored into the current mask level to re-enable the lower priority levels.

Finally, the vector number returned during the IACK cycle provides the association with the request and the physical interrupt signal.

The CLMASK and SLMASK registers are all loaded (if properly enabled) during the interrupt acknowledge read cycle.

For more information on the specific operations typically performed in an interrupt service routine, see Section 10.7.2, “Interrupt Service Routines.”

10.7 Initialization / Application Information

The reset state of the INTC has all requests masked via the IMR. Before any interrupt requests are enabled, the following steps must be taken:

1. The ICONFIG register needs to be set to the desired system configuration.
2. All the ICR*n* registers need to be programmed with the appropriate interrupt levels.
3. The reset value for the Level Mask registers (CLMASK and SLMASK) are set to –1, the value with no levels masked. Typically, these registers do not need to be modified before interrupts are enabled.
4. The appropriate interrupt vector tables and interrupt service routines must be loaded into memory and the memory address pointers for the FIQ and IRQ stacks loaded into the R13_{fiq,irq} registers.
5. Enable the interrupt requests by clearing the appropriate bits in the IMR and the CPSR[F,I].

10.7.1 Typical Applications

In many real-time system designs, a typical configuration for supporting priority-based preemptive task scheduling requires only a single interrupt signal to the processor core. Stated differently, the two levels of interrupt support provided by an ARM7 core (FIQ/IRQ) are not necessarily required, and a single level is sufficient. For ARM7 cores with only a single interrupt level is implemented, the FIQ exception mode is typically the one to be used since it provides significantly more hardware resources for the interrupt processing than IRQ.

By setting ICONFIG[FIQDEF] = 0x00, all interrupt requests are mapped into the FIQ request signal, and the IRQ core functionality is completely unused.

Another common configuration may chose to define certain “non-maskable” interrupt requests. Typically, these requests are programmed as level 15, and would be logically connected to the FIQ core input. In this configuration, priority levels 0-14 would then be available for use of normal (IRQ) requests.

For applications where the wakeup functionality is used, the interrupt controller includes logic that limits the largest value of the 4-bit interrupt level to “maximum – 1” so the controller can always generate an sleep mode exit. Thus, if ipg_lp_int_mask[3:0] is set to 0xf (the maximum), the interrupt controller converts this value to 0xe (maximum - 1) in the wakeup logic. This guarantees that a level 15 interrupt request generates the sleep mode exit.

10.7.2 Interrupt Service Routines

This section focuses on the interaction of the interrupt masking functionality with the service routine. Figure 10-16 presents a timing diagram showing various phases during the execution of an interrupt service routine. It is important to note the time scale in this diagram is *not* meant to be accurate.

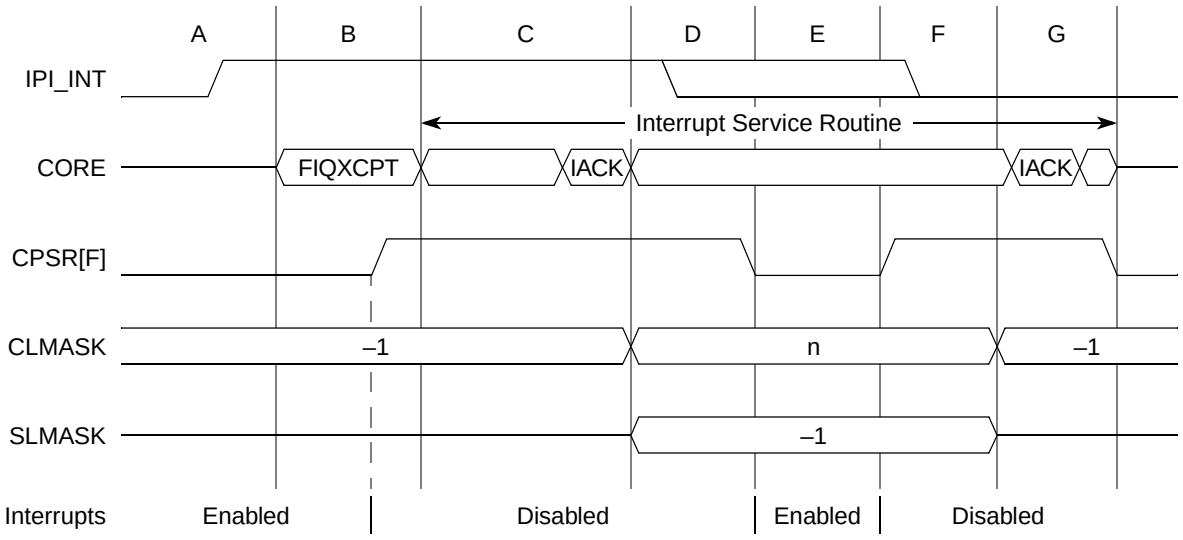


Figure 10-16. INTC Interrupt Service Routine and Masking

Consider the events depicted in each “segment” [A-G] of this diagram.

1. In A, an interrupt request is asserted, which is mapped into the FIQ core signal.
2. As B begins, the interrupt request is recognized and the core begins FIQ exception processing. During the core’s exception processing, the CPSR sets the F bit, disabling all interrupts. At the end of the core’s exception processing, control passes to the interrupt service routine, shown as the beginning of segment C.
3. During C, the ISR performs the interrupt acknowledge read cycle to retrieve the vector number associated with the request. As the interrupt acknowledge read is performed, the vector number is returned to the core, and the CLMASK register raised to level n, the interrupt level being acknowledged. The former contents of the CLMASK are loaded into the SLMASK register at this time, at the end of C.
4. During segment D, the ISR accesses the peripheral to negate the interrupt request source. At the conclusion of segment D, the CPSR[F] flag is cleared to re-enable interrupts.
5. The bulk of the interrupt service routine executes in segment E, with interrupts enabled.
6. Near the end of the service routine, the CPSR[F] flag is again set, disabling interrupt requests, and preparing to perform the context switch.
7. At the end of segment F, the original value in the saved level mask (SLMASK) is restored to the current level mask (CLMASK). Optionally, the service routine can directly load the CLMASK register with any value with pending interrupt requests of certain levels need to be examined.
8. In segment G, the interrupt service routine completes execution. During this period of time (recall interrupts are disabled in the CPSR), it is possible to access the interrupt controller to see if there are any pending properly-enabled requests. Checking for any pending interrupt requests at this time provides the ability to initiate processing of another interrupt without the need to return from the original ISR and then incur the overhead of another interrupt exception.
9. At the conclusion of segment G, the processor core returns to the original interrupted task, or a different task that is ready to execute.

10.7.3 Performance

There are two key performance parameters for the interrupt controller: latency from the assertion of an interrupt request, and vector generation timing.

There are two levels of hardware registers between the assertion of an interrupt request on the ipi_int[63:0] input and the posting of the FIQ/IRQ request to the processor core. Thus, if an interrupt request is first asserted in cycle *i*, then the first assertion of a properly-enabled FIQ/IRQ request to the processor core occurs in cycle *i+2*. The processor's sampling of the FIQ/IRQ request signals is then dependent on the instruction stream being executed. For more information on the core behavior, see the appropriate ARM7 reference manual. As an example, see Section 2.9, page 69 of the *ARM7TDMI-S (Rev 4) Technical Reference Manual* (ARM DDI 0234A).

As the IACK cycle is performed during the interrupt service routine, it appears as a normal IPS read cycle, and requires three platform cycles in the core processor bus data phase (two wait-states).

Chapter 11
Miscellaneous Control Module (MCM)

11.1 Overview

The miscellaneous control module (MCM) provides several control functions for the MAC7100 Family Standard Product Platform (SPP), including program-visible information regarding configuration and revision levels, a reset status register, a software watchdog timer, wake-up control for exiting sleep modes, the address map for the crossbar switch and generic access error information for the processor core.

The MCM also contains control hardware to enable the device to be brought out of a low power mode via an interrupt source through the interrupt controller, with the priority level of the source defined by the MCM wake-up control register (MWCR).

The software watchdog timer (SWT) is contained within the MCM and supports either interrupt generation or reset generation on the occurrence of a time-out event. The SWT also supports a windowed time period, requiring the writing of the watchdog key within a specified time window. The source for the SWT clock is controlled by the clock and reset generator module (CRG).

The MCM also contains the XBS address memory register (AAMR) which is used to steer access to the crossbar slave ports in order to provide remapping operations. Following reset the contents of the AAMR register is determined by the mode entered as a result of the default slave mapping. Following reset, the Flash, external bus and RAM can be remapped between XBS slave port 0 and port 1.

11.2 Features

The MCM includes these features:

- Program-visible information regarding the device configuration and revision
- Reset status register (MRSR)
- Software watchdog timer (SWT) running on independent, asynchronous clock
- Wake-up control for exiting sleep modes
- Address map for crossbar switch (XBS)
- Address information for faulted ARM7TDMI-S processor core memory accesses

11.3 Memory Map / Register Definition

The Miscellaneous Control Module occupies a 128-byte space mapped to the IPS bus as defined in Chapter 8, “Device Memory Map.” Table 11-1 shows a 32-bit view of the MCM memory map.

The MCM does not include any logic which provides access control. Rather, this function is supported using the access control mechanism described in Chapter 16, “AMBA to IP Bus Bridge Module (AIPS).”

Table 11-1. MCM Memory Map

MCM Offset	Register Description		
0x0000	MCM Processor Core Type Register (PCT)	MCM Device Revision Register (REV)	
0x0004	MCM XBS Master Configuration Register (AMC)	MCM XBS Slave Configuration Register (ASC)	
0x0008	MCM IPS On-Platform Module Configuration Register (IOPMC)		
0x000C	Reserved		MCM Reset Status Register (MRSR)
0x0010	Reserved		MCM Wake-up Control Register (MWCR)
0x0014	Reserved	MCM Software Watchdog Timer Control Register (MSWTCR)	
0x0018	Reserved		MCM SWT Service Register (MSWTSR)
0x001C	Reserved		MCM SWT Timer Interrupt Register (MSWTIR)
0x0020	MCM XBS Address Map Register (AAMR)		
0x0024 – 0x006C	Reserved		
0x0070	MCM Core Fault Address Register (CFADR)		
0x0074	Reserved	MCM Core Fault Location Register (CFLOC)	MCM Core Fault Attributes Register (CFATR)
0x0078	Reserved		
0x007C	MCM Core Fault Data Register (CFDTR)		

11.3.1 Register Descriptions

Attempted accesses to reserved addresses result in an error termination, while attempted writes to read-only registers are ignored and do not terminate with an error. Unless noted otherwise, writes to the programming model must match the size of the register, e.g., an n-bit register only supports n-bit writes, etc. Attempted writes of a different size than the register width produce an error termination of the bus cycle and no change to the targeted register.

11.3.1.1 MCM Processor Core Type Register (PCT)

The PCT is a 16-bit read-only register specifying the architecture of the processor core in the device. All writes are ignored.

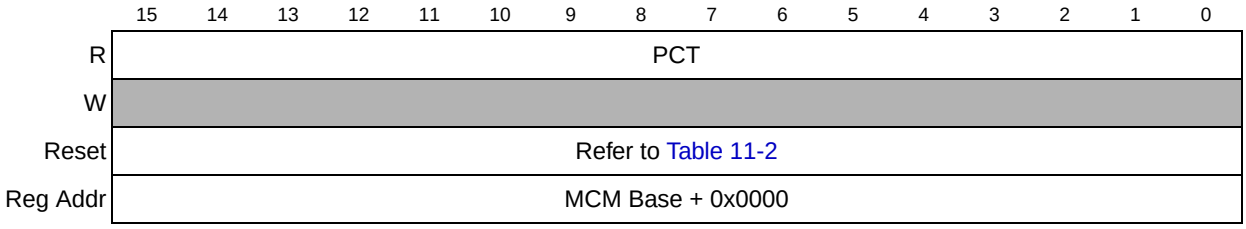


Figure 11-1. MCM Processor Core Type Register (PCT)

Table 11-2. PCT Field Descriptions

Bits	Name	Description
15–0	PCT[15:0]	Processor core type. The MCM is designed to support ARM7, ColdFire and PowerPC cores. MAC7100 Family devices identify an ARM7 core: 0xA700ARM7

11.3.1.2 MCM Device Revision Register (REV)

The device revision register is a 16-bit read-only register that specifies the revision number of the device. All writes are ignored.

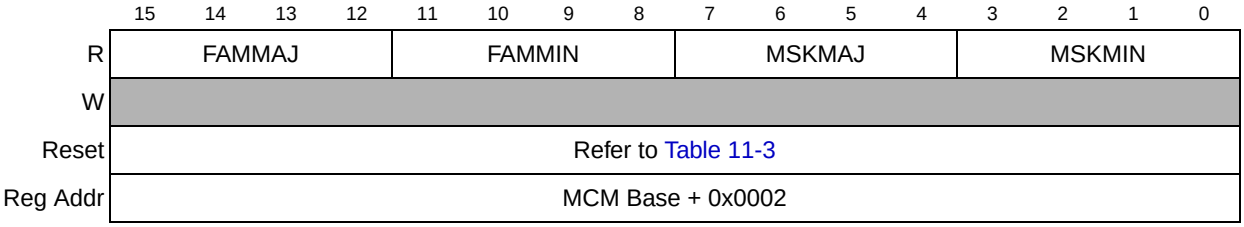


Figure 11-2. MCM Device Revision Register (REV)

Table 11-3. REV Field Descriptions

Bits	Name	Description
15–12	FAMMAJ[3:0]	Family major identifier. 0x7 MAC ARM7 Family
11–8	FAMMIN[3:0]	Family minor identifier. 0x1 MAC7100 Family
7–4	MSKMAJ[3:0]	Mask major identifier. 0x*n* Refer to Table 11-4 for definitions.
3–0	MSKMIN[3:0]	Mask minor identifier. 0x*n* Refer to Table 11-4 for definitions.

Table 11-4. REV Values by Mask Set

Device	Mask Set	FAMMAJ[3:0]	FAMMIN[3:0]	MSKMAJ[3:0]	MSKMIN[3:0]
MAC71*x*1	0L49P	7	1	1	0
	1L49P	7	1	1	1
	0L47W	7	1	2	0
	1L47W	7	1	2	1
MAC71*x*2	0L61W	7	1	3	0
MAC71*x*6	0L38Y	7	1	4	0

11.3.1.3 MCM XBS Master Configuration Register (AMC)

The AMC is an 8-bit read-only register identifying the presence of bus master connections to the AMBA-AHB Cross-Bar Switch (XBS). All writes are ignored.

	7	6	5	4	3	2	1	0
R	XMC7	XMC6	XMC5	XMC4	XMC3	XMC2	XMC1	XMC0
W								
Reset	0	0	0	0	0	0	1	1
Reg Addr	MCM Base + 0x0004							

Figure 11-3. MCM XBS Master Configuration Register (AMC)

Table 11-5. AMC Field Descriptions

Bits	Name	Description
7–0	XMC*n*[1]	XBS master configuration. 0 A bus master connection to XBS input port *n* is absent 1 A bus master connection to XBS input port *n* is present

1 MAC7100 Family devices: `0b0000_0011`.

11.3.1.4 MCM XBS Slave Configuration Register (ASC)

The ASC is an 8-bit read-only register identifying the presence of bus slave connections to the AMBA-AHB Crossbar Switch (XBS). All writes are ignored.

	7	6	5	4	3	2	1	0
R	XSC7	XSC6	XSC5	XSC4	XSC3	XSC2	XSC1	XSC0
W								
Reset	1	0	0	0	1	0	1	0
Reg Addr	MCM Base + 0x0006							

Figure 11-4. MCM XBS Slave Configuration Register (ASC)

Table 11-6. ASC Field Descriptions

Bits	Name	Description
7–0	XSC*n*[1]	XBS slave configuration. 0 A bus slave connection to XBS output port *n* is absent 1 A bus slave connection to XBS output port *n* is present

1 MAC71*x*1 and MAC71*x*6 devices: `0b1000_1010`; MAC71*x*2 devices: `0b1000_1000`

11.3.1.5 MCM IPS On-Platform Module Configuration Register (IOPMC)

The IOPMC is a 32-bit read-only register identifying the presence of SPP peripheral modules connected to the AIPS on-platform IPS bus. Refer to Chapter 16, “AMBA to IP Bus Bridge Module (AIPS),”

particularly Table 16-2 on page 16-254, for details regarding on-platform versus off-platform module configuration. All writes are ignored.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PMC31	PMC30	PMC29	PMC28	PMC27	PMC26	PMC25	PMC24	PMC23	PMC22	PMC21	PMC20	PMC19	PMC18	PMC17	PMC16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Reg Addr	MCM Base + 0x0008															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PMC15	PMC14	PMC13	PMC12	PMC11	PMC10	PMC9	PMC8	PMC7	PMC6	PMC5	PMC4	PMC3	PMC2	PMC1	PMC0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Reg Addr	MCM Base + 0x0008															

Figure 11-5. MCM IPS On-Platform Module Configuration Register (IOPMC)

Table 11-7. IOPMC Field Descriptions

Bits	Name	Description
31–0	PMC*n*	Platform module configuration. 0 A connection to AIPSSPP module *n* is absent 1 A connection to AIPSSPP module *n* is present

11.3.1.6 MCM Reset Status Register (MRSR)

The MRSR is an 8-bit read-only register that contains a bit for each of the reset sources to the device. Only one bit is set at any time in the MRSR, reflecting the cause of the most recent reset as signalled by device reset input signals. All writes are ignored.

	7	6	5	4	3	2	1	0
R	POR	DIR	SWTR	0	0	0	0	0
W								
Reset	Refer to Table 11-8			0	0	0	0	0
Reg Addr	MCM Base + 0x000F							

Figure 11-6. MCM Reset Status Register (MRSR)

Table 11-8. MRSR Field Descriptions

Bits	Name	Description
7	POR	Power-on reset. 0 Last recorded event was not a power-on reset. 1 Last recorded event was a power-on reset.
6	DIR	Device input reset. 0 Last recorded event was not caused by a device input reset. 1 Last recorded event was a reset caused by a device input reset.
5	SWTR	Software watchdog timer reset. 0 Last recorded event was not caused by the MCM software watchdog timer. 1 Last recorded event was a reset caused by the MCM software watchdog timer.
4–0	—	Reserved.

11.3.1.7 MCM Wake-up Control Register (MWCR)

Implementation of low-power sleep modes and exit from these modes via an interrupt require communication between the MCM, INTC and CRG. The MCM wake-up control register (MWCR) provides an 8-bit register to control entry into low-power modes, as well as definition of the interrupt level needed to exit the mode.

The following sequence of operations is generally needed to enable low-power modes.

1. The processor core loads the appropriate data value into the MWCR, setting the ENBWCR bit and the desired interrupt priority level.
2. At the appropriate time, the CPU writes to the CRG in order to stop execution (refer to Section 4.3.6.10.5, “Stop Mode,” on page 4-75 for details). The CRG signals the MCM that a low-power mode is active, which, if enabled by MWCR[ENBWCR], causes the MCM output signal “enter_low_power_mode” to be set. This, in turn, causes the selected low-power mode to be entered, and the appropriate clock signals disabled.
3. After entering the low-power mode, the INTC enables a special combinational logic path which evaluates all unmasked interrupt requests. The device remains in this mode until an event which generates an unmasked interrupt request with a priority level greater than the value programmed in the MWCR[PRILVL] occurs.
4. Once the unmasked interrupt request level arrives, the INTC signals its presence, and the MCM responds by asserting an “exit_low_power_mode” signal.
5. The CRG senses the assertion of the “exit” signal, and re-enables the appropriate clock signals.
6. With the processor core clocks enabled, the core handles the pending interrupt request.

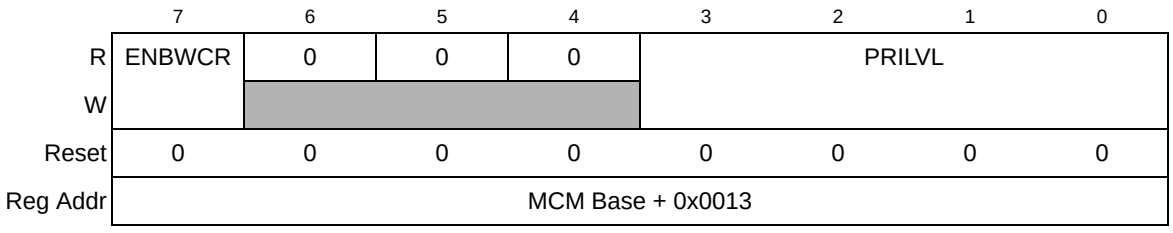


Figure 11-7. MCM Wake-up Control Register (MWCR)

Table 11-9. MWCR Field Descriptions

Bits	Name	Description
7	ENBWCR	Enable WCR 0 MWCR is disabled. 1 MWCR is enabled.
6–4	—	Reserved.
3–0	PRILVL[3:0]	Interrupt priority level. Specifies the interrupt priority level needed to exit the low-power mode. Specifically, an unmasked interrupt request of a priority level greater than the PRILVL value is required to exit the mode. Refer to Chapter 10, "Interrupt Controller Module (INTC)," for more information.

11.3.1.8 MCM Software Watchdog Timer Control Register (MSWTCR)

The software watchdog timer (SWT) prevents system lockup if the software becomes trapped in a loop with no controlled exit, or if a bus transaction becomes "hung." The software watchdog timer can be enabled or disabled through the MSWTCR[SWE]. By default, it is disabled. The SWT operates in a separate, asynchronous clock domain and contains clock domain synchronizers as the communication mechanism between the system clock domain (f_{SYS}) and the software watchdog timer domain. If enabled, the watchdog timer requires the periodic execution of a software watchdog servicing sequence. If this periodic servicing action does not occur, the timer expires, resulting in a watchdog timer interrupt or a hardware reset, as programmed in the MSWTCR[SWRI].

There are three user-defined responses to a time-out:

1. The MSWTCR[SWRI] can specify the assertion of a watchdog timer interrupt.
2. The MSWTCR[SWRI] can specify the immediate assertion of a system reset.
3. The MSWTCR[SWRI] can specify a sequence of responses. Upon the first time-out, a watchdog timer interrupt is asserted. If this time-out condition is not serviced before a second time-out occurs, the watchdog timer asserts the system reset signal. This configuration supports a more graceful response to watchdog time-outs: first attempting an interrupt to notify the system, and if that fails, generating a system reset.

In addition to these three basic modes of operation, the watchdog timer also supports a "windowed" mode of operation. In this mode, the time-out period is divided into 4 equal segments and the actual servicing of the watchdog timer must occur during the last segment, i.e., during the last 25% of the time-out period. If the watchdog timer is serviced anytime in the first 75% of the time-out period, an immediate system reset occurs.

Throughout Section 11.3.1.8, "MCM Software Watchdog Timer Control Register (MSWTCR)," there are numerous references to the generation of a system reset. MCM behavior during this process is detailed below. When the watchdog timer expires and MSWTCR[SWRI] is programmed for a reset (either as the initial or secondary response), the MCM generates a watchdog timer reset output signal which is driven off the device where it is typically combined with other reset signals and driven throughout the system. The combined reset input signal is driven back to the device and MCM, where MRSR[SWTR] can be set and the appropriate action taken by the core and device logic. The watchdog timer logic also sends an interrupt request to the INTC.

To prevent the watchdog timer from interrupting or resetting, the MSWTSR must be serviced by performing the following sequence:

Write `0x55` to the MSWTSR.

Write `0xAA` to the MSWTSR.

Both writes must occur in this order before the time-out, but any number of instructions can be executed between the two writes. This definition allows interrupts and exceptions to occur, if necessary, between the two writes. The timer value is constantly compared with the time-out period specified by MSWTCR[SWT]. Any write to the MSWTCR resets the watchdog timer. In addition, there is a read-only control flag included in the MSWTCR to prevent accidental updates to this control register from changing the desired system configuration.

If the second write occurs at the exact same cycle as the time-out condition is reached, the clear takes precedence and the time-out response suppressed.

The MSWTCR controls the software watchdog timer, time-out periods, and time-out response. The register can be read or written at any time. At system reset, the software watchdog timer is disabled.

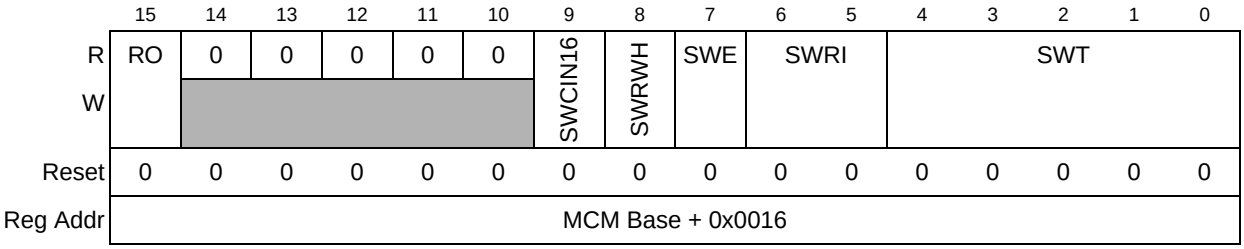


Figure 11-8. MCM Software Watchdog Timer Control Register (MSWTCR)

Table 11-10. MSWTCR Field Descriptions

Bits	Name	Description
15	RO	Read-only 0 MSWTCR can be read or written. 1 MSWTCR can only be read. A system reset is required before this register can again be written. The setting of this bit is intended to prevent accidental writes of the MSWTCR from changing the defined system watchdog configuration.
14–10	—	Reserved.
9	SWCIN16	Force SWT CarryIn16. This control bit is intended for use only when testing the operation of the SWT. When asserted, it forces the actual timer to increment by 65537 (2^{16} + 1) each cycle rather than simply by 1. This allows testing of large SWT time-out values without actually incrementing through the entire dynamic range.
8	SWRWH	Software watchdog run while halted. 0 SWT stops counting if the processor core is halted. 1 SWT continues to count even while the processor core is halted.
7	SWE	Software watchdog enable 0 SWT is disabled. 1 SWT is enabled.

Table 11-10. MSWTCR Field Descriptions (continued)

Bits	Name	Description
6–5	SWRI[1:0]	Software watchdog reset/interrupt 00 If a time-out occurs, the SWT generates an interrupt. Refer to Table 6-2 on page 6-85 and Chapter 10, "Interrupt Controller Module (INTC)," for details on configuring the priority and handling of SWT interrupts. 01 The first time-out generates an interrupt to the processor, and if not serviced, then a second time-out generates a system reset and sets the MRSR[SWTR] flag. 10 If a time-out occurs, the SWT generates a reset. MRSR[SWTR] is set. 11 The SWT functions in a "window" mode of operation. For this mode, the servicing of the MSWSR must occur during the last 25% of the time-out period. Any writes to the MSWSR during the first 75% of the time-out period generate an immediate system reset. Likewise, if the MSWSR is not serviced during the last 25% of the time-out period, then a system reset is generated. For any type of reset response, the MRSR[SWTR] flag is set. As noted previously, the generation of a system reset causes the MCM to assert a signal to the CRG, which controls the reset of on-chip systems and also drives the external $\overline{\text{RESET}}$ signal throughout the system.
4–0	SWT[4:0]	Software watchdog time-out period. This field selects the time-out period for the SWT. At reset, this field is cleared selecting the minimum time-out period, but the SWT is disabled since MSWTCR[SWE] = 0. In general, the value in this field defines the bit position within the 32-bit counter that specifies the time-out period. Thus, if SWT = n, then the time-out period is 2^n system clock (f_{SYS}) cycles. Since it is not practical to operate the software watchdog timer with very short time-out periods, data values of 0 to 7 are forced to a value of 8, defining a minimum time-out period of 256 f_{SYS} cycles. The logic which forces the minimum value to 8 does **not** affect the contents of this field in the register. For SWT = n, then time-out period = 2^n f_{SYS} cycles, n = 8,9,..., 31.

11.3.1.9 MCM Software Watchdog Timer Service Register (MSWTSR)

The software service sequence must be performed using the MSWTSR to prevent a SWT timeout. The service sequence requires two writes to this data register: first a write of 0x55 followed by a write of 0xAA. Both writes must be performed in this order prior to the SWT timeout, but any number of instructions or accesses to the SWSR may occur between the two writes. If a SWT timeout has already occurred, writing to this register has no effect in negating the SWT interrupt or reset. Figure 11-9 illustrates the MSWTSR.

	7	6	5	4	3	2	1	0
R	SWSR							
W								
Reset	— [1]							
Reg Addr	MCM Base + 0x001B							

[1] As reads of this register are meaningless, the $\overline{\text{RESET}}$ value is a don't care / undefined.

Figure 11-9. MCM SWT Service Register (MSWTSR)

If the software watchdog timer is enabled (MSWTCR[SWE] = 1), then any write of a data value other than 0x55 or 0xAA generates an immediate system reset, regardless of the value in the MSWTCR[SWRI] field.

11.3.1.10 MCM Software Watchdog Timer Interrupt Register (MSWTIR)

For certain values of the MSWTCR[SWRI] field, the software watchdog timer generates an interrupt response to a time-out. For these configurations, the MSWTIR provides the program-visible interrupt request from the software watchdog timer.

During the interrupt service routine handling a software watchdog timer request, the interrupt source contained in the MSWTIR must be explicitly cleared.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	SWTIC
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	MCM Base + 0x001F							

Figure 11-10. MCM SWT Timer Interrupt Register (MSWTIR)

Table 11-11. MSWTIR Field Descriptions

Name	Description	Value
7–1	—	Reserved.
0	SWTIC	Software watchdog interrupt flag 0 A SWT interrupt has not occurred. 1 A SWT interrupt has occurred. The interrupt request is negated by writing a 1 to this bit. Writing a 0 has no effect.

11.3.1.11 MCM XBS Address Map Register (AAMR)

For MAC7100 family devices, the AAMR provides a register which implements additional capability for XBS request steering. This register enables the mapping of 512 MByte blocks to various XBS slave ports.

The AAMR is divided into eight 4-bit fields, each of which provides an enable bit and a 3-bit slave number. The upper 3 bits of the XBS master address (HADDR[31:29]) are used to index into the AAMR to select a corresponding 3-bit vector which defines the targeted XBS slave. The AAMR can only be written in its entirety, i.e., only 32-bit writes are supported. Any attempted write of a smaller data size (e.g., 8- or 16-bit value) generates an error response.

Mapping for address slave numbers 2 through 7 are fixed on MAC7100 family devices. Only address ranges 0x2000_0000 and 0x0000_0000 may be assigned to combinations of the external bus, program Flash or SRAM, as shown in Table 11-14. Refer to Section 8.1, “Memory Map Details,” on page 8-93 for detailed limitations on AAMR values in various chip modes.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EA7	A7Slave			EA6	A6Slave			EA5	A5Slave			EA4	A4Slave		
W																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	MCM Base + 0x0020															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EA3	A3Slave			EA2	A2Slave			EA1	A1Slave			EA0	A0Slave		
W																
Reset	0	0	0	0	1	0	1	1	See Table 11-13							
Reg Addr	MCM Base + 0x0020															

Figure 11-11. MCM XBS Address Map Register (AAMR)

Table 11-12. AAMR Field Descriptions

Bits	Name	Description
31, 27, 23, 19, 15, 11, 7, 3	EA*n*	Enable address region *n* 0 The 512 MByte address region associated with HADDR[31:29] = *n* is disabled, and not mapped to an XBS slave. Accesses to this memory region terminate with an error response. 1 The 512 MByte address region associated with HADDR[31:29] = *n* is enabled, and mapped to the XBS slave defined by A*n*Slave.
30–28, 26–24, 22–20, 18–16, 14–12, 10–8, 6–4, 2–0	A*n*Slave[2:0]	Address *n* slave number xxx A*n*Slave defines the XBS slave mapped to the memory region defined by HADDR[31:29] = *n* For MAC7100 family devices: 000 Program Flash (for core requests only) 001 External bus interface[1] 011 SRAM controller 111 AIPS (peripheral control register space))

1 MAC7111, MAC7116, MAC7131 and MAC7136 devices only.

The value of AAMR[7:0] at reset is determined by the selected chip mode as follows:

Table 11-13. AAMR Reset Values

Mode	Initial AAMR	HADDR[31:29] Mapping			
		A7Slave (0xE000 0000)	A3Slave (0x4000 0000)	A1Slave (0x2000 0000)	A0Slave (0x0000 0000)
Normal Single-Chip Mode	0xF000_0B98	Peripheral Bus	SRAM	External Bus	Program Flash
Secured Single-Chip Mode	0xF000_0B08	Peripheral Bus	SRAM	—	Program Flash
Normal Expanded Mode[1]	0xF000_0B89	Peripheral Bus	SRAM	Program Flash	External Bus
Secured Expanded Mode[1]	0xF000_0B09	Peripheral Bus	SRAM	—	External Bus
Normal Data Flash Boot Mode	0xF000_0B80	Peripheral Bus	SRAM	Program Flash	—
Secured Data Flash Boot Mode	0xF000_0B80	Peripheral Bus	SRAM	Program Flash	—

1 MAC7111, MAC7116, MAC7131 and MAC7136 devices only.

Mapping of XBS slave connections may be modified following reset by writing the desired value to the AAMR register. For writes to the AAMR, the hardware invokes the following synchronization mechanism:

1. A write to the AAMR loads the pending address map operand into a temporary register.
2. The pending AAMR write causes the MCM to assert a control signal which forces the XBS into a “halted” state.
3. The IPS write cycle is terminated normally.
4. Once the XBS module indicates it has stalled all bus master transactions and entered the halted state, then MCM loads the pending address map operand into the AAMR, effectively enabling the new mapping.
5. The MCM then releases the halt request signal to the XBS, allowing it to resume operation with the just-loaded address map.

The device is guaranteed to function properly after changing the value of the AAMR if the write to the register is followed by a sufficient number of NOP instructions necessary to flush out the depth of the processor pipeline.

The upper 24 bits of the AAMR register are fixed: AAMR[31:8] = 0xF0000B. This restricts all master bus addresses in the upper 3 GByte of the 4 GByte address space as follows:

- Addresses with HADDR[31:29] = 0x7 are steered to the AIPS controller.
- Addresses with HADDR[31:29] = 0x6, 0x5, 0x4 or 0x3 are not valid and are terminated with an access error.
- Addresses with HADDR[31:29] = 0x2 are steered to the SRAM.
- Addresses with HADDR[31:29] = 0x1 or 0x0 may be steered to the SRAM, EIM, or the program Flash (for core requests only).

The net effect is an address mapping function which steers bus master requests to the selected XBS slave target. Another description of the address mapping function is shown below:

```
if (HADDR[31:29] == 3'b000)
    if (EA0 == 1)
        then route access to the XBS slave defined by A0Slave[2:0]
        else terminate access with an error
if (HADDR[31:29] == 3'b001)
    if (EA1 == 1)
        then route access to the XBS slave defined by A1Slave[2:0]
        else terminate access with an error
if (HADDR[31:29] == 3'b010)
    if (EA2 == 1)
        then route access to the XBS slave defined by A2Slave[2:0]
        else terminate access with an error
if (HADDR[31:29] == 3'b011)
    if (EA3 == 1)
        then route access to the XBS slave defined by A3Slave[2:0]
        else terminate access with an error
```

```
if (HADDR[31:29] == 3'b100)
    if (EA4 == 1)
        then route access to the XBS slave defined by A4Slave[2:0]
        else terminate access with an error
if (HADDR[31:29] == 3'b101)
    if (EA5 == 1)
        then route access to the XBS slave defined by A5Slave[2:0]
        else terminate access with an error
if (HADDR[31:29] == 3'b110)
    if (EA6 == 1)
        then route access to the XBS slave defined by A6Slave[2:0]
        else terminate access with an error
if (HADDR[31:29] == 3'b111)
    if (EA7 == 1)
        then route access to the XBS slave defined by A7Slave[2:0]
        else terminate access with an error
```

The capabilities provided by the AAMR are very flexible; all memory region(s) defined by HADDR[31:29] may be mapped to any XBS slave. There are restrictions on values allowed to be written to AAMR[7:0]; Table 11-14 shows all allowed values. Write attempts with any other AAMR[7:0] values are terminated with an access error. Note that for MAC71*x*2 devices, no protection is provided to prevent A1Slave or A0Slave from being set to the EIM (0b001). If an application sets a slave port to the EIM when the external bus is not present or enabled and then accesses that slave address range, the system will hang.

Table 11-14. AAMR[7:0] Allowed Values

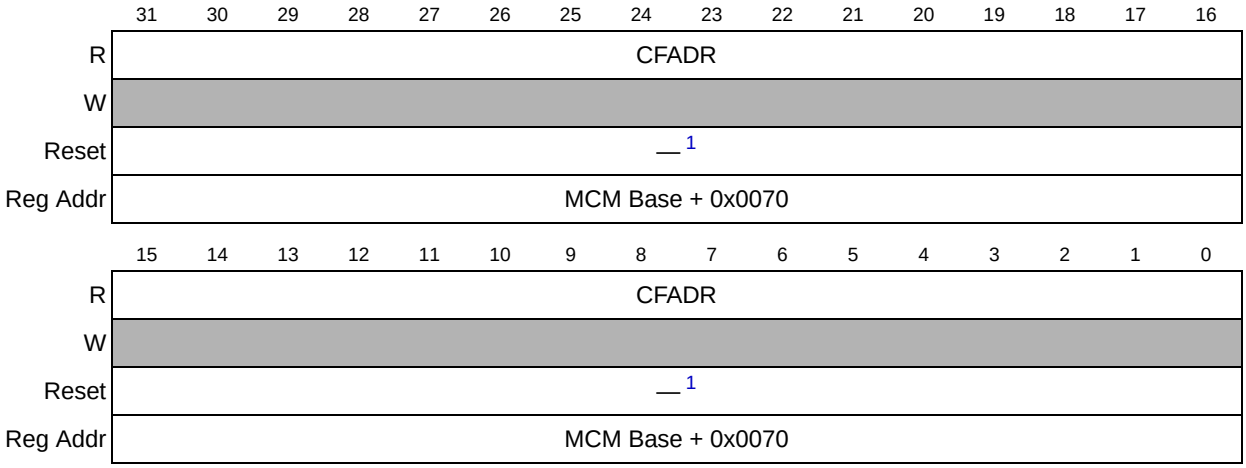
AAMR[7:0] Value	HADDR[31:29] Mapping	
	A1Slave	A0Slave
0x00	—	—
0x08	—	Program Flash
0x09	—	EIM
0x0B	—	SRAM
0x80	Program Flash	—
0x90	EIM	—
0xB0	SRAM	—
0x89	Program Flash	EIM
0x8B	Program Flash	SRAM
0x98	EIM	Program Flash
0x9B	EIM	SRAM
0xB8	SRAM	Program Flash
0xB9	SRAM	EIM
0x99	EIM	EIM
0xBB	SRAM	SRAM

11.3.1.12 MCM Core Fault Address Register (CFADR)

To aid in recovery from certain types of data access errors, the MCM module supports a number of registers which capture access address, attribute and data information on bus cycles terminated with an

error response. These registers can then be read during the resulting exception service routine and the appropriate recovery performed. It is important to note that these registers are used to capture fault recovery information only on data access faults, i.e., no information is captured on instruction fetch faults.

The CFADR is a 32-bit read-only register for capturing the address of the last core data access which was terminated with an error response. All writes are ignored.



1 The content of this register is undefined following $\overline{\text{RESET}}$ and until a core address fault occurs.

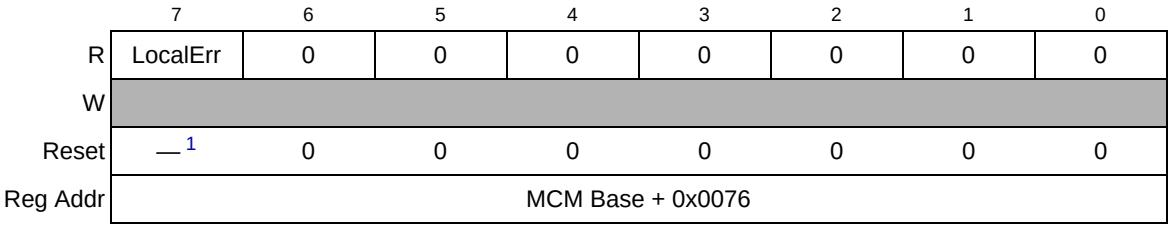
Figure 11-12. MCM Core Fault Address Register (CFADR)

Table 11-15. CFADR Field Descriptions

Bits	Name	Description
31–0	CFADR[31:0]	Core fault address register. This 32-bit register contains the faulting address of the last core data access terminated with an error response.

11.3.1.13 MCM Core Fault Location Register (CFLOC)

The CFLOC is an 8-bit read-only register containing one bit that indicates the exact location of the captured fault information: an XBS master bus data access or a data access to a tightly-coupled core local memory. All writes are ignored.



1 The content of this register is undefined following $\overline{\text{RESET}}$ and until a core address fault occurs.

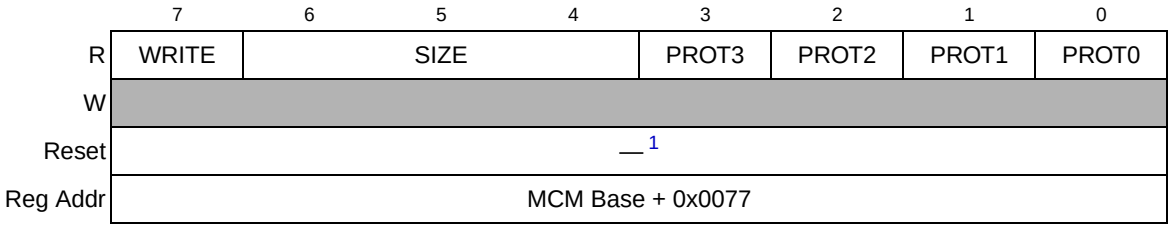
Figure 11-13. MCM Core Fault Location Register (CFLOC)

Table 11-16. CFLOC Field Descriptions

Bits	Name	Description
7	LocalErr	Bus error location indicator. 0 Error occurred on the XBS master bus 1 Error occurred on the ARM7TDMI-S local bus
6–0	—	Reserved.

11.3.1.14 MCM Core Fault Attributes Register (CFATR)

The CFATR is an 8-bit read-only register that captures the bus cycle attributes of the last faulted data access to the on the XBS master bus. All writes are ignored.



1 The content of this register is undefined following $\overline{\text{RESET}}$ and until a core address fault occurs.

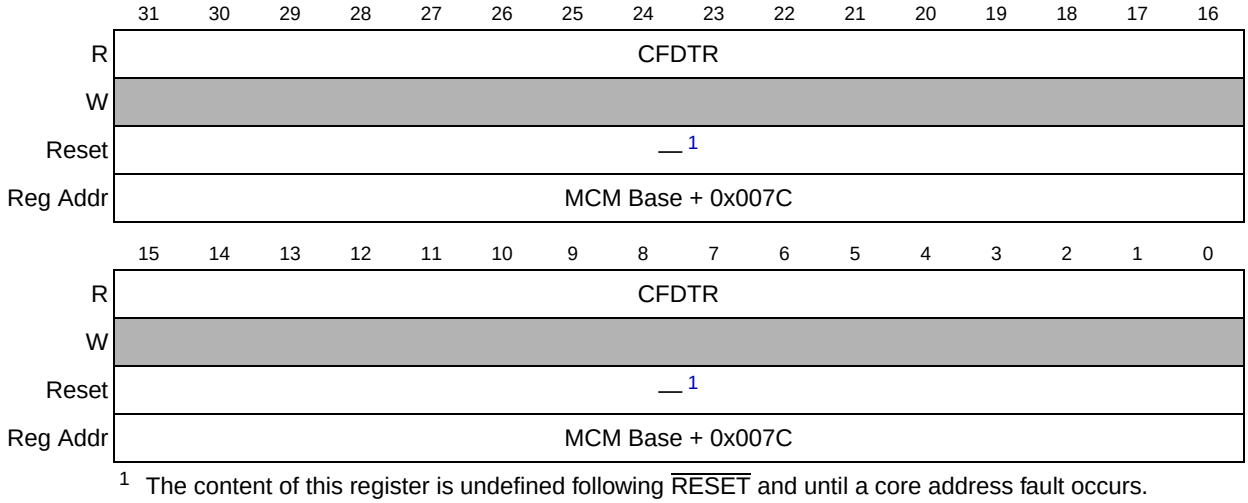
Figure 11-14. MCM Core Fault Attributes Register (CFATR)

Table 11-17. CFATR Field Descriptions

Bits	Name	Description
7	WRITE	Write. 0 Core read access 1 Core write access
6–4	SIZE[2:0]	Size. 000 8-bit core access 001 16-bit core access 010 32-bit core access 011 64-bit core access 1xx Reserved
3	PROT3	Cacheable 0 Non-cacheable 1 Cacheable
2	PROT2	Bufferable 0 Non-bufferable 1 Bufferable
1	PROT1	Mode 0 User mode 1 Supervisor mode
0	PROT0	Type 0 Program 1 Data

11.3.1.15 MCM Core Fault Data Register (CFDTR)

The CFDTR is a 32-bit read-only register that captures the data associated with the last faulted processor write data access from the XBS master bus. The CFDTR is valid only for faulted processor write accesses. The contents of this register are not valid if the last fault occurred on the core local bus (indicated by CFLOC[LocalErr] = 1). This register is not updated on processor read access faults. All writes are ignored.



1 The content of this register is undefined following $\overline{\text{RESET}}$ and until a core address fault occurs.

Figure 11-15. MCM Core Fault Data Register (CFDTR)

Table 11-18. CFDTR Field Descriptions

Bits	Name	Description
31–0	CFDTR[31:0]	Core fault data register. This register contains the data associated with the faulting access of the last processor write access. The register contains the data value taken directly from the device write data bus.

11.4 Initialization / Application Information

11.4.1 Using The PCT And REV Registers

In order to support software compatibility across MAC7100 Family devices (as well as other Freescale microcontrollers that implement the SPP/ISP on-chip architecture), two registers are provided to allow identification of the device that is executing code, thus allowing conditional execution of version-specific routines. Table 11-19 summarizes the five register fields that are available to identify the specific version of a device. Refer to Section 11.3.1.1, “MCM Processor Core Type Register (PCT),” and Section 11.3.1.2, “MCM Device Revision Register (REV),” for more detailed descriptions.

Table 11-19. Processor Type and Device Revision Summary

Processor Core Type Register (PCT)	
PCT[15:0]	Processor core type.
Device Revision Register (REV)	
FAMMAJ[3:0]	Family major identifier.
FAMMIN[3:0]	Family minor identifier.
MSKMAJ[3:0]	Mask major identifier.
MSKMIN[3:0]	Mask minor identifier.

Table 11-20. PCT and REV Values Summary

Mask Set[(1)]	PCT	REV
0L49P	0xA700	0x7110
1L49P	0xA700	0x7111
0L47W	0xA700	0x7120
1L47W	0xA700	0x7121
0L61W	0xA700	0x7130
0L38Y	0xA700	0x7140

1 These are the defined MAC devices at the time of release of this document (MAC7100RM Rev. 1.0 10/2004). As additional devices are defined, this table will be updated with new values.

Chapter 12
Enhanced Direct Memory Access Controller Module (eDMA)

12.1 Overview

The MAC7100 Family of devices include an implementing of the Enhanced Direct Memory Access controller (eDMA). This module is implemented on other Freescale devices such as those in the MPC5500 family. It enables transfer of data between the memory, peripherals and off-chip devices with little intervention from the core, thus helping to increase system performance as well as assisting with the simplification of software development.

The eDMA is a second-generation platform module capable of performing complex data transfers with minimal intervention from a host processor via 16 programmable channels. Intended for use as part of the Standard Product Platform (SPP), the hardware microarchitecture includes the eDMA engine which performs source and destination address calculations, and the actual data movement operations, along with a local memory containing the transfer control descriptors (TCD*n*) for the channels. This SRAM-based implementation is utilized to minimize the overall module size.

The eDMA has been developed to enable instantiation across a range of devices with different feature requirements. This allows the module to be reused both within families of devices and across a wide range of products. The implementation of the eDMA on the MAC7100 family is targeted towards cost sensitive applications while still maintaining a high level of functionality.

The MAC7100 family eDMA implements 16 independently programmable DMA channels. The eDMA enables the definition of transfers from memory to memory, from peripherals to memory, and from peripherals to peripherals.

Figure 12-1 shows a block diagram of the eDMA module and Table 12-1 shows all possible eDMA transfer sources.

As there are more than 16 sources of DMA requests, a multiplexor is used to define which of the sources are used and to which eDMA channels they are assigned. Refer to Chapter 17, “DMA Channel Multiplexer Module (DMAMux),” for further details.

The eDMA is a highly-programmable data transfer engine, which has been optimized to minimize the required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known, and is not defined within the data packet itself. The eDMA hardware supports:

- Connections to the crossbar switch (XBS) for bus mastering the data movement, and to the peripheral slave bus for programming the module
- 32-byte transfer control descriptor (TCD) per channel stored in module memory
- 32 bytes of data registers, used as temporary storage to support burst transfers

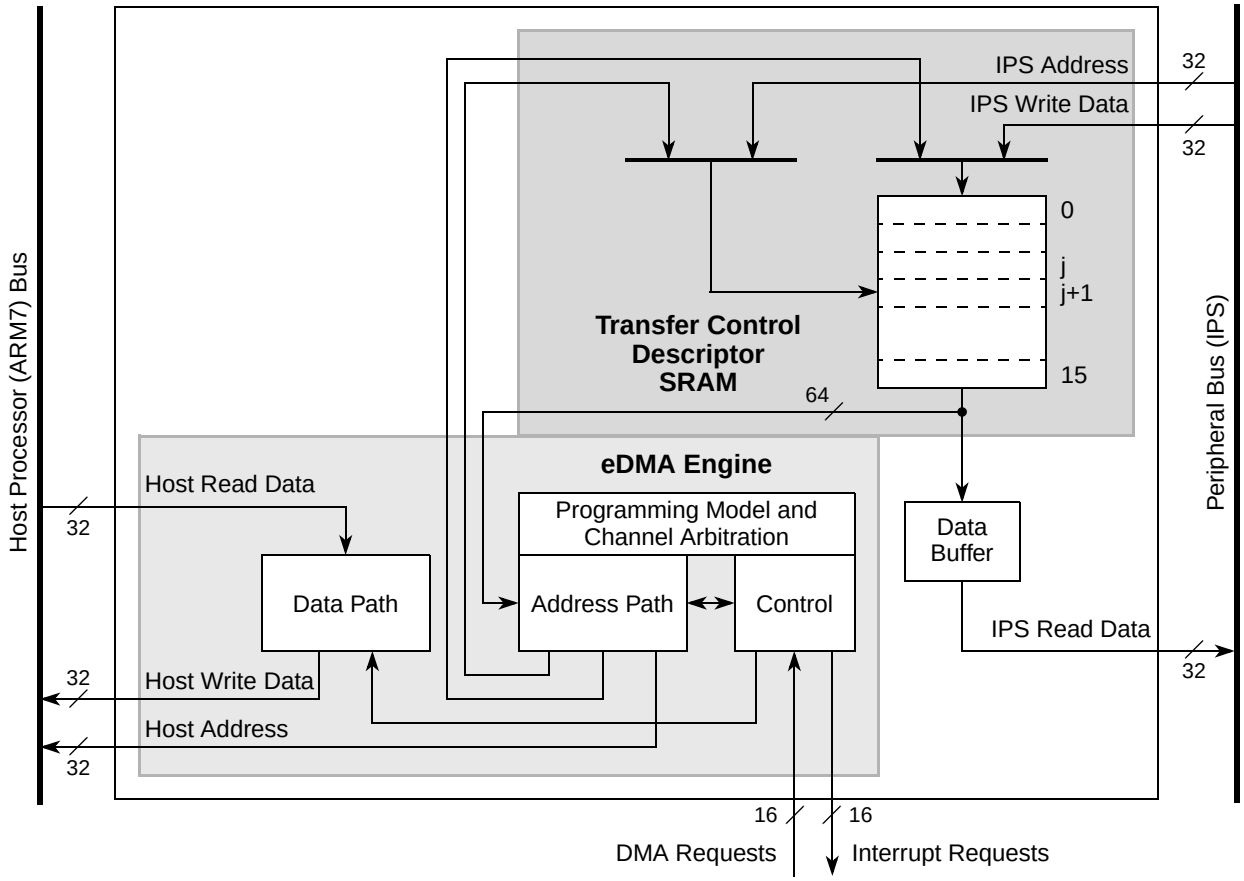


Figure 12-1. eDMA Block Diagram

Table 12-1. eDMA Request Sources

Type	Source [1]	DMA Requests	Comments
Memory	System RAM	—	Transfer between memory addresses
	Program Flash	—	
	Data Flash	—	
	External Bus Interface	—	
Peripherals	eSCI_A, eSCI_B, eSCI_C, eSCI_D	8	Two requests from each eSCI, (one for Tx, one for Rx)
	DSPI_A, DSPI_B	4	Two requests from each DSPI, (one for Tx, one for Rx)
	I^2C	2	Two requests (one for Tx, one for Rx)
	ATD_A, ATD_B	4	Two requests per ATD (one for command, one for result)
	eMIOS	16	One request for each timer channel
Triggered	PIT	8	Any source above can be “throttled” by a PIT channel
External	Port A, B, C, D, E, F, G and H	8	One request for each port, via the DMA Mux “always enabled” sources [2]

1 Not all sources are available on each version of MAC7100 Family devices. Refer to Table 1-1 on page 1-3 for the sources available on each device.

2 While the “always enabled” sources are intended primarily to service PIM ports, which do not have an explicit DMA request capability, they may be used for other general purpose functions such as timed transfers of data from one area of memory to another. Refer to Section 17.5.3, “Always Enabled DMA Request Sources.”

12.2 Features

The eDMA module supports the following features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addressees, transfer size, plus support for enhanced addressing modes
- Transfer control descriptor organized to support two-deep nested transfer operations
  - An inner data transfer loop defined by a “minor” byte transfer count
  - An outer data transfer loop defined by a “major” iteration count
  - Minor loop channel preemption [1]
  - Major and minor loop channel-to-channel linking [1]
- Channel activation via one of three methods:
  - Explicit software initiation
  - Peripheral-paced hardware requests (one per channel)
  - Channel-to-channel linking [1]
- Support for fixed-priority or round-robin [1] channel arbitration
- Channel completion reported via optional interrupt requests
  - One interrupt per channel, optionally asserted at completion of major iteration
  - Error termination interrupts are optionally enabled per channel, and logically summed to present a reduced number of error interrupt requests to the INTC
- Scatter/gather operation [1]

12.3 Memory Map / Register Definition

As shown in Figure 12-1, the eDMA programming model is partitioned into two sections, both mapped into the peripheral bus memory space: the first region defines a number of registers providing control functions, while the second region corresponds to the local transfer control descriptor memory. Table 12-2 shows the eDMA memory map.

Throughout this chapter, *n* is used to reference the channel number. For all unused or reserved register bits, reads return zeroes and writes are ignored. Accessing reserved addresses in the memory map will cause an access error. The eDMA module does not include any logic which provides access control. Rather, this function is supported using the standard access control logic provided by the AIPS module.

1. Channel preemption, channel linking, round-robin arbitration and scatter/gather features are not implemented on mask set L49P devices.

Table 12-2. eDMA Memory Map

eDMA Offset	Register Description			
0x0000	eDMA Control Register (DMACR)			
0x0004	eDMA Error Status Register (DMAES)			
0x0008	Reserved			
0x000C	Reserved		eDMA Enable Request Register (DMAERQ)	
0x0010	Reserved			
0x0014	Reserved		eDMA Enable Error Interrupt Registers (DMAEEI)	
0x0018	eDMA Set Enable Request Register (DMASERQ)	eDMA Clear Enable Request Register (DMACERQ)	eDMA Set Enable Error Interrupt Register (DMASEEI)	eDMA Clear Enable Error Interrupt Register (DMACEEI)
0x001C	eDMA Clear Interrupt Request Register (DMACINT)	eDMA Clear Error Register (DMACERR)	eDMA Set START Bit Register (DMASSRT)	eDMA Clear DONE Status Register (DMACDNE)
0x0020	Reserved			
0x0024	Reserved		eDMA Interrupt Request Register (DMAINT)	
0x0028	Reserved			
0x002C	Reserved		eDMA Error Registers (DMAERR)	
0x0030–0x00FC	Reserved			
0x0100	eDMA Channel 0 Priority Register (DCHPRI0)	eDMA Channel 1 Priority Register (DCHPRI1)	eDMA Channel 2 Priority Register (DCHPRI2)	eDMA Channel 3 Priority Register (DCHPRI3)
0x0104	eDMA Channel 4 Priority Register (DCHPRI4)	eDMA Channel 5 Priority Register (DCHPRI5)	eDMA Channel 6 Priority Register (DCHPRI6)	eDMA Channel 7 Priority Register (DCHPRI7)
0x0108	eDMA Channel 8 Priority Register (DCHPRI8)	eDMA Channel 9 Priority Register (DCHPRI9)	eDMA Channel 10 Priority Register (DCHPRI10)	eDMA Channel 11 Priority Register (DCHPRI11)
0x010C	eDMA Channel 12 Priority Register (DCHPRI12)	eDMA Channel 13 Priority Register (DCHPRI13)	eDMA Channel 14 Priority Register (DCHPRI14)	eDMA Channel 15 Priority Register (DCHPRI15)
0x0110–0x0FFC	Reserved			
0x1000–0x11FC	TCD0-TCD15			

12.3.1 Register Descriptions

12.3.1.1 eDMA Control Register (DMACR)

The 32-bit DMACR defines the basic operating configuration of the eDMA. In the MAC7100 family implementation, when multiple channels are active at the same time, the eDMA arbitrates channel execution using one of two available arbitration modes. Arbitration among the channels can be

programmed for either fixed or round-robin mode. [1] In fixed-priority arbitration, the active channel with the highest priority is executed. In round-robin mode, activated channels are cycled through without regard to priority.

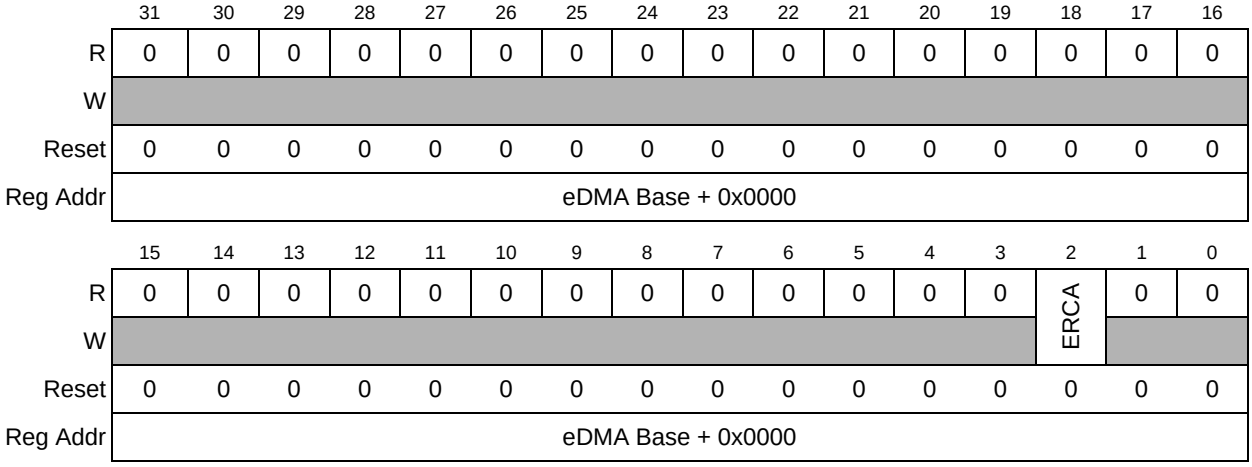


Figure 12-2. eDMA Control Register (DMACR)

Table 12-3. DMACR Field Descriptions

Bits	Name	Description
31–3	—	Reserved, must be written as zero.
2	ERCA	Enable Round-Robin Channel Arbitration [1] 0 Fixed-priority arbitration is used to determine the next channel to execute. 1 Round-robin arbitration is used to determine the next channel to execute.
1–0	—	Reserved, must be written as zero.

[1] Mask set L49P devices do not implement round-robin arbitration, and this bit must be written as zero.

12.3.1.2 eDMA Error Status Register (DMAES)

The DMAES register provides information concerning the last recorded channel error. Channel errors can be caused by a configuration error (an illegal setting in the transfer control descriptor or an illegal priority register setting in fixed arbitration mode) or an error termination to a bus master read or write cycle.

A configuration error is caused when the starting source or destination address, source or destination offsets, minor loop byte count and the transfer size represent an inconsistent state. The addresses and offsets must be aligned on modulo zero transfer size boundaries, and the minor loop byte count must be a multiple of the source and destination transfer sizes. All source reads and destination writes must be configured to the natural boundary of the programmed transfer size, respectively. In fixed arbitration mode, a configuration error is caused by any two channels being assigned the same priority, as all channel priority levels must be unique when fixed arbitration mode is enabled. All configuration error conditions except scatter/gather and minor loop link error are reported as the channel is activated and cause an error interrupt request to be asserted, if enabled. A scatter/gather configuration error is reported when a scatter/gather operation is enabled at major loop completion. If TCD*n*[DLAST_SGA] is not aligned on a

1. Round-robin arbitration is not implemented on mask set L49P devices, thus only fixed-priority mode is available.

32-byte boundary, a configuration error is reported. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion. If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the channel attempts to perform the link if TCD*n*[CITER_E_LINK] is not equal to TCD*n*[BITER_E_LINK].

If a system bus read or write is terminated with an error, the data transfer is stopped (bus operations already in the pipeline will complete) and the appropriate bus error flag set. In this case, the state of the channel TCD is updated by the eDMA engine with the current source address, destination address and minor loop byte count at the point of the fault.

The occurrence of any type of error causes the eDMA engine to immediately stop, and the appropriate channel bit in the eDMA error register to be asserted. At the same time, the details of the error condition are loaded into the DMAES register. The normal eDMA channel completion indicators, setting the transfer control descriptor done flag and the possible assertion of an interrupt request, are not affected when an error is detected.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eDMA Base + 0x0004															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	CPE	0	0	ERRCHN				SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eDMA Base + 0x0004															

Figure 12-3. eDMA Error Status Register (DMAES)

Table 12-4. DMAES Field Descriptions

Bits	Name	Description
31	VLD	Logical OR of all DMAERR status bits. 0 No DMAERR bits are set. 1 At least one DMAERR bit is set.
30–15	—	Reserved.
14	CPE	Channel priority error. 0 No channel priority error. 1 The last recorded error was a configuration error in the channel priorities. All channel priorities are not unique.
13–12	—	Reserved.
11–8	ERRCHN[3:0]	Error channel number. The channel number of the last recorded error (excluding CPE errors).

Table 12-4. DMAES Field Descriptions (continued)

Bits	Name	Description
7	SAE	Source address error. 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCD*n*[SADDR] field. TCD*n*[SADDR] is inconsistent with TCD*n*[SSIZE].
6	SOE	Source offset error. 0 No source offset configuration error. 1 The last recorded error was a configuration error detected in the TCD*n*[SOFF] field. TCD*n*[SOFF] is inconsistent with TCD*n*[SSIZE].
5	DAE	Destination address error. 0 No destination address configuration error. 1 The last recorded error was a configuration error detected in the TCD*n*[DADDR] field. TCD*n*[DADDR] is inconsistent with TCD*n*[DSIZE].
4	DOE	Destination offset error. 0 No destination offset configuration error. 1 The last recorded error was a configuration error detected in the TCD*n*[DOFF] field. TCD*n*[DOFF] is inconsistent with TCD*n*[DSIZE].
3	NCE	NBYTES/CITER configuration error 0 No NBYTES/CITER configuration error. 1 The last recorded error was a configuration error detected in the TCD*n*[NBYTES] or TCD*n*[CITER] fields: TCD*n*[NBYTES] is not a multiple of TCD*n*[SSIZE] and TCD*n*[DSIZE], or TCD*n*[CITER] = 0, or TCD*n*[CITER_E_LINK] is not equal to TCD*n*[BITER_E_LINK].
2	SGE	Scatter/Gather Configuration Error 0 No scatter/gather configuration error. 1 The last recorded error was a configuration error detected in TCD*n*[DLAST_SGA] field. At the beginning of a scatter/gather operation after major loop completion, TCD*n*[E_SG] is enabled and TCD*n*[DLAST_SGA] is not on a 32-byte boundary.
1	SBE	Source bus error 0 No source bus error. 1 The last recorded error was a bus error on a source read.
0	DBE	Destination bus error 0 No destination bus error. 1 The last recorded error was a bus error on a destination write.

12.3.1.3 eDMA Enable Request Register (DMAERQ)

The DMAERQ register provides a single register to enable the request signal for each channel. The state of all channel enables is directly affected by writes to this register. Individual channel enables may also be modified via the DMASERQ and DMACERQ registers, rather than performing a read-modify-write sequence to the DMAERQ register.

Both the eDMA request input signal and the corresponding enable request flag must be asserted before a channel is activated. The state of the eDMA enable request flag does not affect a channel activated through an explicit software initiation or a linked channel request.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERQ15	ERQ14	ERQ13	ERQ12	ERQ11	ERQ10	ERQ9	ERQ8	ERQ7	ERQ6	ERQ5	ERQ4	ERQ3	ERQ2	ERQ1	ERQ0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eDMA Base + 0x000E															

Figure 12-4. eDMA Enable Request Register (DMAERQ)

Table 12-5. DMAERQ Field Descriptions

Bits	Name	Description
15–0	ERQ*n*	Enable eDMA request *n*. 0 The eDMA request signal for channel *n* is disabled. 1 The eDMA request signal for channel *n* is enabled.

As a given channel completes the processing for the major iteration count, a flag in the transfer control descriptor may affect the ending state of the DMAERQ bit for that channel. If the TCD*n*[D_REQ] bit is set, then the corresponding DMAERQ bit is cleared, disabling the DMA request; else if the D_REQ bit is cleared, the state of the DMAERQ bit is unaffected.

12.3.1.4 eDMA Enable Error Interrupt Registers (DMAEEI)

The DMAEEI register provides a single register to enable the error interrupt for each channel. The state of all channel error interrupt enables are directly affected by writes to this register. Individual channel error interrupt enables may also be modified via the DMASEEI and DMACEEI registers, rather than performing a read-modify-write sequence to the DMAEEI register.

Both the eDMA error indicator and the corresponding error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI9	EEI8	EEI7	EEI6	EEI5	EEI4	EEI3	EEI2	EEI1	EEI0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eDMA Base + 0x0016															

Figure 12-5. eDMA Enable Error Interrupt Registers (DMAEEI)

Table 12-6. DMAEEI Field Descriptions

Bits	Name	Description
15–0	EEI*n*	Enable error interrupt *n*. 0 The error signal for channel *n* does not generate an error interrupt. 1 The error signal for channel *n* generates an interrupt request when asserted.

12.3.1.5 eDMA Set Enable Request Register (DMASERQ)

The DMASERQ register provides a simple memory-mapped mechanism to set specific ERQ*n* bit(s) in the DMAERQ register to enable the eDMA request for an individual channel. When the register is written, if the SAER bit is set then all bits in the DMAERQ register will be set. If the SAER bit is clear, then the value of SERQ selects the ERQ*n* bit of the DMAERQ register to be set.

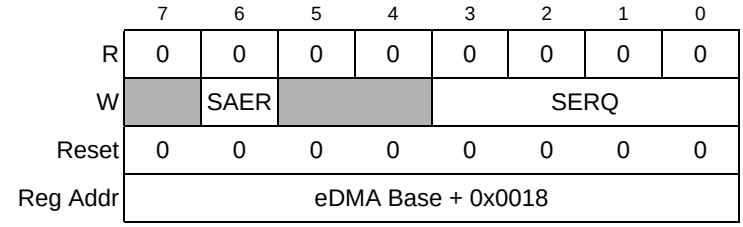


Figure 12-6. eDMA Set Enable Request Register (DMASERQ)

Table 12-7. DMASERQ Field Descriptions

Bits	Name	Description
7	—	Reserved, must be written as zero.
6	SAER	Set all enable requests. 0 Set the DMAERQ bit specified by the SERQ field 1 Set all bits in DMAERQ
5–4	—	Reserved, must be written as zero.
3–0	SERQ[3:0]	Set enable request *n.* 0b*nnnn*Set the corresponding bit in DMAERQ

12.3.1.6 eDMA Clear Enable Request Register (DMACERQ)

The DMACERQ register provides a simple memory-mapped mechanism to clear specific ERQ*n* bit(s) bit in the DMAERQ register to disable the eDMA request for a given channel. When the register is written, if the CAER bit is set then all bits in the DMAERQ register will be cleared. If the CAER bit is clear, then the value of CERQ selects the ERQ*n* bit of the DMAERQ register to be cleared.

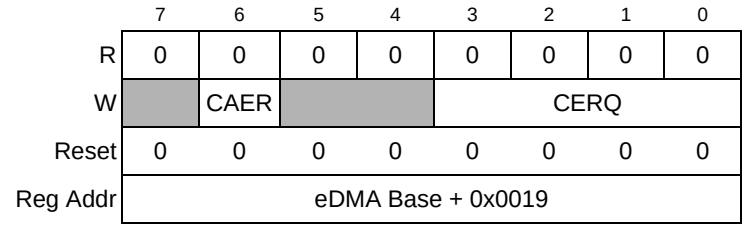


Figure 12-7. eDMA Clear Enable Request Register (DMACERQ)

Table 12-8. DMACERQ Field Descriptions

Bits	Name	Description
7	—	Reserved, must be written as zero.

Table 12-8. DMACERQ Field Descriptions (continued)

Bits	Name	Description
6	CAER	Clear all enable requests. 0 Clear the DMAERQ bit specified by the CERQ field 1 Clear all bits in DMAERQ
5–4	—	Reserved, must be written as zero.
3–0	CERQ[3:0]	Clear enable request *n.* 0b*nnnn*Clear the corresponding bit in DMAERQ

12.3.1.7 eDMA Set Enable Error Interrupt Register (DMASEEI)

The DMASEEI register provides a simple memory-mapped mechanism to set specific EEI*n* bit(s) in the DMAEEI register to enable the error interrupt for a given channel. When the register is written, if the SAEE bit is set then all bits in the DMAEEI register will be set. If the SAEE bit is clear, then the value of SEEI selects the EEI*n* bit of the DMAEEI register to be set.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W		SAEE			SEEI			
Reset	0	0	0	0	0	0	0	0
Reg Addr	eDMA Base + 0x001A							

Figure 12-8. eDMA Set Enable Error Interrupt Register (DMASEEI)

Table 12-9. DMASEEI Field Descriptions

Bits	Name	Description
7	—	Reserved, must be written as zero.
6	SAEE	Set all enable error interrupts. 0 Set the DMAEEI bit specified by the SEEI field 1 Set all bits in DMAEEI
5–4	—	Reserved, must be written as zero.
3–0	SEEI[3:0]	Set enable error interrupt *n*. 0b*nnnn*Set the corresponding bit in DMAEEI

12.3.1.8 eDMA Clear Enable Error Interrupt Register (DMACEEI)

The DMACEEI register provides a simple memory-mapped mechanism to clear specific EEI*n* bit(s) in the DMAEEI register to disable the error interrupt for a given channel. When the register is written, if the CAEE bit is set then all bits in the DMAEEI register will be cleared. If the CAEE bit is clear, then the value of CEEI selects the EEI*n* bit of the DMAEEI register to be cleared.

Figure 12-9. eDMA Clear Enable Error Interrupt Register (DMACEEI)

Table 12-10. DMACEEI Field Descriptions

Bits	Name	Description
7	—	Reserved, must be written as zero.
6	CAEE	Clear all enable error interrupts. 0 Clear the DMAEEI bit specified by the CEEI field 1 Clear all bits in DMAEEI
5–4	—	Reserved, must be written as zero.
3–0	CEEI[3:0]	Clear enable error interrupt *n*. 0b*nnnn*Clear the corresponding bit in DMAEEI

12.3.1.9 eDMA Clear Interrupt Request Register (DMACINT)

The DMACINT register provides a simple memory-mapped mechanism to clear specific INT*n* bit(s) in the DMAINT register to disable the interrupt request for a given channel. When the register is written, if the CAIR bit is set then all bits in the DMAINT register will be cleared. If the CAIR bit is clear, then the value of CINT selects the INT*n* bit of the DMAINT register to be cleared.

Figure 12-10. eDMA Clear Interrupt Request Register (DMACINT)

Table 12-11. DMACINT Field Descriptions

Bits	Name	Description
7	—	Reserved, must be written as zero.
6	CAIR	Clear all enable error interrupts. 0 Clear the DMAINT bit specified by the CINT field 1 Clear all bits in DMAINT
5–4	—	Reserved, must be written as zero.
3–0	CINT[3:0]	Clear individual interrupt request *n*. 0b*nnnn*Clear the corresponding bit in DMAINT

12.3.1.10 eDMA Clear Error Register (DMACERR)

The DMACERR register provides a simple memory-mapped mechanism to clear specific ERR*n* bit(s) in the DMAERR registers to disable the error condition flag for a given channel. When the register is written, if the CAER bit is set then all bits in the DMAERR register will be cleared. If the CAER bit is clear, then the value of CERR selects the ERR*n* bit of the DMAERR register to be cleared.

Figure 12-11. eDMA Clear Error Register (DMACERR)

Table 12-12. DMACERR Field Descriptions

Bits	Name	Description
7	—	Reserved, must be written as zero.
6	CAER	Clear all error indicators. 0 Clear the DMAERR bit specified by the CERR field 1 Clear all bits in DMAERR
5–4	—	Reserved, must be written as zero.
3–0	CERR[3:0]	Clear error indicator *n*. 0b*nnnn*Clear the corresponding bit in DMAERR

12.3.1.11 eDMA Set START Bit Register (DMASSRT)

The DMASSRT register provides a simple memory-mapped mechanism to set the TCD*n*[START] bit of specific channel(s). When the register is written, if the SAST bit is set then the START bits of all transfer control descriptors will be set. If the SAST bit is clear, then the value of SSRT selects the TCD*n* which will have the START bit set.

Figure 12-12. eDMA Set START Bit Register (DMASSRT)

Table 12-13. DMASSRT Field Descriptions

Bits	Name	Description
7	—	Reserved, must be written as zero.

Table 12-13. DMASSRT Field Descriptions

Bits	Name	Description
6	SAST	Set all START bits (activate all channels) 0 Set the TCD*n*[START] bit specified by the SSRT field 1 Set all TCD*n*[START] bits
5–4	—	Reserved, must be written as zero.
3–0	SSRT[3:0]	Set START bit for channel *n*. 0b*nnnn*Set the TCD*n*[START] bit

12.3.1.12 eDMA Clear DONE Status Register (DMACDNE)

The DMACDNE register provides a simple memory-mapped mechanism to clear the TCD*n*[DONE] bit of specific channel(s). When the register is written, if the CADN bit is set then the DONE bits of all transfer control descriptors will be cleared. If the CADN bit is clear, then the value of CDNE selects the TCD*n* which will have the DONE bit cleared.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W		CADN			CDNE			
Reset	0	0	0	0	0	0	0	0
Reg Addr	eDMA Base + 0x001F							

Figure 12-13. eDMA Clear DONE Status Register (DMACDNE)

Table 12-14. DMACDNE Field Descriptions

Bits	Name	Description
7	—	Reserved, must be written as zero.
6	CADN	Clear all DONE status bits 0 Clear the TCD*n*[DONE] bit specified by the CDNE field 1 Clear all TCD*n*[DONE] bits
5–4	—	Reserved, must be written as zero.
3–0	CDNE[3:0]	Clear DONE bit for channel *n*. 0b*nnnn*Clear the TCD*n*[DONE] bit

12.3.1.13 eDMA Interrupt Request Register (DMAINT)

The DMAINT register reflects the presence of an interrupt request for all channels. The eDMA engine signals the occurrence of a programmed interrupt upon the completion of a data transfer as defined in the transfer control descriptor by setting the appropriate bit in this register. The outputs of this register are routed to the INTC module (refer to Chapter 10, “Interrupt Controller Module (INTC),” and Table 6-2 on page 6-85 for more information). During the execution of the interrupt service routine associated with any given channel, it is required that software clear the appropriate bit, negating the interrupt request. Typically, a write to the DMACINT register in the interrupt service routine is used for this purpose.

The state of any channel interrupt request is directly affected by writes to this register; and also affected by writes to the DMACINT register. On writes to DMAINT, a one in any bit position clears the corresponding channel interrupt request. A zero in any bit position has no affect on the corresponding channel interrupt status. The DMACINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the DMAINT registers.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eDMA Base + 0x0026															

Figure 12-14. eDMA Interrupt Request Register (DMAINT)

Table 12-15. DMAINT Field Descriptions

Bits	Name	Description
15–0	INT*n*	eDMA interrupt request *n*. 0 The interrupt request for channel *n* is cleared. 1 The interrupt request for channel *n* is active.

12.3.1.14 eDMA Error Register (DMAERR)

The DMAERR register reflects the presence of an error for all channels. The eDMA engine signals the occurrence of a error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the DMAEEI register, then logically summed across all 16 channels to form a single error interrupt request to the INTC module (refer to Chapter 10, “Interrupt Controller Module (INTC),” and Table 6-2 on page 6-85 for more information). During the execution of the interrupt service routine associated with any eDMA errors, software must clear the appropriate bit, negating the error interrupt request. Typically, a write to the DMACERR register in the interrupt service routine is used for this purpose. The normal eDMA channel completion indicators (setting the transfer control descriptor done flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled, since a non-zero value indicates the presence of a channel error regardless of the state of the DMAEEI register. The state of channel error indicators are affected by writes to this register; and also affected by writes to the DMACERR register. On writes to the DMAERR, a one in any bit position clears the corresponding channel error status. A zero in any bit position has no affect on the corresponding channel current error status.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eDMA Base + 0x002E															

Figure 12-15. eDMA Error Registers (DMAERR)

Table 12-16. DMAERR Field Descriptions

Bits	Name	Description
15–0	ERR*n*	eDMA error *n*. 0 An error in channel *n* has not occurred. 1 An error in channel *n* has occurred.

12.3.1.15 eDMA Channel Priority Registers (DCHPRI*n*)

When the fixed-priority channel arbitration mode is enabled (DMACR[ERCA] = 0), the contents of DCHPRI*n* define the unique priorities associated with each channel. The channel priorities are evaluated by numeric value, i.e., 0 is the lowest priority, 1 is the next priority, then 2, 3, etc. Software must program the channel priorities with unique values, otherwise a configuration error will be reported when a channel is activated. The range of the priority value is limited to the values of 0 through 15.

Channel preemption is enabled on a per-channel basis by setting the DCHPRI*n*[ECP] bit. Channel preemption allows the executing channel data transfers to be temporarily suspended in favor of starting a higher priority channel. Once the preempting channel has completed all of its programmed minor loop transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, its eligible to be preempted again if any higher priority channel is activated. Multiple ECP bits may be set, but the eDMA engine will not perform nested preemption (attempting to preempt a preempting channel). Once a preempting channel begins execution, it cannot be preempted. Preemption is available in fixed arbitration mode only.

NOTE

Channel preemption is not implemented on mask set L49P devices.

	7	6	5	4	3	2	1	0
R	ECP	0	0	0	CHPRI			
W								
Reset	0	0	0	0	*n*			
Reg Addr	eDMA Base + 0x0100 + *n*							

Figure 12-16. eDMA Channel Priority Registers (DCHPRI*n*)

Table 12-17. DCHPRI*n* Field Descriptions

Bits	Name	Description
7	ECP	Enable Channel Preemption [1] 0 Channel *n* cannot be preempted by the activation of a higher priority channel. 1 Channel *n* can be preempted by the activation of a higher priority channel.
6–4	—	Reserved, must be written as zero.
3–0	CHPRI[3:0]	Channel *n* arbitration priority. Channel priority when fixed-priority arbitration is enabled. Set to the channel number on reset.

[1] Mask set L49P devices do not implement channel preemption, and this bit must be written as zero.

12.3.1.16 Transfer Control Descriptors (TCD*n*)

Each channel uses a 32-byte transfer control descriptor to define the desired data movement operation. The TCD structure was previously discussed in detail in Section 12.2, "Features." The channel descriptors are stored in the eDMA local SRAM in sequential order: channel 0, channel 1, ... channel 15. The descriptions of the TCD words are presented as eight 32-bit values. Table 12-18 and Figure 12-17 show the overview and details, respectively, of the TCD*n* structure.

NOTE

The only bits in the TCD local SRAM that are initialized during reset are the DONE, ACTIVE and START bits in each Word 7 (which are all cleared as shown in Figure 12-25). All other TCD fields are not initialized, therefore the entire TCD*n* must be written by software before enabling a channel.

Table 12-18. TCD*n* Memory Map Detail

eDMA Offset	TCD*n* Field Description	
0x1000 + (32 x *n*) + 0x00	Source Address	
0x1000 + (32 x *n*) + 0x04	Transfer Attributes	Signed Source Address Offset
0x1000 + (32 x *n*) + 0x08	Minor Byte Count	
0x1000 + (32 x *n*) + 0x0C	Last Source Address Adjustment	
0x1000 + (32 x *n*) + 0x10	Destination Address	
0x1000 + (32 x *n*) + 0x14	Current Minor Loop Link, Major Loop Count	Signed Destination Address Offset
0x1000 + (32 x *n*) + 0x18	Last Destination Address Adjustment/Scatter Gather Address	
0x1000 + (32 x *n*) + 0x1C	Beginning Minor Loop Link, Major Loop Count	Control and Status

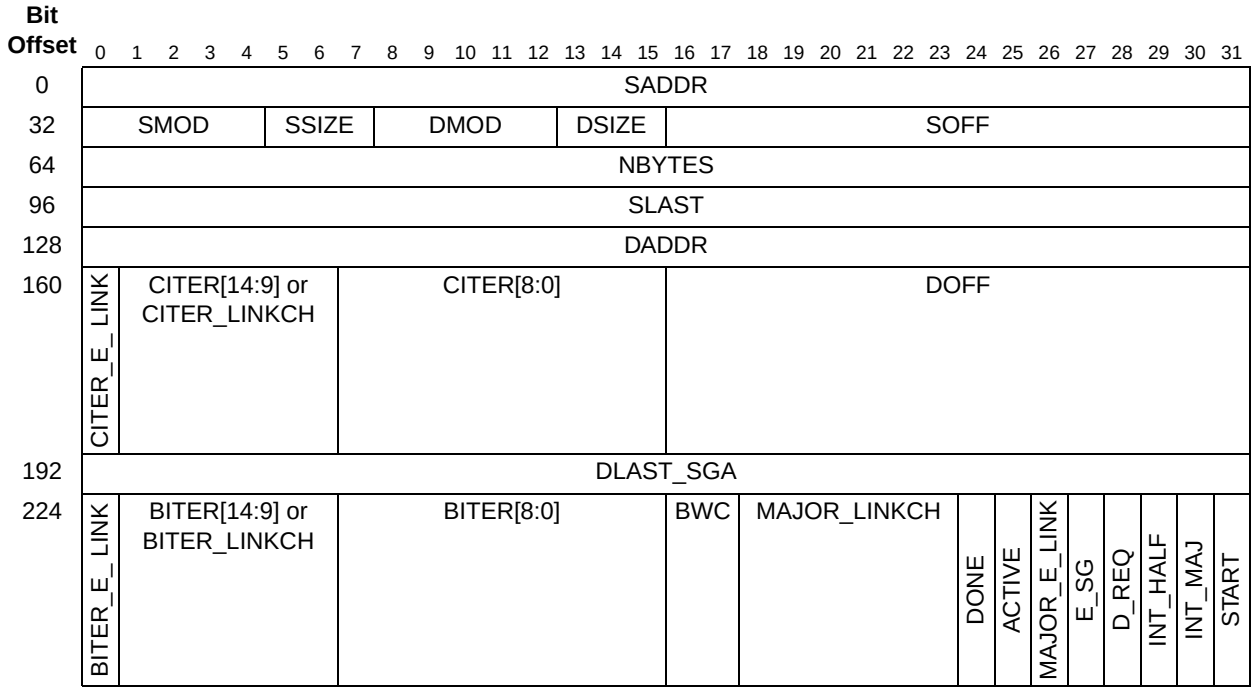


Figure 12-17. TCD*n* Structure Detail

12.3.1.16.1 Transfer Control Descriptor Word 0 (SADDR)

Figure 12-18 and Table 12-19 define word 0 of the TCD*n* structure, the SADDR field.

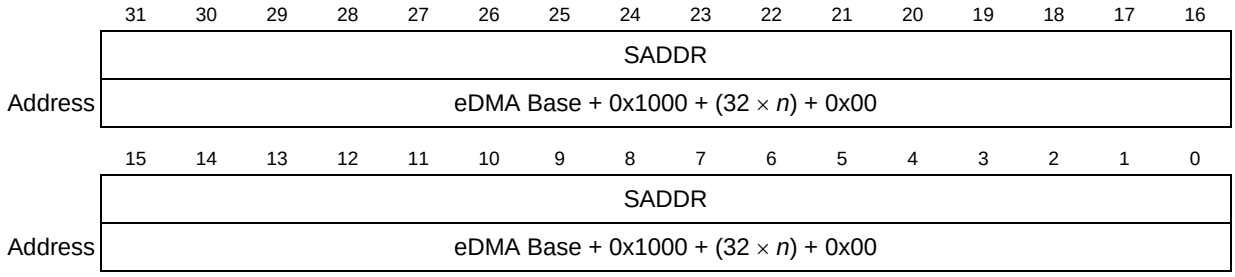


Figure 12-18. TCD*n* Word 0 (TCD*n*[SADDR])

Table 12-19. TCD*n*[SADDR] Field Description

Bits	Name	Description
31–0	SADDR[31:0]	Source address. Memory address of the source data.

12.3.1.16.2 Transfer Control Descriptor Word 1 (SMOD, SSIZE, DMOD, DSIZE, SOFF)

Figure 12-19 illustrates word 1 of the TCD*n* structure, the SOFF and transfer attribute fields.

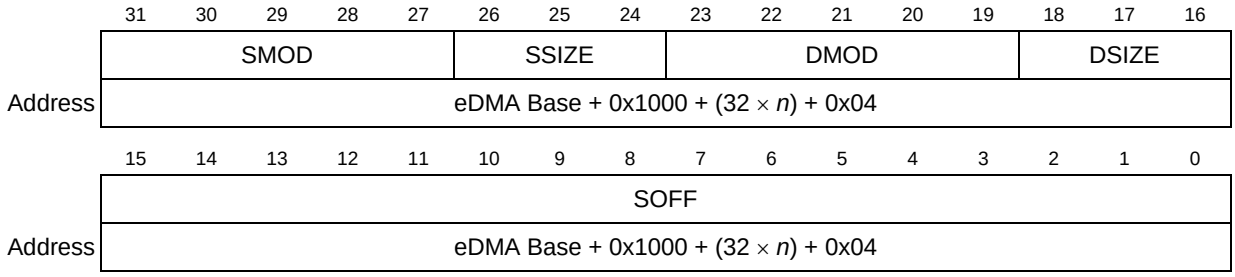


Figure 12-19. TCD*n* Word 0 (TCD*n*[SMOD, SSIZE, DMOD, DSIZE, SOFF])

Table 12-20. TCD*n*[SMOD, SSIZE, DMOD, DSIZE, SOFF] Field Descriptions

Bits	Name	Description
31–27	SMOD[4:0]	Source address modulo. 00000 Source address modulo feature is disabled. non-0 The value defines a specific address bit which is selected to be either the value after SADDR + SOFF calculation is performed, or the original register value. This feature provides the ability to easily implement a circular data queue. For data queues requiring power-of-2 “size” bytes, the queue should be based at a 0-modulo-size address and the SMOD field set to the appropriate size for the queue. The value loaded into this field specifies the number of lower address bits that are allowed to change, freezing the desired number of upper address bits. For a circular queue application, the SOFF field is typically set to the transfer size to implement post-increment addressing, with the SMOD value constraining the addresses to a 0-modulo-size range.

Table 12-20. TCD*n*[SMOD, SSIZE, DMOD, DSIZE, SOFF] Field Descriptions (continued)

Bits	Name	Description
26–24	SSIZE[2:0]	Source data transfer size. Loading any reserved value into this field will produce a configuration error when a channel activation is attempted. 000 8-bit. 001 16-bit. 010 32-bit. 011 Reserved. 100 16-byte burst. 101 Reserved. 110 Reserved. 111 Reserved.
23–19	DMOD[4:0]	Destination address modulo. See the SMOD definition.
18–16	DSIZE[2:0]	Destination data transfer size. See the SSIZE definition.
15–0	SOFF[15:0]	Source address signed offset. Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

12.3.1.16.3 Transfer Control Descriptor Word 2 (NBYTES)

Figure 12-20 illustrates word 2 of the TCD*n* structure, the NBYTES field.

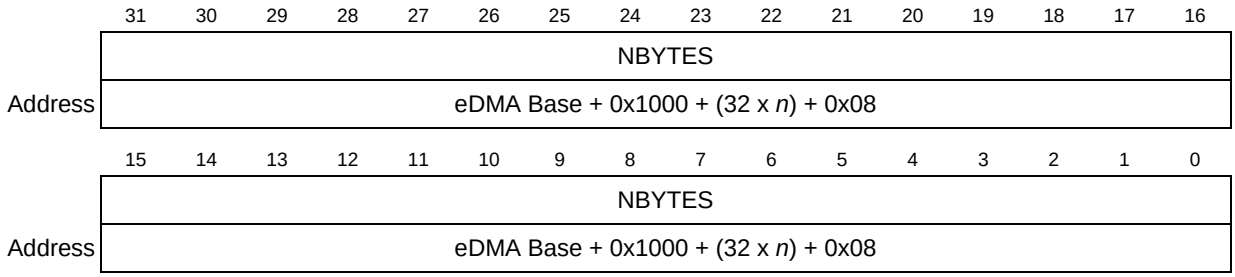


Figure 12-20. TCD*n* Word 2 (TCD*n*[NBYTES])

Table 12-21. TCD*n*[NBYTES] Field Description

Bits	Name	Description
31–0	NBYTES[31:0]	Minor byte transfer count. Number of bytes to be transferred on each activation of the channel. As a channel is activated, the contents of the appropriate TCD*n* is loaded into the eDMA engine, and the appropriate reads and writes performed until the complete byte transfer count has been transferred. This is an indivisible operation and cannot be halted. Once the minor count is exhausted, the current values of the SADDR and DADDR are written back into the local memory, the major iteration count is decremented and restored to the local memory. If the major iteration count is not exhausted, additional processing is performed.

12.3.1.16.4 Transfer Control Descriptor Word 3 (SLAST)

Figure 12-21 illustrates word 3 of the TCD*n* structure, the SLAST field.

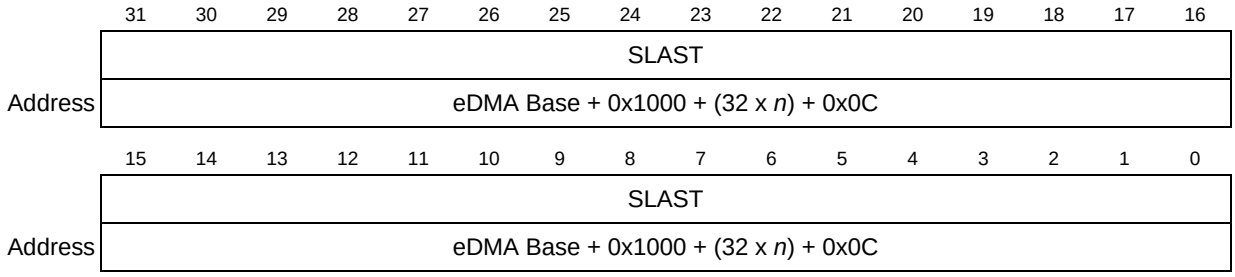


Figure 12-21. TCD*n* Word 3 (TCD*n*[SLAST])

Table 12-22. TCD*n*[SLAST] Field Description

Bits	Name	Description
31–0	SLAST[31:0]	Last source address adjustment. Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to “restore” the source address to the initial value, or adjust the address to reference the next data structure.

12.3.1.16.5 Transfer Control Descriptor Word 4 (DADDR)

Figure 12-22 illustrates word 4 of the TCD*n* structure, the DADDR field.

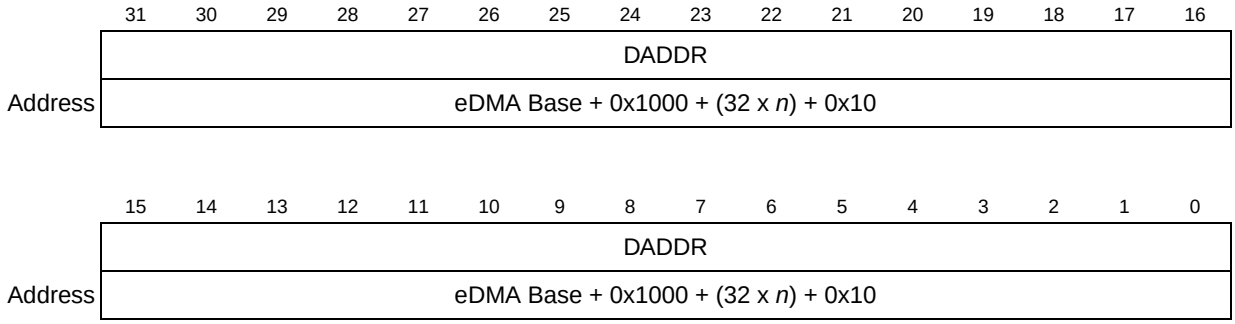


Figure 12-22. TCD*n* Word 4 (TCD*n*[DADDR])

Table 12-23. TCD*n*[DADDR] Field Description

Bits	Name	Description
31–0	DADDR[31:0]	Destination address. Memory address of the data destination.

12.3.1.16.6 Transfer Control Descriptor Word 5 (CITER, DOFF)

Figure 12-23 illustrates word 5 of the TCD*n* structure, the CITER and DOFF fields.

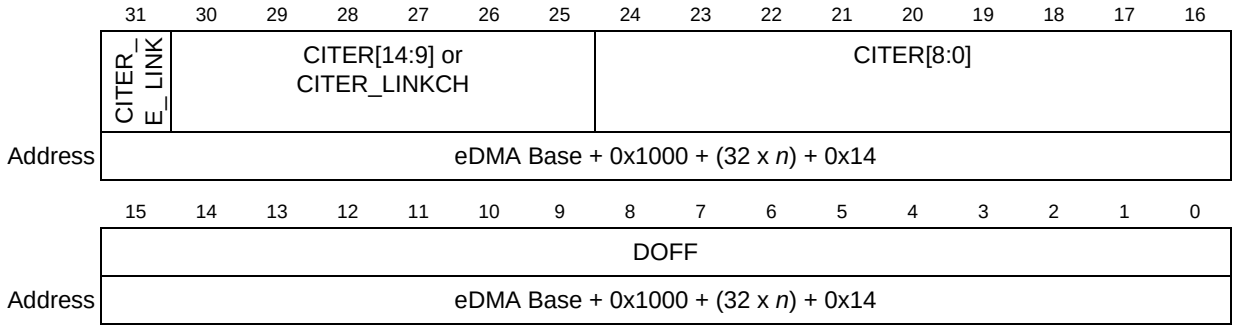


Figure 12-23. TCD*n* Word 5 (TCD*n*[CITER, DOFF])

Table 12-24. TCD*n*[DOFF, CITER] Field Descriptions

Bits	Name	Description
31 [1]	CITER_E_LINK [1]	Enable channel linking on minor loop complete. As the channel completes the minor loop, this flag enables linking to the channel defined by CITER_LINKCH[5:0]. The link target is activated by setting the TCD*n*[START] bit of the specified channel. If channel linking is disabled, a 15-bit iteration count is used rather a 6-bit link channel number and 9-bit iteration count. If the major loop is exhausted on completion of the minor loop, minor linking is suppressed in favor of the MAJOR_E_LINK channel linking. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled. NOTE: This bit must be equal to the BITER_E_LINK bit, otherwise a configuration error will be reported.
30–25 [1]	CITER_LINKCH[5:0] [1]	Minor loop complete link channel. If CITER_E_LINK is set, this 6-bit field specifies the channel that will be started following completion of the minor loop. 00xxxx Linked channel number. 01xxxx Reserved. 10xxxx Reserved. 11xxxx Reserved.
30–16 [1] or 24–16 [1]	CITER[14:0] [1] or CITER[8:0] [1]	Current major iteration count. This value represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. If CITER_E_LINK is clear, a 15-bit counter is used; if CITER_E_LINK is set, a 9-bit counter is used. [1] Once the major iteration count is exhausted, the channel performs a number of operations, such as final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the BITER field. When CITER is initially loaded, it must be set to the same value as that contained in BITER. If the channel is configured to perform a single activation, the initial values of BITER and CITER should be 0x0001.
15–0	DOFF[15:0]	Destination address signed offset. Sign-extended offset added to the current destination address to form the next value as each destination write is completed.

[1] Mask set L49P devices do not implement channel linking, and bit 31 must be written as zero. A 15-bit CITER count is always used.

12.3.1.16.7 Transfer Control Descriptor Word 6 (DLAST_SGA)

Figure 12-24 illustrates word 6 of the TCD*n* structure, the DLAST_SGA field.

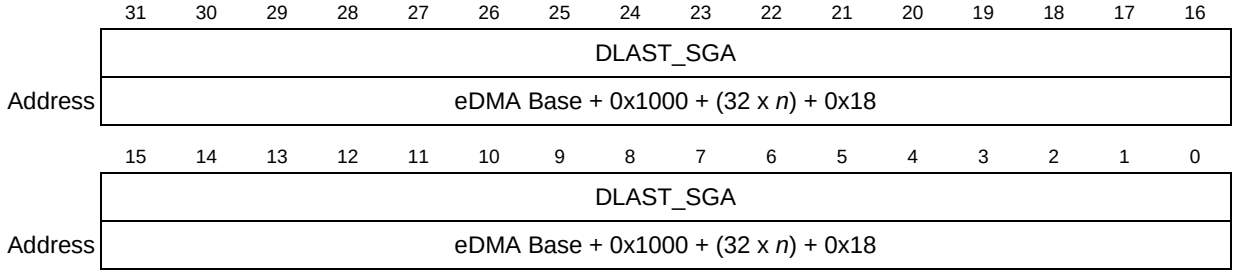


Figure 12-24. TCD*n* Word 6 (TCD*n*[DLAST_SGA])

Table 12-25. TCD*n*[DLAST_SGA] Field Description

Bits	Name	Description
31–0 [1]	DLAST_SGA [31:0] [1]	Last destination address adjustment or the memory address for the next TCD to be loaded for this channel (scatter/gather). If TCD*n*[E_SG] = 0 then the adjustment value is added to the destination address at the completion of the major iteration count. This value can be used to “restore” the destination address to the initial value, or adjust the address to reference the next data structure. If TCD*n*[E_SG] = 1, this address points to the beginning of a 0-modulo-32 region containing the next TCD to be loaded into this channel. The channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32 or a configuration error is reported.

[1] Mask set L49P devices do not implement scatter/gather, thus this field is always used for DLAST.

12.3.1.16.8 Transfer Control Descriptor Word 7 (BITER and Control/Status)

Figure 12-25 illustrates word 7 of the TCD*n* structure, the BITER and Control/Status fields.

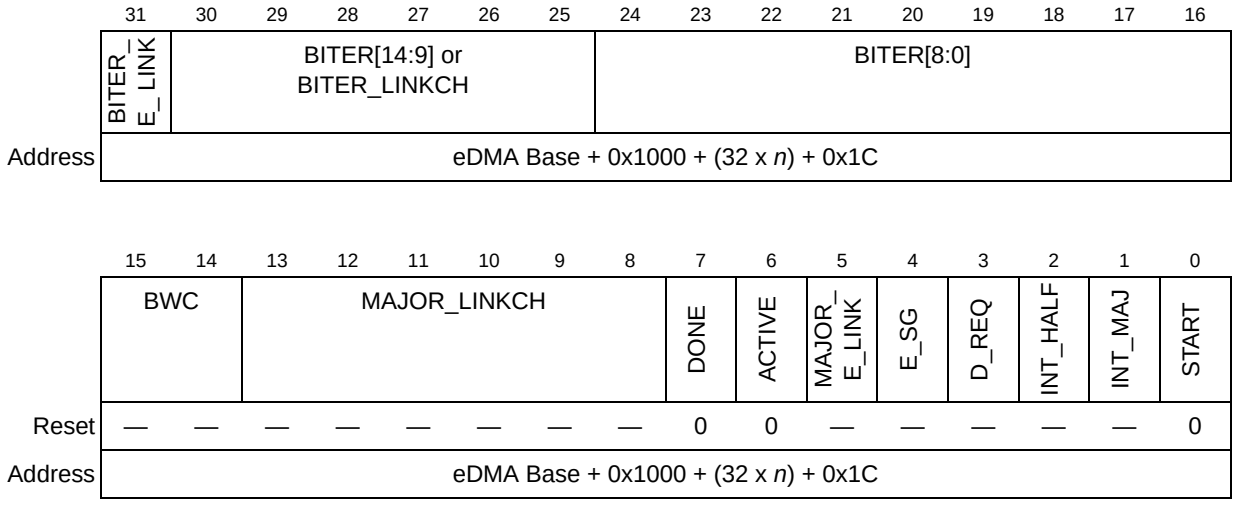


Figure 12-25. TCD*n* Word 7 (TCD*n*[BITER, Control/Status])

Table 12-26. TCD*n*[BITER, Control/Status] Field Descriptions

Bits	Name	Description
31 [1]	BITER_E_LINK [1]	Beginning enable channel linking on minor loop complete. When the TCD is first loaded, this bit must equal TCD*n*[CITER_E_LINK]. As the major iteration count is exhausted, the contents of this bit is reloaded into TCD*n*[CITER_E_LINK]. See TCD word 5 CITER_E_LINK for the full definition.
30–25 [1]	BITER_LINKCH[5:0] [1]	Beginning minor loop complete link channel. [1] When the TCD is first loaded, this field must equal TCD*n*[CITER_LINKCH]. As the major iteration count is exhausted, the contents of this field is reloaded into TCD*n*[CITER_LINKCH]. See TCD word 5 CITER_LINKCH for the full definition.
30–16 [1] or 24–16 [1]	BITER[14:0] [1] or BITER[8:0] [1]	Beginning major iteration count. [1] As the transfer control descriptor is first loaded, this field must be equal to the value in the CITER field. When the major iteration count is exhausted, the contents of this field is reloaded into TCD*n*[CITER]. See TCD word 5 CITER for bit definitions.
15–14	BWC[1:0]	Bandwidth control. This field provides a mechanism to control the amount of bus bandwidth consumed by the eDMA. In general, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field is used to force the eDMA to stall after the completion of each read/write access to control the bus request bandwidth utilized by the eDMA. 00 No eDMA engine stalls (consume 100% bandwidth) 01 Reserved. 10 eDMA engine stalls for 4 cycles after each read/write 11 eDMA engine stalls for 8 cycles after each read/write
13–8 [1]	MAJOR_LINKCH[5:0] [1]	Major loop complete link channel. If TCD[MAJOR_E_LINK] = 0 then no channel-to-channel linking is performed after the major loop counter is exhausted. If TCD[MAJOR_E_LINK] = 1 when the major loop counter is exhausted, the channel specified is activated by setting the TCD*n*[START] bit of that channel. 00xxxx Linked channel number. 11xxxx Reserved.
7	DONE	Channel done. This bit indicates the eDMA has completed the major loop. It is set by the eDMA engine as the CITER count reaches zero; it is cleared by software, or the hardware when the channel is activated.
6	ACTIVE	Channel active. This bit signals the channel is currently in execution. It is set as the execution of each activation of the minor loop begins, and is cleared by the eDMA engine as the minor loop completes or a configuration error is detected.
5 [1]	MAJOR_E_LINK [1]	Enable channel-to-channel linking on major loop complete. As the channel completes the major loop, this flag enables linking to another channel, defined by MAJOR_LINKCH[5:0]. The link target channel is activated by an internal mechanism that sets the TCD*n*[START] bit of the specified channel. In order to set this bit, the TCD*n*[DONE] bit must be clear, as described in Section 12.5.9, “Dynamic Channel Linking and Scatter/Gather Operation.” 0 Channel-to-channel linking is disabled. 1 Channel-to-channel linking is enabled.

Table 12-26. TCD*n*[BITER, Control/Status] Field Descriptions (continued)

Bits	Name	Description
4 [1]	E_SG [1]	Enable scatter/gather processing. As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLAST_SGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure which is loaded as the next TCD for this channel. In order to set this bit, the TCD*n*[DONE] bit must be clear, as described in Section 12.5.9, "Dynamic Channel Linking and Scatter/Gather Operation." 0 Scatter/gather processing is disabled. 1 Scatter/gather processing is enabled.
3	D_REQ	Disable request. If this bit is set, the eDMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero. 0 The channel's DMAERQ bit is not affected. 1 The channel's DMAERQ bit is cleared when the major loop is complete.
2	INT_HALF	Enable an interrupt when major counter is half complete. If this bit is set, the channel generates an interrupt request by setting the appropriate bit in the DMAINT register when the current major iteration count reaches the halfway point (specifically, when CITER = BITER ÷ 2). This halfway point interrupt request is provided to support double-buffered or other types of data movement schemes where the processor needs an early indication of the transfers progress. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled.
1	INT_MAJ	Enable an interrupt when major iteration count completes. If this bit is set, the channel generates an interrupt request by setting the appropriate bit in the DMAINT register when the current major iteration count reaches zero. 0 The end-of-major-loop interrupt is disabled. 1 The end-of-major-loop interrupt is enabled.
0	START	Channel start. If this bit is set, the channel is activated. The eDMA hardware automatically clears this bit after the channel is active. 0 The channel is not explicitly started. 1 The channel is explicitly activated when this bit is written as a one.

[1] Mask set L49P devices do not implement channel linking or scatter/gather; thus bits 31, 13–8, 5 and 4 must be written as zero. A 15-bit BITER count is always used.

12.4 Functional Description

12.4.1 eDMA Microarchitecture

As shown in Figure 12-1, the eDMA module is partitioned into two major modules: the eDMA engine and the transfer control descriptor local memory. Additionally, the eDMA engine is further partitioned into four submodules, which are detailed below.

- **eDMA Engine**
  - *Address Path*: This block implements registered versions of two channel transfer control descriptors: channel "x" and channel "y," and handles all master bus address calculations. All channels provide the exact same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. Once a channel is

activated, it runs until the minor loop is completed unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution. [1]

When any channel is selected to execute, the contents of its TCD is read from the local memory and loaded into the address path channel "x" registers for a normal start and into channel "y" registers for a preemption start. Once the minor loop completes execution, the address path hardware writes the new values for TCD*n*[SADDR, DADDR, CITER] back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*[CITER] field, channel linking if required, and fetching of the next TCD*n* from memory as part of a scatter/gather operation if required. [1]

— *Data Path*: This block implements the bus master read/write datapath. It includes 32 bytes of register storage and the necessary multiplex logic to support any required data alignment. The host read data bus is the primary input, and the host write data bus is the primary output.

The address and data path blocks directly support the 2-stage pipelined host bus. The address path block represents the 1st stage of the bus pipeline (the address phase), while the data path block implements the 2nd stage of the pipeline (the data phase).

— *Programming Model and Channel Arbitration*: This block implements the first section of eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the peripheral bus (not shown). The DMA request inputs and interrupt request outputs are also connected to this block (via the control logic).

— *Control*: This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read / destination write operations until the number of bytes specified in the minor loop byte count has been moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two 16-bit reads are performed followed by one 32-bit write.

- **Transfer Control Descriptor SRAM**
  — *Memory Controller*: This logic implements the required dual-ported controller, handling accesses from both the eDMA engine as well as references from the peripheral bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
  — *Memory Array*: TCD storage is implemented using a single-port, synchronous RAM array.

12.4.2 eDMA Basic Data Flow

The basic flow of a data transfer can be partitioned into three parts. As shown in Figure 12-26, the first part involves the channel activation. In the diagram, this example uses the assertion of a DMA request signal to activate channel *n*. Channel activation via software and the TCD*n*[START] bit follows the same basic flow as an that for peripheral requests. The DMA request input signal is registered internally and then routed to through the eDMA engine, first through the control module, then into the programming model and channel arbitration. In the next cycle, the channel arbitration is performed, either using the

1. Mask set L49P devices do not implement channel preemption or scatter/gather.

fixed-priority or round-robin [1] algorithm. After the arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine address path channel “x” or “y” registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel “x” or “y” registers.

In the second part of the basic data flow shown in Figure 12-27, the modules associated with the data transfer (address path, data path and control) sequence through the required source reads and destination writes to perform the data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the host bus during the destination write. This source read/destination write processing continues until the minor byte count has been transferred.

Once the minor byte count has been moved, the final phase of the basic data flow is performed. In this part, the address path logic performs the required updates to certain fields in the appropriate TCD*n*, e.g., SADDR, DADDR, CITER. If the major iteration count is exhausted, then there are additional operations which are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Additionally, assertion of an optional interrupt request occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor. [2] The updates to the TCD memory and the assertion of an interrupt request are shown in Figure 12-28.

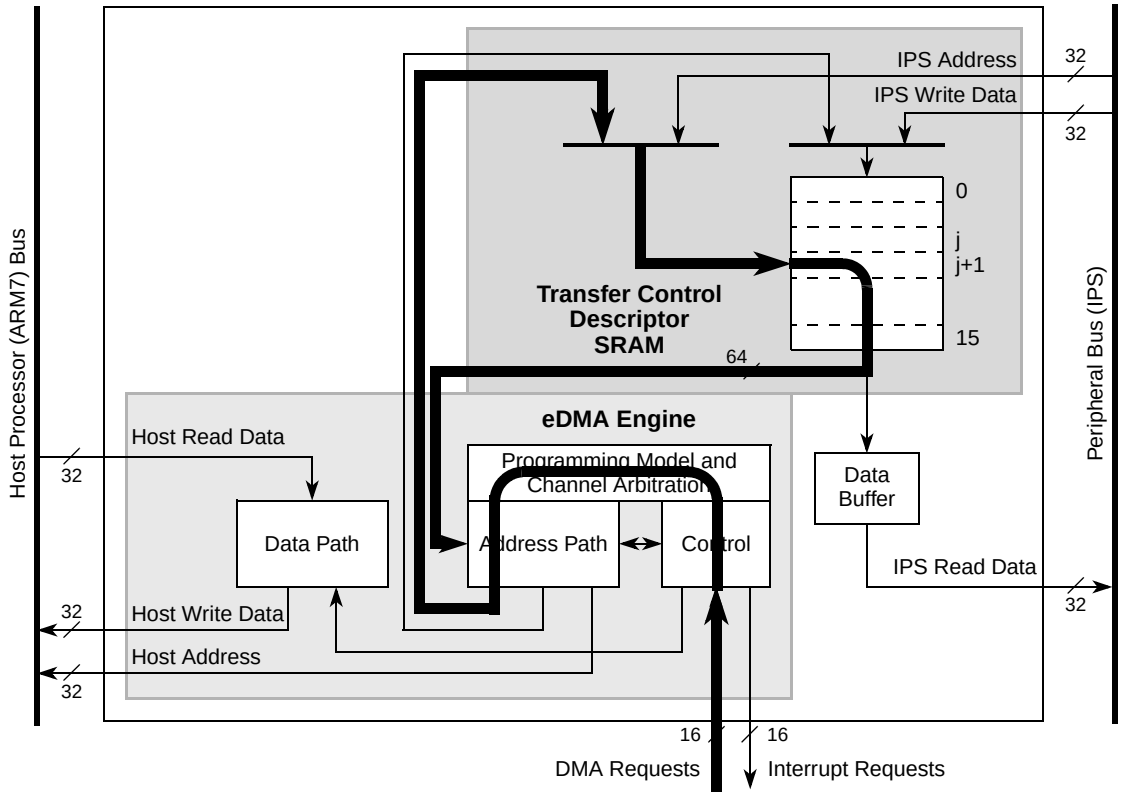


Figure 12-26. eDMA Operation – Part 1

1. Mask set L49P devices do not implement round-robin arbitration.
2. Mask set L49P devices do not implement scatter/gather.

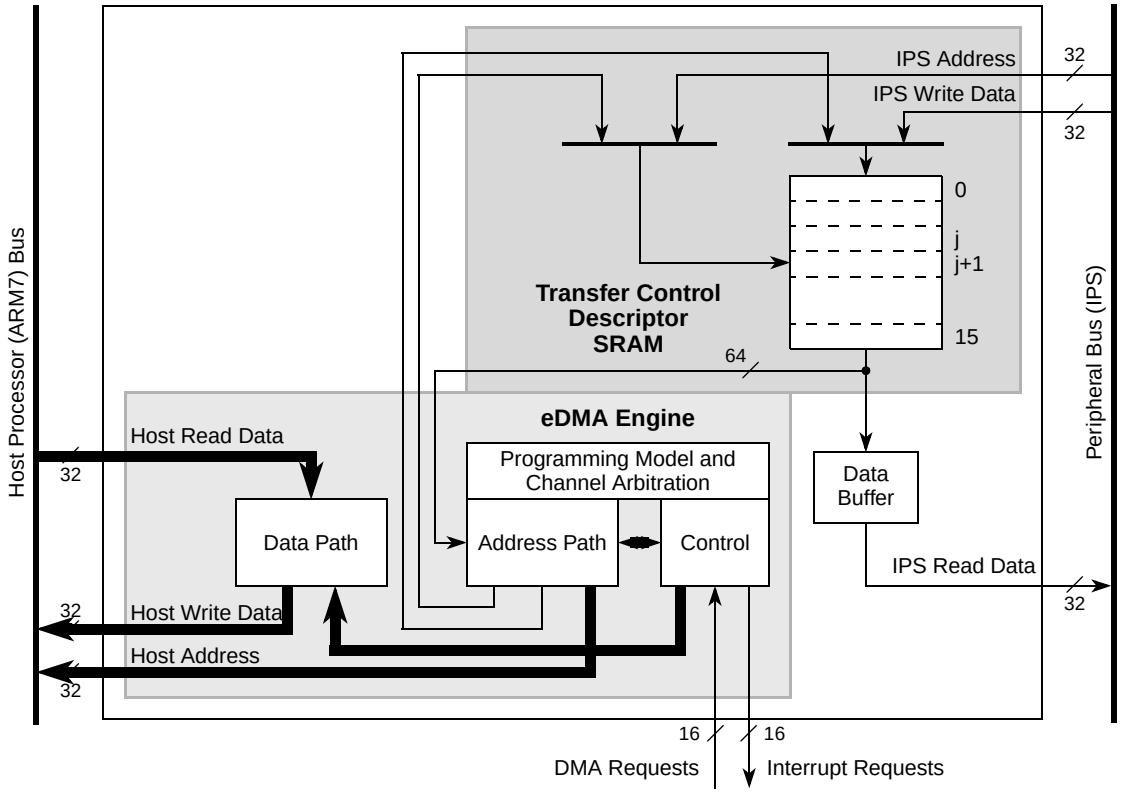


Figure 12-27. eDMA Operation – Part 2

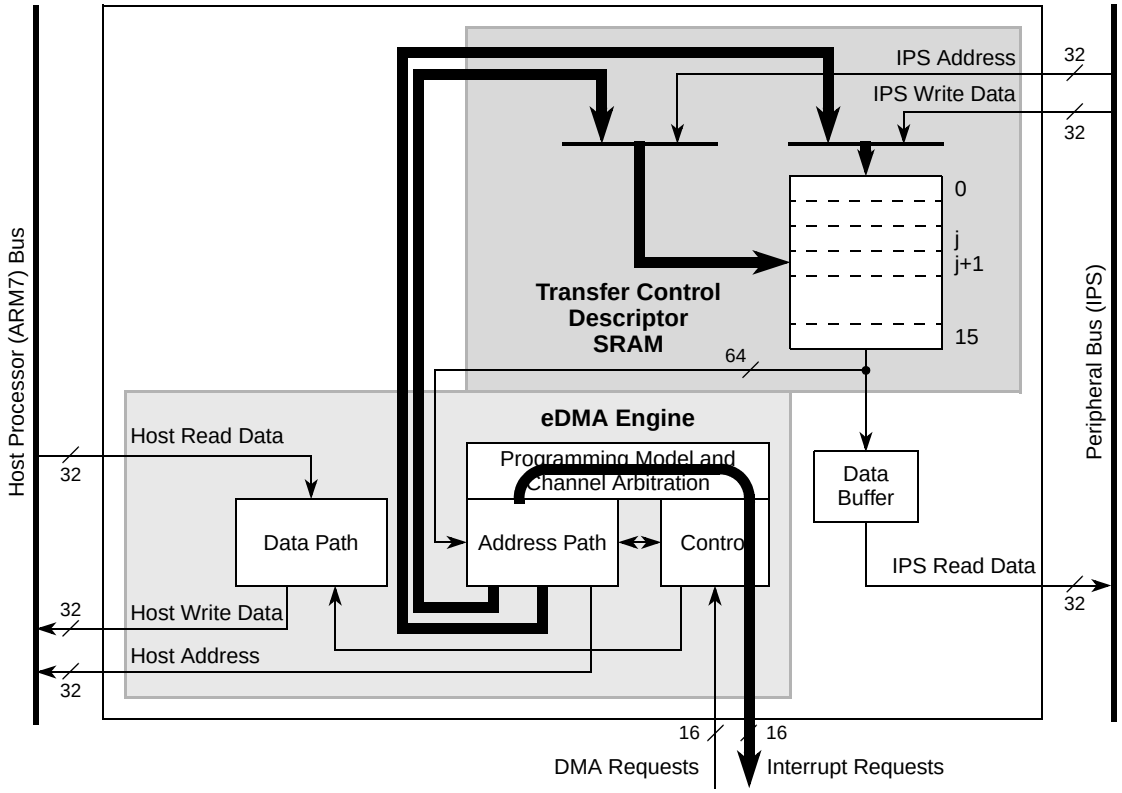


Figure 12-28. eDMA Operation – Part 3

12.5 Initialization / Application Information

12.5.1 eDMA Transfer Control Descriptor Header File Example

The structure of the transfer control descriptor is fundamental to the operation of the eDMA module. For more details, consult Section 12.3.1.16, “Transfer Control Descriptors (TCDn).” An example of using a C language specification to define the structure of each TCD is shown below (note that “`int`” refers to a 32-bit variable unless noted otherwise, and “`short`” is a 16-bit variable):

```
typedef struct {
  vuint32_t saddr          : 32;// Source Address
  vuint32_t smod           :  5;// Source Address Modulo
  vuint32_t ssize          :  3;// Source Data Transfer Size
  vuint32_t dmod           :  5;// Destination Address Modulo
  vuint32_t dsize          :  3;// Destination Data Transfer Size
  vuint32_t soff           : 16;// Source Address Signed Offset
  vuint32_t nbytes         : 32;// Minor byte transfer count
  vuint32_t slast          : 32;// Last source address adjustment
  vuint32_t daddr          : 32;// Destination Address
  vuint32_t citer_e_link   :  1;// Enable minor loop channel linking
  vuint32_t citer_linkch   :  6;// Minor loop compl link ch or CITER[14:9]
  vuint32_t citer          :  9;// Current major iteration count [8:0]
  vuint32_t doff           : 16;// Destination address signed offset
  vuint32_t dlast_sga      : 32;// Last dest or scatter/gather addr adjust
  vuint32_t biter_e_link   :  1;// Beginning enable minor loop ch linking
  vuint32_t biter_linkch   :  6;// Begin minor loop compl link ch/BITER[14:9]
  vuint32_t biter          :  9;// Beginning major iteration count [8:0]
  vuint32_t bwc            :  2;// Bandwidth control
  vuint32_t major_linkch   :  6;// Major loop complete link channel
  vuint32_t done           :  1;// Channel done
  vuint32_t active         :  1;// Channel active
  vuint32_t major_e_link   :  1;// Enable major loop complete channel link
  vuint32_t e_sg           :  1;// Enable scatter/gather processing
  vuint32_t d_req          :  1;// Disable request
  vuint32_t int_half       :  1;// Enable int when maj ctr is half complete
  vuint32_t int_maj        :  1;// Enable int when maj iteration completes
  vuint32_t start          :  1;// Channel start
} TCD;
```

Note: Mask set L49P devices do not implement channel linking or scatter/gather; thus citer_e_link, biter_e_link, major_linkch, major_e_link and e_sg must be written as zero. citer_linkch / citer and biter_linkch / biter are concatenated to form 15-bit major loop iteration counters.

Figure 12-29. eDMA Transfer Control Descriptor Header File Example

12.5.2 eDMA Basic Channel Operation

The basic operation of a channel is defined as:

1. The channel is initialized by software loading EDMA_TCD*n* into the eDMA programming model.
2. The channel is activated, either explicitly by software or a peripheral request.
3. The contents of the transfer control descriptor for the activated channel is read from the local memory and loaded into the eDMA engine registers.
4. The eDMA engine executes the data transfer defined by the minor loop, reading from the source and writing to the destination.

   One iteration of the major loop is executed per activation. One iteration of the major loop is equal to the complete execution of the minor loop. The number of iterations in the minor loop varies, and is computed by dividing NBYTES by the larger of SSIZE and DSIZE (e.g.: NBYTES = 16, source size is byte, destination size is word; minor loop iterations = 16 ÷ max(1, 4) resulting in four iterations of the sequence [read byte – read byte – read byte – read byte – write word]).
5. At the conclusion of the minor loop execution, certain fields of the transfer control descriptor are restored to the local memory.

The process (steps 2-5) is repeated until the major loop iteration count is exhausted. At that time, additional processing steps are completed (e.g., the optional assertion of an interrupt request signaling the transfer completion, final adjustments to the source and destination addresses, etc.) A more detailed description of the channel processing is listed in the pseudo-code below. This simplified example is intended to represent the most basic of data transfers, where the source and destination sizes are equal. Additionally, the detailed processing associated with the error handling is omitted.

```
/* the given eDMA channel is activated by the software assertion of the
   tcd[channel].start bit or the assertion of an enabled request from a device */

/* begin by reading the transfer control descriptor from the local RAM
   into the local eDMA engine registers */
edma = read_from_local_memory [channel];

/* check the transfer control descriptor for consistency */
if (edma.config_error == 0) {

    / * begin execution of the minor loop transfers */
    while (edma.nbytes > 0) {
        edma.active = 1;         /* set   active flag */
        edma.done   = 0;         /* clear done   flag */

        /* process the source address read */
        /* convert the source transfer size into a byte count */
        switch (edma.ssize) {
        case 0:                          /*  8-bit transfer */
            xfr_size = 1;
            break;
        case 1:                          /* 16-bit transfer */
```

```
            xfr_size = 2;
            break;
        case 2:                         /* 32-bit transfer */
            xfr_size = 4;
            break;
        case 3:                         /* 16-byte burst transfer */
            xfr_size = 16;
            break;
        }

        /* read "xfr_size" bytes from the source operand */
        edma.data = read_from_host (edma.saddr, xfr_size);

        /* generate the next-state source address */
        /* sum the current saddr with the signed source offset */
        ns_addr = edma.saddr + (int) edma.soff;

/* if enabled, apply the power-of-2 modulo to the next-state source addr */
        if (edma.smod != 0) {
            /* modulo addressing is enabled */
            address_select = (1 << edma.smod) - 1;
        } else
            address_select = 0xffff_ffff;
        edma.saddr = ns_addr           &  address_select
                   | edma.saddr & ~address_select;

        /* process the destination address write */
        /* convert the destination transfer size into a byte count */
        switch (edma.dsize) {
        case 0:                         /*  8-bit transfer */
            xfr_size = 1;
            break;
        case 1:                         /* 16-bit transfer */
            xfr_size = 2;
            break;
        case 2:                         /* 32-bit transfer */
            xfr_size = 4;
            break;
        case 3:                         /* 16-byte burst transfer */
            xfr_size = 16;
            break;
        }

        /* write "xfr_size" bytes to the destination operand */
        write_to_host (edma.daddr, xfr_size) = edma.data;

        /* generate the next-state destination address */
        /* sum the current daddr with the signed destination offset */
        ns_addr = edma.daddr + (int) edma.doff;
```

```
/* if enabled, apply the power-of-2 modulo to the next-state dst addr */
        if (edma.dmod != 0) {
            /* modulo addressing is enabled */
            address_select = (1 << edma.dmod) - 1;
        } else
            address_select = 0xffff_ffff;
        edma.daddr = ns_addr           &  address_select
                   | edma.daddr & ~address_select;

/* the bandwidth control field determines when the next read/write occurs */
        if (edma.bwc > 1)
            stall_edma (1 << edma.bwc);

        /* decrement the minor loop byte count */
        edma.nbytes = edma.nbytes - xfr_size;

    }   /* end of minor loop */

    edma.citer--;                    /* decrement major loop iteration count */

/* if the major loop is not yet exhausted, update certain TCD values in the RAM */
    if (edma.citer != 0) {
        write_to_local_memory [channel].saddr = edma.saddr;
        write_to_local_memory [channel].daddr = edma.daddr;
        write_to_local_memory [channel].citer = edma.citer;

        /* if minor loop linking is enabled, make the channel link */
        if (edma.citer_e_link)
            tcd[citer_linkch].start = 1;     /* activate the specified channel */

/* check for interrupt assertion if half of the major iterations are done */
        if (edma.int_half && (edma.citer == (edma.biter >> 1)))
            generate_interrupt (channel);

        edma.active = 0;         /* clear the channel busy flag */
     } else { /* major loop is complete, edma.citer == 0 */
/* since the major loop is complete, perform the final address adjustments */

       /* sum the current {src,dst} addresses with "last" adjustment */
       write_to_local_memory [channel].saddr = edma.saddr + edma.slast;
       write_to_local_memory [channel].daddr = edma.daddr + edma.dlast;
       /* restore the major iteration count to the beginning value */
       write_to_local_memory [channel].citer = edma.biter;

       /* check for interrupt assertion at completion of the major iteration */
       if (edma.int_maj)
         generate_interrupt (channel);
```

```
/* check if the request is to be disabled at completion of the major iteration */
       if (edma.d_req)
           dmaerq [channel] = 0;

       edma.active = 0;         /* clear the channel busy flag */
       edma.done = 1;           /* set the channel done flag */

} else {
       /* configuration error detected, abort the channel */
       edma.error_status = error_type; /* record the error */
       edma.active = 0;                /* clear the channel busy flag */
       /* check for interrupt assertion on error */
       if (edma.int_err)
           generate_interrupt (channel);
}
```

12.5.3 eDMA Initialization Sequence

A typical initialization of the eDMA might have the following sequence.

1. Write the EDMA_CR if a configuration other than default is desired.
2. Write channel priority levels into EDMA_CPR*n* if a configuration other than default is desired.
3. Enable error interrupts in the EDMA_EEIR if required.
4. Write the 32 byte TCD*n* for each channel that may request service.
5. Enable any hardware service requests via the EDMA_ERQR.
6. Request channel service either by software (setting the TCD*n*[START] bit) or by hardware (slave device asserting a eDMA request signal).

Once any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine will read the entire TCD, including the primary transfer control parameters shown in Table 12-27, for the selected channel into its internal address path module. As the TCD is being read, the first transfer is initiated on the host bus unless a configuration error is detected. Transfers from the source (as defined by the source address, TCD*n*[SADDR]) to the destination (as defined by the destination address, TCD*n*[DADDR]) continue until the specified number of bytes (TCD*n*[NBYTES]) have been transferred. When the transfer is complete, the eDMA engine copies of TCD*n*[SADDR], TCD*n*[DADDR], and TCD*n*[CITER] are written back to the TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing is executed, i.e. interrupts, major loop channel linking, and scatter/gather operations, if enabled.

Table 12-27. eDMA TCD Primary Control and Status Fields

TCD*n* Field Name	Description
START	Control bit to explicitly start channel when using a software initiated DMA service (automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution

Table 12-27. eDMA TCD Primary Control and Status Fields (continued)

TCD*n* Field Name	Description
DONE	Status bit indicating major loop completion (must be cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a software initiated DMA service
BWC	Control bits for “throttling” bandwidth of a channel
E_SG [1]	Control bit to enable scatter/gather processing
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

[1] Not available on mask set L49P devices.

Figure 12-30 shows how each DMA request initiates one minor loop transfer (iteration) without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

Figure 12-30. eDMA Example – Multiple Loop Iterations

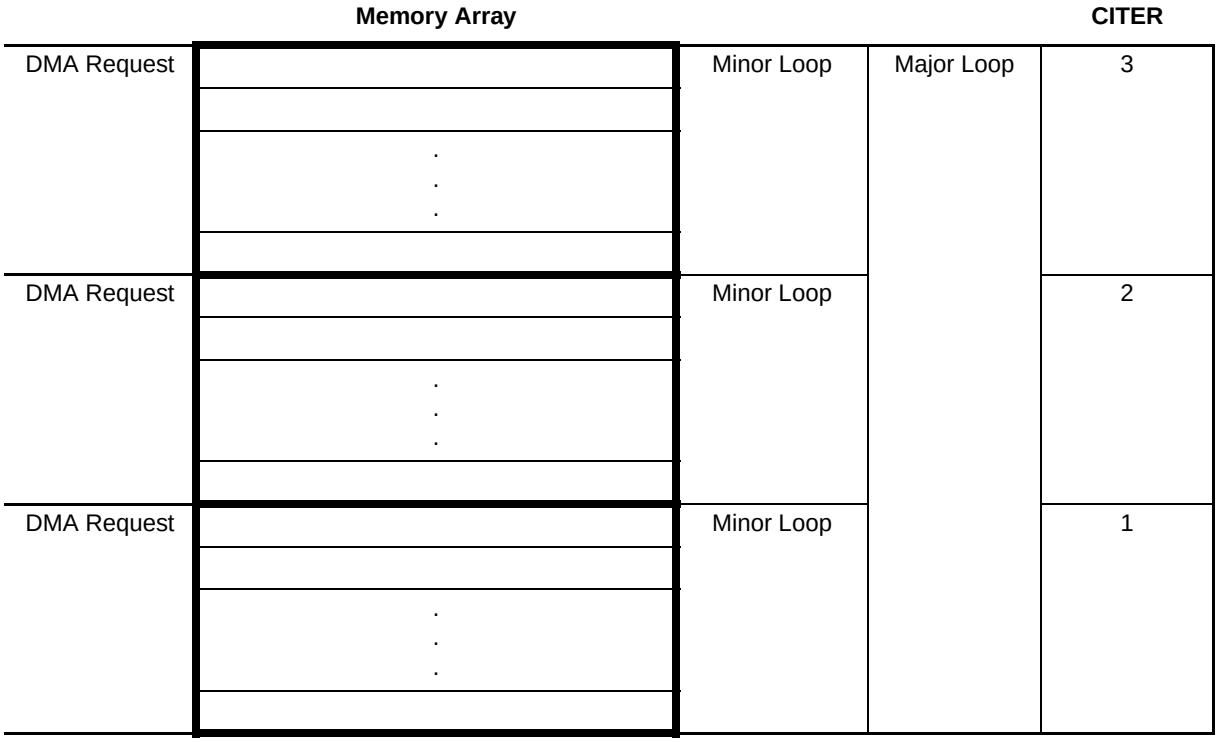


Table 12-28 lists the memory array terms and how the TCD settings interrelate.

Table 12-28. eDMA Memory Array Terms

*x*ADDR [1]: (Starting Address)	*x*SIZE [1] (size of one data transfer)	First Minor Loop (NBYTES often the same value as *x*SIZE)	Offset (*x*OFF [1]): number of bytes added to current address after each transfer (often the same value as *x*SIZE)
. . . .	. . . .	Minor Loops	Each DMA source (S) and destination (D) has its own: • Address (*x*ADDR) • Size (*x*SIZE) • Offset (*x*OFF) • Modulo (*x*MOD [1]) • Last Address Adjustment (*x*LAST)
*x*LAST [1]: Number of bytes added to current address after major loop (typically used to loop back)	. . . .	Last Minor Loop	Peripheral queues typically have size and offset equal to NBYTES.

1 Where *x* = S or D for Source or Destination, respectively.

12.5.4 eDMA Programming Errors

The eDMA performs various tests on each TCD*n* to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of error: Channel Priority Error, or DMAES[CPE].

For all error types other than channel priority errors, the channel number causing the error is recorded in the DMAES. If the error source is not removed before the next activation of the problem channel, the error will be detected and recorded again.

If priority levels are not unique, the highest channel priority that has an active request will be selected, but the lowest numbered channel with that priority will be selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts and error reporting will be associated with the selected channel.

12.5.5 eDMA Arbitration Mode Considerations

NOTE

Mask set L49P devices do not implement channel preemption or round-robin arbitration; thus none of this sub-section applies to those devices.

12.5.5.1 Fixed-Priority Arbitration

In this mode, the channel service request from the highest priority channel is selected to execute. It is possible that higher priority channels will take all the bandwidth of the eDMA, thus starving lower priority channels if the high priority channels continuously request service. The advantage of fixed arbitration is that preemption is available and latency can be small for channels that need to be serviced quickly.

12.5.5.2 Round-Robin Arbitration

In this mode, channels are serviced according to channel number in a circular fashion without regard to priority. Channel requests are serviced starting with the highest channel number and rotating through to the lowest. Because channels are serviced in round-robin manner, any channel that generates DMA service requests faster than the combined service rate of all requesting channels may lose requests. This scenario ensures that all channels will be guaranteed service at some point, regardless of request rates. However, the potential latency could be quite high. All channels are treated equally. Priority levels are not used and preemption is not available in round-robin mode.

12.5.6 eDMA Transfers

12.5.6.1 Single Request

To perform a simple transfer of ‘n’ bytes of data with one activation, set the major loop to one (TCD*n*[CITER] = TCD*n*[BITER] = 1). The data transfer will begin after the channel service request is acknowledged and the channel is selected to execute. Once the transfer is complete, the TCD*n*[DONE] bit will be set and an interrupt will be generated if properly enabled.

For example, the following TCD*n* entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a word wide port located at 0x2000. The address offsets are programmed in increments to match the size of the transfer; one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

TCD*n*[BITER]	=	1
TCD*n*[CITER]	=	1
TCD*n*[NBYTES]	=	16
TCD*n*[SADDR]	=	0x1000
TCD*n*[SOFF]	=	1
TCD*n*[SSIZE]	=	0
TCD*n*[SLAST]	=	–16
TCD*n*[DADDR]	=	0x2000
TCD*n*[DOFF]	=	4
TCD*n*[DSIZE]	=	2
TCD*n*[DLAST_SGA]	=	–16

TCD*n*[INT_MAJ]	=	1
All other TCD*n* fields	=	0
TCD*n*[START]	=	1 (written after all other fields are initialized)

This will generate the following sequence of events:

1. CPU write to the TCD*n*[START] bit (via peripheral bus) requests channel service,
2. the channel is selected by arbitration for servicing,
3. eDMA engine writes: TCD*n*[DONE] = 0, TCD*n*[START] = 0, TCD*n*[ACTIVE] = 1,
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file,
5. the source to destination transfers are executed as follows:
   a) read_byte(0x1000), read_byte(0x1001), read_byte(0x1002), read_byte(0x1003),
   b) write_word(0x2000) → *first iteration of the minor loop*
   c) read_byte(0x1004), read_byte(0x1005), read_byte(0x1006), read_byte(0x1007),
   d) write_word(0x2004) → *second iteration of the minor loop*
   e) read_byte(0x1008), read_byte(0x1009), read_byte(0x100a), read_byte(0x100b),
   f) write_word(0x2008) → *third iteration of the minor loop*
   g) read_byte(0x100c), read_byte(0x100d), read_byte(0x100e), read_byte(0x100f),
   h) write_word(0x200c) → *last iteration of the minor loop* → *major loop complete*
6. eDMA engine writes: TCD*n*[SADDR] = 0x1000, TCD*n*[DADDR] = 0x2000, TCD*n*[CITER] = 1 (from TCD*n*[BITER]),
7. eDMA engine writes: TCD*n*[ACTIVE] = 0, TCD*n*[DONE] = 1, DMAINT[*n*] = 1,
8. the channel retires.

The eDMA goes idle or services next channel.

12.5.6.2 Multiple Requests

The next example is the same as previous with the exception of transferring 32 bytes via two hardware requests. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After hardware requests for the channel are enabled in the DMAERQ register, channel service requests are initiated by the slave device.

TCD*n*[CITER]	=	2
TCD*n*[BITER]	=	2
TCD*n*[SLAST]	=	–32
TCD*n*[DLAST_SGA]	=	–32

This will generate the following sequence of events:

1. first hardware (eDMA peripheral request) request for channel service,
2. the channel is selected by arbitration for servicing,
3. eDMA engine writes: TCD*n*[DONE] = 0, TCD*n*[START] = 0, TCD*n*[ACTIVE] = 1,

4. eDMA engine reads: channel TCD*n* data from local memory to internal register file,
5. the source to destination transfers are executed as follows:
   a) read_byte(0x1000), read_byte(0x1001), read_byte(0x1002), read_byte(0x1003),
   b) write_word(0x2000) → *first iteration of the minor loop*
   c) read_byte(0x1004), read_byte(0x1005), read_byte(0x1006), read_byte(0x1007),
   d) write_word(0x2004) → *second iteration of the minor loop*
   e) read_byte(0x1008), read_byte(0x1009), read_byte(0x100a), read_byte(0x100b),
   f) write_word(0x2008) → *third iteration of the minor loop*
   g) read_byte(0x100c), read_byte(0x100d), read_byte(0x100e), read_byte(0x100f),
   h) write_word(0x200c) → *last iteration of the minor loop*
6. eDMA engine writes: TCD*n*[SADDR] = 0x1010, TCD*n*[DADDR] = 0x2010, TCD*n*[CITER] = 1,
7. eDMA engine writes: TCD*n*[ACTIVE] = 0,
8. the channel retires → *one iteration of the major loop*

The eDMA goes idle or services next channel.

1. second hardware (eDMA peripheral request) requests channel service,
2. the channel is selected by arbitration for servicing,
3. eDMA engine writes: TCD*n*[DONE] = 0, TCD*n*[START] = 0, TCD*n*[ACTIVE] = 1,
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file,
5. the source to destination transfers are executed as follows:
   a) read_byte(0x1010), read_byte(0x1011), read_byte(0x1012), read_byte(0x1013),
   b) write_word(0x2010) → *first iteration of the minor loop*
   c) read_byte(0x1014), read_byte(0x1015), read_byte(0x1016), read_byte(0x1017),
   d) write_word(0x2014) → *second iteration of the minor loop*
   e) read_byte(0x1018), read_byte(0x1019), read_byte(0x101a), read_byte(0x101b),
   f) write_word(0x2018) → *third iteration of the minor loop*
   g) read_byte(0x101c), read_byte(0x101d), read_byte(0x101e), read_byte(0x101f),
   h) write_word(0x201c) → *last iteration of the minor loop* → *major loop complete*
6. eDMA engine writes: TCD*n*[SADDR] = 0x1000, TCD*n*[DADDR] = 0x2000, TCD*n*[CITER] = 2 (from TCD*n*[BITER]),
7. eDMA engine writes: TCD*n*[ACTIVE] = 0, TCD*n*[DONE] = 1, DMAINT[*n*] = 1,
8. the channel retires → *major loop complete*

The eDMA goes idle or services the next channel.

12.5.6.3 Modulo Operation

The modulo feature of the eDMA provides the ability to easily implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for both the source and destination in the TCD*n*, and it specifies which lower address bits are allowed to increment from their original value after the

address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

Table 12-29 shows how the transfer addresses are specified based on the setting of the MOD field. In this example, a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567x) retain their original value. The source address is set to 0x12345670, the offset is set to 4 bytes and the *x*MOD field is set to 4, allowing for a 2^4-byte (16-byte) size queue.

Table 12-29. eDMA Modulo Feature Example

Transfer Number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

12.5.7 eDMA TCD*n* Status Monitoring

12.5.7.1 Minor Loop Complete

There are two methods to test for minor loop completion when using software initiated service requests. The first method is to read the TCD*n*[CITER] field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD*n*[START] bit AND the TCD*n*[ACTIVE] bit. The minor loop complete condition is indicated by both bits reading zero after the TCD*n*[START] was written to a one. Polling the TCD*n*[ACTIVE] bit may be inconclusive because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

	TCD*n*[START]	TCD*n*[ACTIVE]	TCD*n*[DONE]	State
1.	1	0	0	channel service request via software
2.	0	1	0	channel is executing
3.	0	0	0	channel has completed the minor loop and is idle
4.	0	0	1	channel has completed the major loop and is idle

The best method to test for minor loop completion when using hardware initiated service requests is to read the TCD*n*[CITER] field and test for a change. The hardware request and acknowledge handshakes signals are not visible in the programming model.

The TCD status bits execute the following sequence for a hardware activated channel:

	TCD*n*[START]	TCD*n*[ACTIVE]	TCD*n*[DONE]	State
1.	0	0	0	channel service request via hardware (peripheral request asserted via DMA Mux)
2.	0	1	0	channel is executing
3.	0	0	0	channel has completed the minor loop and is idle
4.	0	0	1	channel has completed the major loop and is idle

For both activation types, the major loop complete status is explicitly indicated via the TCD*n*[DONE] bit.

The TCD*n*[START] bit is cleared automatically when the channel begins execution regardless of how the channel was activated.

12.5.7.2 Active Channel TCD*n* Reads

The eDMA will return the “true” TCD*n*[SADDR], TCD*n*[DADDR], and TCD*n*[NBYTES] values for CPU reads while a channel is executing. The “true” values of SADDR, DADDR, and NBYTES are the values the eDMA engine is currently using in the internal register file and not the values in the TCD local memory for that channel. The addresses (SADDR and DADDR) and NBYTES (decrements to zero as the transfer progresses) values can give an indication of the progress of the transfer. All other values are read directly from the TCD local memory.

12.5.7.3 Preemption Status

Preemption is only available when fixed arbitration is selected as the channel arbitration mode. [1] A preemptable situation is one in which a preemption-enabled channel is running and a higher priority request becomes active. When the eDMA is not operating in fixed-channel arbitration mode, the determination of the relative priority of the actively running and the outstanding requests become undefined. Channel priorities are treated as equal (or more exactly, constantly rotating) when round-robin arbitration mode is selected.

The TCD*n*[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one iteration of the major loop. If two TCD*n*[ACTIVE] bits are set at the same time in the global TCD map, it indicates that a higher priority channel is actively preempting a lower priority channel.

12.5.8 Channel Linking

Channel linking is a mechanism where one channel sets the TCD*n*[START] bit of another channel (or itself) thus initiating a service request for that channel. [2] This operation is automatically performed by the eDMA at the conclusion of the major or minor loop when properly enabled.

1. Mask set L49P devices do not implement channel preemption.
2. Mask set L49P devices do not implement channel linking.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*[CITER_E_LINK] field is used to determine whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop, except the last pass. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, with the initial fields of:

TCD*n*[CITER_E_LINK]	=	1
TCD*n*[CITER_LINKCH]	=	0xC
TCD*n*[CITER]	=	0x4
TCD*n*[MAJOR_E_LINK]	=	1
TCD*n*[MAJOR_LINKCH]	=	0x7

will execute as:

1. minor loop done → set TCD12[START] bit
2. minor loop done → set TCD12[START] bit
3. minor loop done → set TCD12[START] bit
4. minor loop done, major loop done → set TCD7[START] bit

When minor loop linking is enabled (TCD*n*[CITER_E_LINK] = 1), the TCD*n*[CITER] field uses a nine bit value to form the current iteration count.

When minor loop linking is disabled (TCD*n*[CITER_E_LINK] = 0), the TCD*n*[CITER] field uses a 15-bit value to form the current iteration count. The bits associated with the TCD*n*[CITER_LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

NOTE

TCD*n*[CITER_E_LINK] and TCD*n*[BITER_E_LINK] must be equal or a configuration error will be reported. The CITER and BITER count widths must be equal in order to calculate the major loop, half-way done interrupt point.

Table 12-30 summarizes how an eDMA channel can be linked to other channels at the end of minor and major loops.

Table 12-30. Channel Linking Parameters

Desired Link Behavior	TCD*n* Control Field Name	Description
Link at end of Minor Loop	CITER_E_LINK	Enable channel-to-channel linking on minor loop completion (current iteration).
	CITER_LINKCH	Link channel number when minor loop completion (current iteration) linking is enabled.
Link at end of Major Loop	MAJOR_E_LINK	Enable channel-to-channel linking on major loop completion.
	MAJOR_LINKCH	Link channel number when major loop completion linking is enabled.

12.5.9 Dynamic Channel Linking and Scatter/Gather Operation

Dynamic channel linking and dynamic scatter/gather is the technique of changing the TCD*n*[MAJOR_E_LINK] or TCD*n*[E_SG] bits during channel execution. [1] These bits are read from the TCD local memory at the end of channel execution, thus allowing software to enable either feature during channel execution.

Because software is allowed to change the configuration during execution, a coherency sequence must be followed. Consider the scenario where software attempts to execute a dynamic channel link by enabling the TCD*n*[MAJOR_E_LINK] bit at the same time the eDMA engine is retiring the channel. The TCD*n*[MAJOR_E_LINK] would be set in the programmer model, but it would be indeterminate whether the actual link was made before the channel retired.

The following coherency sequence is recommended when executing a dynamic channel link or dynamic scatter/gather request:

1. Set the TCD*n*[MAJOR_E_LINK] bit,
2. Read the TCD*n*[MAJOR_E_LINK] bit,
3. Test the TCD*n*[MAJOR_E_LINK] request status:
   a) If the bit is set, the dynamic link attempt was successful,
   b) If the bit is clear, the attempted dynamic link did not succeed.

This same coherency model is true for dynamic scatter/gather operations. For both dynamic requests, the TCD local memory controller forces the TCD*n*[MAJOR_E_LINK] and TCD*n*[E_SG] bits to zero on any writes to a TCD*n* once the TCD*n*[DONE] bit for that channel is set, indicating the major loop is complete.

NOTE

Driver code must clear the TCD*n*[DONE] bit before writing the TCD*n*[MAJOR_E_LINK] or TCD*n*[E_SG] bits. The TCD*n*[DONE] bit is cleared automatically by the eDMA engine once a channel begins execution.

1. Mask set L49P devices do not implement channel linking or scatter/gather processing.
2. Mask set L49P devices do not implement scatter/gather processing.

Chapter 13
External Interface Module (EIM)

NOTE

This chapter does not apply to MAC71*x*2 devices, as those devices do not implement the EIM.

13.1 Overview

The external interface connects the device to off-chip resources and is only available on the MAC7111, MAC7116, MAC7131 and MAC7136 devices. There are two portions to this module, a user-programmable chip select portion and the bus controller. The bus controller uses a simple, flexible bus protocol that supports a variety of external memory devices with little or no external logic needed, and is based on the external bus of the 68K/Coldfire® family of devices. All transfers on the external bus are controlled by the MCU masters (CPU or eDMA), as shown in Figure 13-1 (external bus masters are not supported).

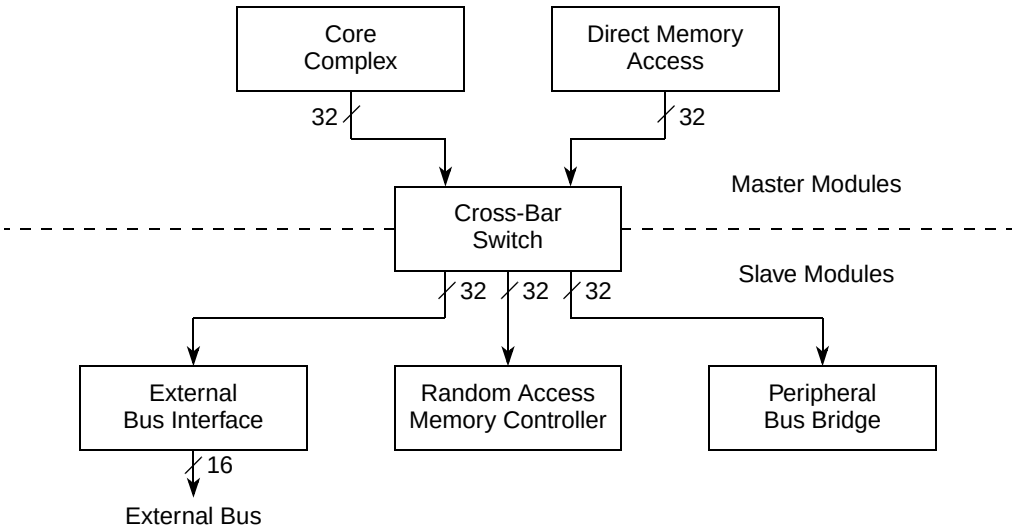


Figure 13-1. MAC7100 Family Bus Architecture Block Diagram

The interface is implemented with a 16-bit data bus capable of supporting byte, half word (2-byte) and word (4-byte) transfers. The port width of the data bus can be selected as either 8-bits or 16-bits in wide for each of the three chip select address ranges. At reset, the level on the PA14 pin is read to determine the width of the external data bus for $\overline{CS0}$ in global chip select mode (refer to Section 13.6.1.2, “Global Chip Select,” for detailed information). Transfers which are greater than the selected port width of the data bus are automatically broken into a sequence of individual transfers. The address bus has up to 22 address lines, ADDR[21:0], allowing access of up to 4 Mbytes of external memory space. The external bus uses the signals listed in Table 13-1 below.

The three chip select signals can be configured for different transfers types including auto acknowledge with a defined number of wait states. Each can be assigned to different address location, but if none of the

chip selects address/mask registers match an external access, no chip select is asserted and the access is internally terminated with a bus error. [1]

$\overline{CS0}$ is the only chip select that is active after reset, enabling the device boot routines to be executed from an off-chip source. $\overline{CS0}$ can be configured for port size and auto acknowledge of transfers based the values of PA[15:14] at reset. Refer to Table 13-9 for possible configurations.

$\overline{CS0}$ is asserted for all external accesses until the control registers (CSAR*n*, CSMR*n* and CSCR*n*) of the chip selects are configured and their valid bits are set. In order for $\overline{CS1}$ and $\overline{CS2}$ to function, $\overline{CS0}$ must first be configured and validated, and there should be no overlapping memory areas between chip selects.

The EIM supports a normal three clock-cycle access, but a two cycle fast termination can also be used if the transfer acknowledge signal is asserted before the rising edge of the second CLKOUT cycle.

The EIM provides support for burst transfers of up to 16 bytes of data, but depends on external devices to generate additional control information to support this operation as no transfer start, in progress or size signals are provided on MAC7100 Family devices.

13.2 Features

- Simple, flexible bus protocol
  - — Provides glueless interface to a variety of external memory devices
  - — Provides multiple, independently programmable chip selects
- The external bus supports 8- or 16-bit, bi-directional external data connections
- The external bus operates at a 1-to-1 clock frequency with the device bus master
- Key chip select features include:
  - — Up to three independent, user-programmable chip select signals ($\overline{CS}$[2:0]) that can interface with external SRAM, (P/E/EE)ROM, Flash, and peripherals
  - — Address masking for 64 Kbyte to 4 Gbyte memory block sizes
- Key bus operation features include:
  - — Up to 22 bits of address and 16 bits of data
  - — Access 8- and 16-bit data port sizes
  - — Generates byte, halfword (16-bit), word (32-bit), and line (16-byte) size transfers
  - — Burst and burst-inhibited transfer support
  - — Optional internal termination for external bus cycles

13.3 Modes of Operation

The external bus interface is in “global chip select mode” after reset. In this mode, which uses chip select zero, $\overline{CS0}$, all references to the external bus interface module use one chip select with a predetermined port size and termination method. This provides boot functionality. After the EIM is initialized, a wide variety of programmable options are available to taylor the external bus interface to various external bus resources.

1. For mask set L49P devices, $\overline{CS0}$ is asserted and a simple transfer is initiated which must be externally terminated.

13.4 Signal Description

Table 13-1 lists the signals used by the EIM and their basic characteristics. Note that these signal functions are enabled automatically when the device is reset into normal or secured expanded mode. In all other chip modes the EIM may be enabled by using the PIM module to configure the pins associated with the external bus into peripheral mode (refer to Chapter 18, "Port Integration Module (PIM)").

Table 13-1. EIM Signal Properties

Signal Name	Mnemonic	Function	Input / Output	CLKOUT Edge
Clock	CLKOUT	System clock (f_{SYS}) to synchronize external logic	Out	—
Address Bus	ADDR[21:0]	Address bus for devices on the EIM bus	Out	Rising
Data Bus	DATA[15:0]	Read/Write data bus for devices on the EIM bus	In/Out	Rising
Read/Write	R/$\overline{W}$	Identifies read and write data transfers	Out	Rising
Address Strobe	$\overline{AS}$ [1]	Indicates a bus cycle has been initiated and the address is stable	Out	Falling
Transfer Acknowledge	$\overline{TA}$ [1]	Assertion terminates transfer	In	Rising
Output Enable	$\overline{OE}$ [2]	Output enable	Out	Falling
Chip Selects	$\overline{CS}$[2:0]	Enables peripherals at programmed addresses. $\overline{CS0}$ provides relocatable boot ROM capability	Out	Falling
Byte Selects	$\overline{BS}$[1:0] [2]	Select individual bytes in memory.	Out	Falling

1 MAC7100 family devices multiplex $\overline{TA}$ and $\overline{AS}$ on one pin; thus, an application may use only one of these functions at a time. Mask set L49P devices implement only the $\overline{TA}$ functionality.

2 These signals change after the falling edge. However, in the *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC), they are specified off of the rising edge because CLKOUT is squared up internally.

13.4.1 CLKOUT

CLKOUT provides the device system clock (f_{SYS}) for use with the external bus or to drive external synchronous devices. Refer to Section 18.7.3, "PD2 / CLKOUT Configuration," on page 18-313 for details on controlling this signal.

13.4.2 Address Bus (ADDR[21:0])

The 22-bit address bus provides the address for the current transfer. The EIM outputs the address and increments the lower bits during burst transfers, allowing the address bus to be directly connected to external memory without external counters.

13.4.3 Data Bus (DATA[15:0])

These input/output signals provide a non-multiplexed general purpose data path between the MAC7100 and all devices on the external bus. DATA[15:0] provides a path to the MAC7100 for bus read operations and a path from the MAC7100 for bus write operations. The $\overline{OE}$ signal is the output enable for this bus.

13.4.4 Read/Write Signal (R/$\overline{W}$)

This signal defines the data transfer direction for the current bus cycle. A high level indicates a read cycle while a low level indicates a write cycle.

13.4.5 Address Strobe ($\overline{AS}$) [1]

Address strobe ($\overline{AS}$) is asserted to indicate when the address is stable at the start of a bus cycle. The address and attributes are guaranteed to be valid during the entire period that $\overline{AS}$ is asserted. This signal is asserted and negated on the falling edge of the clock. Since MAC7100 devices multiplex the $\overline{AS}$ and $\overline{TA}$ signals on one pin, the $\overline{AS}$ function cannot be used if external bus cycle termination is required. Refer to Section 18.7.4, “TA / AS Configuration,” on page 18-314 for details on controlling this signal.

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

This active-low synchronous input signal indicates the completion of a data transfer operation. It is sampled by the EIM after a transfer starts (as indicated by an active chip select). There are two methods of terminating an EIM bus cycle: 1) by externally asserting $\overline{TA}$, and 2) by setting the automatic acknowledge (AA) bit in the chip select control registers (CSCRs). Since MAC7100 devices multiplex the $\overline{AS}$ and $\overline{TA}$ signals on one pin, automatic acknowledge must be used when the $\overline{AS}$ function is selected. [1] Refer to Section 18.7.4, “TA / AS Configuration,” on page 18-314 for details on controlling this signal.

13.4.7 Output Enable ($\overline{OE}$)

This active-low signal is the output enable to control the output drive buffers of a device connected to the EBI. The output enable signal is active during bus read transfers and is inactive at all other times.

13.4.8 Chip Selects ($\overline{CS}$[2:0])

The EIM provides up to three programmable chip selects that can directly interface with SRAM, EPROM, EEPROM, and peripherals. The chip selects are used to define bus cycles. Each chip select can be programmed for a base address location and for masking addresses, port size, and burst-capability indication, wait-state generation, and internal/external termination.

Reset clears all chip select programming and configures global chip select mode, which uses $\overline{CS0}$. Refer to Section 13.6.1.2, “Global Chip Select,” for details.

13.4.9 Byte Selects ($\overline{BS}$[1:0])

The active-low byte select outputs provide control for peripherals and memory. During transfers, these outputs indicate which bytes within the transfer are being selected and which bytes of the data bus will be used for the transfer. $\overline{BS1}$ controls DATA[15:8] and $\overline{BS0}$ controls DATA[7:0]. Refer to Table 13-8 for more details.

1. Mask set L49P devices do not implement the $\overline{AS}$ functionality.

13.5 Memory Map / Register Definition

This section describes the EIM chip select module, including the operation and programming model of the the chip select address, mask and control registers. Table 13-2 shows the EIM register memory map. Reading reserved locations returns zeros.

Table 13-2. EIM Memory Map

EIM Offset	Register Description	
	[31:16]	[15:0]
0x0080	Chip select address register—bank 0 (CSAR0)	Reserved [1]
0x0084	Chip select mask register—bank 0 (CSMR0)	
0x0088	Reserved [1]	Chip select control register—bank 0 (CSCR0)
0x008C	Chip select address register—bank 1 (CSAR1)	Reserved [1]
0x0090	Chip select mask register—bank 1 (CSMR1)	
0x0094	Reserved [1]	Chip select control register—bank 1 (CSCR1)
0x0098	Chip select address register—bank 2 (CSAR2)	Reserved [1]
0x009C	Chip select mask register—bank 2 (CSMR2)	
0x00A0	Reserved [1]	Chip select control register—bank 2 (CSCR2)

[1] Addresses not assigned to a register and undefined register bits are reserved for expansion. Write accesses to these reserved address spaces and reserved register bits have no effect.

NOTE

Refer to 18.5.1.7 PIM Global Configuration Register (PIMCONFIG) on page 18-290 for details on controlling the clock signal to the EIM. Registers cannot be accessed if the EIM clock is disabled.

13.5.1 Register Descriptions

The chip select module is programmed through the chip select address, mask and control registers, CSAR0–CSAR2, CSMR0–CSMR2, and CSCR0–CSCR2.

13.5.1.1 EIM Chip Select Address Registers (CSAR*n*)

These registers specify the base address for each $\overline{\text{CS}}n$.

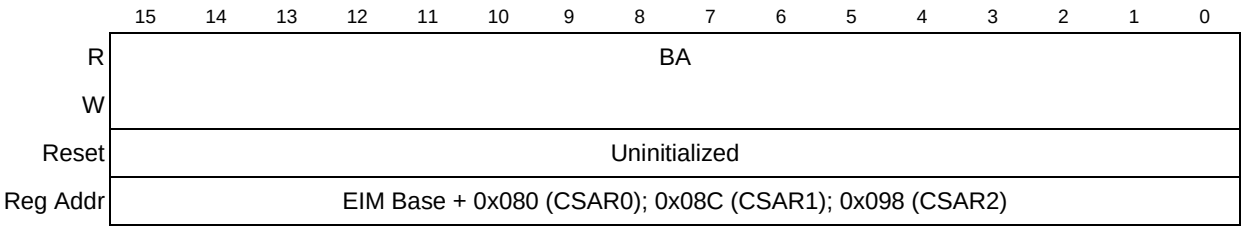


Figure 13-2. EIM Chip Select Address Registers (CSAR*n*)

Table 13-3. CSAR*n* Field Description

Bits	Name	Description
15–0	BA[15:0]	Base address. Defines the base address for memory dedicated to chip select $\overline{CSn}$. BA is compared to bits 31–16 on the internal address bus to determine if chip select memory is being accessed.

13.5.1.2 EIM Chip Select Mask Registers (CSMR*n*)

These registers specify the address mask and allowed access types for each $\overline{CSn}$.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BAM															
W																
Reset CSCR0	0xFFFF															
Reset CSCR[2:1]	Uninitialized															
Reg Addr	EIM Base + 0x084 (CSMR0); 0x090 (CSMR1); 0x09C (CSMR2)															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	WP	0	AM	C/I	SC	SD	UC	UD	V
W																
Reset CSCR0	—	—	—	—	—	—	—	0	—	0	0	0	0	0	0	0
Reset CSCR[2:1]	Uninitialized															0
Reg Addr	EIM Base + 0x084 (CSMR0); 0x090 (CSMR1); 0x09C (CSMR2)															

Figure 13-3. EIM Chip Select Mask Registers (CSMR*n*)

Table 13-4. CSMR*n* Field Descriptions

Bits	Name	Description
31–16	BAM[15:0]	Base address mask. Defines the chip select address range. The block size for $\overline{CSn}$ is 2^n where n = (number of CSMR*n*[BAM] bits set) + 16. Refer to Table 13-5 for examples. 0 Corresponding CSAR*n* bit position is used by $\overline{CSn}$ match logic. 1 Corresponding CSAR*n* bit position is ignored by $\overline{CSn}$ match logic.
15–9	—	Reserved.
8	WP	Write protect. Controls write accesses to the address range in the corresponding CSAR. Attempting to write to the range of addresses for which CSAR*n*[WP] = 1 results in the appropriate chip select not being selected. No exception occurs. 0 Both read and write accesses are allowed. 1 Only read accesses are allowed.
7	—	Reserved.
6	AM	Alternate master. When AM = 0 during an eDMA access, SC, SD, UC, and UD are don't cares in the chip select decode.

Table 13-4. CSMR*n* Field Descriptions (continued)

Bits	Name	Description
5–1	C/I, SC, SD, UC, UD	Address space mask bits. These bits determine whether the specified accesses can occur to the address space defined by the BAM for this chip select. Note that the AM bit may be used to cause the EIM to ignore these bits for eDMA accesses. C/I CPU space and interrupt acknowledge cycle mask SC Supervisor code address space mask SD Supervisor data address space mask UC User code address space mask UD User data address space mask 0 The address space assigned to $\overline{CS}n$ is available to the specified access type. 1 The address space assigned to $\overline{CS}n$ is not available (masked) to the specified access type. If this address space is accessed, chip select is not activated and a regular external bus cycle occurs.
0	V	Valid bit. Indicates that the corresponding CSAR*n*, CSMR*n*, and CSCR*n* contents are initialized as required and the associated $\overline{CS}n$ should be asserted on matches. Reset clears all CSMR*n*[V] bits and enables global chip select mode for $\overline{CS0}$ as described in Section 13.6.1.2, “Global Chip Select”. CSMR1[V] and CSMR2[V] are ignored until CSMR0[V] is set for the first time following reset. 0 Chip select configuration is invalid 1 Chip select configuration is valid

Table 13-5. CSMR*n*[BAM] / CSAR*n* Configuration Examples

$\overline{CS}n$	CSMR*n*[V]	CSMR*n*[BAM]	Block Size [1]	CSAR*n* (binary)	Address Range (hex)
$\overline{CS0}$	1	0001	128 Kbytes	0000_0000_0000_0000	0000_0000 – 0001_FFFF
$\overline{CS1}$	0	xxxx	—	xxxx_xxxx_xxxx_xxxx	—
$\overline{CS2}$	0	xxxx	—	xxxx_xxxx_xxxx_xxxx	—
$\overline{CS0}$	1	003F	4 Mbytes	0000_0000_00xx_xxxx	0000_0000 – 003F_FFFF
$\overline{CS1}$	1	001F	2 Mbytes	0000_0001_000x_xxxx	0100_0000 – 011F_FFFF
$\overline{CS2}$	1	000F	1 Mbyte	0000_0010_0000_xxxx	0200_0000 – 020F_FFFF

[1] While the EIM utilizes 32 bits of internal address information to perform matches, a maximum of 22 bits are provided externally on MAC7100 family devices. Defining chip select blocks larger than the number of external address bits (BAM > 0x003F for a 22-bit address bus) will result in the maximum size physical block being mirrored across the larger logical block on modulo physical block size boundaries.

13.5.1.3 EIM Chip Select Control Registers (CSCR*n*)

These registers control the auto-acknowledge, port size, byte enables and burst capability associated with each chip select address range. Note that to support global chip select mode for $\overline{CS0}$, the CSCR0 reset values differ from the other CSCR*n*s (refer to Section 13.6.1.2, “Global Chip Select,” for details).

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	WS				0	AA	PS		BEM	BSTR	BSTW	0	0	0
W																
Reset CSCR0	—		1	1	1	1	—	See Table 13-6			1	0	0	—		
Reset CSCR[2:1]	Uninitialized															
Reg Addr	EIM Base + 0x08A (CSCR0); 0x096 (CSCR1); 0x0A2 (CSCR2)															

Figure 13-4. EIM Chip Select Control Registers (CSCR*n*)

Table 13-6. CSCR*n* Field Descriptions

Bits	Name	Description
15–14	—	Reserved.
13–10	WS[3:0]	Wait states. The number of wait states inserted before an internal transfer acknowledge is generated (WS = 0 inserts zero wait states, WS = 0xF inserts 15 wait states). If AA = 0, $\overline{TA}$ must be asserted by the external system regardless of the number of wait states generated. In that case, the external transfer acknowledge ends the cycle. An external $\overline{TA}$ supersedes the generation of an internal $\overline{TA}$.
9	—	Reserved.
8	AA	Auto-acknowledge enable. Determines the assertion of the internal transfer acknowledge for accesses specified by the chip select address. Note that if AA = 1 for a corresponding $\overline{CS}n$ and the external system asserts an external $\overline{TA}$ before the wait-state countdown asserts the internal $\overline{TA}$, the cycle is terminated. Burst cycles increment the address bus between each internal termination. For $\overline{CS0}$, this bit initially reflects the state of the PA15 / DATA15 / $\overline{AA}$ pin during the assertion of $\overline{RESET}$ (if the pin is low, this bit is set; if the pin is high, this bit is clear). 0 No internal $\overline{TA}$ is asserted. Cycle is terminated externally. 1 Internal $\overline{TA}$ is asserted as specified by WS.
7–6	PS[1:0]	Port size. Specifies the width of the data bus associated with each chip select. It determines where data is driven during write cycles and where data is sampled during read cycles. See Section 13.6.1.1, “8- and 16-Bit Port Sizing,” for details. For $\overline{CS0}$, this field initially reflects the state of the PA14 / DATA14 / $\overline{PS}$ pin during the assertion of $\overline{RESET}$ (if the pin is low, this field is set to 0b01; if the pin is high, this field is set to 0b10). 00 32-bit port size. Not Valid 1x 16-bit port size. Valid data sampled and driven on DATA[15:0] 01 8-bit port size. Valid data sampled and driven on DATA[15:8]
5	BEM	Byte enable mode. Specifies the byte enable operation. Certain SRAMs have byte enables that must be asserted during reads as well as writes. BEM can be set in the relevant CSCR*n* to provide the appropriate mode of byte enable in support of these SRAMs. 0 $\overline{BS}n$ is not asserted for read. $\overline{BS}n$ is asserted for data write only. 1 $\overline{BS}n$ is asserted for read and write accesses.

Table 13-6. CSCR*n* Field Descriptions (continued)

Bits	Name	Description
4	BSTR	Burst read enable. Specifies whether burst reads are used for memory associated with each $\overline{CS}n$. 0 Data exceeding the specified port size is broken into individual, port-sized non-burst reads. For example, a word read from an 8-bit port is broken into four 8-bit reads. 1 Enables data burst reads larger than the specified port size, including word reads from 8- and 16-bit ports, halfword reads from 8-bit ports, and line reads from 8-, 16-, and 32-bit ports.
3	BSTW	Burst write enable. Specifies whether burst writes are used for memory associated with each $\overline{CS}n$. 0 Break data larger than the specified port size into individual port-sized, non-burst writes. For example, a word write to an 8-bit port takes four byte writes. 1 Enables burst write of data larger than the specified port size, including word writes to 8 and 16-bit ports, halfword writes to 8-bit ports and line writes to 8-, 16-, and 32-bit ports.
2–0	—	Reserved.

13.6 Functional Description

13.6.1 Chip Select Operation

When an internal bus access is initiated to the memory map area defined for the external bus interface (refer to Chapter 8, “Device Memory Map”), the EIM first compares the address with the base address and mask configurations programmed for $\overline{CS}$[2:0] (configured in CSAR0/CSMR0–CSAR2/CSMR2). The following rules are used to determine how the EIM responds to the bus cycle:

- If the address and attributes do not match any CSAR*n*/CSMR*n* pair, no chip select is asserted and the access is internally terminated with a bus error. [1]
- If the address and attributes match multiple CSAR*n*/CSMR*n* pairs, the matching $\overline{CS}n$ signals are asserted; however, the CSCR*n* settings for the chip selects are ignored, and a burst-inhibited bus cycle with external termination on a 16-bit port is executed.

Based on these rules, the chip select and control strobes are generated as described below.

1. For mask set L49P devices, if the address and attributes do not match in any CSAR*n*/CSMR*n* pair, the EIM runs an external burst-inhibited bus cycle with a default of external termination on a 16-bit port.

Table 13-7. Chip Select Module Control Signal Functions

Signal	Description
Chip Selects ($\overline{CS}$[2:0])	Each $\overline{CSn}$ can be independently programmed for an address location and mask, port size, read/write burst capability, wait-state generation, and internal/external termination. Only $\overline{CS0}$ is initialized at reset and operates in global chip select mode to allow boot ROM to reside at an external address space. In order for $\overline{CS1}$ and $\overline{CS2}$ to function properly, $\overline{CS0}$ must first be initialized and validated, and there should be no address range overlaps between chip selects.
Output Enable ($\overline{OE}$)	Interfaces to memory or to peripheral devices and enables a read transfer. It is asserted and negated on the falling edge of the clock. $\overline{OE}$ is asserted only when one of the chip selects matches for the current address decode.
Byte Strobes $\overline{BS}$[1:0]	These signals are individually programmed through the byte-enable mode bit, CSCR*n*[BEM], described in Section 13.5.1.3, "EIM Chip Select Control Registers (CSCRn)." The signals provide byte data select strobes, which are decoded from the transfer size, A1, and A0 in addition to the programmed port size and burst-ability of the memory accessed, as shown in Table 13-8.

Table 13-8. EIM Byte Select Signals Generation

Transfer Size	Port Size	A1	A0	$\overline{BS1}$	$\overline{BS0}$
				DATA[15:8]	DATA[7:0]
BYTE	8 bits	0	0	0	1
		0	1	0	1
		1	0	0	1
		1	1	0	1
	16 bits	0	0	0	1
		0	1	1	0
		1	0	0	1
		1	1	1	0
HALFWORD	8 bits	0	0	0	1
		0	1	0	1
		1	0	0	1
		1	1	0	1
	16 bits	0	0	0	0
		1	0	0	0
WORD	8 bits	0	0	0	1
		0	1	0	1
		1	0	0	1
		1	1	0	1
	16 bits	0	0	0	0
		1	0	0	0
LINE	8 bits	0	0	0	1
		0	1	0	1
		1	0	0	1
		1	1	0	1
	16 bits	0	0	0	0
		1	0	0	0

13.6.1.1 8- and 16-Bit Port Sizing

Static bus sizing is programmable through the CSCR*n*[PS] bits. See Section 13.5.1.3, "EIM Chip Select Control Registers (CSCRn)," for more information. Figure 13-5 shows the correspondence between the data bus and the external byte strobe control lines ($\overline{BS}$[1:0]). Note that all byte lanes are driven, although the state of unused byte lanes is undefined.

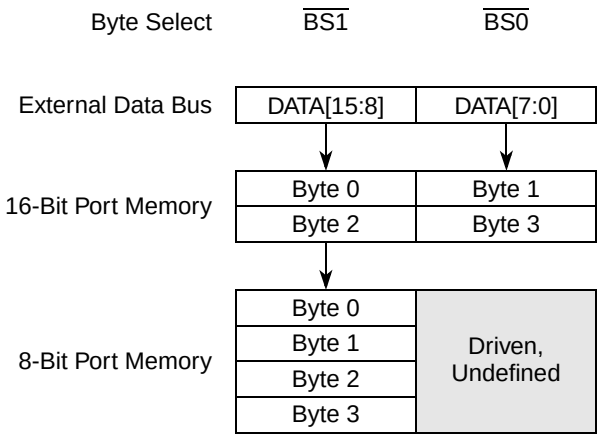


Figure 13-5. EIM Connections for External Memory Port Sizes

NOTE

If the Nexus interface is used in the primary position (PA[6:0]), then the port size selected for the external bus must be 8 bits.

13.6.1.2 Global Chip Select

The global chip select mode allows address decoding for boot ROM before system initialization. Thus, $\overline{CS0}$ operation differs from other chip select outputs after reset.

Immediately after system reset, $\overline{CS0}$ is asserted for all external bus interface accesses (refer to Table 8-5 on page 8-96 and Table 8-6 on page 8-97). At reset, the port size function of the global chip select is determined by the logic levels of the input on PA14, while PA15 controls the auto acknowledge function. Table 13-9 list the various reset encodings for the configuration signals multiplexed on PA14 and PA15. The configuration registers for $\overline{CS}$[2:1] are ignored, and thus no matches can occur, until CSMR0[V] is set. $\overline{CS0}$ may be returned to the global chip select mode only via a system reset.

Table 13-9. PA[15:14] Global Chip Select Configuration

PA[15:14] at $\overline{RESET}$	Auto Acknowledge	Data Port Size
00	No	8-bit
01	No	16-bit
10	Yes	8-bit
11	Yes	16-bit

13.6.2 External Bus Operation

The EIM uses the CLKOUT signal as its internal clock. Therefore, the external bus operates at the same speed as the internal bus clock rate, where all bus operations are synchronous to the rising edge of CLKOUT, and some of the bus control signals ($\overline{AS}$, $\overline{BS}$[1:0], $\overline{OE}$, and $\overline{CS}$[2:0]) are synchronous to the falling edge, shown in Figure 13-6.

NOTE

Throughout this section, $\overline{AS}$ and $\overline{TA}$ signals are discussed as though both signals are available simultaneously. Since MAC7100 devices multiplex the $\overline{AS}$ and $\overline{TA}$ signals on one pin (refer to 18.5.1.12 PIM Configure TA / AS Pin Register (CONFIG_TA) on page 18-294), only one function is available during a data transfer.

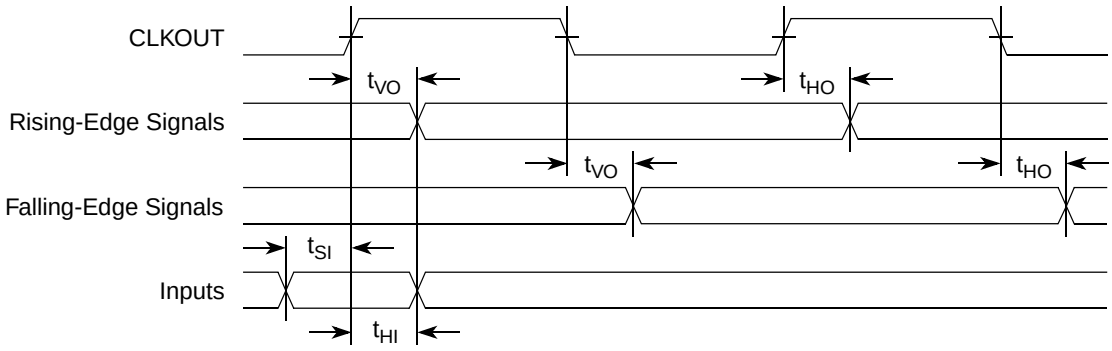


Figure 13-6. EIM Signal Relationship to CLKOUT

13.6.2.1 Data Transfers

The EIM bus supports byte, halfword, and word operand transfers and allows accesses to 8- and 16-bit data ports. Halfword and word transfers must be halfword aligned if the chip select that matches on an address defines a 16-bit port, otherwise the access will be terminated with an error. Aspects of the transfer, such as the port size, the number of wait states for the external slave being accessed, and whether internal transfer termination is enabled, can be programmed in the chip-select control registers (CSCR*n*s).

Figure 13-6 shows the byte lanes that external memory should be connected to and the sequential transfers if a word is transferred for three port sizes. For example, an 8-bit memory should be connected to DATA[15:8] ($\overline{BS1}$). A word transfer takes four transfers on DATA[15:8], starting with the MSB and going to the LSB.

Data transfers between the MAC7100 and other devices involve the following signals:

- Address bus (ADDR[21:0]) and Data bus (DATA[16:0])
- Attribute (R/$\overline{W}$), strobe ($\overline{AS}$, $\overline{CS}$[2:0], $\overline{OE}$, $\overline{BS}$[1:0]) and control ($\overline{TA}$) signals

The address bus, write data, and all attribute signals change on the rising edge of CLKOUT. $\overline{AS}$ changes on the falling edge. Read data is latched on the rising edge. The timing relationship of $\overline{AS}$, $\overline{CS}$[2:0],

$\overline{BS}$[1:0], and $\overline{OE}$ with respect to CLKOUT is similar in that all transitions occur during the low phase of CLKOUT (refer to Figure 13-7 below). However, due to differences in on-chip signal routing, signals may not assert simultaneously.

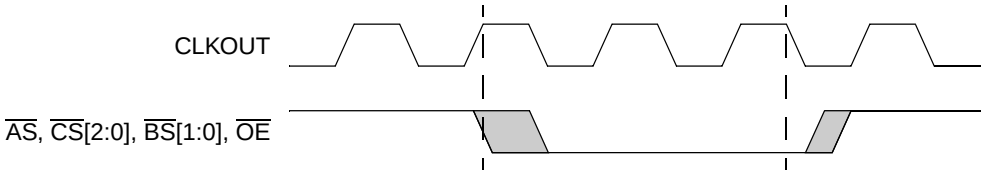


Figure 13-7. EIM Strobe Output Timing Diagram

13.6.2.2 Bus Cycle Execution

When an internal bus cycle is initiated, the EIM first compares the address of that bus cycle with the base address and mask configurations programmed for chip selects 0–2 (configured in CSCR0–CSCR2). If the driven address matches with one of the programmed chip selects, the appropriate chip select is asserted using the specifications programmed by the user in the respective configuration register. Otherwise, the following occurs:

- If the address and attributes do not match any CSCR*n*, no chip select is asserted and the access is internally terminated with a bus error. [1]
- Should an address and attribute match in multiple CSCRs, the matching chip-select signals are driven; however, the EIM runs an external burst-inhibited bus cycle with external termination on a 16-bit port.

Table 13-10 shows the type of access as a function of matches in the CSCR*n*s.

Table 13-10. Accesses by Matches in CSCR*n*s

Number of CSCR*n* Matches	Type of Access
0	None (internal bus error) [1]
1	Defined by matching CSCR*n*
Multiple	External, burst-inhibited, 16-bit

Basic operation of the MAC7100 bus is a three-clock bus cycle.

1. During the first clock, the address, attributes are driven. $\overline{AS}$ is asserted at the falling edge of the clock to indicate that address and attributes are valid and stable.
2. Data and $\overline{TA}$ are sampled during the second clock of a bus-read cycle. During a read, the external device provides data and is sampled at the rising edge at the end of the second bus clock. This data is concurrent with $\overline{TA}$, which is also sampled at the rising edge of the clock.
3. During a write, the device drives data from the rising clock edge at the end of the first clock to the rising clock edge at the end of the bus cycle. Wait states can be added between the first and second clocks by delaying the assertion of $\overline{TA}$. $\overline{TA}$ can be configured to be generated internally through the CSCRs. If $\overline{TA}$ is not generated internally, the system must provide it externally.

1. For mask set L49P devices, if the address and attributes do not match in any CSCR*n*, the EIM runs an external burst-inhibited bus cycle with a default of external termination on a 32-bit port.

4. The last clock of the bus cycle uses what would be an idle clock between cycles to provide hold time for address, attributes and write data. Figure 13-10 and Figure 13-12 show the basic read and write operations.

13.6.2.3 Data Transfer Cycle States

The data transfer operation in the EIM is controlled by an on-chip state machine. Each bus clock cycle is divided into two states. Even states occur when CLKOUT is high and odd states occur when CLKOUT is low. The state transition diagram for basic and fast termination read and write cycles are shown in Figure 13-8. Table 13-11 describes the states as they appear in subsequent timing diagrams.

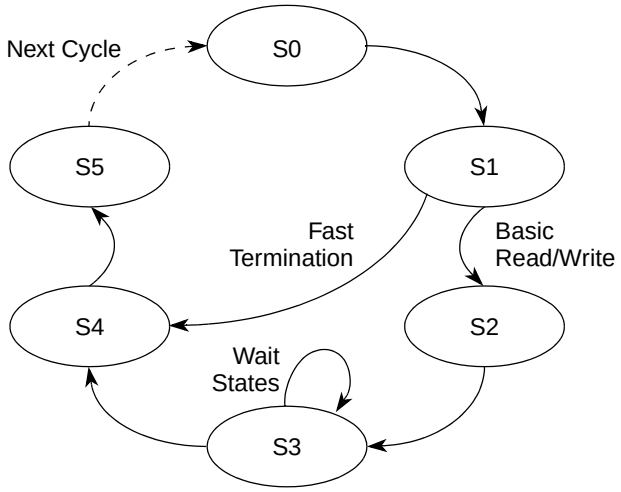


Figure 13-8. EIM Data Transfer State Transition Diagram

Table 13-11. Bus Cycle States

State	Cycle	CLKOUT	Description
S0	All	High	The read or write cycle is initiated in S0. On the rising edge of CLKOUT, the EIM places a valid address on the address bus and drives R/$\overline{W}$ high for a read and low for a write, if it is not already in the appropriate state.
S1	All	Low	$\overline{AS}$ asserts on the falling edge of CLKOUT, indicating that the address and attributes are stable. Appropriate $\overline{CS}$[2:0], $\overline{BS}$[1:0], and $\overline{OE}$ signals assert on the CLKOUT falling edge.
	Fast Termination		$\overline{TA}$ must be asserted during S1. Data is made available by the external device and is sampled on the rising edge of CLKOUT with $\overline{TA}$ asserted.
S2	Read/write (skipped if fast term.)	High	$\overline{TS}$ is negated on the rising edge of CLKOUT in S2.
	Write		Data is driven on the bus on the rising edge of CLKOUT.
S3	Read/write (skipped if fast term.)	Low	The EIM waits for $\overline{TA}$ assertion. If $\overline{TA}$ is not sampled as asserted before the rising edge of CLKOUT at the end of the first clock cycle, the EIM inserts wait states (full clock cycles) until $\overline{TA}$ is sampled as asserted.
	Read		Data is made available by the external device on the falling edge of CLKOUT and is sampled on the rising edge of CLKOUT with $\overline{TA}$ asserted.

Table 13-11. Bus Cycle States (continued)

State	Cycle	CLKOUT	Description
S4	All	High	The external device should negate $\overline{TA}$.
	Read (including fast term.)		The external device can stop driving data after the rising edge of CLKOUT. However data could be driven through the end of S5. The data bus must be in three-state ≤ two CLKOUT cycles after the start of S4. This applies to basic read cycles, fast termination cycles, and the last transfer of a burst.
S5	S5	Low	$\overline{CS}$[2:0], $\overline{BS}$[1:0], and $\overline{OE}$ are negated on CLKOUT falling edge of S5. The EIM stops driving address lines and R/$\overline{W}$ on CLKOUT rising edge, terminating the read or write cycle. Note that the rising edge of CLKOUT may be the start of S0 for the next access cycle.
	Read		The external device stops driving data between S4 and S5.
	Write		The data bus returns to high impedance on the rising edge of CLKOUT.

13.6.2.4 Read Cycle

During a read cycle, the EIM receives data from memory or from a peripheral device. Figure 13-9 is the read cycle flowchart. The read cycle timing diagram is shown in Figure 13-10.

NOTE

In the following timing diagrams, $\overline{TA}$ waveforms apply for chip selects programmed to enable either internal or external termination. $\overline{TA}$ assertion is the same in either case.

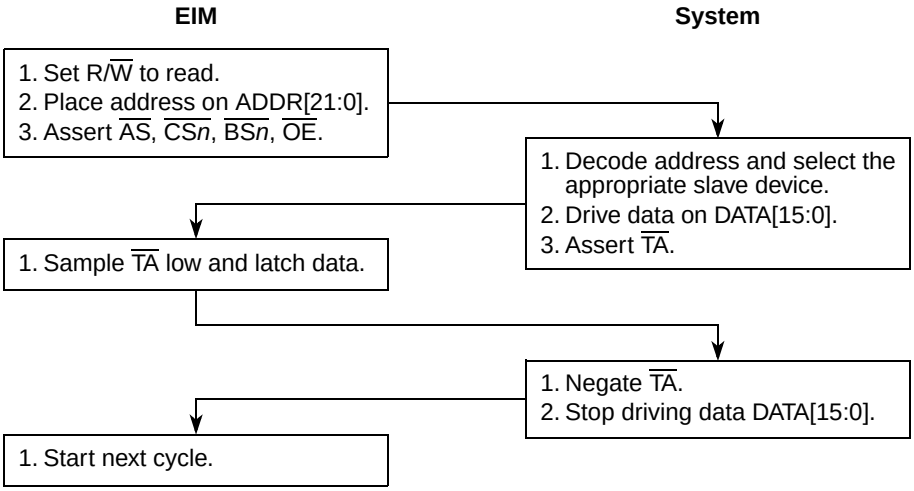


Figure 13-9. EIM Read Cycle Flowchart

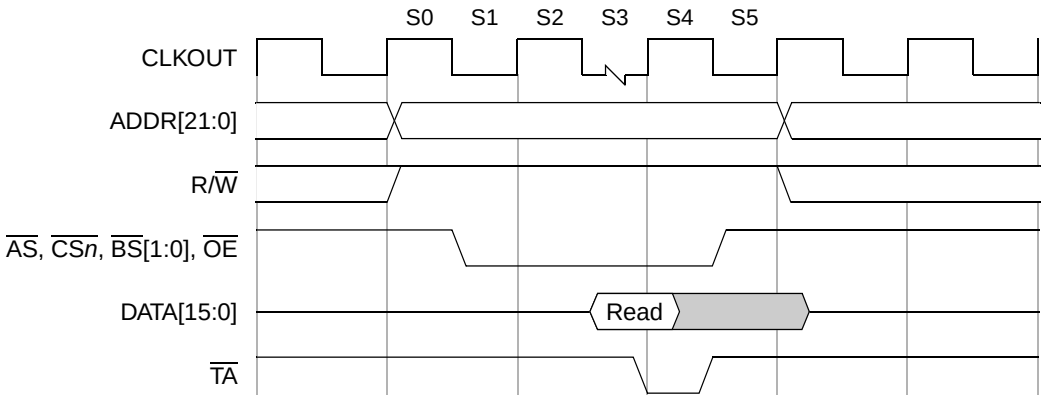


Figure 13-10. EIM Basic Read Bus Cycle

Note the following characteristics of a basic read:

- In S3, data is made available by the external device on the falling edge of CLKOUT and is sampled on the rising edge of CLKOUT with $\overline{TA}$ asserted.
- In S4, the external device can stop driving data after the rising edge of CLKOUT. However data could be driven up to S5.
- For a read cycle, the external device stops driving data between S4 and S5.

13.6.2.5 Write Cycle

During a write cycle, the EIM sends data to the memory or to a peripheral device. The write cycle flowchart is shown in Figure 13-11. The write cycle timing diagram is shown in Figure 13-12.

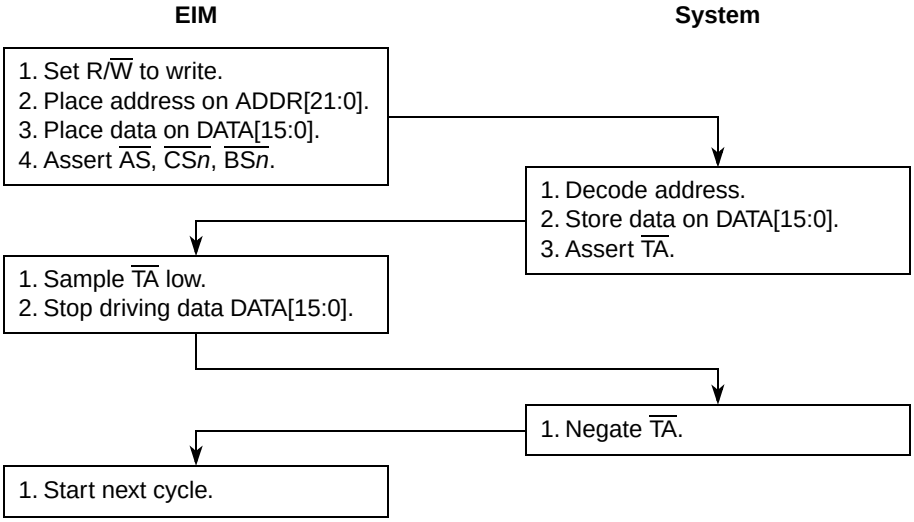


Figure 13-11. EIM Write Cycle Flowchart

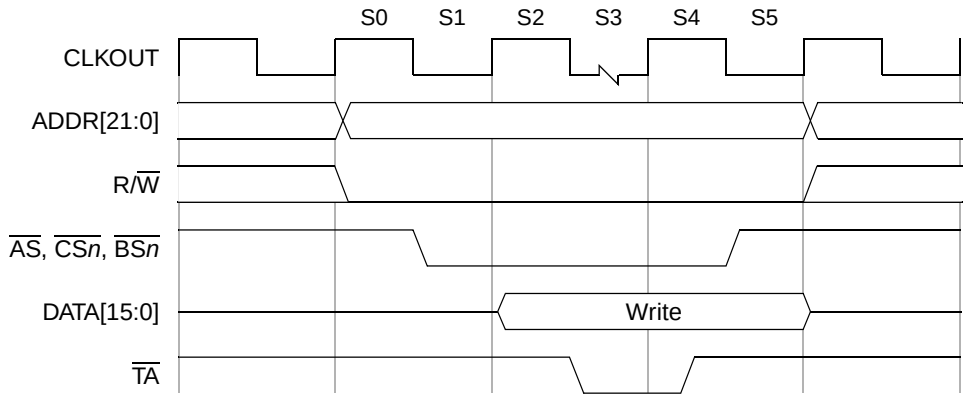


Figure 13-12. EIM Basic Write Bus Cycle

13.6.2.6 Fast Termination Cycles

Two clock cycle transfers are supported on the MAC7100 bus. In most cases, this is impractical to use in a system because the termination must take place in the same half clock during which CS*n* are asserted. As this is atypical, it is not referred to as the zero-wait-state case but is called the fast-termination case. Fast termination cycles occur when the external device or memory asserts TA less than one clock after CS*n* is asserted. This means that the EIM samples TA on the rising edge of the second cycle of the bus transfer.

Figure 13-13 shows a read cycle with fast termination. Note that fast termination cannot be used with internal termination. Figure 13-14 shows a write cycle with fast termination. Since TA and AS can not be used at the same time on MAC7100 devices and external termination must be used for fast cycles, AS is not shown in Figure 13-13 or Figure 13-14.

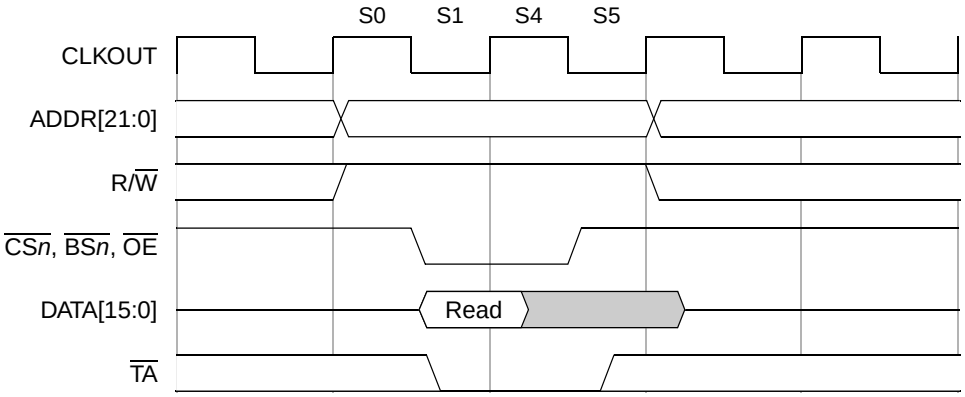


Figure 13-13. EIM Read Cycle with Fast Termination

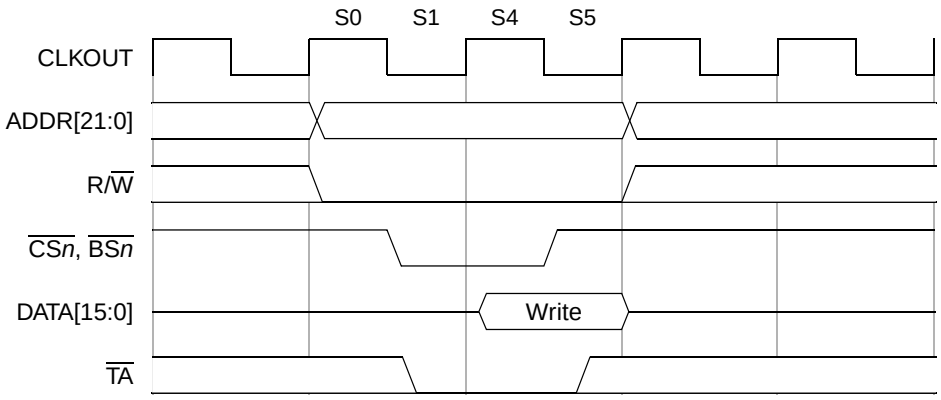


Figure 13-14. EIM Write Cycle with Fast Termination

13.6.2.7 Back-to-Back Bus Cycles

The EIM runs back-to-back bus cycles whenever possible. For example, when a word read is started on a halfword-size bus, the processor performs two back-to-back halfword read accesses. Figure 13-15 shows a read back-to-back with a write.

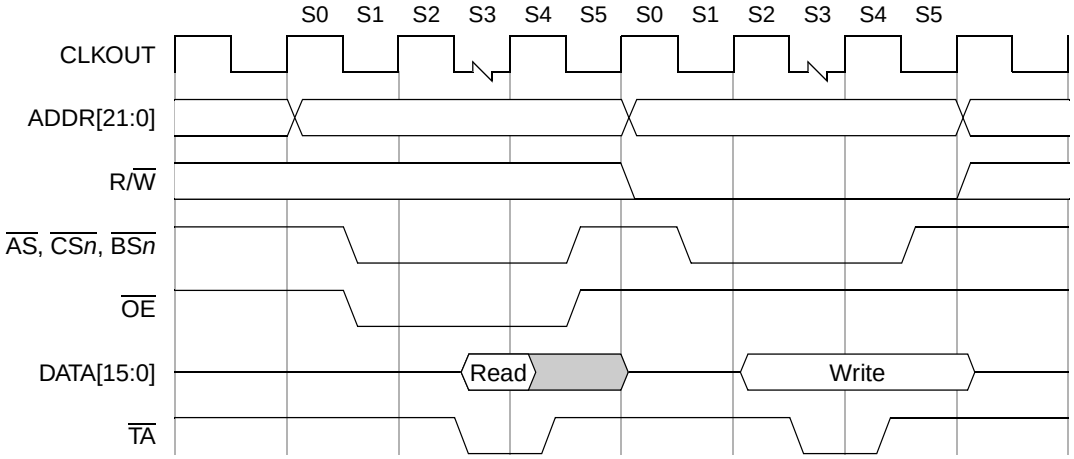


Figure 13-15. EIM Back-to-Back Bus Cycles

Basic read/write cycles are shown in Figure 13-15, but there is no restriction on the type of cycles that may be back-to-back. Software cannot control execution of back-to-back cycles.

13.6.2.8 Burst Cycles

The EIM can be programmed to initiate burst cycles if its transfer size exceeds the size of the port it is transferring to. For example, a halfword transfer to an 8-bit port requires two byte burst cycles. A line transfer to a 16-bit port requires eight halfword burst cycles.

The EIM bus can support 2-1-1-1-1-1-1-1 burst cycles to optimize eDMA transfers. A user can add wait states by delaying termination of the cycle.

The CSCRs are used to enable bursts for reads, writes, or both. MAC7100 memory space can be declared burst-inhibited for reads and writes by clearing the appropriate CSCR*n*[BSTR, BSTW]. A line access to a

burst-inhibited region first accesses the external bus encoded as a line access. The address changes if internal termination is used but does not change if external termination is used, as shown in Figure 13-16 and Figure 13-17.

13.6.2.8.1 Line Transfers

A line is a 16-byte-aligned, 16-byte value. Despite the alignment, a line access may not begin on the aligned address; therefore, the bus interface supports line transfers on multiple address boundaries. Table 13-12 shows allowable patterns for line accesses.

Table 13-12. Allowable Line Access Patterns

ADDR[3:2]	Word Accesses
00	0–4–8–C
01	4–8–C–0
10	8–C–0–4
11	C–0–4–8

13.6.2.8.2 Line Read Bus Cycles

Figure 13-16 shows a line access read with zero wait states. The access starts like a basic read bus cycle with the first data transfer sampled on the rising edge of S4, but the next pipelined burst data is sampled a cycle later on the rising edge of S6. Each subsequent pipelined data burst is single cycle until the last one, which can be held for up to two CLKOUT cycles after $\overline{TA}$ is asserted. Note that $\overline{AS}$ and $\overline{CS}n$ are asserted throughout the burst transfer. This example shows the timing for external termination, which differs from the internal termination example in Figure 13-17 only in that the address lines change only at the beginning and end of the transfer.

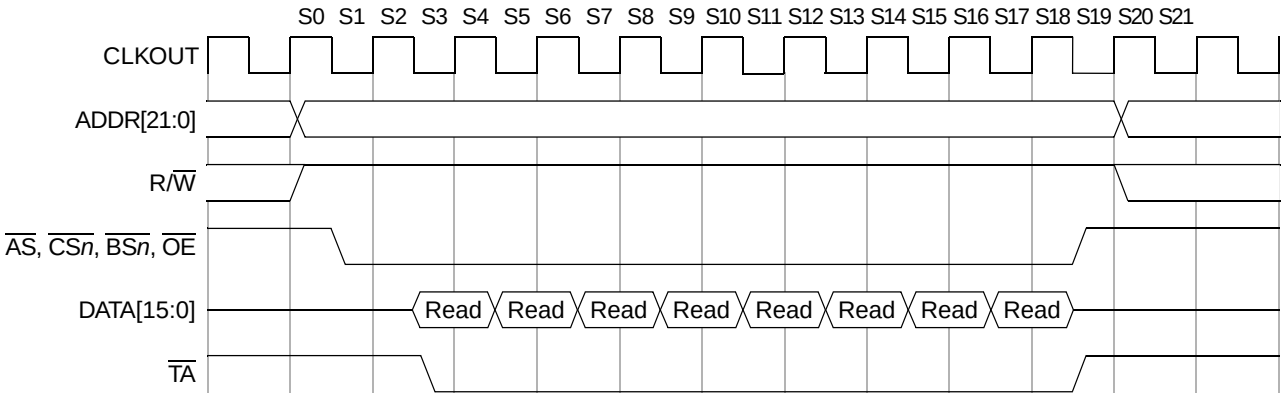


Figure 13-16. EIM Line Read Burst (2-1-1-1-1-1-1-1), External Termination

A line read access may have wait states programmed in CSCR*n* to give the peripheral or memory more time to return read data. This access follows the same execution as a zero-wait state read burst with the exception of added wait states.

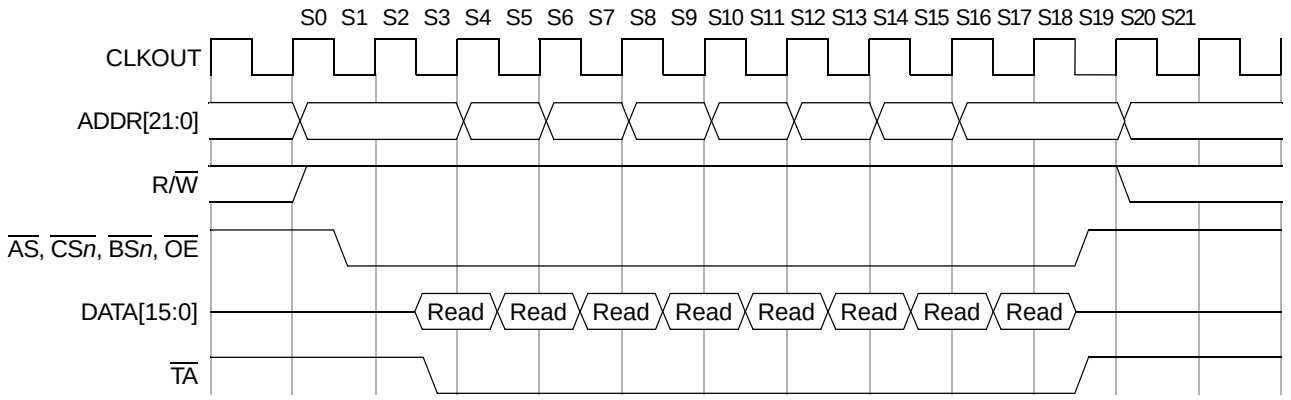


Figure 13-17. EIM Line Read Burst (2-1-1-1-1-1-1-1), Internal Termination

A burst-inhibited line read access may use fast termination. The external device executes a basic read cycle while determining that a line is being transferred. The external device uses fast termination for subsequent transfers.

13.6.2.8.3 Line Write Bus Cycles

Figure 13-18 shows a line access write with zero wait states and external termination. It begins like a basic write bus cycle with data driven one clock after TS. The next pipelined burst data is driven a cycle after the write data is registered (on the rising edge of S6). Each subsequent burst takes a single cycle. Note that as with the line read example in Figure 13-16, AS and CS*n* remain asserted throughout the burst transfer. Figure 13-19 shows a line access write with zero wait states and internal termination. Note that when external termination is used, the address lines change.

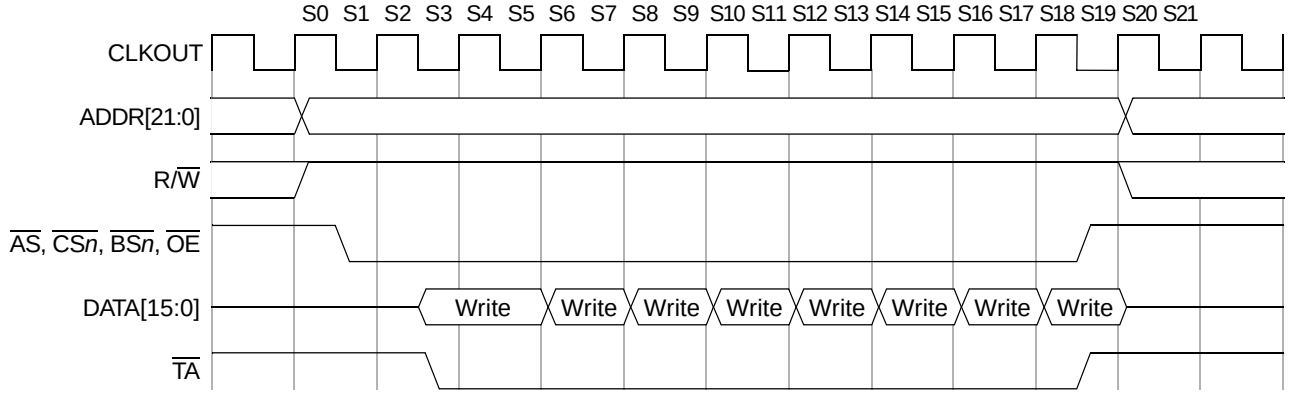


Figure 13-18. EIM Line Write Burst (2-1-1-1-1-1-1-1), External Termination

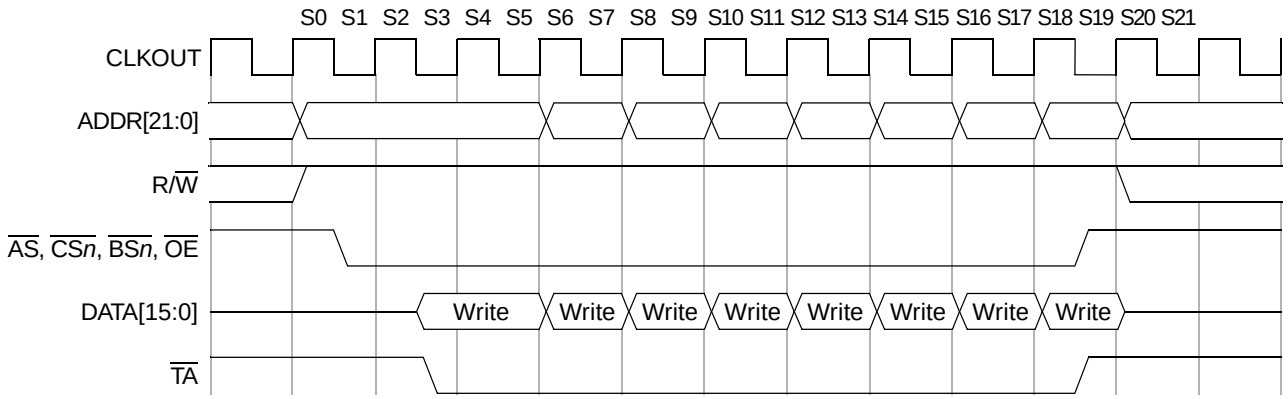


Figure 13-19. EIM Line Write Burst (2-1-1-1-1-1-1-1), Internal Termination

A line write access may have wait states programmed in EIM_CSCR*n* to give the peripheral or memory more time to catch write data. This access follows the same execution as a zero-wait state write burst with the exception of added wait states. A burst-inhibited line write access may use fast termination. The external device executes a basic write cycle while determining that a line is being transferred. The external device uses fast termination to end each subsequent transfer.

13.7 Initialization / Application Information

13.7.1 Using Global Chip Select Mode

As described in Table 8-5 on page 8-96, Section 11.3.1.11, "MCM XBS Address Map Register (AAMR)," on page 11-132, Section 13.5.1, "Register Descriptions," and Section 13.6.1.2, "Global Chip Select," $\overline{CS0}$ is configured as listed below following a system reset:

1. CSAR0 is ignored, base address = 0x0000_0000;
2. CSMR0[BAM] is ignored, block size = 512 Mbytes;
3. CSMR0[WP, AM, C/I, SC, SD, UC, UD] are ignored, no access protection;
4. CSCR0[WS] = F, 15 wait states if auto-acknowledge is selected;
5. CSCR0[BEM] = 1, $\overline{BSn}$ is asserted for read and write accesses;
6. CSCR0[BSTR, BSTW] = 00, burst reads and writes are not enabled;
7. CSCR0[AA, PS] set by PA[15:14] according to Table 13-9.

If the global chip select reset configuration is sufficient for system operating requirements, no other initialization of the EIM registers must be performed.

13.7.2 Configuring Chip Selects

If the global chip select configuration is not sufficient to support the desired system design, one or more of the chip select signals may be configured via the appropriate CSAR*n*/CSMR*n*/CSCR*n* register sets. Until CSMR0[V] is set, the configuration registers for $\overline{CS}$[2:1] are ignored. Therefor, if a required external address range is covered by $\overline{CS0}$ in global chip select mode, CSAR0/CSMR0/CSCR0 must be

programmed to continue decoding for that address range and the CSMR0[V] bit set in order to also use $\overline{CS}$[2:1]. If $\overline{CS0}$ is not needed after initial boot load is complete, CSMR0[V] may be set and then immediately cleared, and then $\overline{CS}$[2:1] may be configured for use.

In order to utilize the full capability of the chip select module, the following initialization steps should be followed:

1. Write the ADDR[31:16] base for $\overline{CS}$[2:1] into CSAR2 and/or CSAR1 as required,
2. Write the appropriate attributes for $\overline{CS}$[2:1] into CSCR2 and/or CSCR1 as required,
3. Write the appropriate address mask and access protection configuration for $\overline{CS}$[2:1] into CSMR2 and/or CSMR1 as required, with the V bit set,
4. Write the ADDR[31:16] base for $\overline{CS0}$ into CSAR0,
5. Write the appropriate attributes for $\overline{CS0}$ into CSCR0,
6. Write the appropriate address mask and access protection configuration for $\overline{CS0}$ into CSMR0, with the V bit clear,
7. Execute code as required to guarantee that pipelined accesses to external memory are complete,
8. Set the CSMR0[V] bit.

Programming two or more $\overline{CS}n$ configuration registers to decode to overlapping address spaces with identical protection restrictions is not prohibited in hardware. However, the resulting external bus cycle attributes, described in Table 13-10, may not be appropriate for the system design. Future MAC7100 family device implementations may place limitations on overlapping $\overline{CS}n$ configurations, so the use of such configurations are discouraged.

13.7.3 Dynamic Chip Select Configuration

There are no hardware protections against changing any CSAR*n*/CSMR*n*/CSCR*n* field contents while the associated CSMR*n*[V] bit is set. Thus, it is possible to dynamically change the configuration of a $\overline{CS}n$ signal, although the operation of the external bus interface may be indeterminate if this is done. Thus, this type of an operation is strongly discouraged. If it is necessary to modify the configuration of any $\overline{CS}n$ signal after initialization, the associated CSMR*n*[V] bit should be cleared (writing the same values into the other fields as were previously configured), the CSAR*n*/CSMR*n*/CSCR*n* fields modified as appropriate, and the CSMR*n*[V] set during the final write to the register set.

Chapter 14
Cross-Bar Switch Module (XBS)

14.1 Overview

This section provides information on the layout, configuration, and programming of the cross-bar switch bus. The cross-bar switch bus connects the bus masters and bus slaves using a cross-bar switch structure. This structure allows bus masters to simultaneously access different bus slaves with no interference while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitrations methods and attributes may be programmed on a slave by slave basis.

The MAC7111, MAC7116, MAC7131 and MAC7136 devices have two masters and three slaves (2Mx3S) utilized on the cross-bar switch. The two masters are the ARM7TDMI-S core and the Enhanced Direct Memory Access controller (eDMA). The slaves are the peripheral bus (IPS), the SRAM controller, and the external bus interface (EIM).

For the MAC7101, MAC7106, MAC7112, MAC7121, MAC7122, MAC7126, MAC7141 and MAC7142 devices, the cross-bar switch uses two masters and two slaves (2Mx2S). The two masters are the ARM7TDMI-S core and the enhanced direct memory access controller (eDMA). The slaves are the peripheral bus (IPS) and the SRAM controller.

Figure 14-1 is a block diagram of the MAC7100 family bus architecture showing the 2Mx3S cross-bar switch bus configuration.

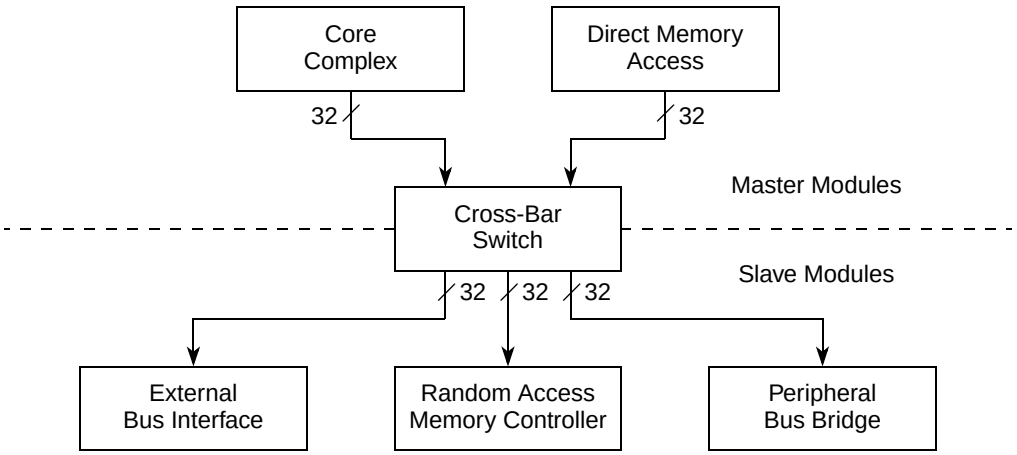


Figure 14-1. MAC7100 Family Bus Architecture Block Diagram

14.2 Features

The cross-bar switch bus includes these distinctive features:

- Symmetric cross-bar bus switch implementation
  - — Allows concurrent accesses from different masters to different slaves
  - — Slave arbitration attributes configured on a slave by slave basis

- 32 bits wide and supports byte, half-word (2 byte), word (4 byte), and 16 byte burst transfers
- Operates at a 1-to-1 clock frequency with the bus masters

14.3 Modes of Operation

The cross-bar switch bus provides two arbitration modes: fixed or round-robin. The arbitration mode may be set on a slave by slave basis. Slaves configured for fixed arbitration mode have a unique arbitration level assigned to each bus master.

Fixed Priority: The highest priority active master accessing a particular slave is granted the master bus switch-path to that slave. A higher priority master will block access to a given slave from a lower priority master as long as the higher priority master continuously requests that slave. See Section 14.5.1.1, “Fixed-Priority Operation.”

Round-Robin Priority: Active masters accessing a particular slave are initially granted the slave based on their master port number. Master priority is then modified in a wrap-around manner to give all masters fair access to the slave. See Section 14.5.1.2, “Round-Robin Priority Operation.”

14.4 Memory Map / Register Definition

There are two registers that reside in each slave port of the cross-bar switch bus. Read and write transfers both require two IP bus clock cycles. The registers can only be read from and written to in supervisor mode. Additionally, these registers can only be read from or written to by 32-bit accesses. Non 32-bit accesses to legal registers are ignored.

A bus error response is returned if an unimplemented location is accessed within the cross-bar switch bus.

The slave registers also feature a bit which, when set, will prevent the registers from being written. The registers will still be readable, but future write attempts will have no effect on the registers and will be terminated with a bus error response to the master initiating the write. The core, for example, will take a bus error interrupt.

Table 14-1 shows the memory map for the cross-bar switch bus program-visible registers.

Table 14-1. XBS Memory Map

XBS Offset	Register Description	Access
0x0100	Priority Register for EIM Slave port (PR_EIM)	word (4byte)
0x0110	Control Register for EIM Slave port (CR_EIM)	word (4byte)
0x0300	Priority Register for SRAM Slave port (PR_SRAM)	word (4byte)
0x0310	Control Register for SRAM Slave port (CR_SRAM)	word (4byte)
0x0700	Priority Register for Peripheral Controller Slave port (PR_PC)	word (4byte)
0x0710	Control Register for Peripheral Controller Slave port (CR_PC)	word (4byte)

14.4.1 Register Descriptions

14.4.1.1 XBS Priority Registers (PR_*port*)

The priority registers (PR_*port*) set the priority of each master port on a per slave port basis and resides in each slave port.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	XBS Base + 0x0100 (PR_EIM: Slave port to external interface module); XBS Base + 0x0300 (PR_SRAM: Slave port to SRAM controller); XBS Base + 0x0700 (PR_PC: Slave port to peripheral controller)															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	DMA	0	0	0	Core
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Reg Addr	XBS Base + 0x0100 (PR_EIM: Slave port to external interface module); XBS Base + 0x0300 (PR_SRAM: Slave port to SRAM controller); XBS Base + 0x0700 (PR_PC: Slave port to peripheral controller)															

Figure 14-2. XBS Priority Registers (PR_*port*)

Table 14-2. PR_*port* Field Descriptions

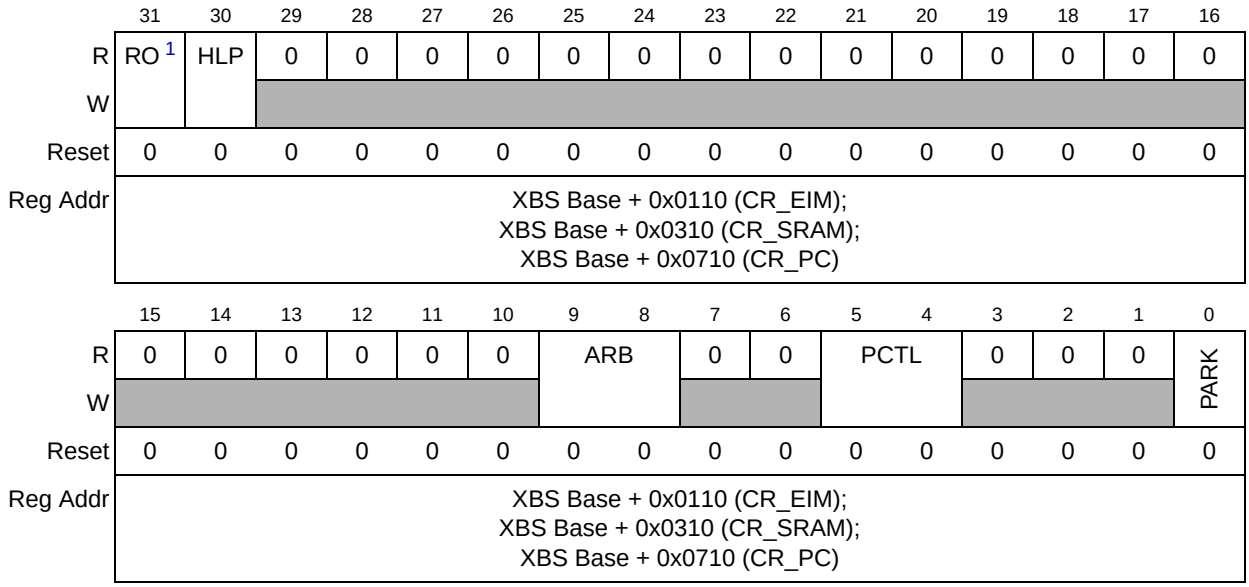
Bits	Name	Description
31–5	—	Reserved.
4	DMA_MSTR	DMA master priority. This bit sets the arbitration priority for this port on the associated slave port.This bit is initialized by hardware reset. 0 This master has the highest priority when accessing the slave port. 1 This master has the lowest priority when accessing the slave port.
3–1	—	Reserved.
0	CORE_MSTR	Core master priority. This bit sets the arbitration priority for this port on the associated slave port. This bit is initialized by hardware reset. 0 This master has the highest priority when accessing the slave port. 1 This master has the lowest priority when accessing the slave port.

The priority register can only be accessed with 32-bit accesses. Once the read only bit (RO) has been set in the slave control register, the priority register can only be read; attempts to write to it will have no effect on the PR_*port* and result in a bus error response to the master initiating the write.

Additionally, no two available master ports may be programmed with the same priority level. Attempts to program two or more available masters with the same priority level will result in an error response and the PR_*port* will not be updated.

14.4.1.2 XBS Control Registers (CR_*port*)

The XBS control registers (CR_*port*) controls several features of each slave port.



1 Once this bit is written to a 1 only hardware reset will return it to a 0.

Figure 14-3. XBS Control Registers (CR_*port*)

Table 14-3. CR_*port* Descriptions

Bits	Name	Description
31	RO	Read only. This bit is used to force all of a slave port's registers to be read only. Once set it can only be cleared by hardware reset. This bit is initialized by hardware reset. 0 All this slave port's registers can be written. 1 All this slave port's registers are read only and cannot be written (attempted writes have no effect and result in an error response).
30	HLP	Halt low priority. This bit is used to set the initial arbitration priority for low power mode requests. Note that setting this bit will not effect the request for low power mode from attaining highest priority once it has control of the slave ports. This bit is initialized by hardware reset. 0 The low power mode request has the highest priority for arbitration on this slave port 1 The low power mode request has the lowest initial priority for arbitration on this slave port.
29–10	—	Reserved.
9–8	ARB[1:0]	Arbitration mode. These bits are used to select the arbitration policy for the slave port. These bits are initialized by hardware reset. 00 Fixed Priority. 01 Round Robin (rotating) Priority. 10 Reserved 11 Reserved
7–6	—	Reserved.

Table 14-3. CR_*port* Descriptions (continued)

Bits	Name	Description
5–4	PCTL[1:0]	Parking control. These bits determine the parking control used by this slave port. The low power park feature can result in an overall power savings if the slave port is not saturated; however, it will force an extra clock of latency whenever any master tries to access it when it is not in use because it will not be parked on any master. These bits are initialized by hardware reset. 00 When no master is making a request the arbiter will park the slave port on the master port defined by the PARK bit field. 01 When no master is making a request the arbiter will park the slave port on the last master to be in control of the slave port. 10 When no master is making a request the arbiter will park the slave port on no master and will drive all outputs to a constant safe state. 11 Reserved
3–1	—	Reserved.
0	PARK	This bit is used to determine which master port this slave port parks on when no masters are actively making requests and the PCTL bits are set to 00. This bit is initialized by hardware reset. 0 Park on CORE Master Port 1 Park on DMA Master Port

The CR_*port* can only be accessed with 32-bit accesses. Once the read only (RO) bit has been set in the CR_*port*, the CR_*port* can only be read; attempts to write to it will have no effect on the CR_*port* and result in an error response.

14.5 Functional Description

14.5.1 Arbitration

The cross-bar switch bus supports two arbitration schemes: a simple fixed-priority comparison algorithm and a simple round-robin fairness algorithm. The arbitration scheme is independently programmable for each slave port.

14.5.1.1 Fixed-Priority Operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the PR_*port* (priority registers). If two masters both request access to a slave port the master with the highest priority in the selected priority register will gain control over the slave port.

Any time a master makes a request to a slave port, the slave port checks to see if the new requesting master's priority level is higher than that of the master that currently has control over the slave port (unless the slave port is in a parked state). The slave port does an arbitration check at every bus transfer boundary to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port the new requesting master will be granted control over the slave port at the next clock edge.

The exception to this rule is if the master that currently has control over the slave port is running a fixed length burst transfer or a locked transfer. In this case the new requesting master will have to wait until the end of the burst transfer or locked transfer before it will be granted control of the slave port.

If the new requesting master's priority level is lower than that of the master that currently has control of the slave port the new requesting master will be forced to wait until the master that currently has control of the slave port either runs an IDLE cycle or runs a non IDLE cycle to a location other than the current slave port.

14.5.1.2 Round-Robin Priority Operation

When operating in round-robin mode, each master is assigned a relative priority based on the master port number. This priority is based on how far ahead the master port number of the requesting master is to the master port number of the current bus master for this slave. Master port numbers are compared modulo the total number of bus masters, i.e. take the requesting master port number minus the current bus master's port number modulo the total number of bus masters. The master port whose priority is the highest based on this comparison will be granted control over the slave port at the next bus transfer boundary.

For the case of only the two bus masters on this device, this means when operating in round-robin mode, if the core complex is the current bus master for a given slave, then the eDMA has the highest priority when requesting that slave. Likewise, if the eDMA is the current bus master for a given slave, then the core complex has the highest priority when requesting that slave.

Once granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line will be granted access to the slave port at the next transfer boundary.

Parking may still be used in a round-robin mode, but will not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff will occur to the next master in line after one cycle of arbitration. If the slave port is put into low power park mode the round-robin pointer will be reset to point at master port 0, giving it the highest priority.

14.5.1.3 Priority Assignment

Each master port needs to be assigned a unique 1-bit priority level. If an attempt is made to program multiple master ports with the same priority level within the priority registers (PR_*port*) the cross-bar switch bus will respond with a bus error (refer to Table 6-1 on page 6-85) and the registers will not be updated.

14.6 Initialization / Application Information

No initialization is required by or for the cross-bar switch bus. Hardware reset ensures all the register bits used by the cross-bar switch bus are properly initialized.

Chapter 15
Common Flash Module (CFM)

15.1 Overview

The CFM provides up to 1 MByte of 32-bit program Flash memory with an optional 32 Kbytes of 16-bit Data Flash memory. The Flash memory serves as electrically erasable and programmable, non-volatile memory which is ideal for program and data storage for single-chip applications allowing for field reprogramming without requiring external programming voltage sources. For more information on the non-volatile memory options by device, refer to Table 1-1 on page 1-3.

The CFM contains a common Flash bus interface, IP bus interface, Flash command controller, Flash memory controller, and multiple Flash memory blocks, as shown in Figure 15-1 and Figure 15-2. The program Flash is connected directly to the core via a tightly coupled memory bus (CFM bus) for the fastest possible execution of code. The program Flash can also be accessed through the Flash command or memory controller via the peripheral bus for program, erase and verify operations as well as reading of its contents. The data Flash is intended to provide a dedicated area of memory to be used for the emulation of EEPROM functionality, and is accessed only via the peripheral bus. EEPROM emulation can be achieved using several possible software schemes by using the read-while-write capability offered between the program and data Flash memories. Execution of code from the data Flash via the peripheral bus is supported, although at lower performance than from the program Flash via the CFM bus, due to the lower speed of the peripheral bus and 16-bit width of the data Flash.

On MAC7100 Family devices the program Flash is implemented with four or eight 16-bit wide arrays of up to 128 Kbytes, providing up to 1 MByte of 32-bit wide program Flash. While the initial access across the CFM bus to the program Flash requires two f_{SYS} cycles, the arrays are interleaved such that back-to-back access to the physical blocks may be performed with an effective access time of one f_{SYS} cycle per word, after the initial access. The 32 Kbyte data Flash is implemented using a single 16-bit wide array.

The program Flash includes a configuration field to hold information controlling the state of the program and data Flash protection, access and security, as well as a security key to unlock the device, if enabled. The securing of the whole MCU is controlled by the security settings used in the CFM. The device can only be unlocked if the program Flash is in normal mode and unlocking the device with the security key has been permitted, or by using a JTAG lockout recovery mechanism to mass erase and blank verify the contents of the non-volatile memory. In order to avoid erroneous program and erase operations, the program and data Flash memories are divided into sectors which can individually be protected against those operations. The use of supervisor and data access protection further restricts these memories by allowing reads from the defined sector only when the device is in supervisor mode (supervisor access) or when the access is not for an instruction fetch (data access).

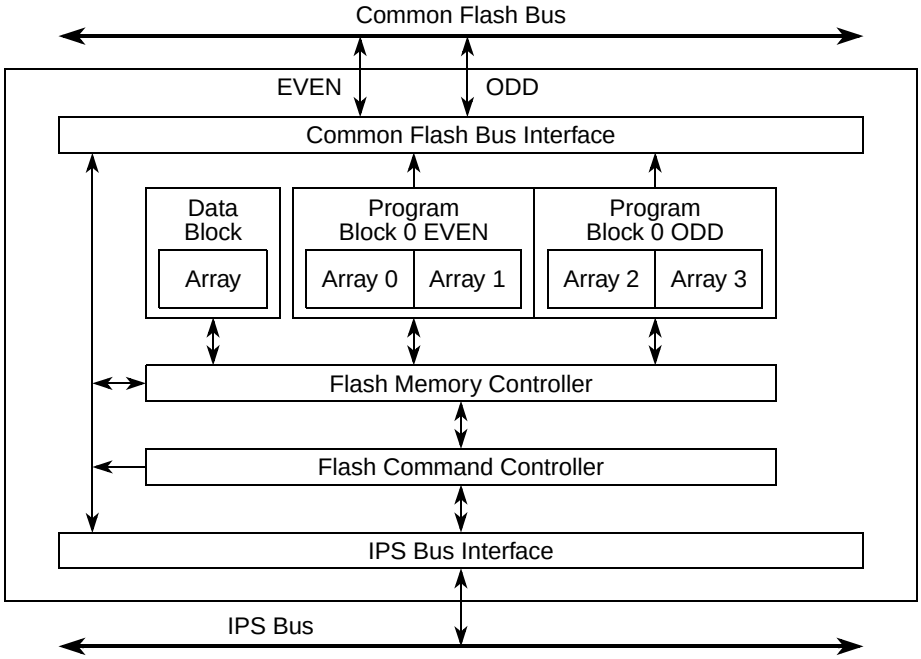


Figure 15-1. CFM Block Diagram, MAC71*x*1 and MAC71*x*2 Devices

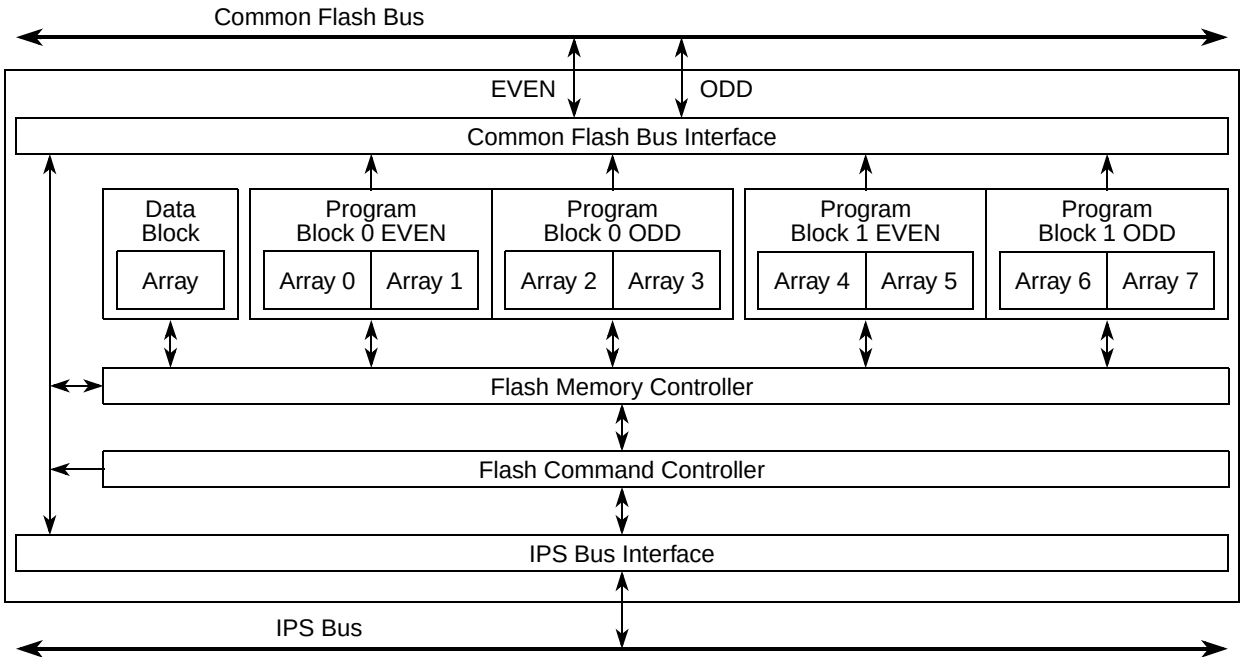


Figure 15-2. CFM Block Diagram, MAC71*x*6 Devices

In normal operation, a Flash read through the tightly coupled interface takes one or two f_{SYS} clock cycles to access each program Flash block, depending on the factory setting of the CFMCLKSEL[CLKSEL] bit. [1] Accesses may take longer if it is necessary to wait for a backdoor access or a CFM command to complete. Program Flash physical blocks are interleaved between odd and even addresses to form a program Flash logical block. Interleaving allows back-to-back read operations from the Common Flash bus to the

1. The CFMCLKSEL[CLKSEL] feature is not available on mask set L49P devices.

program Flash memory at an effective access rate of one f_{SYS} cycle per word after the initial two cycle access (if CFMCLKSEL[CLKSEL] is clear, if it is set the initial access is also one cycle [1]). Program Flash memory read operations via the IP bus take two f_{IPS} cycles per word. Data Flash memory read operations, available only via the IP bus, take two f_{IPS} cycles per half word and four f_{IPS} cycles per word. Register read operations from the IP bus execute with no wait states. The IP bus will take precedence over a transaction in progress. There is no arbitration; accesses are handled in the order they are received regardless of which bus they come from.

Write operations to the Flash memory and registers are only allowed from the IP bus. All program, erase, and verify operations are performed by the Flash memory controller through instructions from the Flash command controller. It is not possible to read from any Flash logical block while the same logical block is being erased, programmed, or verified. The CFM provides the ability to perform read-while-write operations on one program Flash logical block under control of software routines executing out of another program Flash logical block. The CFM also provides the ability to perform read-while-write operations between program and data Flash blocks. All Program Flash logical blocks can be programmed, erased, or verified concurrently. However, program, page erase and page erase verify operations cannot execute concurrently on the program and data Flash blocks.

Flash logical blocks are divided into multiple logical pages which can be erased separately. An erased bit reads 0b1 and a programmed bit reads 0b0. See Figure 15-3, Figure 15-4 and Figure 15-5 for the relationship between logical blocks, pages and addresses.

15.2 Features

- 1 MByte, 512 Kbytes or 256 Kbytes of 32-bit Program Flash memory.
- 32 Kbytes of 16-bit data Flash memory (optional).
- Concurrent Flash memory read operations on any two program Flash physical blocks from the common Flash bus and IP bus.
- Concurrent program, erase, or verify operations on all program Flash physical blocks.
- Automated program, erase, and verify operations.
- Single power supply for program and erase operations.
- Read-while-write capability.
- Software programmable interrupts on command completion, access violations, or protection violations.
- Fast page erase operation.
- Fast word program operation for program Flash blocks.
- Fast half word program operation for data Flash blocks.
- Protection scheme to prevent accidental program or erase of Flash memory.
- Access restriction control for supervisor/user and data/instruction operations.
- Security feature to prevent unauthorized access to the Flash memory.
- Data signature generation for programming integrity verification. [1]

1. Not implemented on mask set L49P and L47W devices.

15.3 Memory Map / Register Definition

The memory maps for the CFM Flash memory are shown in Figure 15-3, Figure 15-4 and Figure 15-5. The starting address of the program Flash memory is determined by the program Flash array base address as defined by the system level configuration (see Chapter 8, “Device Memory Map”). The Flash memory map shows how pairs of 32-bit program Flash physical arrays (even and odd) interleave every 4 bytes to form a contiguous memory space as shown in Table 15-1 below.

Table 15-1. CFM Flash Blocks Address Map Summary

Device Group	Offset (refer to Table 8-1 on page 8-93 for Base Address)		Data Flash Offset (refer to Table 8-10 on page 8-99 for Base Address)
	Program Flash Block 0	Program Flash Block 1	
MAC71*x*2	0x0000_0000 to 0x0003_FFFF	—	0x0000 to 0x7FFF
MAC71*x*1	0x0000_0000 to 0x0007_FFFF	—	0x0000 to 0x7FFF
MAC71*x*6	0x0000_0000 to 0x0007_FFFF	0x0008_0000 to 0x000F_FFFF	0x0000 to 0x7FFF

The starting address of the data Flash memory, independent of the program Flash array, is determined by the data Flash base address as defined by the system level configuration (see Section 8.1.6, “Peripheral Bus Memory Map,” on page 8-99). The data Flash array is not interleaved and forms a contiguous memory space as shown in Table 15-1.

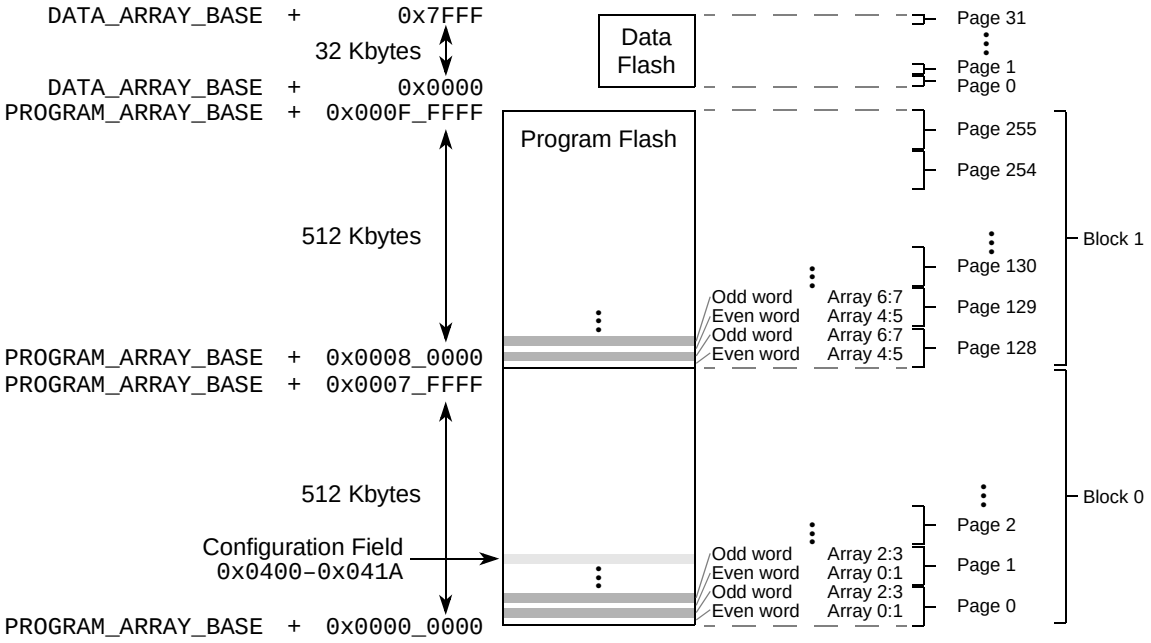


Figure 15-3. CFM Flash Memory Map Overview (MAC71*x*6 Devices)

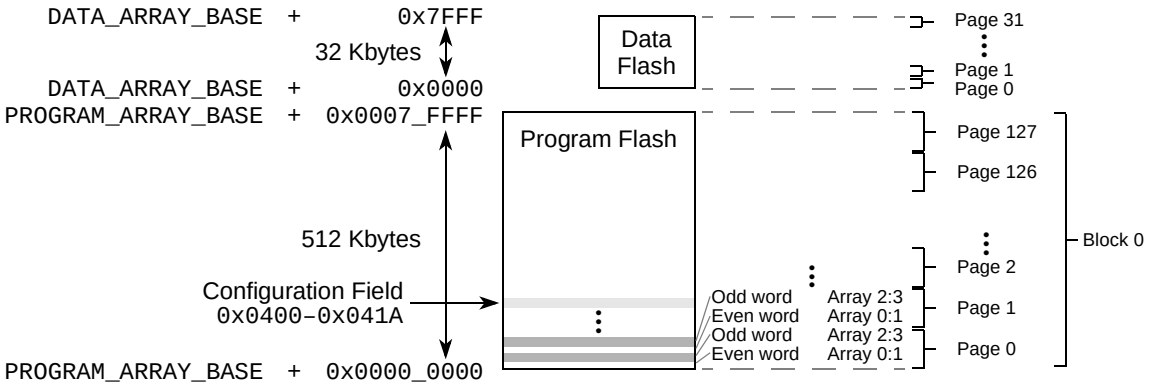


Figure 15-4. CFM Flash Memory Map Overview (MAC71*x*1 Devices)

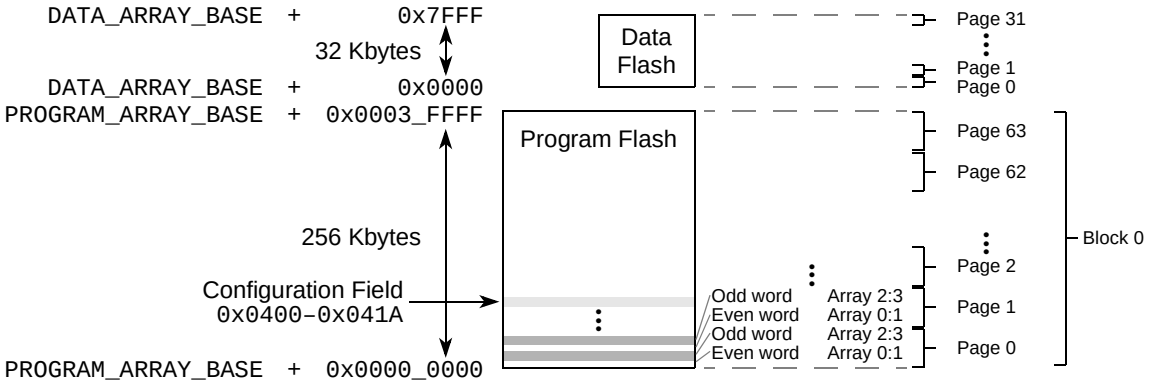


Figure 15-5. CFM Flash Memory Map Overview (MAC71*x*2 Devices)

Security information that allows the MCU to prevent intrusive access to the Flash memory is stored in the Flash configuration field. The Flash configuration field is composed of 27 bytes of reserved memory space within the program Flash memory which contains information that determines the CFM protection and access restriction scheme out of reset. A description of each byte found in the Flash configuration field is given in Table 15-2.

Table 15-2. CFM Flash Configuration Field

Address Offset from PROGRAM_ARRAY_BASE	Size (bytes)	Function	Detailed Description	Factory Default
0x0400 – 0x0407	8	Backdoor Comparison Key	Section 15.4.2.1	0xFFFF_FFFF_FFFF_FFFF
0x0408 – 0x040B	4	Program Flash Protection Bytes	Section 15.3.1.4	0xFFFF_FFFF
0x040C – 0x040F	4	Program Flash SUPV Access Bytes	Section 15.3.1.6	0xFFFF_FFFF
0x0410 – 0x0413	4	Program Flash DATA Access Bytes	Section 15.3.1.8	0xFFFF_FFFF
0x0414 – 0x0417	4	Flash Security Word	Section 15.3.1.3	0xFFFF_FFFF
0x0418	1	Data Flash Protection Byte	Section 15.3.1.5	0xFF
0x0419	1	Data Flash SUPV Access Byte	Section 15.3.1.7	0xFF
0x041A	1	Data Flash DATA Access Byte	Section 15.3.1.9	0xFF

The CFM has hardware interlocks which protect data from accidental corruption using program or erase operations. A flexible scheme allows the protection of any combination of Flash logical sectors as described in section Section 15.3.1.4, “CFM Program Flash Protection Register (CFMPROT),” for

program Flash memory and section Section 15.3.1.5, “CFM Data Flash Protection Register (CFMDFPROT),” for data Flash memory. A similar scheme is available to control supervisor/user and data/instruction access to these Flash logical sectors.

The CFM contains a set of control and status registers located at the register base address as defined by the system level configuration. A summary of the CFM register memory map is given in Table 15-3.

Table 15-3. CFM Memory Map

CFM Offset	Register Description			
	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	CFMMCR		CFMCLKD	Reserved [1]
0x0004	Reserved [1]			
0x0008	CFMSEC			
0x000C	Reserved [1]			
0x0010	CFMPROT			
0x0014	CFMSACC			
0x0018	CFMDACC			
0x001C	Reserved [1]			
0x0020	CFMUSTAT	Reserved [1]		
0x0024	CFMCMD	Reserved [1]		
0x0028 – 0x002C	Reserved [1]			
0x0030	CFMDATA0 [2]			
0x0034	Reserved [1]			
0x0038	CFMDATA1 [3]			
0x003C	Reserved [1]			
0x0040	Reserved [1]		CFMDISU [3]	
0x0044	CFMDFPROT	CFMDFSACC	CFMDFDACC	Reserved [1]
0x0048	Reserved [1]		CFMCLKSEL [4]	

1 Access to reserved address locations generate a cycle termination transfer error.

2 Mask set L49P and L47W devices do not implement this register, and the offset must be treated as reserved.

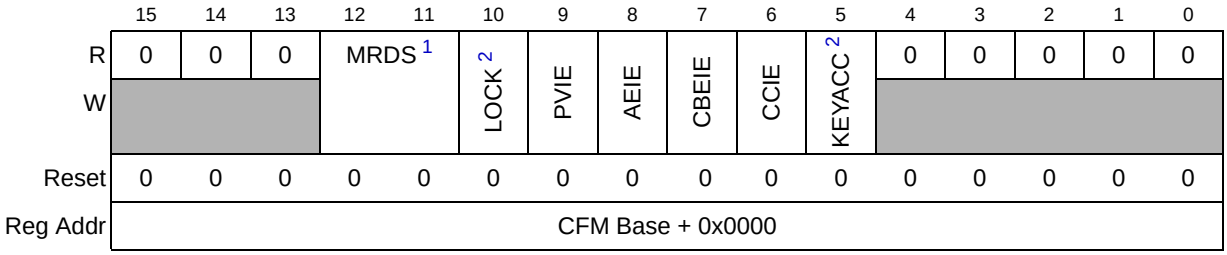
3 Mask set L49P, L47W and L61W devices do not implement this register, and the offset must be treated as reserved.

4 Mask set L49P devices do not implement this register, and the offset must be treated as reserved.

15.3.1 Register Descriptions

15.3.1.1 CFM Module Configuration Register (CFMMCR)

The CFMMCR is used to configure and control the operation of the CFM peripheral bus interface. Bits 12:5 are readable and writable, with restrictions, while the remaining bits read zero and writes are ignored.



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	MRDS [1]		LOCK [2]	PVIE	AEIE	CBEIE	CCIE	KEYACC [2]	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	CFM Base + 0x0000															

1 1519bMask set L49P and L47W devices do not implement this field, and it must be treated as reserved.

2 1519bWrites of zero to LOCK are ignored; once set it can be cleared only by a reset.

3 1519bWrites to KEYACC are ignored if CFMSEC[KEYEN] ≠ 0b10.

Figure 15-6. CFM Module Configuration Register (CFMMCR)

Table 15-4. CFMMCR Field Descriptions

Bits	Name	Description
15–13	—	Reserved.
12–11	MRDS [1]	Margin read setting. This field is used to set the sense amp margin level for reads of the Flash array, and may be used in conjunction with the data signature command to verify programming integrity, as described in Section 15.4.1.5.6, “Data Signature.” 00 Flash array reads use the normal margin. 01 Flash array reads are sensitive to the programmed-level margin. 10 Flash array reads are sensitive to the erased-level margin. 11 Reserved.
10	LOCK	Write lock control. This bit is always readable and writes of zero are ignored; thus, once set it may only be cleared by $\overline{\text{RESET}}$. 0 CFMPROT, CFMSACC, CFMDACC CFMDFPROT, CFMDFSACC, and CFMDFDACC registers are writable. 1 CFMPROT, CFMSACC, CFMDACC, CFMDFPROT, CFMDFSACC, and CFMDFDACC registers are write-locked.
9	PVIE	Protection violation interrupt enable. The PVIE bit is always readable and writable. The PVIE bit enables an interrupt in case the protection violation flag, PVIOL in the CFMUSTAT register, is set. 0 PVIOL interrupt disabled. 1 Interrupt requested when the PVIOL flag is set.
8	AEIE	Access error interrupt enable. The AEIE bit is always readable and writable. The AEIE bit enables an interrupt in case the access error flag, ACCERR in the CFMUSTAT register, is set. 0 ACCERR interrupt disabled. 1 Interrupt requested when the ACCERR flag is set.
7	CBEIE	Command buffer empty interrupt enable. The CBEIE bit is always readable and writable. The CBEIE bit enables an interrupt in case the command buffer empty flag, CBEIF in the CFMUSTAT register, is set. 0 CBEIF interrupt disabled. 1 Interrupt requested when the CBEIF flag is set.

Table 15-4. CFMMCR Field Descriptions (continued)

Bits	Name	Description
6	CCIE	Command complete interrupt enable. The CCIE bit is always readable and writable. The CCIE bit enables an interrupt in case the command completion flag, CCIF in the CFMUSTAT register, is set. 0 CCIF interrupt disabled. 1 Interrupt requested when the CCIF flag is set.
5	KEYACC	Enable Security Key Writing. The KEYACC bit is always readable and only writable if the CFMSEC[KEYEN] field is set to enable backdoor key access (0b10). 0 Writes to CFM Flash memory are interpreted as the start of a command write sequence. 1 Writes to CFM Flash memory are interpreted as keys to release security.
4–0	—	Reserved.

[1] Mask set L49P and L47W devices do not implement this field, and it must be treated as reserved.

15.3.1.2 CFM Clock Divider Register (CFMCLKD)

The CFMCLKD register is used to control the period of the clock used for timed events in program and erase algorithms. All CFMCLKD register bits are readable while bits [6:0] are write once and bit 7 is not writable.

The CFMCLKD register bits PRDIV8 and DIV must be set with appropriate values before programming or erasing the CFM Flash memory Section 15.4.1.4, “Initializing the CFMCLKD Register.”

	7	6	5	4	3	2	1	0
R	DIVLD	PRDIV8	DIV					
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	CFM Base + 0x0002							

Figure 15-7. CFM Clock Divider Register (CFMCLKD)

Table 15-5. CFMCLKD Field Descriptions

Bits	Name	Description
7	DIVLD	Clock divider loaded 0 CFMCLKD register has not been written. 1 CFMCLKD register has been written to since the last reset.
6	PRDIV8	Enable Prescaler by 8 0 The IP bus clock is fed directly into the clock divider. 1 A prescaler divides the IP bus clock by 8 before feeding into the clock divider.
5–0	DIV[5:0]	Clock divider bits. PRDIV8 and DIV set the divide ratio used to generate f_{NVMOP} from the IP bus clock. Refer to Section 15.4.1.4, “Initializing the CFMCLKD Register,” for more detailed information.

15.3.1.3 CFM Security Register (CFMSEC)

The CFMSEC register is used to store the Flash security word and CFM security state. Register bits [31:29,1:0] are readable while remaining bits read zero and all bits are not writable. CFMSEC is loaded from the Flash configuration field in the program Flash block at offset 0x0414 during the reset sequence.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	KEYEN		SECSTAT	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	—[1]		—[2]	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	CFM Base + 0x0008															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NV														SEC	
W																
Reset	—[1]															
Reg Addr	CFM Base + 0x0008															

1 Reset state loaded from Flash configuration field during reset. Refer to Table 15-2

2 Reset state determined by security state of CFM.

Figure 15-8. CFM Security Register (CFMSEC)

Table 15-6. CFMSEC Field Descriptions

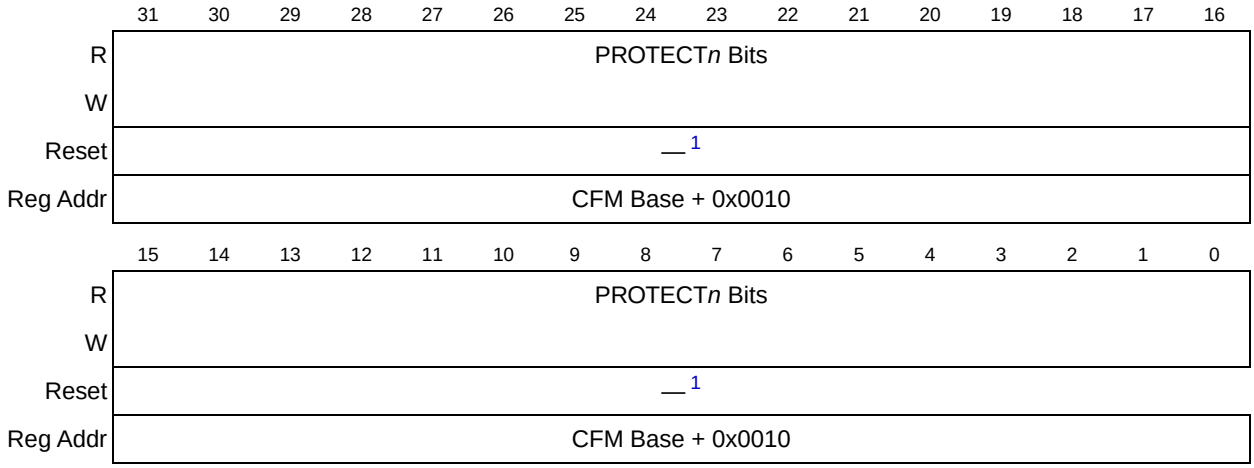
Bits	Name	Description
31–30	KEYEN[1:0]	Backdoor key access state. Enable/disable security unlock key. 0x Backdoor key access disabled. 10 Backdoor key access enabled. 11 Backdoor key access disabled.
29	SECSTAT	Flash memory security status. 0 Flash security is disabled. 1 Flash security is enabled.
28–16	—	Reserved.
15–2	NV[13:0]	Non-volatile flags. The NV bits are available as user-defined flags.
1–0	SEC[1:0]	Security state. Define the security state of the Flash array. 0x Flash memory secured. 10 Flash memory not secured. 11 Flash memory secured. Refer to Section 15.4.2, “Flash Security Operation,” for details.

15.3.1.4 CFM Program Flash Protection Register (CFMPROT)

The CFMPROT register defines which program Flash logical sectors are protected against program and erase operations. All CFMPROT register bits are readable and only writable when LOCK = 0.

The program Flash memory is divided into logical sectors for the purpose of data protection using the CFMPROT register. The program Flash memory consists of 15 lower and 15 upper 4 Kbyte sectors and 2 middle sectors of 452 Kbytes as shown in Figure 15-10, 196 Kbytes as shown in Figure 15-11 or 68 Kbytes as shown in Figure 15-12.

In order to change the program Flash memory protection on a temporary basis, the CFMPROT register should be written after the LOCK bit in the CFMMCR has been cleared. To change the program Flash memory protection that will be loaded during the reset sequence, the program Flash logical sector containing the Flash configuration field must first be unprotected, then the program Flash protection bytes must be programmed with the desired value.



1 Reset state loaded from Flash configuration field during reset. Refer to Table 15-2.

Figure 15-9. CFM Program Flash Protection Register (CFMPROT)

Table 15-7. CFMPROT Field Descriptions

Bits	Name	Description
31–0	PROTECT*n*	Each program Flash logical sector can be protected from program and erase operations by setting the PROTECT*n* bit. 0 Program Flash logical sector M is not protected. 1 Program Flash logical sector M is protected.

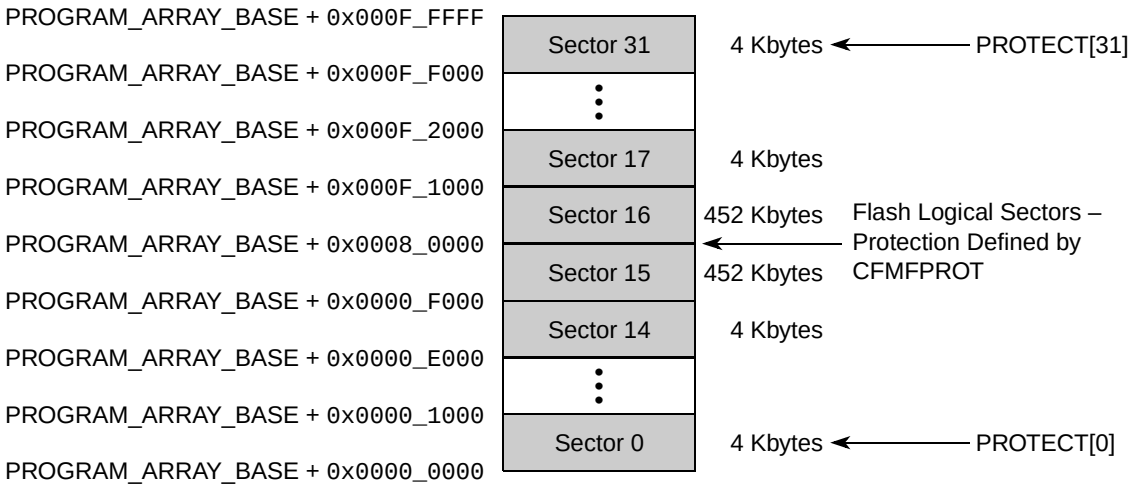


Figure 15-10. CFMPROT Protection Diagram, MAC71*x*6 Devices

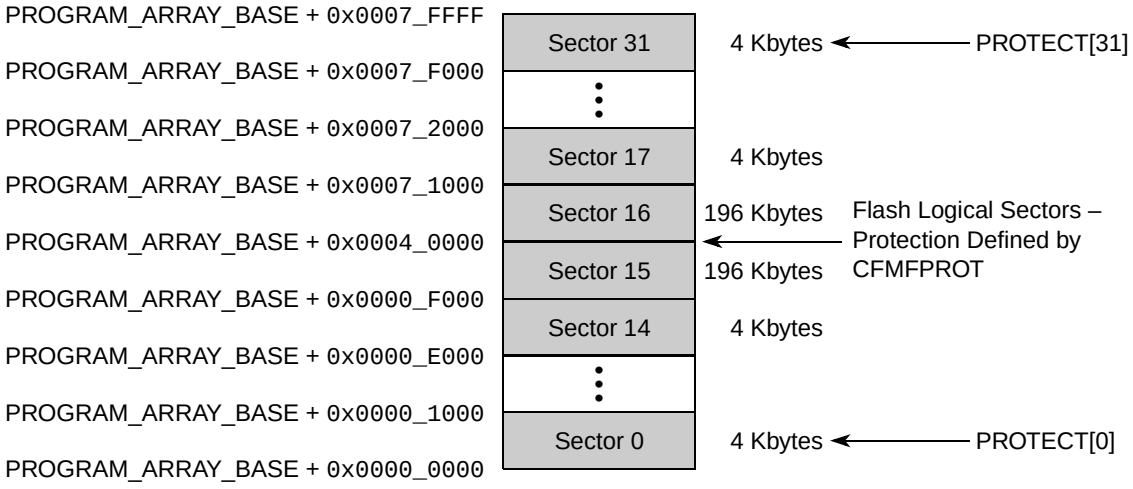


Figure 15-11. CFMPROT Protection Diagram, MAC71*x*1 Devices

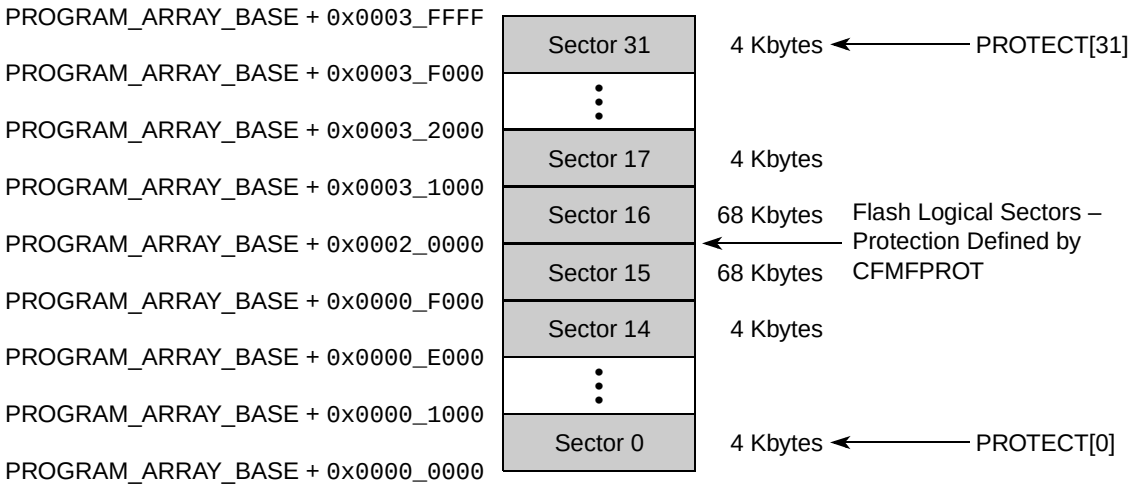


Figure 15-12. CFMPROT Protection Diagram, MAC71*x*2 Devices

15.3.1.5 CFM Data Flash Protection Register (CFMDFPROT)

The CFMDFPROT register defines which data Flash logical sectors are protected against program and erase operations. All CFMDFPROT register bits are readable and only writable when LOCK = 0.

The data Flash memory is divided into logical sectors for the purpose of data protection using the CFMDFPROT register. The data Flash memory consists of 2 lower 2 Kbyte sectors, 2 upper 2 Kbyte sectors, followed by a lower 4 Kbyte sector, an upper 4 Kbyte sector, and 2 middle sectors each with 8 Kbytes as shown in Figure 15-14.

In order to temporarily change the data Flash memory protection, the CFMDFPROT register should be written after the LOCK bit in the CFMMCR has been cleared. To change the data Flash memory protection that will be loaded during the reset sequence, the program Flash logical sector containing the Flash configuration field must first be unprotected, then the data Flash protection byte must be programmed with the desired value.

	7	6	5	4	3	2	1	0
R	PROTECT_DF							
W								
Reset	—[1]							
Reg Addr	CFM Base + 0x0044							

[1] Reset state loaded from Flash configuration field during reset. Refer to Table 15-2.

Figure 15-13. CFM Data Flash Protection Register (CFMDFPROT)

Table 15-8. CFMDFPROT Field Descriptions

Bits	Name	Description
7–0	PROTECT_DF*n*	Each data Flash logical sector can be protected from program and erase operations by setting the PROTECT_DF[M] bit. 0 Data Flash logical sector M is not protected. 1 Data Flash logical sector M is protected.

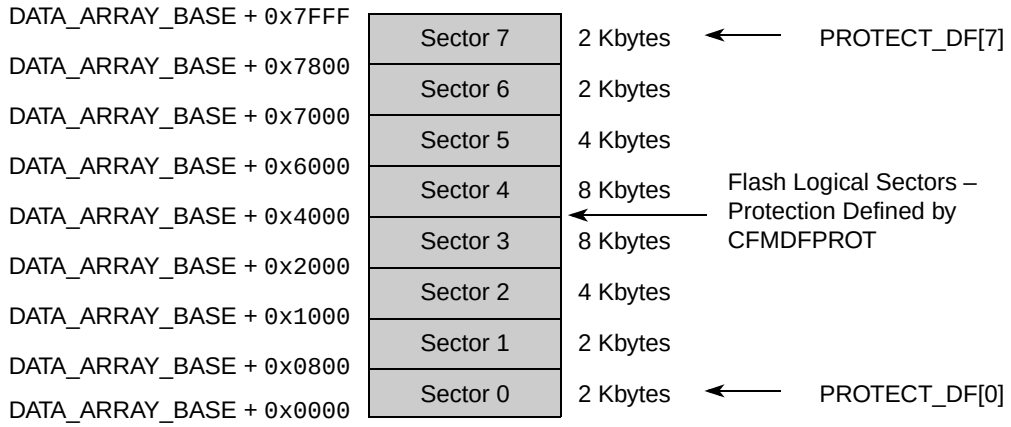
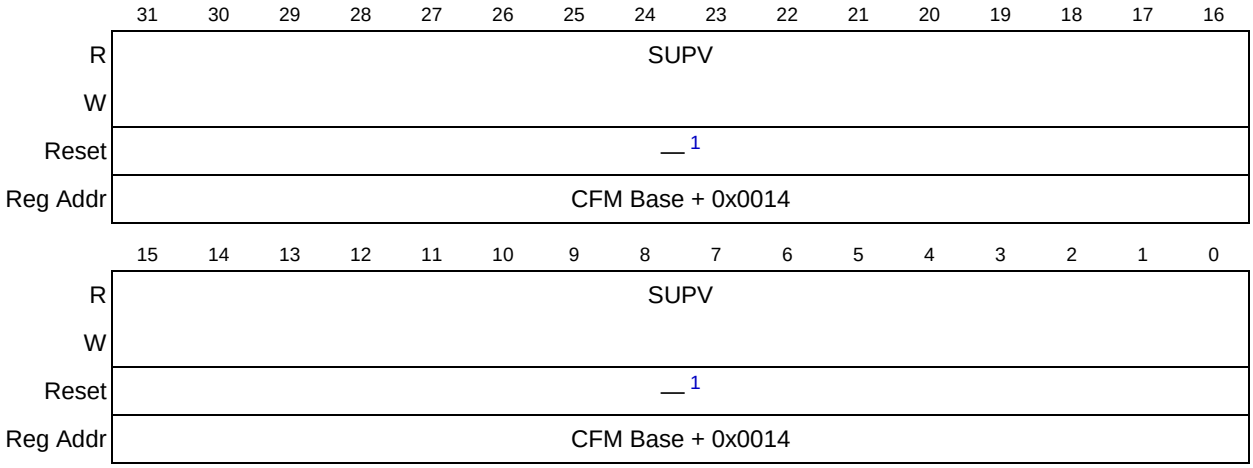


Figure 15-14. CFMDFPROT Protection Diagram

15.3.1.6 CFM Program Flash Supervisor Access Register (CFMSACC)

The CFMSACC register is used to control supervisor/user access to the program Flash memory. All CFMSACC register bits are readable and only writable when LOCK = 0.

In order to change the program Flash supervisor access on a temporary basis, the CFMSACC register should be written after the LOCK bit in the CFMMCR has been cleared. To change the program Flash supervisor access that will be loaded during the reset sequence, the program Flash logical sector containing the Flash configuration field must first be unprotected, then the program Flash supervisor access bytes must be programmed with the desired value. Each program Flash logical sector may be mapped into supervisor or unrestricted address space (see Figure 15-10, Figure 15-11 and Figure 15-12 for details on program Flash sector mapping).



1 Reset state loaded from Flash configuration field during reset. Refer to Table 15-2.

Figure 15-15. CFM Program Flash Supervisor Access Register (CFMSACC)

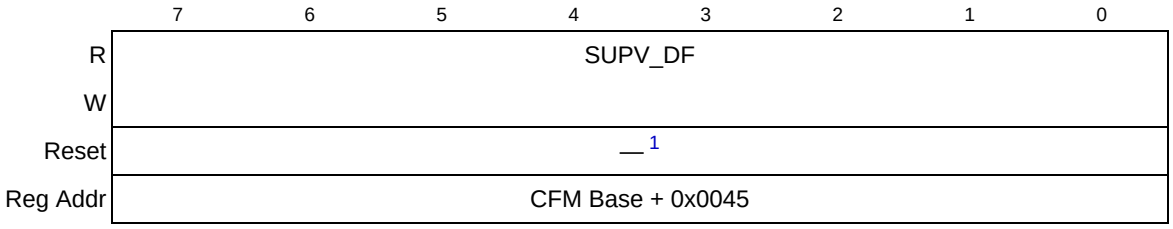
Table 15-9. CFMSACC Field Descriptions

Bits	Name	Description
31–0	SUPV*n*	Program Flash address space assignment for supervisor/user access 0 Program Flash logical sector M is placed in unrestricted address space. 1 Program Flash logical sector M is placed in supervisor address space.

15.3.1.7 CFM Data Flash Supervisor Access Register (CFMDFSACC)

The CFMDFSACC register is used to control supervisor/user access to the data Flash memory. All CFMDFSACC register bits are readable and only writable when LOCK = 0.

In order to change the data Flash supervisor access on a temporary basis, the CFMDFSACC register should be written after the LOCK bit in the CFMMCR has been cleared. To change the data Flash supervisor access that will be loaded during the reset sequence, the program Flash logical sector containing the Flash configuration field must first be unprotected, then the data Flash supervisor access byte must be programmed with the desired value. Each data Flash logical sector may be mapped into supervisor or unrestricted address space (see Figure 15-14 for details on data Flash sector mapping).



1 Reset state loaded from Flash configuration field during reset. Refer to Table 15-2.

Figure 15-16. CFM Data Flash Supervisor Access Register (CFMDFSACC)

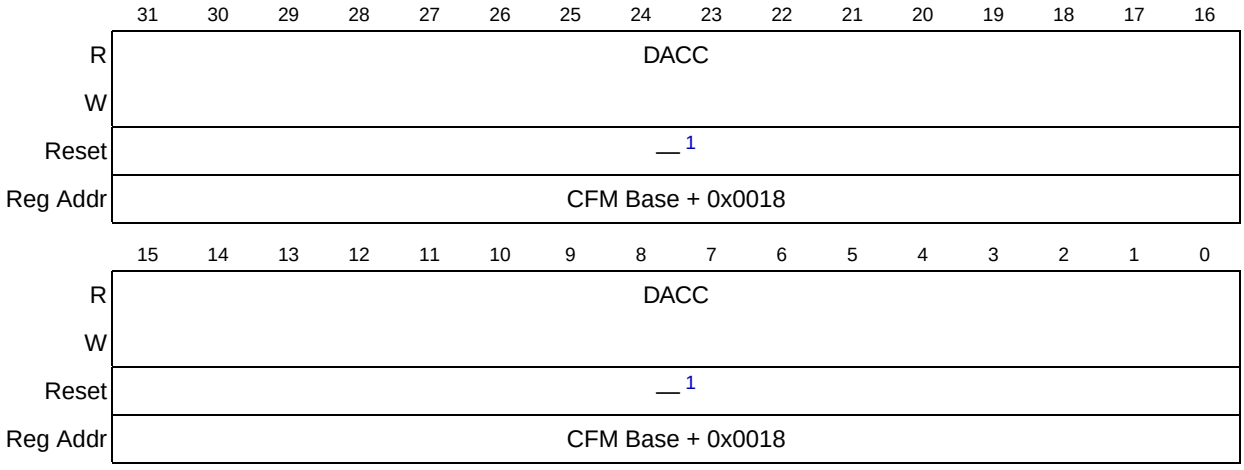
Table 15-10. CFMDFSACC

Bits	Name	Description
7–0	SUPV_DF*n*	Data Flash address space assignment for supervisor/unrestricted access. 0 Data Flash logical sector M is placed in unrestricted address space. 1 Data Flash logical sector M is placed in supervisor address space.

15.3.1.8 CFM Program Flash Data Access Register (CFMDACC)

The CFMDACC register is used to control data/instruction access to the program Flash memory All CFMDACC register bits are readable and only writable when LOCK = 0.

In order to change the program Flash data access on a temporary basis, the CFMDACC register should be written after the LOCK bit in the CFMMCR has been cleared. To change the program Flash data access that will be loaded during the reset sequence, the program Flash logical sector containing the Flash configuration field must first be unprotected, then the program Flash data access bytes must be programmed with the desired value. Each program Flash logical sector may be mapped into data or both data and instruction address space (see Figure 15-10, Figure 15-11 and Figure 15-12 for details on program Flash sector mapping).



1 Reset state loaded from Flash configuration field during reset. Refer to Table 15-2.

Figure 15-17. CFM Program Flash Data Access Register (CFMDACC)

Table 15-11. CFMDACC Field Descriptions

Bits	Name	Description
31–0	DACC*n*	Program Flash memory address space assignment for data/instruction access 0 Program Flash logical sector M is placed in data address space. 1 Program Flash logical sector M is placed in data and instruction address space.

15.3.1.9 CFM Data Flash Data Access Register (CFMDFDACC)

The CFMDFDACC register is used to control data/instruction access to the data Flash memory. All CFMDFDACC register bits are readable and only writable when LOCK = 0.

In order to change the data Flash data access on a temporary basis, the CFMDFDACC register should be written after the LOCK bit in the CFMMCR has been cleared. To change the data Flash data access that will be loaded during the reset sequence, the program Flash logical sector containing the Flash configuration field must first be unprotected, then the data Flash data access byte must be programmed with the desired value. Each data Flash logical sector may be mapped into data or data and instruction address space (see Figure 15-14 for details on data Flash sector mapping).

	7	6	5	4	3	2	1	0
R	DACC_DF							
W								
Reset	—[1]							
Reg Addr	CFM Base + 0x0046							

[1] Reset state loaded from Flash configuration field during reset. Refer to Table 15-2.

Figure 15-18. CFM Data Flash Data Access Register (CFMDFDACC)

Table 15-12. CFMDFDACC Field

Bits	Name	Description
7–0	DACC_DF*n*	Data Flash address space assignment for data/instruction access 0 Data Flash logical sector M is placed in data address space. 1 Data Flash logical sector M is placed in data and instruction address space.

15.3.1.10 CFM User Status Register (CFMUSTAT)

The CFMUSTAT register defines the Flash command controller status and Flash memory access, protection and verify status. CFMUSTAT register bits CBEIF, PVIOL, ACCERR, and BLANK are readable and writable while CCIF is readable but not writable, and remaining bits read zero and are not writable. Note that only one CFMUSTAT register bit can be cleared at a time.

	7	6	5	4	3	2	1	0
R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
W								
Reset	1	1	0	0	0	0	0	0
Reg Addr	CFM Base + 0x0020							

Figure 15-19. CFM User Status Register (CFMUSTAT)

Table 15-13. CFMUSTAT Field Descriptions

Bits	Name	Description
7	CBEIF	Command buffer empty interrupt flag. The CBEIF flag, set by the Flash command controller, indicates that the address, data and command buffers are empty so that a new command write sequence can be started. The CBEIF flag is cleared by writing a 0b1 to CBEIF as part of a command write sequence. Writing a 0b0 to the CBEIF flag has no effect on CBEIF but can be used to abort a command write sequence. The CBEIF flag can generate an interrupt if the CBEIE bit in the CFMMCR is set. 0 Buffers are full. 1 Buffers are ready to accept a new command write sequence.
6	CCIF	Command complete interrupt flag. The CCIF flag, set by the Flash command controller, indicates that there are no more commands pending. The CCIF flag is cleared by the Flash command controller when CBEIF is cleared and sets upon completion of all active and pending commands. Writing to the CCIF flag has no effect on CCIF. The CCIF flag can generate an interrupt if the CCIE bit in the CFMMCR is set. 0 Command in progress. 1 All commands are completed.
5	PVIOL	Protection violation. The PVIOL flag, set by the Flash command controller, indicates an attempt was made to program or erase an address in a protected Flash logical sector. The PVIOL flag is cleared by writing a “1" to PVIOL. Writing a 0b0to the PVIOL flag has no effect on PVIOL. While the PVIOL flag is set, it is not possible to launch a command or start a command write sequence. 0 No protection violation has been detected. 1 Protection violation has occurred.
4	ACCERR	Access Error. The ACCERR flag, set by the Flash command controller, indicates an illegal access was made to the Flash memory or registers caused by an illegal command write sequence. The ACCERR flag is cleared by writing a 0b1 to the ACCERR flag. Writing a 0b0 to the ACCERR flag has no effect on ACCERR. While the ACCERR flag is set, it is not possible to launch a command or start a command write sequence. See section Section 15.4.1.6, “Flash Normal Mode Illegal Operations,” for details on what action sets the ACCERR flag. 0 No access error has been detected. 1 Access error has occurred.
3	—	Reserved.

Table 15-13. CFMUSTAT Field Descriptions (continued)

Bits	Name	Description
2	BLANK	All Flash memory locations or the selected Flash logical page have been verified as erased. The BLANK flag, set by the Flash command controller, indicates that a blank check or page erase verify operation has checked all Flash memory locations or the selected Flash logical page and found them to be erased. The BLANK flag is cleared by writing a 0b1 to BLANK. Writing a 0b0 to the BLANK flag has no effect on BLANK. 0 If a blank check or page erase verify command has been executed, and the CCIF flag is set, then a zero in the BLANK flag indicates that all Flash memory locations are not erased or the selected Flash logical page is not erased. 1 All Flash memory locations or selected logical page verify as erased.
1–0	—	Reserved.

15.3.1.11 CFM Command Register (CFMCMD)

The CFMCMD register is the Flash command register. All CFMCMD register bits are readable and writable except bit 7 which reads zero and is not writable.

	7	6	5	4	3	2	1	0
R	0	CMD						
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	CFM Base + 0x0024							

Figure 15-20. CFM Command Register (CFMCMD)

Table 15-14. CFMCMD Field Descriptions

Bits	Name	Description
7	—	Reserved.
6–0	CMD[6:0]	Command. Valid Flash memory commands are shown below. Writing a command other than those listed during a command write sequence will cause the CFMUSTAT[ACCERR] flag to be set. 05 Blank Check 06 Page Erase Verify 20 Program Word/Half Word 40 Page Erase 41 Mass Erase 65 Data Signature [1]

[1] Mask set L49P devices do not implement this command; writing it will set CFMUSTAT[ACCERR].

15.3.1.12 CFM Data Registers (CFMDATA1/0)

The CFMDATA register is the Flash data register for reading the data signature command response. All CFMDATA1/0 register bits read-only. Attempts to write to the CFMDATA1/0 registers result in a cycle

termination transfer error. When a data signature command is executed, the response is placed in the CFMDATA1/0 registers and remains valid until the start of the next command write sequence. When the next command write sequence begins, the CFMDATA1/0 register values return to all 0. Note that following a data signature operation on the data Flash, CFMDATA0[31:16] is always zero.

NOTE

Mask set L49P, and L47W devices do not implement this register. Mask set L61W devices only implement CFMDATA0.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	CFM Base + 0x0030 or 0x0038															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	CFM Base + 0x0030 or 0x0038															

Figure 15-21. CFM Data Registers (CFMDATA1/0)

15.3.1.13 CFM Disable Upper Block Register (CFMDISU)

The CFMDISU register reflects the factory setting to enable/disable the upper program Flash logical block. All CFMDISU register bits are programmed at the factory and are read only. Attempts to write to the CFMDISU register result in a cycle termination transfer error.

NOTE

Mask set L49P, L47W and L61W devices do not implement this register.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DISU															
W																
Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]
Reg Addr	CFM Base + 0x0042															

[1] Set at factory during manufacturing test.

Figure 15-22. CFM Disable Upper Register (CFMDISU)

Table 15-15. CFMDISU Field Descriptions

Bits	Name	Description
15–0	DISU[15:0]	Disable Upper Program Flash Logical Block 0xA5A5 Upper 512 Kbyte Program Flash block disabled All other Upper 512 Kbyte Program Flash block enabled

15.3.1.14 CFM Clock Select Register (CFMCLKSEL)

The CFMCLKSEL register reflects the factory setting for read access latency from the system bus to the program Flash block. All CFMCLKSEL register bits are readable and not writable.

NOTE

Mask set L49P devices do not implement this register.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CLKSEL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	F[1]	F[1]
Reg Addr	CFM Base + 0x004A															

[1] Set at factory during manufacturing test.

Figure 15-23. CFM Clock Select Register (CFMCLKSEL)

Table 15-16. CFMCLKSEL Field Descriptions

Bits	Name	Description
15–2	—	Reserved.
1–0	CLKSEL[1:0]	Program Flash read access latency. This field is programmed at the factory during manufacturing testing, and indicates read access latency to the program Flash block via the tightly coupled memory bus. 00 Two-cycle program Flash block read access (2-1-1-1 burst) 01 Single-cycle program Flash block read access (1-1-1-1 burst) 1x Reserved.

15.4 Functional Description

The Flash module operates in one of two modes: normal or security. The following sections describe the various operations that are available or specifically prohibited in each mode.

15.4.1 Flash Normal Mode

In Flash normal mode, the user can access the CFM registers and the program Flash memory via the IP bus (see Section 15.3, “Memory Map / Register Definition”) with minimal restrictions, and execute commands to erase, program and verify the contents of the Flash.

15.4.1.1 Read Operation

A valid read operation occurs whenever a transfer request is initiated on the Common Flash bus or the IP bus, the address is equal to an address within the valid range of the CFM Flash memory space and the read/write control indicates a read cycle.

15.4.1.1.1 Bus Priority during Read Operations

If a read access is simultaneously requested on the Common Flash bus and the IP bus to the same Flash physical block, the IP bus will be granted access and signal the core platform to hold off additional read accesses to the same Flash physical block until the CFM completes the IP bus read access.

15.4.1.2 Write Operation

A valid write operation occurs whenever a transfer request is initiated on the IP bus, the address is equal to an address within the valid range of the CFM Flash memory space and the read/write control indicates a write cycle. The action taken on a valid Flash array write depends on the subsequent user command issued as part of a valid command write sequence. Only 32-bit write operations are allowed to the program Flash memory space and 16-bit write operations to the data Flash memory space. Byte and half word write operations to the program Flash memory space and byte or word write operations to the data Flash memory space will result in a cycle termination transfer error.

15.4.1.2.1 Bus Arbitration During Write Operations

Once a command has been successfully launched as below, the CFM will signal the Core platform to hold off read accesses to any active Flash physical block until all active and buffered commands have completed (CCIF = 1). A Flash write operation from the IP bus will hold off the Core platform until it is completed.

15.4.1.3 Command Launch Sequence

The Flash command controller is used to supervise the command write sequence to execute blank check, page erase verify, program, page erase, mass erase and data signature algorithms. A specific sequence, consisting of three steps, must be strictly followed, with writes to the CFM not permitted between the steps. However, Flash register and array reads are allowed during a command write sequence. The basic command write sequence is as follows:

1. Write to one or more addresses in the Flash memory.
2. Write a valid command to the CFMCMD register.
3. Launch the command by writing a 0b1 to CFMUSTAT[CBEIF] (which clears the flag).

Before starting a command write sequence, the CFMUSTAT[ACCERR and PVIOL] flags must be clear and the CBEIF flag should be tested to determine the state of the address, data and command buffers. If the CBEIF flag is set, indicating the buffers are empty, a new command write sequence can be executed.

When the CBEIF flag is cleared, the CFMUSTAT[CCIF] flag will be cleared by the Flash command controller, indicating that the command was successfully launched. The CBEIF flag will typically be set again to indicate that the address, data and command buffers are ready for a new command write sequence

to begin. A buffered command will wait for the active command to be completed before being launched. The CFMUSTAT[CCIF] flag will set upon completion of all active and buffered commands.

The CFMCMD register as well as the associated address and data registers operate as a buffer and a register (2-stage FIFO), so that a new command along with the necessary data and address can be written to the buffer while the previous command is still in progress. This buffering operation provides time optimization when programming more than one word on a physical row in the Flash memory, as the high voltage generation can be kept active in between two programming operations, thereby saving the time overhead needed for setup of the high voltage charge pumps. Buffer empty as well as command completion are signalled by flags in the CFMUSTAT register with interrupts generated, if enabled.

A command write sequence can be aborted at any time prior to clearing the CFMUSTAT[CBEIF] flag by writing a 0b0 to CBEIF. The CFMUSTAT[ACCERR] flag will be set after successfully aborting a command write sequence, and ACCERR must be cleared prior to starting a new command write sequence.

15.4.1.4 Initializing the CFMCLKD Register

The Flash command controller uses an independent timebase, f_{NVMOP}, to execute CFM algorithms for program and erase operations. f_{NVMOP} is derived from the IP bus clock via a selectable prescaler and a programmable divider, as shown in Figure 15-24.

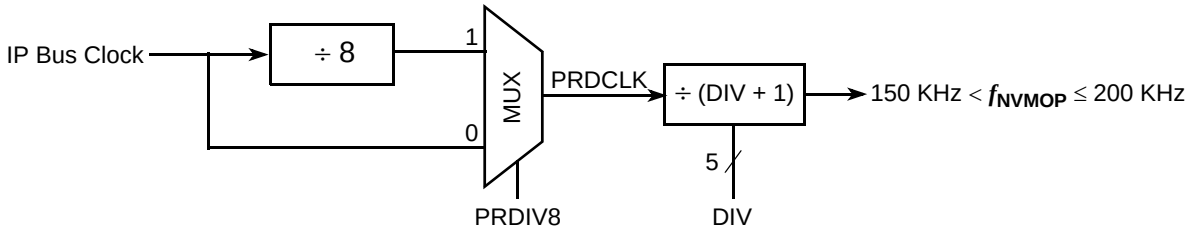


Figure 15-24. CFM f_{NVMOP} Generation Logical Block Diagram

Prior to issuing any commands, it is necessary to write the CFMCLKD register to set the ratio between the IP bus clock frequency, f_{IPS} (equal to the system clock frequency, f_{SYS}, divided by two), and f_{NVMOP} such that f_{NVMOP} is within the range of 150 KHz to 200 KHz. The equation used to calculate f_{NVMOP} is:

$$f_{NVMOP} = \frac{PRDCLK}{DIV + 1} \qquad \textit{Eqn. 15-1}$$

If the CFMCLKD[DIVLD] bit is zero, the CFMCLKD register has not been written since the last reset. When the CFMCLKD register is written to load the PRDIV8 and DIV values, the DIVLD bit is set automatically. No command can be executed if the CFMCLKD register has not been initialized (see Section 15.4.1.6, “Flash Normal Mode Illegal Operations”).

Flash software code for the MAC7100 family may use a macro to compute the clock divider. Pseudocode illustrating such a macro is shown below.

1. Determine if the prescaler is required (CFMCLKD[PRDIV8] value):

```
if (ips_clock > 12.8 MHz)
    PRDIV8 = 1
    PRDCLK = ips_clock/8
else
    PRDIV8 = 0
    PRDCLK = ips_clock
```

2. Calculate CFMCLKD[DIV] based on selected PRDCLK: [1]

```
if (PRDCLK[KHz] is divisible by 200 KHz)
    DIV = (PRDCLK[KHz] / 200 KHz) - 1
else
    DIV = INT(PRDCLK[KHz] / 200 KHz)
```

For example, if the input clock frequency is 33 MHz, CFMCLKD[DIV] field should be set to 0x14 and PRDIV8 set to 0b1. The resulting f_{NVMOP} is 196.4 KHz. This configures the Flash memory program and erase algorithm to run 1.78% slower than the optimum target program/erase rate:

$$\frac{200 - 196.4}{200} \times 100 = 1.78\% \qquad \textit{Eqn. 15-2}$$

Table 15-17 lists some example f_{NVMOP} frequencies that are appropriate for 40 MHz and 50 MHz system operation, and their deviation from the optimum program/erase frequency.

Table 15-17. CFMCLKD Register Values For 40 MHz and 50 MHz f_{SYS}

f_{IPS} ($f_{SYS} \div 2$)	PRDIV8	PRDCLK	DIV	f_{NVMOP}	200 KHz Δ
25 MHz	1	3.125 MHz	15	195,312.50 Hz	2.34%
25 MHz	1	3.125 MHz	19	156,250.00 Hz	21.88%
20 MHz	1	2.500 MHz	12	192,307.69 Hz	3.85%
20 MHz	1	2.500 MHz	15	156,250.00 Hz	21.88%

NOTE

Program and Erase command execution time are increased proportionally with the period of f_{NVMOP}.

Setting CFMCLKD to a value such that $f_{NVMOP} < 150$ KHz should be avoided, as this can destroy the Flash memory due to overstress. $f_{NVMOP} = 200$ KHz gives the fastest program and erase performance. Setting CFMCLKD to a value such that $f_{NVMOP} > 200$ KHz can result in incomplete programming or erasure of the Flash memory array cells.

1. INT(X) means taking the integer part of X; for example, INT(4.32) = 4.

15.4.1.5 Program, Erase, and Verify Operations

The following sections describe the function of each CFM command and the detailed write sequence required for each. Table 15-18 summarizes the available Flash normal mode commands.

Table 15-18. CFM Flash Memory Commands

CFMCMD	Command	Description
0x05	Blank Check	Verify that the entire Flash memory is erased. If all bits are erased, the CFMUSTAT[BLANK] bit (see Figure 15-19) will be set upon command completion.
0x06	Page Erase Verify	Verify that a Flash logical page is erased. If all bits in the Flash logical page are erased, the CFMUSTAT[BLANK] bit (see Figure 15-19) will be set upon command completion.
0x20	Program	Program Flash: Program a 32-bit word. Data Flash: Program a 16-bit half word.
0x40	Page Erase	Erase a Flash logical page. The logical page protection must be disabled prior to executing this command. Refer to Section 15.4.2, “Flash Security Operation,” for special consideration following a page erase of page 0.
0x41	Mass Erase	Erase the entire Flash memory. All Flash memory (data and program) protection must be disabled prior to executing this command. Refer to Section 15.4.2, “Flash Security Operation,” for special consideration following mass erase.
0x65 [1]	Data Signature	Generate a data signature from the selected portion of Flash memory. The response is returned in the CFMDATA1/0 register(s). [1]

[1] Mask set L49P and L47W devices do not implement the data signature command, and writing it will set CFMUSTAT[ACCERR]. Mask set L61W devices implement only the CFMDATA0 register.

15.4.1.5.1 Blank Check

The blank check operation may be used to verify that all Flash memory addresses in the CFM are erased.

An example flow to execute the blank check command is shown in Figure 15-25. The blank check command write sequence is as follows:

1. Write to any Flash memory address to start the command write sequence for the blank check command. The specific address and data written during the blank check command write sequence will be ignored.
2. Write the blank check command, 0x05, to the CFMCMD register.
3. Write 0b1 to CBEIF to clear it, and to launch the blank check command.

Since all Flash physical blocks are verified simultaneously, the number of f_{IPS} clock cycles required to execute the blank check operation on a fully erased Flash memory is equal to the number of word addresses in a program Flash logical block plus fifteen f_{IPS} clock cycles as measured from the time the CFMUSTAT[CBEIF] flag is cleared until the CFMUSTAT[CCIF] flag is set. Upon completion of the blank check operation (CCIF = 1), the CFMUSTAT[BLANK] flag will set if the entire Flash memory is erased. If any Flash memory location is not erased, the blank check operation will terminate and the BLANK flag will remain clear.

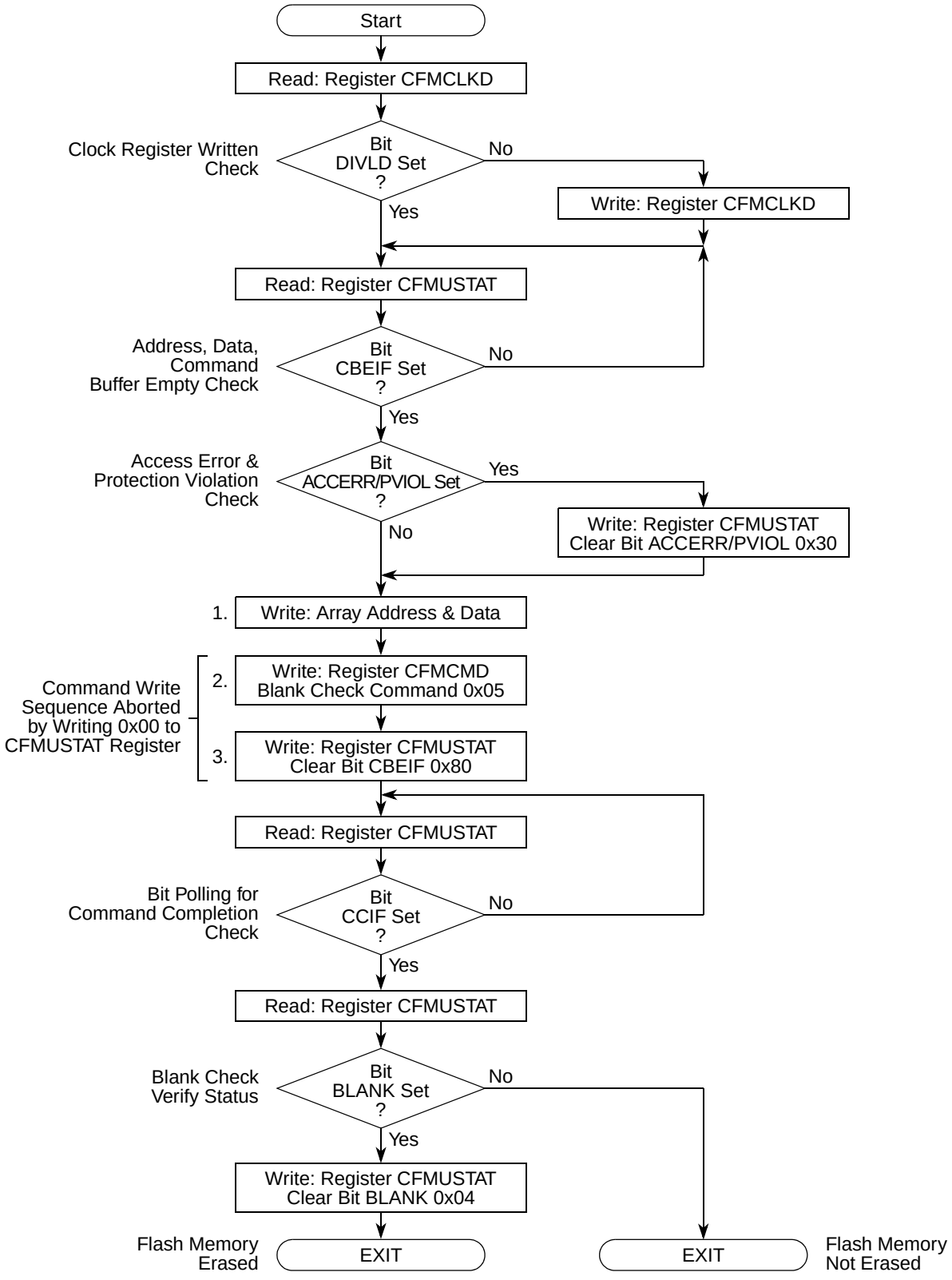


Figure 15-25. CFM Example Blank Check Command Flow

15.4.1.5.2 Page Erase Verify

The page erase verify operation will verify that all memory addresses in a Flash logical page are erased. Figure 15-26 below shows the logical page addresses for the data Flash and various program Flash arrays, overlaid with the logical sectors protected via the CFMDFPROT and CFMPROT registers.

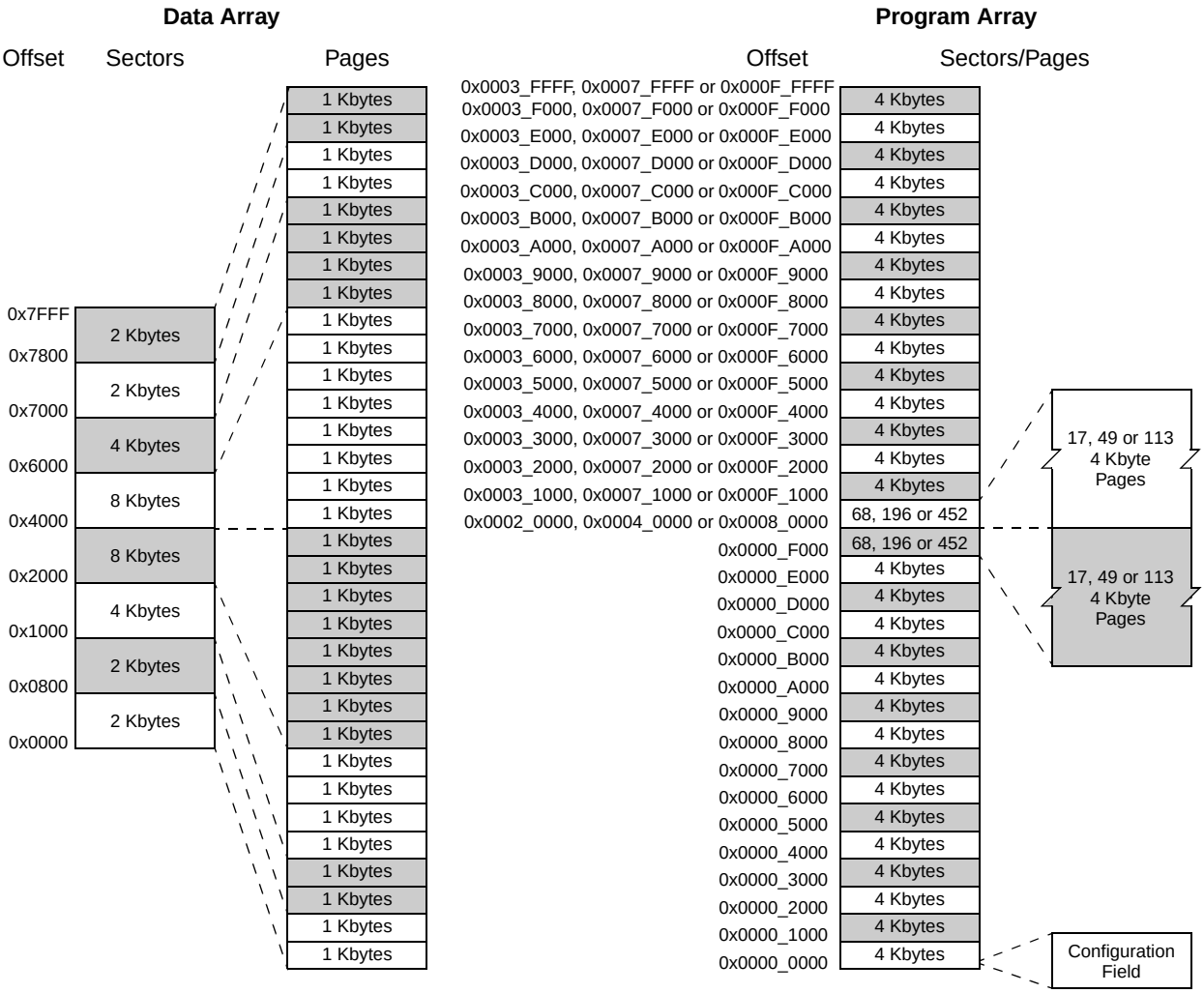


Figure 15-26. CFM Flash Logical Page and Security Sector Mapping

An example flow to execute the page erase verify operation is shown in Figure 15-27. The page erase verify command write sequence for the program Flash memory is as follows:

1. Write to any word address in a program Flash logical page to start the command write sequence for the page erase verify command. The address written will determine the program Flash logical page to be verified while the data written during the page erase verify command write sequence will be ignored. For 1 Mbyte devices, if the same relative page in program Flash logical block 0 and block 1 needs to be page erase verified, the first write must be to program Flash logical block 0 odd and the second write must be to the same relative address in program Flash logical block 1 even (see

Figure 15-3). For example, to simultaneously page erase verify logical pages 0 and 128, write to offsets 0x0000_0004 and 0x0008_0000; to page erase verify logical pages 1 and 129, write to offsets 0x0000_1004 and 0x0008_1000.

2. Write the page erase verify command, 0x06, to the CFMCMD register.
3. Write 0b1 to CBEIF to clear it, and launch the page erase verify command.

Since the word addresses in even and odd physical program Flash blocks are interleaved, pages from adjacent interleaved program Flash physical blocks will automatically be erase verified at the same time. The number of f_{IPS} clock cycles required to execute the page erase verify operation on a fully erased program Flash logical page is equal to the number of word addresses in a program Flash logical page plus fifteen f_{IPS} clock cycles as measured from the time the CFMUSTAT[CBEIF] flag is cleared until the CFMUSTAT[CCIF] flag is set.

The page erase verify command write sequence for the data Flash memory is as follows:

1. Write to any half word address in a data Flash logical page to start the command write sequence for the page erase verify command. The address written will determine the data Flash logical page to be verified while the data written during the page erase verify command write sequence will be ignored.
2. Write the page erase verify command, 0x06, to the CFMCMD register.
3. Write 0b1 to CBEIF to clear it, and to launch the page erase verify command.

The number of f_{IPS} clock cycles required to execute the page erase verify operation on a fully erased data Flash logical page is equal to the number of half word addresses in a data Flash logical page plus fifteen f_{IPS} clock cycles as measured from the time the CBEIF flag is cleared until the CCIF flag is set in the CFMUSTAT register.

Upon completion of any page erase verify operation (CCIF = 1), the BLANK flag in the CFMUSTAT register will be set if all addresses in the selected Flash logical page are verified to be erased. If any address in the selected Flash logical page is not erased, the page erase verify operation will terminate and the BLANK flag will remain clear.

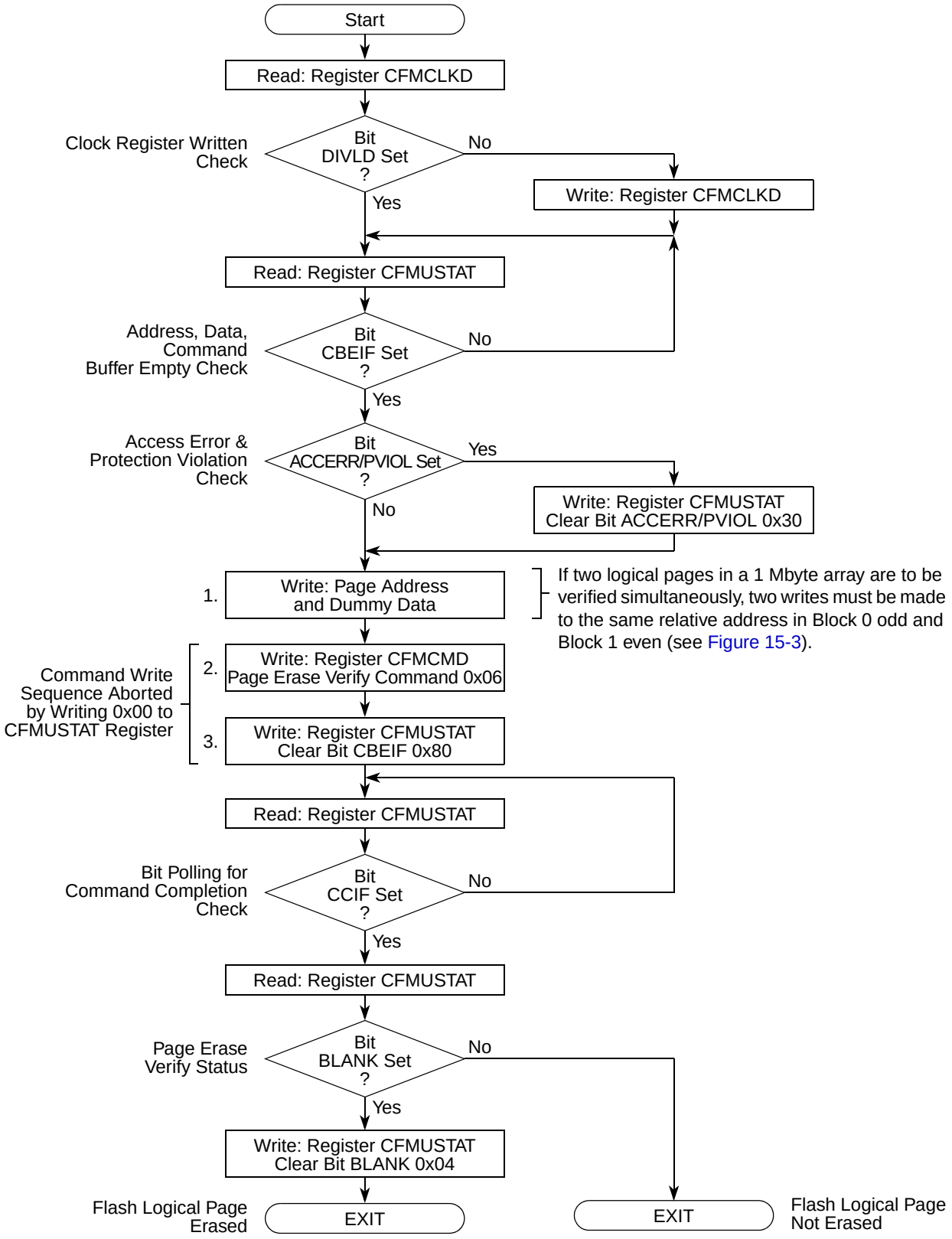


Figure 15-27. CFM Example Page Erase Verify Command Flow

15.4.1.5.3 Program

The program operation will program a previously erased address in the Flash memory using an embedded algorithm.

An example flow to execute the program operation is shown in Figure 15-28. The program command write sequence for the program Flash memory is as follows:

1. Write to a word address in a program Flash physical block to start the command write sequence for the program command. The word address written will determine the program Flash physical block address to program while the data written during the program command write sequence will determine the data stored at that address. The same relative address in multiple program Flash physical blocks may be programmed simultaneously by writing to the relative address in Flash physical block order: Block 0 even, Block 0 odd, Block 1 even, Block 1 odd (for example, offset 0x0000_0000, 0x0000_0004, 0x0008_0000, 0x0008_0004)). The Flash physical block written to in the first array write limits the ability to simultaneously program in block order only those Flash physical blocks that remain.
2. Write the program command, 0x20, to the CFMCMD register.
3. Write 0b1 to CBEIF to clear it, and to launch the program command.

The program command write sequence for the data Flash memory is as follows:

1. Write to any half word address in a data Flash logical block to start the command write sequence for the program command. The half word address written will determine the data Flash logical block address to be programmed while the data written during the program command write sequence will determine the data stored at that address.
2. Write the program command, 0x20, to the CFMCMD register.
3. Write 0b1 to CBEIF to clear it, and to launch the program command.

If the address to be programmed is in a protected sector of the Flash memory, the PVIOL flag in the CFMUSTAT register will set and the program command will not launch. Once the program command has successfully launched, the CCIF flag in the CFMUSTAT register will set after the program operation has completed unless a new command write sequence has been buffered.

NOTE

Attempting to program a Flash location that is not blank prior to issuing the program command can over-stress the Flash memory, which could potentially destroy the memory. It is the responsibility of the user to follow the recommended command sequence and verify that a Flash location is blank before issuing a program command.

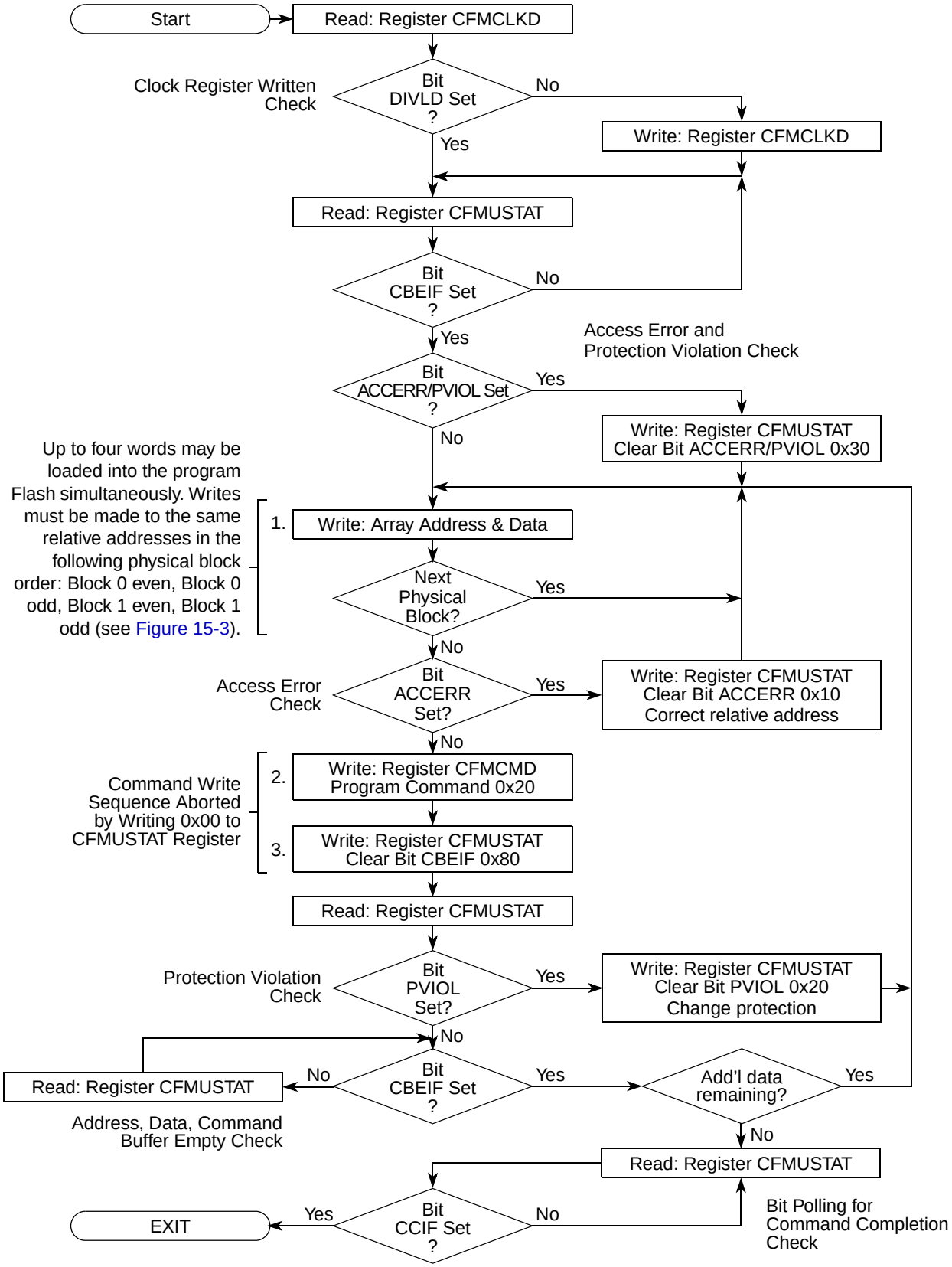


Figure 15-28. CFM Example Program Command Flow

15.4.1.5.4 Page Erase

The page erase operation will erase all memory addresses in a Flash logical page using an embedded algorithm. After a page erase of page 0, the Flash will be secured unless the user writes to the security configuration word in Flash. The Flash logical page to be erased must be in an unprotected sector of the Flash memory. Therefore it may be necessary to unprotect the appropriate sector, as described in Section 15.3.1.4, “CFM Program Flash Protection Register (CFMPROT),” and Section 15.3.1.5, “CFM Data Flash Protection Register (CFMDFPROT),” prior to performing the page erase operation. Figure 15-26 above shows the correspondence between logical pages and protection sector partitioning.

An example flow to execute the page erase operation is shown in Figure 15-29 and Figure 15-30. The page erase command write sequence for the program Flash memory is as follows:

1. Write to any word address in a program Flash logical page to start the command write sequence for the page erase command. The word address written will determine the program Flash logical page to erase while the data written during the page erase command write sequence will be ignored. If the same relative page in program Flash logical block 0 and program Flash logical block 1 of a 1 Mbyte device needs to be erased, the first write must be to program Flash logical block 0 and the second write must be to the same relative address in program Flash logical block 1 (see Figure 15-3). For example, to simultaneously erase logical pages 0 and 128, write to offsets 0x0000_0004 and 0x0008_0000; to erase logical pages 1 and 129, write to offsets 0x0004_1004 and 0x0008_1000.
2. Write the page erase command, 0x40, to the CFMCMD register.
3. Write 0b1 to CBEIF to clear it, and to launch the page erase command.

The page erase command write sequence for the data Flash memory is as follows:

1. Write to any half word address in a data Flash logical page to start the command write sequence for the page erase command. The half word address written will determine the data Flash logical page to be erased while the data written during the page erase command write sequence will be ignored.
2. Write the page erase command, 0x40, to the CFM_CMDR register.
3. Write 0b1 to CBEIF to clear it, and to launch the page erase command.

If the Flash logical page to be erased is in a protected sector of the Flash memory, the PVIOL flag in the CFMUSTAT register will set and the page erase command will not launch. Once the page erase command has successfully launched, the CCIF flag in the CFMUSTAT register will set after the page erase operation has completed unless a new command write sequence has been buffered.

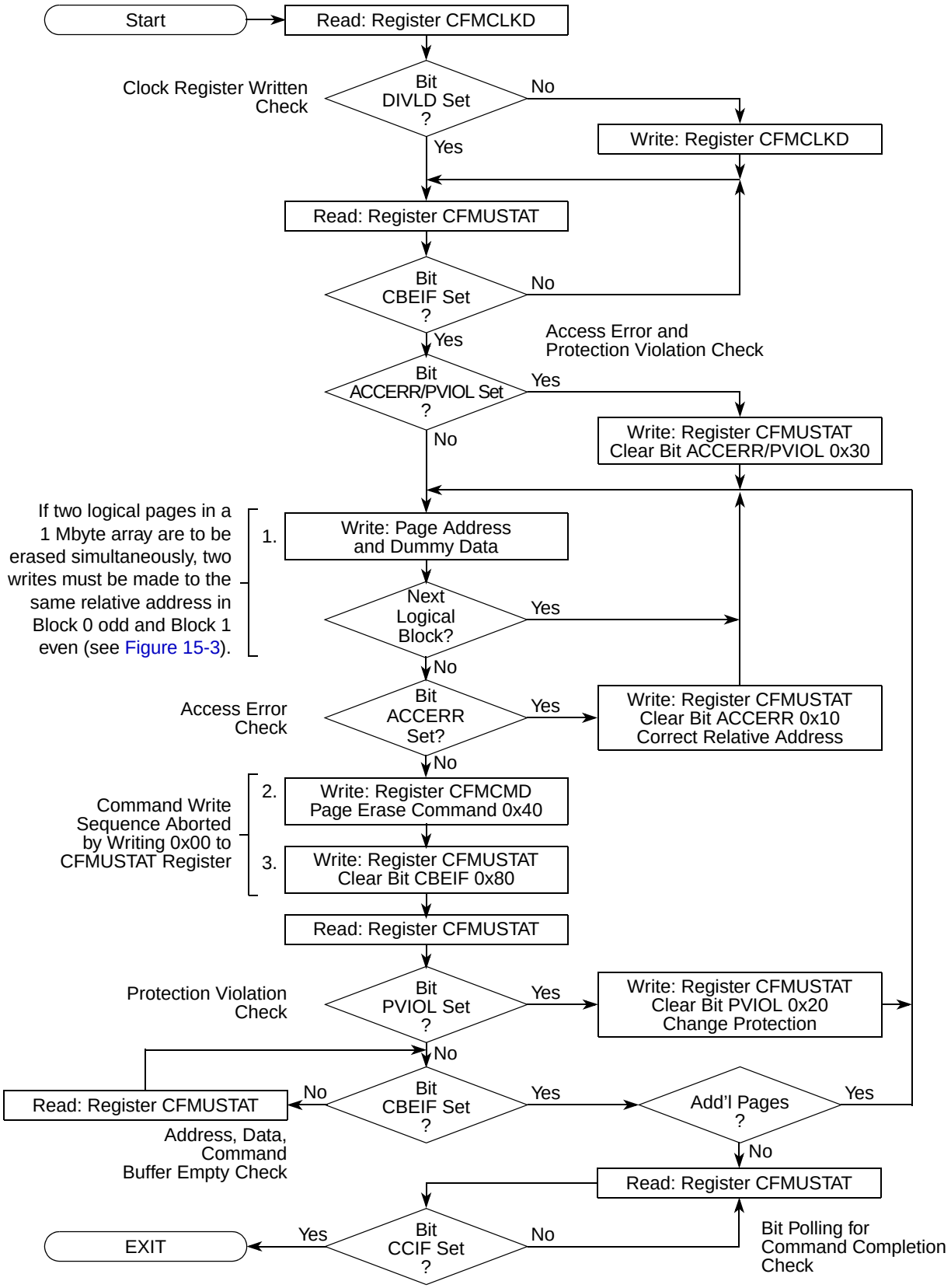


Figure 15-29. CFM Example Page Erase Command Flow for Program Flash

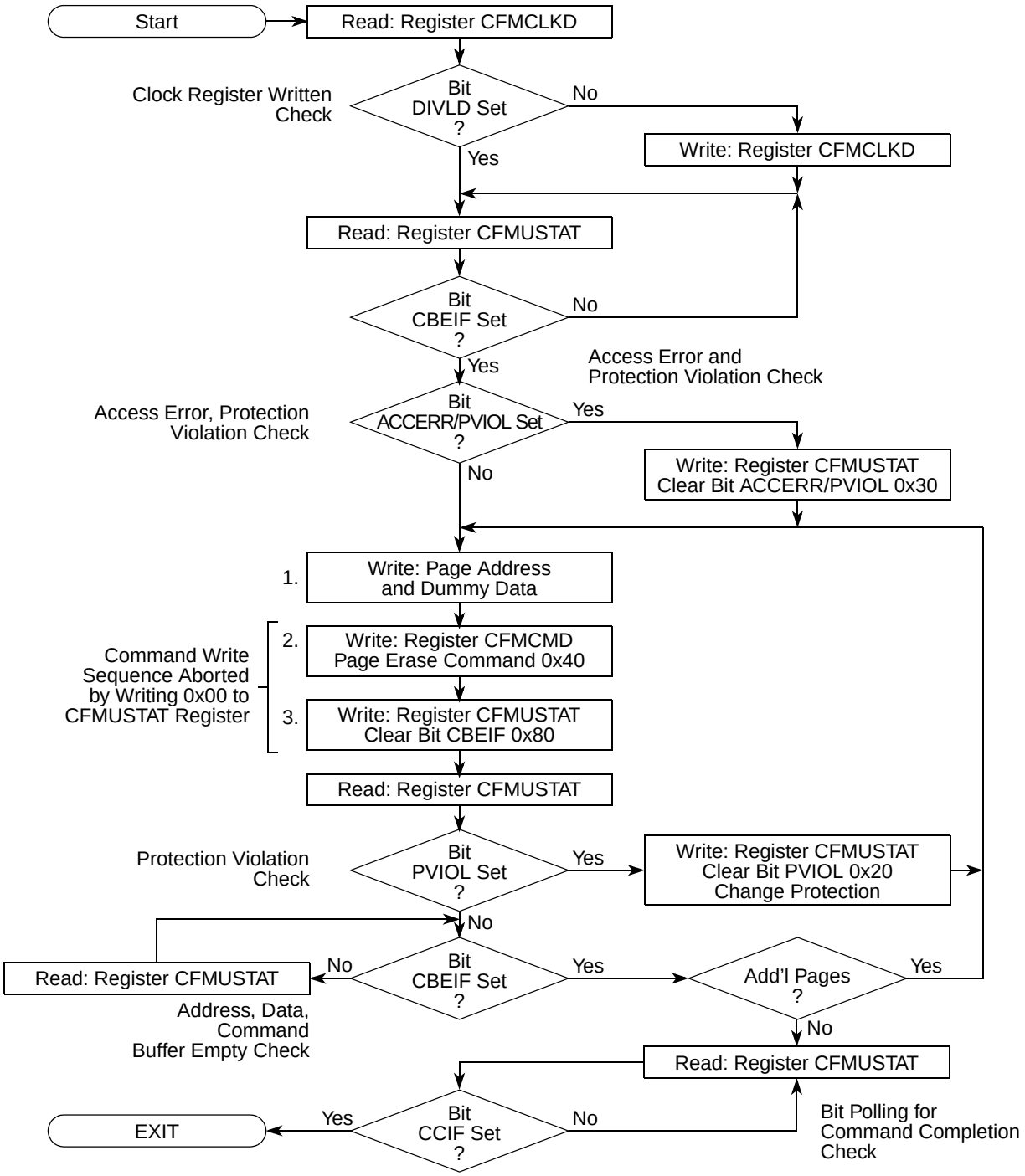


Figure 15-30. CFM Example Page Erase Command Flow for Data Flash

15.4.1.5.5 Mass Erase

The mass erase operation will erase all Flash memory addresses using an embedded algorithm. All of the Flash (data and program) must be unsecured before this operation. After a mass erase, the Flash will be secured unless user writes to the security configuration word in Flash.

An example flow to execute the mass erase command is shown in Figure 15-31. The mass erase command write sequence is as follows:

1. Write to any Flash memory address to start the command write sequence for the mass erase command. The specific address and data written during the mass erase command write sequence will be ignored.
2. Write the mass erase command, 0x41, to the CFMCMD register.
3. Write 0b1 to CBEIF to clear it, and to launch the mass erase command.

If any Flash logical sector is protected, the PVIOL flag in the CFMUSTAT register will set during the command write sequence and the mass erase command will not launch. Once the mass erase command has successfully launched, the CCIF flag in the CFMUSTAT register will set after the mass erase operation has completed unless a new command write sequence has been buffered.

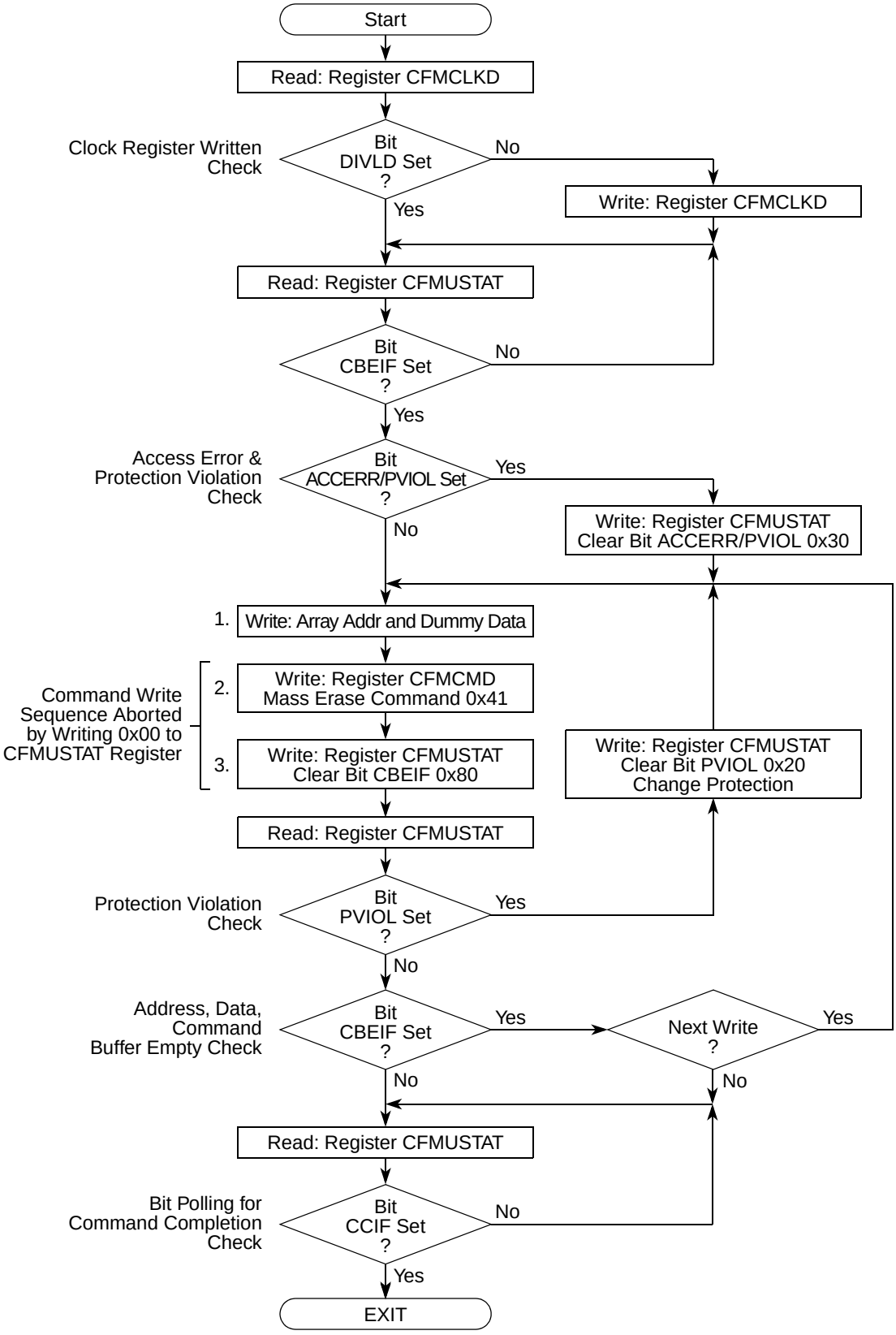


Figure 15-31. CFM Example Mass Erase Command Flow

15.4.1.5.6 Data Signature

NOTE

Mask set L49P and L47W devices do not implement this feature.

The data signature operation compresses data from all selected Flash memory addresses into a 32-bit response that is returned to the CFMDATA0 and CFMDATA1 registers. The data signature can be compared to the expected response to determine the integrity of the data stored in the selected Flash addresses. An example flow to execute the data signature operation on the program Flash is shown in Figure 15-32. The data signature command write sequence for program Flash is:

1. Write to any word address in a program Flash logical block to start the command write sequence for the data signature command. The address written determines the starting program Flash word address for the data signature operation, while the data written during the data signature command write sequence determines the number of consecutive word addresses to compress within the program Flash block. [1] For 1 Mbyte program Flash arrays, multiple writes may be required to initialize the operation, depending on the starting address and size of the data to be compressed. Table 15-19 lists the various cases for starting address and block size and required initialization write operations. If the proper sequence is not followed, only data in Block 0 is compressed.
2. Write the data signature command, 0x65, to the CFMCMD register.
3. Write 0b1 to CBEIF to clear it, and to launch the data signature command.

Table 15-19. CFM Multiple-Block Data Signature Operations

Signature Block Characteristics	Initialization Write Cycles Required						Result
	First		Second		Third		
	Offset	Data	Offset	Data	Offset	Data	
Resides only in Block 0	0000_*nnnn* to 0007_*nnnn*	Word Count [1]	—	—	—	—	CFMDATA0
Resides only in Block 1	0008_*nnnn* to 000F_*nnnn*	Word Count [1]	—	—	—	—	CFMDATA1
Resides in Block 0 and 1, even-word starting address	0000_*nnn*0, 0000_*nnn*8, to 0007_*nnn*0, 0007_*nnn*8	Word Count [1] (used for each block)	0000_*nnn*4, 0000_*nnn*C, to 0007_*nnn*4, 0007_*nnn*C	xxxx_xxxx (ignored)	0008_*nnn*0, 0008_*nnn*8, to 000F_*nnn*0, 000F_*nnn*8	xxxx_xxxx (ignored)	CFMDATA0 and CFMDATA1
Resides in Block 0 and 1, odd-word starting address	0000_*nnn*4, 0000_*nnn*C, to 0007_*nnn*4, 0007_*nnn*C	Word Count [1] (used for each block)	0008_*nnn*0, 0008_*nnn*8, to 000F_*nnn*0, 000F_*nnn*8	xxxx_xxxx (ignored)	0008_*nnn*4, 0008_*nnn*C, to 000F_*nnn*4, 000F_*nnn*C	xxxx_xxxx (ignored)	CFMDATA0 and CFMDATA1

Note that the parallel design of 1 Mbyte Flash arrays allows the signature for the entire array to be calculated in the same number of cycles as is required for 1/2 of the array. However, in order to take

1. If the sum of the address/offset and data written during the command write sequence results in an address/offset that is beyond the end of the program Flash logical block, no address wrap is performed. The data signature operation terminates when the end of the program Flash block is reached.

advantage of this design, the same number of words at the same relative offset within each physical block must be compressed simultaneously. If an application data set does not fit such a symmetrical schema, multiple data signature commands must be used to generate the desired results.

The data signature operation for the data Flash follows the same flow as shown in Figure 15-32 for the program Flash, except it should be noted that CFMDATA0[31:16] are always zero following the command, and may be ignored. The data signature command write sequence for the data Flash is:

1. Write to any half-word address in a data Flash logical block to start the command write sequence for the data signature command. The half-word address written will determine the starting data Flash logical block address for the data signature operation while the data written during the data signature command write sequence will determine the number of consecutive half-word addresses to compress within the data Flash block. [1]
2. Write the data signature command, 0x65, to the CFMCMD register.
3. Write 0b1 to CBEIF to clear it, and to launch the data signature command.

The number of f_{IPS} clock cycles required to execute the data signature operation is equal to the number of addresses to compress plus fifteen f_{IPS} clock cycles, measured from the time the CBEIF flag is cleared until the CCIF flag is set in the CFMUSTAT register. While the data signature command is executing, the CBEIF flag remains clear to indicate that a new command write sequence should not be buffered behind a data signature command write sequence. Upon completion of the data signature operation, the CBEIF and CCIF flags will set, and the data signature response is stored in the CFMDATA0 / CFMDATA1 registers. The data signature response will remain in the CFMDATA0 / CFMDATA1 registers until a new command write sequence is started.

1. If the sum of the address and data written during the command write sequence results in an address that is beyond the end of the data Flash logical block, no address wrap is performed. The data signature operation terminates when the end of the data Flash block is reached.

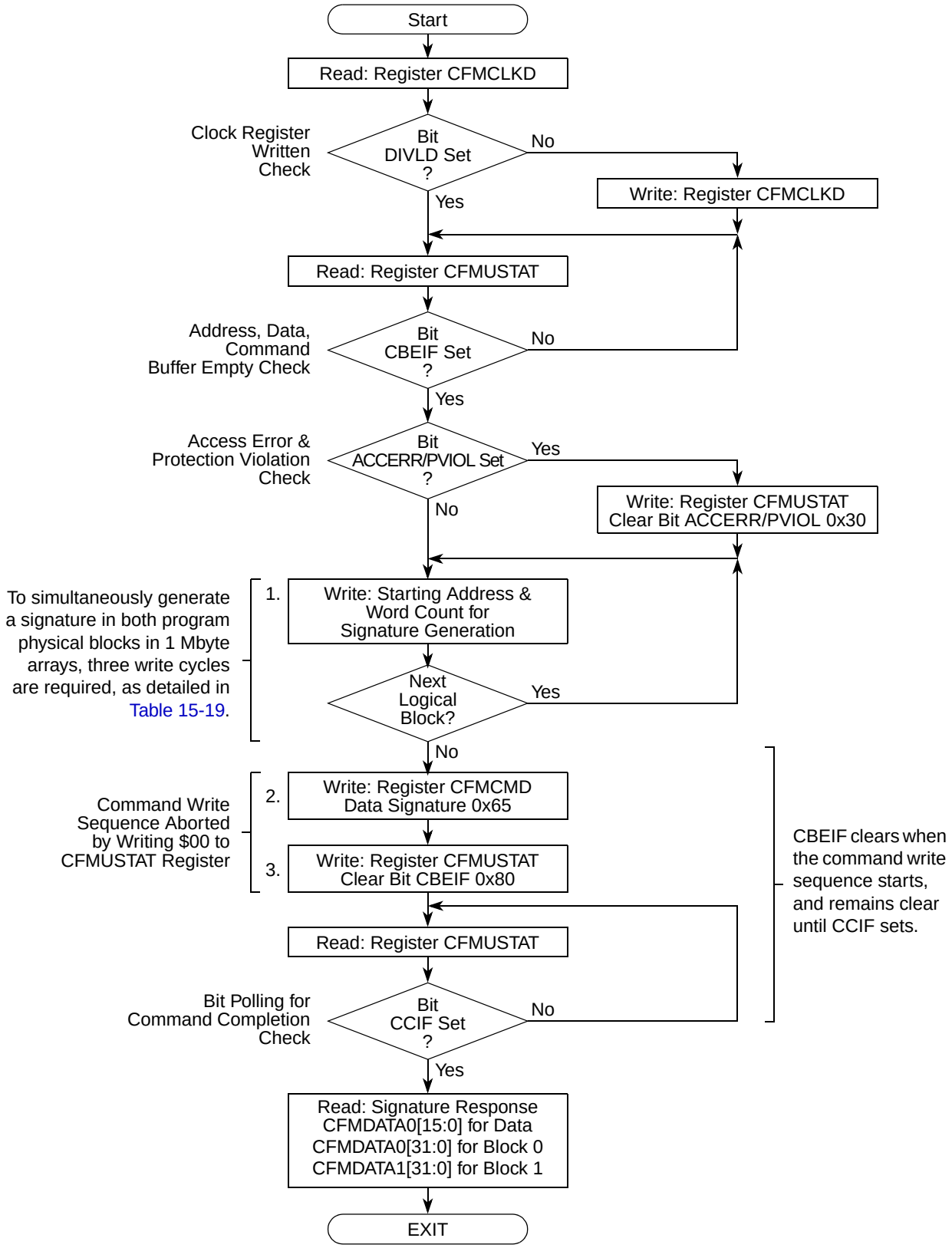

Start
Read: Register CFMCLKD
Clock Register Written Check
Bit DIVLD Set ?
No
Write: Register CFMCLKD
Yes
Read: Register CFMUSTAT
Address, Data, Command Buffer Empty Check
Bit CBEIF Set ?
No
Yes
Access Error & Protection Violation Check
Bit ACCERR/PVIOL Set ?
Yes
Write: Register CFMUSTAT Clear Bit ACCERR/PVIOL 0x30
No
To simultaneously generate a signature in both program physical blocks in 1 Mbyte arrays, three write cycles are required, as detailed in Table 15-19.
1.
Write: Starting Address & Word Count for Signature Generation
Next Logical Block?
Yes
No
Command Write Sequence Aborted by Writing $00 to CFMUSTAT Register
2.
Write: Register CFMCMD Data Signature 0x65
3.
Write: Register CFMUSTAT Clear Bit CBEIF 0x80
CBEIF clears when the command write sequence starts, and remains clear until CCIF sets.
Read: Register CFMUSTAT
Bit Polling for Command Completion Check
Bit CCIF Set ?
No
Yes
Read: Signature Response
CFMDATA0[15:0] for Data
CFMDATA0[31:0] for Block 0
CFMDATA1[31:0] for Block 1
EXIT

Figure 15-32. CFM Example Data Signature Command Flow

15.4.1.6 Flash Normal Mode Illegal Operations

The CFMUSTAT[ACCERR] flag will be set during the command write sequence if any of the following illegal operations are performed, causing the command write sequence to immediately abort:

- Writing to the Flash memory before initializing CFMCLKD.
- Writing to the Flash memory while CBEIF is not set.
- Writing to a program Flash block with a data size other than 32 bits.
- Writing to a data Flash block with a data size other than 16 bits.
- After writing to program Flash block 0 even, writing an additional word to the Flash memory during the program Flash command write sequence other than program Flash block 0 odd, Flash block 1 odd.
- Writing a second half word to a data Flash block after writing to a data Flash block.
- Writing to both a program Flash and a data Flash block during the same command write sequence if the command is not mass erase or blank check.
- Writing an invalid Flash normal mode command to the CFMCMD register.
- Writing to any CFM register other than CFMCMD after writing to the Flash memory.
- Writing a second command to the CFMCMD register before executing the previously written command.
- Writing to any CFM register other than CFMUSTAT (to clear CBEIF) after writing to the command register, CFMCMD.
- The part enters STOP mode and any command is in progress. Upon entering STOP mode, any active command is aborted.
- Aborting a command write sequence by writing a 0b0 to the CBEIF flag after writing to the Flash memory or after writing a command to the CFMCMD register but before the command is launched.

The CFMUSTAT[PVIOL] flag will be set during the command write sequence if any of the following illegal operations are performed causing the command write sequence to immediately abort:

- A program command if the address to program is in a protected Flash logical sector.
- A page erase command if the address to erase is in a protected Flash logical sector.
- A mass erase command while any protection is enabled in either the Program or Data Flash blocks.

If a read operation is attempted on a Flash logical block while a command is active on that logical block (CCIF = 0), the read operation will return invalid data and the CFMUSTAT[ACCERR] flag will not be set.

15.4.1.7 Stop Mode

If a command is active (CCIF = 0) when the MCU enters STOP mode, the Flash command controller and Flash memory controller will perform the following:

1. The active command will be aborted, and the data being programmed or erased is lost.
2. The high voltage circuitry to the Flash arrays will be switched off.
3. Any buffered command (CBEIF = 0) will not be executed once the MCU exits STOP mode.

4. The CCIF and ACCERR flags will be set if a command is active when the MCU enters STOP mode.

NOTE

As active commands are immediately aborted when the MCU enters STOP mode, it is strongly recommended that a STOP instruction is not executed during program and erase operations.

If a command is not active (CCIF = 1) when the MCU enters STOP mode, the ACCERR flag will not set.

15.4.2 Flash Security Operation

The CFM provides security information to the SPP that supports protection of the Flash contents, if required. The security information is stored within a word in the Flash configuration field (refer to Table 15-2). This security word is read automatically after each reset and stored in the CFMSEC register. After a mass erase or page erase of page 0, the Flash will be secured until the CFMSEC register is written to disable security. If the application does not use a program command to set the security configuration word to disable security prior to the next reset, the Flash will be secured and must be unsecured using the procedures described below.

In Flash normal mode, the user can bypass the security via a backdoor access sequence using an 8-byte long key. Upon successful completion of the backdoor access sequence, the SECSTAT bit in the CFMSEC register is cleared indicating that the MCU is unsecured.

The CFM may be unsecured via one of the following methods:

1. Executing a backdoor access sequence.
2. Passing a blank check operation on the Flash memory.
3. Executing the JTAG lockout recovery sequence.

15.4.2.1 Backdoor Access Sequence

If the KEYEN bits in the CFMSEC register are set, the user can bypass security by:

1. Setting the KEYACC bit in the CFMMCR.
2. Writing the correct 8-byte Backdoor Comparison Key to the Flash memory at offset 0x0400 - 0x0407. This operation must be composed of two 32-bit writes to address 0x0400 and 0x0404 in that order. The two backdoor write cycles can be separated by any number of IP bus cycles.

NOTE

Any attempt to use a key of all zeros or all ones will lock the backdoor access sequence until the CFM is reset.

3. Clearing the KEYACC bit.
4. If all 8 bytes written match the Flash memory content at offset 0x0400 - 0x0407, then security is bypassed until the next reset.

In the unsecured state the user has full control of the contents of the 8-byte Backdoor Comparison Key by programming the bytes at offset 0x0400 - 0x0407 of the Flash configuration field. If at any time a key of

all zeros or all ones is received, the backdoor access sequence is terminated and cannot be successfully restarted until after the CFM is reset.

Note that the security of the CFM as defined in the Flash security word at address offset 0x0414 is not changed by the executing the backdoor access sequence to unsecure the device. After the next reset sequence, the CFM is secured again and the same backdoor key is in effect unless the Flash configuration field was changed by program or erase prior to reset. The backdoor access sequence to unsecure the device has no effect on the program and erase protections defined in the CFM protection registers.

The contents of the Flash security word at address offset 0x0414 must be changed by programming that address when the device is unsecured and the sector containing the Flash configuration field is unprotected.

15.4.2.2 Blank Check

A secured CFM can be unsecured by verifying that the entire Flash memory is erased. If required, the mass erase command can be executed on the Flash memory. The blank check command must then be executed on the Flash memory. The CFM will be unsecured if the blank check operation determines that the entire Flash memory is erased. After the next reset sequence, the security state of the CFM is determined by the Flash security word at address offset 0x0414. For further details on security, see the MCU security specification.

15.4.2.3 JTAG Lockout Recovery

If a program-controlled mechanism is not available to use the backdoor access sequence to unsecure the Flash, a development system can utilize the JTAG interface to perform a lockout recovery. In a manner similar to ARM CPU instruction execution flows described in Section 15.4.1.5, “Program, Erase, and Verify Operations,” a sequence of JTAG commands can be used to:

1. Execute a mass erase command,
2. execute a blank check,
3. program the Flash Security Word (offset 0x0414) bits corresponding to the CFMSEC[SEC] field to 0b10 (see Section 15.3.1.3), and
4. $\overline{\text{RESET}}$ the device into expanded mode to execute boot code from external memory.

Refer to Section B.4.4.3, “Memory Mapped Register Access via JTAG,” on page B-612 for more information on using the JTAG interface to read and write memory-mapped CFM module registers.

15.5 Initialization / Application Information

15.5.1 Using The Data Signature Command

NOTE

Mask set L49P and L47W devices do not implement this feature.

The Data Signature command (see Section 15.4.1.5.6, “Data Signature”) can be used in conjunction with the margin-sensitive read feature (see the MRDS description in Section 15.3.1.1, “CFM Module

Configuration Register (CFMMCR)") in order to validate the contents of the program and/or data Flash arrays. A validation sequence can be used in both the end-equipment manufacturing process (to verify initial programming) and in-field operation (to verify data retention), as required by the application.

Two features of the CFM provide robust Flash contents validation via:

- Dedicated hardware for fast CRC polynomial calculations
- Selectable read sense-amp level shifting to detect "weak" bit levels in either the programmed or erased state

Chapter 16
AMBA to IP Bus Bridge Module (AIPS)

16.1 Overview

The AMBA to IP Bus Bridge (AIPS) module provides an interface between the 32-bit high speed Standard Product Platform (SPP) bus and the lower bandwidth 32-bit Intelligent Peripheral Subsystem (IPS) bus. The MAC7100 Family implements two SPP masters, up to 6 SPP slaves and up to 23 IPS peripheral slaves.

16.2 Features

The following list summarizes key features of the AIPS as implemented on MAC7100 Family devices:

- Supports access to configuration registers of peripheral modules via the IPS bus
- Supports access to each IPS peripheral via a 16 Kbyte address space per module
- Supports access to two larger IPS peripheral spaces, one for program Flash and one for data Flash
- Occupies 64 Mbytes of total address space.
- Provides configurable per-slave and per-master access protections.
- Peripheral read transactions require a minimum of 2 master bus clocks, and write transactions require a minimum of 3 master bus clocks.

16.3 Modes of Operation

The AIPS provides the interface between the SPP system bus and the IPS bus as shown in Figure 16-1. An SPP master reads and writes IPS peripheral registers through the AIPS. The AIPS generates module enables, the module address, transfer attributes, byte enables and write data as inputs to the IPS peripherals. Internal IPS peripheral registers are selected based on the address driven on the IPS bus. The AIPS captures read data from the IPS interface and drives it on the SPP bus.

As shown in Figure 16-2 below, the AIPS module occupies a 64 MByte portion of the MCU address space. A 512 KByte portion of this space is allocated to SPP peripherals. The remaining 63.5 MBytes are available for IPS peripherals. The register maps of the IPS peripherals are located on 16 Kbyte boundaries. Each IPS peripheral is allocated one 16 Kbyte block of the memory map, and is activated by one of the module enables from the AIPS. Two global IPS module enables are available for the remaining 63 Mbytes of address space, and are used by MAC7100 Family devices to access the CFM programming interface and Data Flash. The address assignments for IPS peripherals is presented in Chapter 8, “Device Memory Map.”

The AIPS indicates to IPS peripherals if an access is in supervisor or user mode. The AIPS may block user mode accesses to certain IPS peripherals or it may allow the individual IPS peripherals to determine if user mode accesses are allowed. In addition, peripherals may be designated as write-protected. The AIPS supports the notion of “trusted” masters for security purposes. Masters may be individually designated as trusted for reads, writes, or both reads and writes, as well as defining that all accesses from a master are in user-mode privilege level. Refer to Section 16.4.1, “Register Descriptions,” below for more information.

All peripheral devices require aligned accesses equal to or smaller in size than the peripheral size. An exception to this rule is supported for 32-bit peripherals, which allows memory arrays to be accessed via the IPS bus.

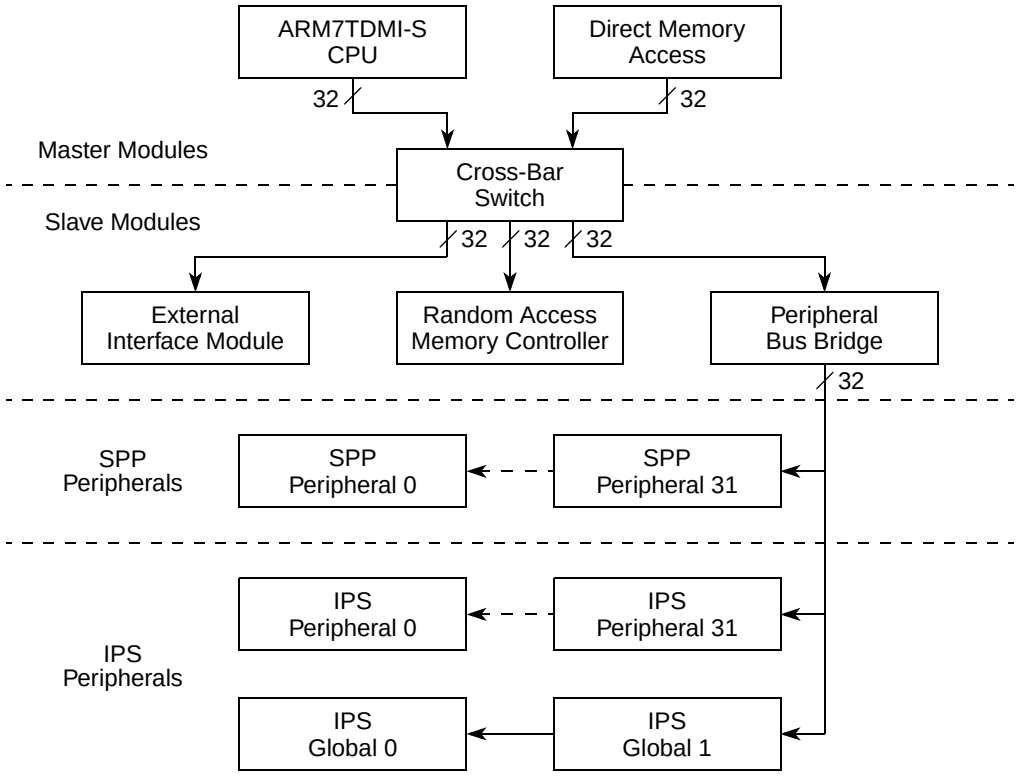


Figure 16-1. AIPS Interface Block Diagram

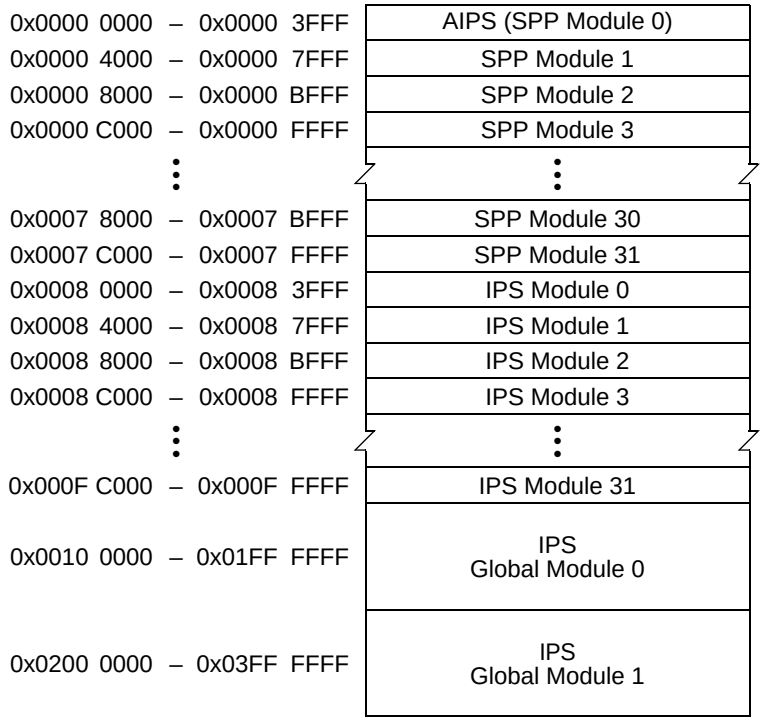


Figure 16-2. AIPS Peripheral Module Address Assignment

16.4 Memory Map / Register Definition

The memory map for AIPS registers is shown in Table 16-1. Note that the AIPS memory map is designed for future expandability, and thus the registers and fields implemented on MAC7100 family devices is not contiguous. The master protection register (MPRA) defines access privileges associated with bus masters, while access levels supported by each peripheral are defined by platform access control registers (PACRA and PACRC) and off-platform access control registers (OPACRA through OPACRE).

AIPS registers must be read and written using only 32-bit aligned accesses. The AIPS module registers are mapped into the address space controlled by PAC0: 0xFC00 0000 to 0xFC00 3FFF. Read accesses to the AIPS registers require two system clocks (f_{SYS}), while write accesses required three system clocks.

On MAC7100 Family devices, the processor complex is master zero and the eDMA module is master one. The mapping of access control fields to peripheral modules is shown in Table 16-2. Note that not all peripherals can be accessed in all operational modes or on all members of the MAC7100 family.

Table 16-1. AIPS Memory Map

AIPS Offset	Register Name	Field Name							
		[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	MPRA	MPROT0	MPROT1	Reserved					
0x0004		Reserved							
0x0008		Reserved							
0x000C		Reserved							
0x0020	PACRA	PAC0	PAC1	PAC2	Reserved				
0x0024		Reserved							
0x0028	PACRC	PAC16	PAC17	PAC18	Reserved				
0x002C		Reserved							
0x0040	OPACRA	OPAC0	OPAC1	OPAC2	OPAC3	OPAC4	OPAC5	OPAC6	OPAC7
0x0044	OPACRB	OPAC8	Reserved		OPAC11	Reserved	OPAC13	OPAC14	Reserved
0x0048	OPACRC	Reserved	OPAC17	OPAC18	OPAC19	OPAC20	Reserved		OPAC23
0x004C	OPACRD	OPAC24	OPAC25	OPAC26	Reserved	OPAC28	Reserved		
0x0050	OPACRE	OPAC32	OPAC33	Reserved					
0x0054 to 0x3FFF		Reserved							

Table 16-2. AIPS Access Control Fields Module Assignments

Access Control Field	Module	Address Range
MPROT0	**ARM7TDMI**-S – ARM7 CPU	—
MPROT1	**eDMA** – Enhanced Direct Memory Access Controller	—
PAC0	**AIPS** – AMBA to IP Bus Bridge	0xFC00 0000 – 0xFC00 3FFF
PAC1	**XBS** – Crossbar Switch Module	0xFC00 4000 – 0xFC00 7FFF
PAC2	**EIM** – External Interface Module	0xFC00 8000 – 0xFC00 BFFF
	Reserved	0xFC00 C000 – 0xFC03 FFFF
PAC16	**MCM** – Miscellaneous Control Module	0xFC04 0000 – 0xFC04 3FFF
PAC17	**eDMA** – Enhanced Direct Memory Access Controller	0xFC04 4000 – 0xFC04 7FFF
PAC18	**INTC** – Interrupt Controller	0xFC04 8000 – 0xFC04 BFFF
	Reserved	0xFC04 C000 – 0xFC07 FFFF
OPAC0	**SSM** – System Service Module	0xFC08 0000 – 0xFC08 3FFF
OPAC1	**DMA Mux** – Direct Memory Access Controller Mux	0xFC08 4000 – 0xFC08 7FFF
OPAC2	**CRG** – Clock and Reset Generator	0xFC08 8000 – 0xFC08 BFFF
OPAC3	**PIT** – Programmable Interval Timer	0xFC08 C000 – 0xFC08 FFFF
OPAC4	**VREG** – Voltage Regulator	0xFC09 0000 – 0xFC09 3FFF
OPAC5	**FlexCAN_A** – CAN Controller A	0xFC09 4000 – 0xFC09 7FFF
OPAC6	**FlexCAN_B** – CAN Controller B	0xFC09 8000 – 0xFC09 BFFF
OPAC7	**FlexCAN_C** – CAN Controller C	0xFC09 C000 – 0xFC09 FFFF
OPAC8	**FlexCAN_D** – CAN Controller D	0xFC0A 0000 – 0xFC0A 3FFF
	Reserved	0xFC0A 4000 – 0xFC0A BFFF
OPAC11	**I^2C** – Inter-IC Bus Controller	0xFC0A C000 – 0xFC0A FFFF
	Reserved	0xFC0B 0000 – 0xFC0B 3FFF
OPAC13	**DSPI_A** – Serial Peripheral Interface A	0xFC0B 4000 – 0xFC0B 7FFF
OPAC14	**DSPI_B** – Serial Peripheral Interface B	0xFC0B 8000 – 0xFC0B BFFF
	Reserved	0xFC0B C000 – 0xFC0C 3FFF
OPAC17	**eSCI_A** – Enhanced Serial Communication Interface A	0xFC0C 4000 – 0xFC0C 7FFF
OPAC18	**eSCI_B** – Enhanced Serial Communication Interface B	0xFC0C 8000 – 0xFC0C BFFF
OPAC19	**eSCI_C** – Enhanced Serial Communication Interface C	0xFC0C C000 – 0xFC0C FFFF
OPAC20	**eSCI_D** – Enhanced Serial Communication Interface D	0xFC0D 0000 – 0xFC0D 3FFF
	Reserved	0xFC0D 4000 – 0xFC0D BFFF
OPAC23	**eMIOS** – Enhanced Modular I/O Subsystem	0xFC0D C000 – 0xFC0D FFFF
OPAC24	**ATD_A** – Analog-to-Digital Converter A	0xFC0E 0000 – 0xFC0E 3FFF
OPAC25	**ATD_B** – Analog-to-Digital Converter B	0xFC0E 4000 – 0xFC0E 7FFF
OPAC26	**PIM** – Port Integration Module	0xFC0E 8000 – 0xFC0E BFFF
	Reserved	0xFC0E C000 – 0xFC0E FFFF
OPAC28	**CFM** – Common Flash Module registers	0xFC0F 0000 – 0xFC0F 3FFF
	Reserved	0xFC0E 4000 – 0xFC0F FFFF
OPAC32	**Program Flash Array** – Programming interface	0xFC10 0000 – 0xFCFF FFFF
OPAC33	**Data Flash Array**	0xFE00 0000 – 0xFFFF FFFF

16.4.1 Register Descriptions

16.4.1.1 AIPS Master Protection Registers (MPR*x*)

As shown in Table 16-1, the MPR*x* registers (only one of which is implemented on MAC7100 devices) contain up to eight 4-bit master protection fields (MPROT*n*), one per bus master, defining the access privilege level associated with a bus master.

	3	2	1	0
R	0	MTR	MTW	MPL
W				
Reset	0	1	1	1
Field Offset	Refer to Table 16-1			

Figure 16-3. AIPS MPR*x* Master Protection Fields (MPROT*n*)

Table 16-3. AIPS MPR*x* MPROT*n* Bit Descriptions

Bits	Name	Description
3	—	Reserved.
2	MTR	Master trusted for reads. This bit determines whether the master is trusted for read accesses 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
1	MTW	Master trusted for writes. This bit determines whether the master is trusted for write accesses 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
0	MPL	Master privilege level. This bit determines how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode regardless of the master's access attribute. 1 Accesses from this master are not forced to user-mode. The master's access attribute is used directly to determine the peripheral's access attribute

Accesses to registers or register fields which correspond to master or peripheral locations which are not implemented return zeros on reads, and are ignored on writes.

16.4.1.2 AIPS Peripheral Access Control Registers (PACR*x*)

As shown in Table 16-1, the PACR*x* registers (two of which are implemented on MAC7100 devices) contain up to eight 4-bit peripheral access control fields (PAC*n*), one per SPP peripheral. Each PAC*n* defines the access levels supported by the associated module. All reserved field and bit positions are unimplemented and read as zero. Writes are ignored. Each PAC*n* field has the following format:

	3	2	1	0
R W	0	SP	WP	TP
Reset, PACR0	0	1	0	1
Reset, PACR*n*	0	1	0	0
Field Offset	Refer to Table 16-1			

Figure 16-4. AIPS PACR*x* Peripheral Access Control Fields (PAC*n*)

Table 16-4. AIPS PACR*x* PAC*n* Bit Descriptions

Bits	Name	Description
3	—	Reserved.
2	SP	Supervisor protect. This bit determines whether the peripheral requires supervisor privilege level for access. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate the supervisor access attribute, and the MPR*x*.MPROT*n*.MPL control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.
1	WP	Write protect. This bit determines whether the peripheral allows write accesses 0 This peripheral allows write accesses. 1 This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.
0	TP	Trusted protect. This bit determines whether the peripheral allows accesses from an untrusted master. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.

16.4.1.3 AIPS Off-Platform Peripheral Access Control Registers (OPACR*x*)

As shown in Table 16-1, the OPACR*x* registers (five of which are implemented on MAC7100 devices) contain up to eight 4-bit off-platform access control fields (OPAC*n*), one per off-platform peripheral. Each OPAC*n* defines the access levels supported by the associated module. Each OPAC*n* has a format identical to the PAC*n* fields described in Section 16.4.1.2, “AIPS Peripheral Access Control Registers (PACRx).”

Each OPAC*n* field corresponds to the assigned IPS peripheral defined in Table 16-2; OPAC0 corresponds to the System Service Module (SSM), etc., with OPAC32 corresponding to Program Flash programming interface, and OPAC33 corresponding to the Data Flash read interface. All reserved locations read as zero and writes are ignored.

16.5 Functional Description

The AIPS serves as bus protocol translator and interface between the SPP masters on the system bus and the IPS peripherals. Accesses which fall within the address space of the AIPS are decoded to provide individual module selects for peripheral devices on the IPS bus.

16.5.1 Access Protections

The AIPS provides programmable access protections for both masters and peripherals. It allows the privilege level of a master to be overridden, forcing it to user-mode privilege, and allows masters to be designated as trusted or untrusted. Peripherals may require supervisor privilege level for access, may restrict access to a trusted master only, and may be write-protected.

16.5.2 Access Support

Aligned word and halfword accesses, as well as byte accesses are supported for 32-bit peripherals. Peripheral registers must not be misaligned, although no explicit checking is performed by the AIPS. Note that not all peripherals support halfword and byte accesses to all of their registers.

16.5.3 Read Cycles

Two clock SPP peripheral read accesses are possible when the requested access size is 32-bits or smaller. Note that some IPS peripherals run at half-speed and all devices can take much longer if there is contention for a resource.

16.5.4 Write Cycles

Three clock SPP peripheral write accesses are possible when the requested access size is 32-bits or smaller. Note that some IPS peripherals run at half-speed and all devices can take much longer if there is contention for a resource.

16.5.5 Aborted Cycles

The AIPS follows a standard procedure when a system bus cycle is aborted and the abort is initiated by the AIPS itself or the targeted IP bus peripheral. The AIPS either blocks initiation or immediately terminates any IPS bus activity that is ongoing.

There are several conditions that can cause the AIPS to abort the current operation and report an error:

1. Where the targeted IPS bus peripheral asserts a bus error. In this case the AIPS immediately terminates access to the targeted IPS bus peripheral and follows the abort procedure described above. Whether the current IPS bus access is a multi-cycle access or a single cycle access has no bearing on the behavior of the AIPS. The AIPS responds identically in both cases.

2. When an access is attempted to an IPS bus peripheral whose corresponding PAC*n* or OPAC*n* settings do not allow the access, thus causing a permissions violation. In this case, the AIPS does not initiate any IPS bus activity, but instead responds by following the abort procedure described above.
3. When an access is attempted to a location at which there is no IPS bus peripheral. In this case the AIPS does not initiate any IPS bus activity but instead responds by following the same abort procedure described above for a permissions violation.

16.6 Initialization / Application Information

The AIPS is configured at reset with all masters trusted for both reads and writes, all masters set to act as a supervisor, and all peripherals supervisor protected. After reset, a bus master may be used to change the Master Protection Registers (MPR*x*), the Peripheral Access Control Registers (PACR*x*), and the Off-Platform Peripheral Access Control Registers (OPACR*x*) as needed.

Chapter 17
DMA Channel Multiplexer Module (DMAMux)

17.1 Overview

The Enhanced Direct Memory Access (eDMA) controller implemented on MAC7100 Family devices has 16 channels, but it is possible to generate direct memory access requests from considerably more than 16 sources. The DMA Channel Multiplexer (DMAMux) module enables the configuration of which peripheral DMA request sources are connected to which eDMA controller channel. The DMAMux enables flexibility in the assignment of these channels, with any source being able to be connect to any eDMA channel. Additionally, the DMAMux allows eDMA channels 0 to 7 to be triggered either by their respective DMA request line directly or in conjunction with the Programmable Interrupt Timer (PIT). This allows the eDMA to be used to perform periodic transfers between peripherals such as the PIM and on-chip memory or the external bus interface.

The DMAMux allows for software selection of 16 out of 42 possible DMA request sources.

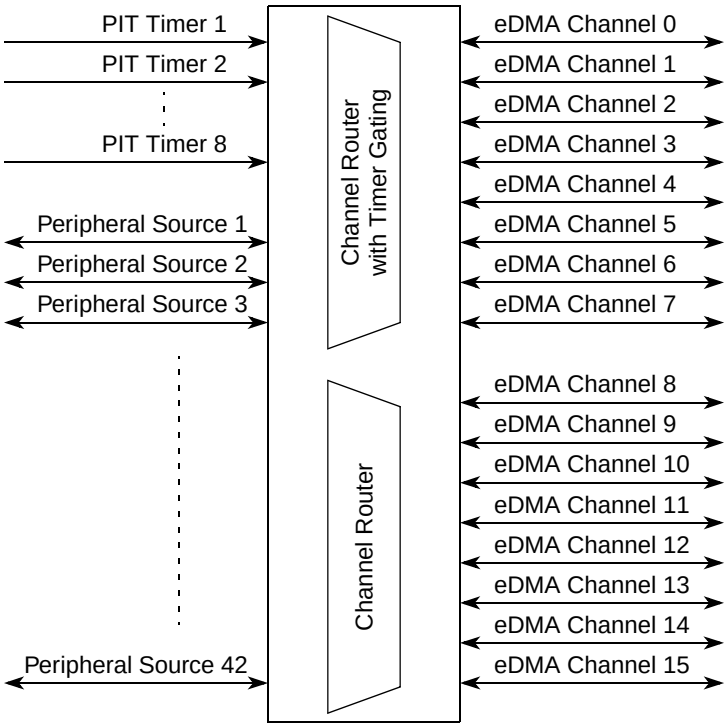


Figure 17-1. DMAMux Block Diagram

17.2 Features

The DMAMux includes these distinctive features:

- 16 independently selectable DMA request channel routers
  - — Each channel router can be assigned to 1 of 42 possible DMA request sources

— 8 channels with normal or periodic triggering capability
— 8 channels with normal operation

17.3 Modes of Operation

eDMA Channels 0 to 7 may be used in the following modes, while Channels 8 to 15 may be configured only in Disabled or Normal Mode.

- Disabled Mode

  In this mode, the eDMA channel is disabled. Since disabling and enabling of eDMA channels is done primarily via the eDMA registers, this mode is used mainly as the reset state for a DMA channel in the DMAMux. It may also be used to temporarily suspend an eDMA channel while reconfiguration of the system takes place (changing the period of a DMA trigger, for example).
- Normal Mode

  In this mode, a DMA request source (e.g., SCI transmit, SCI receive, etc.) is routed directly to the specified eDMA channel. The operation of the DMAMux in this mode is completely transparent to the system.
- Periodic Trigger Mode

  In this mode, a DMA request source only requires eDMA service periodically, such as when a transmit buffer becomes empty or a receive buffer becomes full. In this mode a timer in the PIT is used to “throttle” a peripheral module service request via the DMAMux.

17.4 Memory Map / Register Definition

Table 17-1 shows the memory map for the DMAMux. Note that all addresses are offsets; the absolute address may be computed by adding the specified offset to the base address of the DMAMux module as defined in Chapter 8, “Device Memory Map.”

All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, CHCONFIG0 through CHCONFIG4 are accessible by a 32-bit READ/WRITE to address DMAMux base + 0x0000, but performing a 32-bit access to address DMAMux base + 0x0001 is illegal.

Table 17-1. DMAMux Memory Map

DMAMux Offset	Register Description	Access
0x0000	Channel 0 Configuration (CHCONFIG0)	R/W
0x0001	Channel 1 Configuration (CHCONFIG1)	R/W
0x0002	Channel 2 Configuration (CHCONFIG2)	R/W
0x0003	Channel 3 Configuration (CHCONFIG3)	R/W
0x0004	Channel 4 Configuration (CHCONFIG4)	R/W
0x0005	Channel 5 Configuration (CHCONFIG5)	R/W
0x0006	Channel 6 Configuration (CHCONFIG6)	R/W
0x0007	Channel 7 Configuration (CHCONFIG7)	R/W
0x0008	Channel 8 Configuration (CHCONFIG8)	R/W
0x0009	Channel 9 Configuration (CHCONFIG9)	R/W

Table 17-1. DMAMux Memory Map (continued)

0x000A	Channel 10 Configuration (CHCONFIG10)	R/W
0x000B	Channel 11 Configuration (CHCONFIG11)	R/W
0x000C	Channel 12 Configuration (CHCONFIG12)	R/W
0x000D	Channel 13 Configuration (CHCONFIG13)	R/W
0x000E	Channel 14 Configuration (CHCONFIG14)	R/W
0x000F	Channel 15 Configuration (CHCONFIG15)	R/W

17.4.1 Register Descriptions

17.4.1.1 DMAMux Channel Configuration Registers (CHCONFIG*n*)

Each of the total of 16 eDMA channels can be independently enabled/disabled and associated with any of the 42 total DMA request sources in the system.

	7	6	5	4	3	2	1	0
R	ENBL	TRIG	SOURCE					
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	DMAMux Base + *n*							

Figure 17-2. DMAMux Channel Configuration Registers (CHCONFIG*n*)

Table 17-2. CHCONFIG*n* Field Descriptions

Bits	Name	Description
7	ENBL	DMA request channel enable. 0 DMA channel is disabled 1 DMA channel is enabled
6	TRIG	DMA request channel periodic trigger enable (channels 1–8 only). 0 Normal mode 1 Periodic trigger mode

Table 17-2. CHCONFIG*n* Field Descriptions (continued)

Bits	Name	Description
5–0	SOURCE[5:0]	DMA request channel source. Specifies which DMA request source, if any, is routed to a particular eDMA channel. 00 Channel route is unassigned (disabled) 01 I^2C Transmit 02 I^2C Receive 03 DSPI_A Transmit 04 DSPI_A Receive 05 DSPI_B Transmit 06 DSPI_B Receive 07 eSCI_A Transmit 08 eSCI_A Receive 09 eSCI_B Transmit 0A eSCI_B Receive 0B eSCI_C Transmit 0C eSCI_C Receive 0D eSCI_D Transmit 0E eSCI_D Receive 0F eMIOS Channel 0 10 eMIOS Channel 1 11 eMIOS Channel 2 12 eMIOS Channel 3 13 eMIOS Channel 4 14 eMIOS Channel 5 15 eMIOS Channel 6 16 eMIOS Channel 7 17 eMIOS Channel 8 18 eMIOS Channel 9 19 eMIOS Channel 10 1A eMIOS Channel 11 1B eMIOS Channel 12 1C eMIOS Channel 13 1D eMIOS Channel 14 1E eMIOS Channel 15 1F ATD_A Result 20 ATD_A Command 21 ATD_B Result 22 ATD_B Command 23 Always Enabled 0 24 Always Enabled 1 25 Always Enabled 2 26 Always Enabled 3 27 Always Enabled 4 28 Always Enabled 5 29 Always Enabled 6 2A Always Enabled 7 2B–FF DMA request channel is not used (disabled) NOTE: Setting multiple CHCONFIG*n* registers with the same SOURCE value (other than 0x00) will result in unpredictable behavior.

17.5 Functional Description

The primary purpose of the DMAMux is to provide flexibility in the use of the available eDMA channels. As such, configuration of the DMAMux is intended to be a static procedure performed only during execution of the system initialization code. However, if the procedure outlined in Section 17.6.6, “Switching DMA Request Source to eDMA Channel Assignment,” is followed, the configuration of the DMAMux may be changed during the normal operation of the system.

NOTE

Because of the dynamic nature of the system (i.e. eDMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

Functionally, the DMAMux channels may be divided into two classes: Channels 0 to 7, which implement the normal routing functionality plus the periodic triggering capability, and Channels 8 to 15, which implement only the normal routing functionality.

17.5.1 eDMA Channels 0 to 7

In addition to the normal routing functionality, Channels 0 to 7 of the DMAMux provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames or packets at fixed intervals without the need for processor intervention. The trigger is generated by the Periodic Interrupt Timer (PIT); as such, the configuration of the periodic triggering interval is done via registers in the PIT. Refer to Chapter 25, "Periodic Interrupt Timer Module (PIT)," for more information.

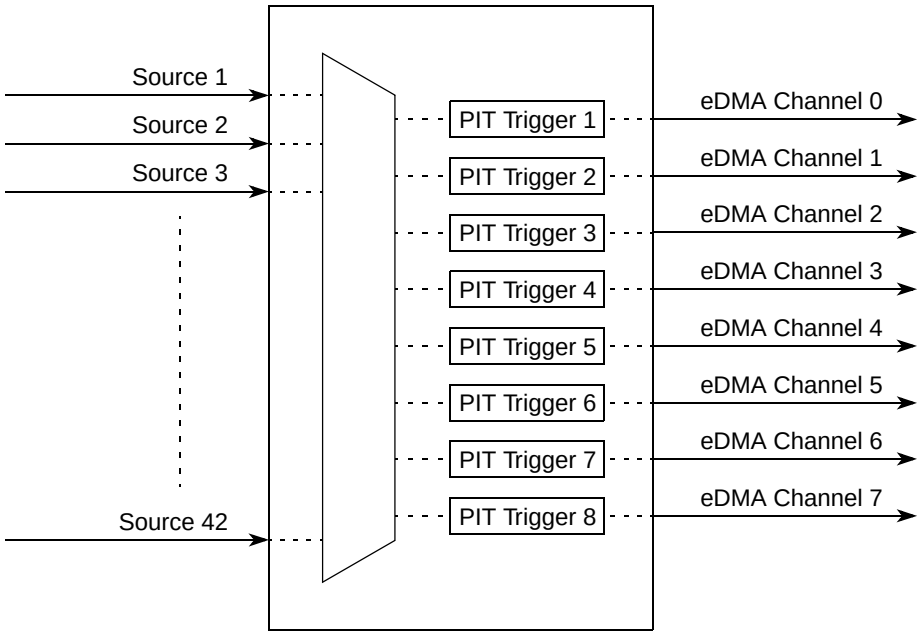


Figure 17-3. DMAMux Channel 0 to 7 Block Diagram

The DMA request channel periodic triggering capability allows the system to "schedule" regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. The periodic trigger mode works by gating the request from the peripheral to the eDMA until a periodic trigger event has occurred. This is illustrated in Figure 17-4.

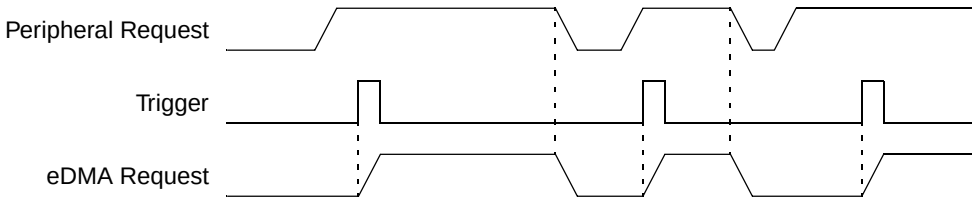


Figure 17-4. DMAMux Channel Triggering: Normal Operation

Once the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral re-asserts its request and the next trigger event occurs. If a periodic trigger occurs but the peripheral is not requesting a transfer at that time, no request is sent to the eDMA module for that period. This case is illustrated in Figure 17-5.

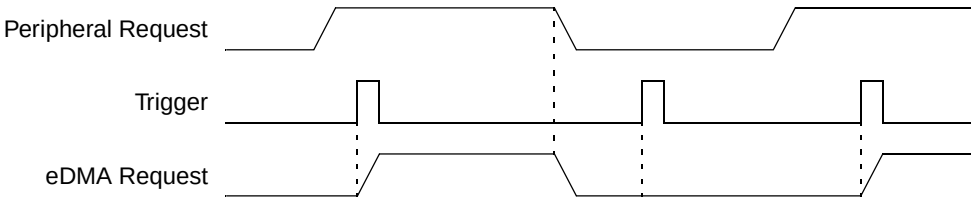


Figure 17-5. DMAMux Channel Triggering: Gated Request

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices. For example, the transmit side of an SPI may be assigned to an eDMA channel with a periodic trigger, as described above. Once initialized, the SPI will request DMA transfers (presumably from memory) as long as the transmit buffer is empty. By using a trigger on this channel, the SPI transfers could be automatically performed every *n* µs. On the receive side of the SPI, the SPI and eDMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.
- Use the GPIO ports to drive or sample waveforms. By configuring the eDMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in memory. Conversely, using the eDMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in memory.

As described in Chapter 25, "Periodic Interrupt Timer Module (PIT)," the assignment of PIT timers to DMAMux channels is fixed as shown in the table below.

Table 17-3. DMAMux Channel to PIT Timer Assignments

DMAMux Channel	PIT Timer
Channel 0	Timer 1
Channel 1	Timer 2
Channel 2	Timer 3
Channel 3	Timer 4
Channel 4	Timer 5
Channel 5	Timer 6
Channel 6	Timer 7
Channel 7	Timer 8

17.5.2 eDMA Channels 8 to 15

DMAMux Channels 8 to 15 provide normal request routing as described in Section 17.3, "Modes of Operation."

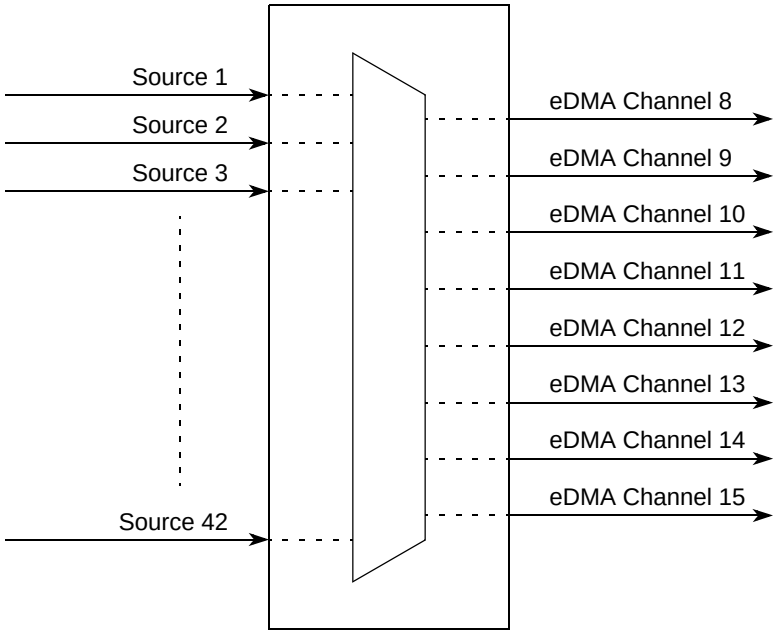


Figure 17-6. DMAMux Channel 8 to 15 Block Diagram

17.5.3 Always Enabled DMA Request Sources

In addition to the 34 peripheral sources that can be generate DMA service requests, there are 8 additional DMA request sources that are always enabled. Unlike the peripheral DMA request sources, where the peripheral controls the flow of data during DMA transfers, the always enabled sources provide no such “throttling” of the data transfers. These sources are most useful in the following cases:

- DMA transfers to/from GPIO – Moving data from/to one or more GPIO pins, either at maximum throughput or periodically (using the DMAMux/PIT periodic triggering capability).
- DMA transfers from memory to memory – Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- DMA transfers from memory to/from the external bus – Similar to memory to memory transfers, this is typically done as quickly as possible.
- DMA transfers that require software activation – Any transfer that should be explicitly started by software but is more efficiently handled by the eDMA controller than the core processor.

In cases where software should initiate the start of a DMA transfer, an always enabled DMA request source can be used to provide maximum flexibility. When activating an eDMA channel via software, subsequent executions of the minor loop require that a new start event be sent. This can either be a new software activation, or a transfer request from the DMAMux. The options for doing this are:

- Transfer all data in a single minor loop. By configuring the eDMA to transfer all of the data in a single minor loop (i.e., major loop counter = 1), no re-activation of the channel is necessary. The disadvantage is the reduced granularity in determining the load that the DMA transfer will place on the system. For this option, the DMA request channel should be disabled in the DMAMux.
- Use explicit software re-activation. In this configuration, the eDMA is initialized to transfer the data using both minor and major loops, but the processor is required to re-activate the channel (by

writing to the eDMA registers) after every minor loop. For this option, the DMA request channel should be disabled in the DMAMux.

- Use an always enabled DMA request source. In this configuration, the eDMA is initialized to transfer the data using both minor and major loops, and the DMAMux performs the channel re-activation. For this option, the DMA request channel should be enabled and configured to use an always enabled request. Note that the re-activation of the channel can be continuous (periodic triggering is disabled) or can use the DMAMux/PIT periodic triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

17.6 Initialization / Application Information

17.6.1 Simple Setup

1. Setup the desired channel in the DMA
2. Connect the channel from step #1 to the desired peripheral by writing the CHCONFIG*x* register in the DMAMux
3. Enable the desired peripheral by clearing the appropriate MDIS bit
4. Configure the eDMA interface of the peripheral by writing the appropriate register(s) in the peripheral. Please refer to the section for the particular peripheral for more information.

17.6.1.1 Configure eDMA Channel 0 to Service eSCI_A Transmit Requests

1. Set up the desired channel in the DMA
   - — Configure TCD0 in the eDMA to transfer from the ESCI_A's SCIDRH/L registers to memory
   - — Configure error and interrupt handling for DMA Channel 0 by writing the appropriate registers in the eDMA
   - — Write 0x00 to the DMASERQ register in the DMA to enable Channel 0
2. Connect the channel from step 1 to the desired peripheral by writing the CHCONFIG*x* register in the DMAMux
   - — Write 0x89 to CHCONFIG0 register to assign eSCI_A transmit request to eDMA Channel 0
3. Enabled the desired peripheral by clearing the appropriate MDIS bit
   - — Turn on the module by clearing the [MDIS] bit
4. Setup the DMA interface of the peripheral by writing the appropriate register(s) in the peripheral. Refer to Chapter 21, "Enhanced Serial Communications Interface Module (eSCI)," for more information.
   - — Select a baud rate. Write this value to the ESCI baud registers (SCIBDH/L) to start the baud rate generator. Writing to the SCIBDH has no effect without also writing to SCIBDL
   - — Write to SCIACR1 to configure word length, parity, and other configuration bits (LOOPS, RSRC, M, WAKE, ILT, PE, PT)
   - — Enable the transmitter, interrupts, receive, and wake up as required, by writing to the SCIACR2[TIE, TCIE, RIE, ILIE, TE, RE, RWU, SBK] bits. The TXDMA bit should be set

when the eSCI is used with a DMA. A preamble or idle character will then be shifted out of the transmitter shift register

17.6.2 Using the "Always Enabled" Feature to Periodically Drive GPIO Pins

1. Temporarily disable the desired DMA channel in the DMAMux by writing 0x00 to the appropriate CHCONFIGx register.
2. Setup the timer interval by writing the correct value to the TLVAL register. The timer reload value is calculated with the following formula:

$$\frac{\text{trigger period}}{\text{clock period}} - 1 = \text{timer reload value} \qquad \textit{Eqn. 17-1}$$

   Example: The system clock (f_{SYS}) is 50Mhz (t_{fSYS} = 20ns), which results in a peripheral bus clock (f_{IPS}) of 25Mhz (t_{fIPS} = 40ns). The desired trigger period is 25µs.

$$\frac{25\mu s}{40ns} - 1 = 625 - 1 = 624 = 0x0000\_0270 \qquad \textit{Eqn. 17-2}$$

3. Enable the PIT trigger by setting the corresponding bit in the PITEN register.
4. Configure the desired pin(s) to be general purpose outputs by writing 0x40 to the appropriate CONFIG*x* register(s).
5. Configure the transfer control descriptor (TCD) for the selected DMA channel by writing to the TCD*x* in the eDMA. The source address will probably be a location in the Flash or SRAM, and the destination address will be the PORTDATA register (with a minor counter = 1) or the PINDATA*x* registers (with a minor counter > 1). Note that if a pin is configured as either peripheral mode or general purpose input mode, then writing to the PORTDATA register will have no effect on that particular pin. In this manner, it is possible to simultaneously write to less than 16 pins in a port, using the PORTDATA register. If, however, other pins in the chosen port are also configured as general purpose output but should not be changed by eDMA transfers, the PINDATA*x* registers must be used to control the pins. In this case, the pins serviced by the eDMA must be contiguous (for example, PA[6:3]).
6. Configure interrupt or error handling features for the DMA channel as appropriate, by writing the appropriate registers in the eDMA.
7. Write the appropriate value to the DMASERQ register to enable the eDMA channel.
8. Enable the eDMA channel requests in the DMAMux (with triggering) by setting the appropriate CHCONFIG*x*[ENBL, TRIG] bits. Choose an "always enabled" source (i.e., 0x23 to 0x2A). This is done by writing 0xE3 to 0xEA, respectively, to the CHCONFIG*x* register. Eight unique "always enabled" requests are available in order to allocate one per eDMA channel and / or PIM port (for example, 0x2A for channel 0, 0x2B for channel 1, etc.).

17.6.3 Disabling a Source

At reset the DMAMux module clears the SOURCE field of all CHCONFIG*n* registers; thus, all peripheral module DMA service requests are effectively disabled until explicitly enabled by writing the appropriate SOURCE value into a CHCONFIG*n* register. Additionally, DMA requests from any peripheral may be disabled via the DMAMux without changing the peripheral module configuration.

17.6.4 Enable Source With Periodic Triggering

1. Determine which eDMA channel 0 to 7 will service the source.
2. Clear the ENBL and TRIG bits of the appropriate CHCONFIG*n* register.
3. Ensure that the eDMA channel is properly configured in the eDMA module. The eDMA channel may be enabled (within the eDMA module) at this point.
4. In the PIT, configure the associated timer (see Table 17-3) for the desired trigger service interval.
5. Select the source to be routed to the eDMA channel. Write to the corresponding CHCONFIG*n* register with the appropriate SOURCE encoding and the ENBL and TRIG bits set.

17.6.4.1 DSPI Channel Configured For Periodic Service

Below is an example of configuring DSPI_B Transmit for service by eDMA Channel 3 in periodic trigger mode.

1. Write 0x00 to CHCONFIG3 (DMAMux base + 0x0003).
2. Configure Channel 3 in the eDMA controller, including enabling the channel.
3. Configure Timer 4 in the Periodic Interrupt Timer (PIT) for the desired trigger interval.
4. Write 0xC5 to CHCONFIG3 (DMAMux base + 0x0003).

The following code example illustrates steps #1 and #4 above:

```
In file registers.h:
#define DMAMUX_BASE_ADDR  0xFC084000  /* Example only! */
        .                             /* Assumes char is 8-bits */
        .
        .
volatile  unsigned  char  *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
        .
        .
        .
In file main.c:
#include "registers.h"
        .
        .
        .
*CHCONFIG3 = 0x00;
*CHCONFIG3 = 0xC5;
        .
        .
        .
```

17.6.5 Enable Source With Transparent Triggering

1. Determine which eDMA channel will service the source.
2. Clear the ENBL and TRIG bits of the appropriate CHCONFIG*n* register.

3. Ensure that the eDMA channel is properly configured in the eDMA module. The eDMA channel may be enabled (within the eDMA module) at this point.
4. Select the source to be routed to the eDMA channel. Write to the corresponding CHCONFIG*n* register with the appropriate SOURCE encoding, the ENBL bit set and the TRIG bit cleared.

17.6.5.1 DSPI Channel Configured for Immediate Service

Below is an example of configuring DSPI_B Transmit for service by eDMA Channel 3 in transparent trigger mode.

1. Write 0x00 to CHCONFIG3 (DMAMux base + 0x0003)
2. Configure Channel 3 in the eDMA controller, including enabling the channel.
3. Configure DSPI_B transmitter to generate DMA service requests.
4. Write 0x85 to CHCONFIG3 (DMAMux base + 0x0003)

The following code example illustrates steps #1 and #3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR  0xFC084000  /* Example only! */
        .                             /* Assumes char is 8-bits */
        .
        .
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
        .
        .
        .
In File main.c:
#include "registers.h"
        .
        .
        .
*CHCONFIG3 = 0x00;
*CHCONFIG3 = 0x85;
        .
        .
        .
```

17.6.6 Switching DMA Request Source to eDMA Channel Assignment

If it is desired to “dynamically” change a request source to eDMA channel assignment, the following procedure should be used:

1. Disable the eDMA channel in the eDMA module and re-configure the channel for the new source.
2. Clear the ENBL and TRIG bits of the appropriate CHCONFIG*n* register.
3. In the PIT, configure the desired service interval in the associated Timer *n* (if necessary).
4. Select the source to be routed to the eDMA channel. Write to the corresponding CHCONFIG*n* register with the ENBL bit set and the appropriate SOURCE and TRIG values.

17.6.6.1 Switch eDMA Channel 9 from DSPI_A Transmit to eSCI_D Transmit

1. Via the eDMA configuration registers, disable eDMA Channel 9
2. Re-configure and enable eDMA Channel 9 to handle the eSCI_D transmit buffer(s).
3. Write 0x00 to CHCONFIG9 (DMAMux base + 0x0009)
4. Via the eSCI_D configuration registers, enable DMA requests for the transmitter as desired.
5. Write 0x8D to CHCONFIG9 (DMAMux base + 0x0009). In this case, setting the TRIG bit will have no effect, because channels 8 to 15 do not support the periodic triggering mode.

The following code example illustrates steps #2 and #4 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only! */
        .                             /* Assumes char is 8-bits */
        .
        .
volatile unsigned char *CHCONFIG9  = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
        .
        .
        .
In File main.c:
#include "registers.h"
        .
        .
        .
*CHCONFIG9 = 0x00;
*CHCONFIG9 = 0x8D;
        .
        .
        .
```

Chapter 18
Port Integration Module (PIM)

18.1 Overview

The Port Integration Module (PIM) provides the interface between the physical pins and either the general purpose I/O signals or the various peripherals which provide alternate signal functions, as shown in Figure 18-1 below. MAC7100 Family devices implement up to 8 ports, all of which are 16-bits wide. Each pin can be independently configured to be in either port mode or peripheral mode as listed in Table 18-1. When peripheral mode is selected, the pin's output and configuration are both controlled by the peripheral module. Each pin is also capable of being used as an interrupt source with a filter available to help eliminate instances of noise generated false interrupts. The PIM also provides configuration registers to disable the clock to the EIM module (if present) in order to reduce power consumption when the external bus is not in use, and for five non-port pins: $\overline{TA}$ / $\overline{AS}$, and the E-ICE JTAG pins (TCK, TMS, TDI and TDO).

Table 18-1. GPIO and Peripheral Signal Associations

Port	Peripheral
Port A	External Bus Data
Port B	I^2C, DSPI_A, DSPI_B
Port C	External Bus Address and Control
Port D	External Bus Address and Control, Interrupt Controller
Port E	ATD_A
Port F	eMIOS
Port G	SCI_A, SCI_B, SCI_C, SCI_D, CAN_A, CAN_B, CAN_C, CAN_D
Port H	ATD_B
Port I	DSPI_A, DSPI_B

In order to improve software performance, all pins can be controlled either in a port-wide or single-bit manner. Registers are provided which allow each pin to be independently controlled by writing to a separate register, but mirror registers are also available to simplify read and write operations by accessing all pins in a port at one time.

Following a reset into any mode other than normal or secured expanded mode, [1] all pins are configured as general purpose input ports. [2] Following a reset into normal or secured expanded mode, [1] the PA[15:0], PC[15:0] and PD[15:5,1:0] pins are controlled by the EIM (refer to Chapter 13, "External Interface Module (EIM)," for details) and are not available for general purpose functions.

1. Available only on the MAC7111, MAC7116, MAC7131 and MAC7136.
2. On L49P mask set devices, PD2 always operates as the CLKOUT output signal after reset.

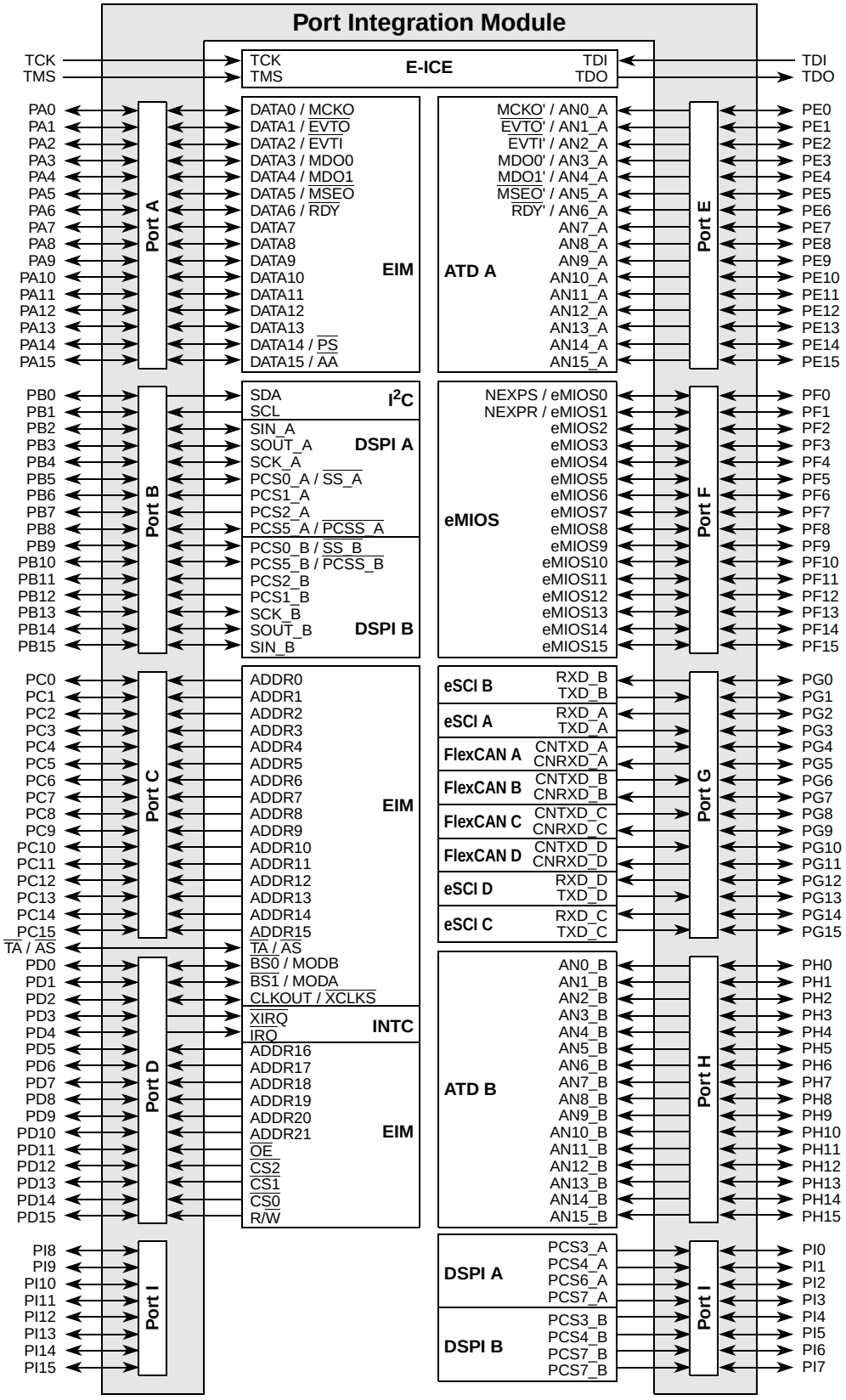


Figure 18-1. Port Integration Module Block Diagram

18.2 Features

The Port Integration Module includes these distinctive features:

- 8 independent 16-bit ports, with each pin having these characteristics:
  - — Selectable Peripheral or GPIO mode
  - — Input/Output selection
  - — 5 V output drive with two selectable drive strengths
  - — 5 V digital and analog inputs
  - — Selectable pull-up or pull-down resistors
  - — Selectable open drain for wired-or connections
  - — Selectable interrupt capability (with glitch filtering and interrupt mask)
- Port-wide and single-pin access methods
- Port C and Port H address map swapping for simplified MAC7101/MAC7106/ MAC7111/MAC7116 software compatibility
- Control of $\overline{TA}$ / $\overline{AS}$ and JTAG (TCK, TMS, TDI and TDO) pin characteristics
  - — Enable/disable or function selection
  - — Output drive type and strength selection
  - — Selectable pull-up or pull-down
- Control of EIM clock for reduced power consumption

18.3 Modes of Operation

Each of the 16 pins on a port can be independently selected between the following modes:

- Peripheral mode — In this mode, the associated peripheral (See Figure 18-1) is driving the pin and has control of the configuration (pull-up/pull-down, etc.) of that pin. Except for the drive strength, changing the configuration of the pin via the port configuration registers has no effect until the pin is switched into GPIO Mode.
- GPIO mode — In this mode, the pin is driven and controlled by the port data and configuration registers. The output and all control signals from the associated peripheral are ignored, and any inputs to the peripheral that would normally be driven externally onto the pin are statically held at a pre-determined state (low for most peripherals).

18.4 Signal Description

Table 18-2 details the behavior of PIM pins following reset. In single chip modes the port pins function as GPIO input signals until configured for an alternate function, while in expanded chip modes three of the ports provide the external bus. For detailed information on configuring pins for peripheral functions, refer to the associated peripheral module chapter.

Table 18-2. PIM / Peripheral Signal Properties

GPIO Function	Peripheral Function	Debug Function	Read on Reset	Peripheral Module	Reset State [1, 2]
PA15	DATA15	—	$\overline{AA}$	External Interface Module (Chapter 13)	I [1] or I/O [2]
PA14	DATA14	—	$\overline{PS}$		I [1] or I/O [2]
PA[13:7]	DATA[13:7]	—	—		I [1] or I/O [2]
PA[6:0]	DATA[6:0]	Nexus (pri.) [3]	—		I [1] or I/O [2]
PB15	SIN_B	—	—	DSPI Module B (Chapter 22)	I
PB14	SOUT_B	—	—		I
PB13	SCK_B	—	—		I
PB12	PCS1_B	—	—		I
PB11	PCS2_B	—	—		I
PB10	PCS5_B / $\overline{PCSS\_B}$	—	—		I
PB9	PCS0_B / $\overline{SS\_B}$	—	—		I
PB8	PCS5_A / $\overline{PCSS\_A}$	—	—	DSPI Module B (Chapter 22)	I
PB7	PCS2_A	—	—		I
PB6	PCS1_A	—	—		I
PB5	PCS0_A / $\overline{SS\_A}$	—	—		I
PB4	SCK_A	—	—		I
PB3	SOUT_A	—	—		I
PB2	SIN_A	—	—		I
PB1	SCL	—	—	I^2C Bus Module (Chapter 24)	I
PB0	SDA	—	—		I
PC[15:0]	ADDR[15:0]	—	—	External Interface Module (Chapter 13)	I [1] or I/O [2]
PD15	R/$\overline{W}$	—	—		I [1] or I/O [2]
PD14	$\overline{CS0}$	—	—		I [1] or I/O [2]
PD13	$\overline{CS1}$	—	—		I [1] or I/O [2]
PD12	$\overline{CS2}$	—	—		I [1] or I/O [2]
PD11	$\overline{OE}$	—	—		I [1] or I/O [2]
PD[10:5]	ADDR[21:16]	—	—		I [1] or I/O [2]
PD4	$\overline{XIRQ}$	—	—	Interrupt Controller Module (Chapter 10)	I
PD3	$\overline{IRQ}$	—	—		I
PD2 [4]	CLKOUT	—	$\overline{XCLKS}$	External Interface Module (Chapter 13)	I [1] or I/O [2]
PD1	$\overline{BS0}$	—	MODA		I [1] or I/O [2]
PD0	$\overline{BS1}$	—	MODB		I [1] or I/O [2]
PE[15:7]	AN[15:7]_A	—	—	ATD Module A (Chapter 19)	I
PE[6:0]	AN[6:0]_A	Nexus (alt.) [3]	—		I
PF[15:2]	eMIOS[15:2]	Debug Status [5]	—	eMIOS Module (Chapter 20)	I
PF1	eMIOS1	Debug Status [5]	NEXPR		I
PF0	eMIOS0	Debug Status [5]	NEXPS		I
PG15	TXD_C	—	—	eSCI Module C (Chapter 21)	I
PG14	RXD_C	—	—		I

Table 18-2. PIM / Peripheral Signal Properties (continued)

GPIO Function	Peripheral Function	Debug Function	Read on Reset	Peripheral Module	Reset State [1, 2]
PG13	TXD_D	—	—	eSCI Module D (Chapter 21)	I
PG12	RXD_D	—	—		I
PG11	CNRX_D	—	—	FlexCAN Module D (Chapter 23)	I
PG10	CNTX_D	—	—		I
PG9	CNRX_C	—	—	FlexCAN Module C (Chapter 23)	I
PG8	CNTX_C	—	—		I
PG7	CNRX_B	—	—	FlexCAN Module B (Chapter 23)	I
PG6	CNTX_B	—	—		I
PG5	CNRX_A	—	—	FlexCAN Module A (Chapter 23)	I
PG4	CNTX_A	—	—		I
PG3	TXD_A	—	—	eSCI Module A (Chapter 21)	I
PG2	RXD_A	—	—		I
PG1	TXD_B	—	—	eSCI Module B (Chapter 21)	I
PG0	RXD_B	—	—		I
PH[15:0]	AN[15:0]_B	—	—	ATD Module B (Chapter 19)	I
PI[15:8] [6]	—	—	—	—	I
PI[7:4] [6]	PCS[7:6, 4:3]_B	—	—	DSPI Module B (Chapter 22)	I
PI[3:0] [6]	PCS[7:6, 4:3]_A	—	—	DSPI Module A (Chapter 22)	I
—	$\overline{TA}$ / $\overline{AS}$ [7]	—	—	External Interface Module (Chapter 13)	I, pull-up
—	TCK [7]	—	—	E-ICE JTAG Port (Appendix A)	I, pull-down
—	TMS [7]	—	—		I, pull-up
—	TDI [7]	—	—		I, pull-up
—	TDO [7]	—	—		O, full drive

1 Following reset into single chip mode (refer to Chapter 7, “Modes of Operation”), all port pins are in GPIO mode, configured as inputs with no pull-up/pull-down and open drain disabled.

2 Following reset into expanded chip mode (refer to Chapter 7, “Modes of Operation”), all pins that have an associated EIM function are in external bus mode and GPIO functionality for those pins is not available. All other port pins are configured as described in Note 1.

3 Port F0 and F1 are sampled during the assertion of $\overline{RESET}$ to determine if the Nexus port is enabled and where it should be located. If Nexus is enabled, the associated GPIO or external bus mode pin function is not available. Refer to Appendix A, “Debug Interface,” for more information.

4 The PD2 function is not available on L49P mask set devices; after reset the pin is always an output driving the CLKOUT signal.

5 Optional debug status port is not implemented on L49P mask set devices. Refer to Section 26.4.1.6 on page 26-570.

6 Port I functions are available only on mask set L38Y devices in the 208-pin MAP BGA package (MAC7136).

7 The $\overline{TA}$, TCK, TMS, TDI and TDO signals are not controlled by the PIM on mask set L49P devices.

18.5 Memory Map / Register Definition

Table 18-3 and Table 18-4 show the memory map for the Port Integration Module. Note that all addresses are offsets; the absolute address may be computed by adding the specified offset to the base address of the PIM defined in Chapter 8, “Device Memory Map.”

All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. For example, the CONFIG0_A register is accessible via a 16-bit read or write to address ‘PIM base + 0x0000’, but performing a 16-bit access to ‘PIM base + 0x0001’ is illegal. Any access to an offset labeled as reserved will cause a bus cycle abort.

Table 18-3. PIM Memory Map — Global Registers

	PIM Offset	Register Description	Access
Global Control [1]	0x03C0	PIM Global Interrupt Status (GLBINT)	R
	0x03C2	PIM Global Configuration (PIMCONFIG)	R/W
	0x03C4	TDI Pin Configuration (CONFIG_TDI)	R/W
	0x03C6	TDO Pin Configuration (CONFIG_TDO)	R/W
	0x03C8	TMS Pin Configuration (CONFIG_TMS)	R/W
	0x03CA	TCK Pin Configuration (CONFIG_TCK)	R/W
	0x03CC	$\overline{TA}$ / $\overline{AS}$ Pin Configuration (CONFIG_TA)	R/W
	0x03CE	Reserved	—
	0x03D0 – 0x03DF	Reserved	—
32-bit Port Input [2]	0x03E0	Port A/B 32-bit Input Register (PORT32IR_AB)	R
	0x03E4	Port C/D 32-bit Input Register (PORT32IR_CD)	R
	0x03E8	Port E/F 32-bit Input Register (PORT32IR_EF)	R
	0x03EC	Port G/H 32-bit Input Register (PORT32IR_GH)	R
	0x03F0	Port B/C 32-bit Input Register (PORT32IR_BC)	R
	0x03F4	Port D/E 32-bit Input Register (PORT32IR_DE)	R
	0x03F8	Port F/G 32-bit Input Register (PORT32IR_FG)	R
	0x03FC	Port H/I 32-bit Input Register (PORT32IR_HI)	R

1 Not implemented on L49P mask set devices, and must be treated as reserved.

2 Not implemented on L49P, L47W or L61W mask set devices, and must be treated as reserved.

NOTE

The global registers and functions are not implemented on all devices, and these address offsets must be treated as reserved.

Table 18-4. PIM Memory Map — Port *x* Registers

	PIM Offset	Register Description	Access
Port A	0x0000	Port A Pin 0 Configuration (CONFIG0_A)	R/W
	0x0002	Port A Pin 1 Configuration (CONFIG1_A)	R/W
	0x0004	Port A Pin 2 Configuration (CONFIG2_A)	R/W
	0x0006	Port A Pin 3 Configuration (CONFIG3_A)	R/W
	0x0008	Port A Pin 4 Configuration (CONFIG4_A)	R/W
	0x000A	Port A Pin 5 Configuration (CONFIG5_A)	R/W
	0x000C	Port A Pin 6 Configuration (CONFIG6_A)	R/W
	0x000E	Port A Pin 7 Configuration (CONFIG7_A)	R/W
	0x0010	Port A Pin 8 Configuration (CONFIG8_A)	R/W
	0x0012	Port A Pin 9 Configuration (CONFIG9_A)	R/W
	0x0014	Port A Pin 10 Configuration (CONFIG10_A)	R/W
	0x0016	Port A Pin 11 Configuration (CONFIG11_A)	R/W
	0x0018	Port A Pin 12 Configuration (CONFIG12_A)	R/W
	0x001A	Port A Pin 13 Configuration (CONFIG13_A)	R/W
	0x001C	Port A Pin 14 Configuration (CONFIG14_A)	R/W
	0x001E	Port A Pin 15 Configuration (CONFIG15_A)	R/W
	0x0020	Port A Interrupt Flag (PORTIFR_A)	R/W
	0x0022	Reserved	—
	0x0024	Port A Data (PORTDATA_A)	R/W
	0x0026	Port A Input (PORTIR_A)	R
	0x0028	Port A Pin 0 Data (PINDATA0_A)	R/W
	0x0029	Port A Pin 1 Data (PINDATA1_A)	R/W
	0x002A	Port A Pin 2 Data (PINDATA2_A)	R/W
	0x002B	Port A Pin 3 Data (PINDATA3_A)	R/W
	0x002C	Port A Pin 4 Data (PINDATA4_A)	R/W
	0x002D	Port A Pin 5 Data (PINDATA5_A)	R/W
	0x002E	Port A Pin 6 Data (PINDATA6_A)	R/W
	0x002F	Port A Pin 7 Data (PINDATA7_A)	R/W
	0x0030	Port A Pin 8 Data (PINDATA8_A)	R/W
	0x0031	Port A Pin 9 Data (PINDATA9_A)	R/W
	0x0032	Port A Pin 10 Data (PINDATA10_A)	R/W
	0x0033	Port A Pin 11 Data (PINDATA11_A)	R/W
	0x0034	Port A Pin 12 Data (PINDATA12_A)	R/W
	0x0035	Port A Pin 13 Data (PINDATA13_A)	R/W
	0x0036	Port A Pin 14 Data (PINDATA14_A)	R/W
	0x0037	Port A Pin 15 Data (PINDATA15_A)	R/W
	0x0038 – 0x003F	Reserved	—

Table 18-4. PIM Memory Map — Port *x* Registers (continued)

	PIM Offset	Register Description	Access
Port B	0x0040	Port B Pin 0 Configuration (CONFIG0_B)	R/W
	0x0042	Port B Pin 1 Configuration (CONFIG1_B)	R/W
	0x0044	Port B Pin 2 Configuration (CONFIG2_B)	R/W
	0x0046	Port B Pin 3 Configuration (CONFIG3_B)	R/W
	0x0048	Port B Pin 4 Configuration (CONFIG4_B)	R/W
	0x004A	Port B Pin 5 Configuration (CONFIG5_B)	R/W
	0x004C	Port B Pin 6 Configuration (CONFIG6_B)	R/W
	0x004E	Port B Pin 7 Configuration (CONFIG7_B)	R/W
	0x0050	Port B Pin 8 Configuration (CONFIG8_B)	R/W
	0x0052	Port B Pin 9 Configuration (CONFIG9_B)	R/W
	0x0054	Port B Pin 10 Configuration (CONFIG10_B)	R/W
	0x0056	Port B Pin 11 Configuration (CONFIG11_B)	R/W
	0x0058	Port B Pin 12 Configuration (CONFIG12_B)	R/W
	0x005A	Port B Pin 13 Configuration (CONFIG13_B)	R/W
	0x005C	Port B Pin 14 Configuration (CONFIG14_B)	R/W
	0x005E	Port B Pin 15 Configuration (CONFIG15_B)	R/W
	0x0060	Port B Interrupt Flag (PORTIFR_B)	R/W
	0x0062	Reserved	—
	0x0064	Port B Data (PORTDATA_B)	R/W
	0x0066	Port B Input (PORTIR_B)	R
	0x0068	Port B Pin 0 Data (PINDATA0_B)	R/W
	0x0069	Port B Pin 1 Data (PINDATA1_B)	R/W
	0x006A	Port B Pin 2 Data (PINDATA2_B)	R/W
	0x006B	Port B Pin 3 Data (PINDATA3_B)	R/W
	0x006C	Port B Pin 4 Data (PINDATA4_B)	R/W
	0x006D	Port B Pin 5 Data (PINDATA5_B)	R/W
	0x006E	Port B Pin 6 Data (PINDATA6_B)	R/W
	0x006F	Port B Pin 7 Data (PINDATA7_B)	R/W
	0x0070	Port B Pin 8 Data (PINDATA8_B)	R/W
	0x0071	Port B Pin 9 Data (PINDATA9_B)	R/W
	0x0072	Port B Pin 10 Data (PINDATA10_B)	R/W
	0x0073	Port B Pin 11 Data (PINDATA11_B)	R/W
	0x0074	Port B Pin 12 Data (PINDATA12_B)	R/W
	0x0075	Port B Pin 13 Data (PINDATA13_B)	R/W
	0x0076	Port B Pin 14 Data (PINDATA14_B)	R/W
	0x0077	Port B Pin 15 Data (PINDATA15_B)	R/W
	0x0078 – 0x007F	Reserved	—

Table 18-4. PIM Memory Map — Port *x* Registers (continued)

	PIM Offset	Register Description	Access
Port C	0x0080	Port C Pin 0 Configuration (CONFIG0_C)	R/W
	0x0082	Port C Pin 1 Configuration (CONFIG1_C)	R/W
	0x0084	Port C Pin 2 Configuration (CONFIG2_C)	R/W
	0x0086	Port C Pin 3 Configuration (CONFIG3_C)	R/W
	0x0088	Port C Pin 4 Configuration (CONFIG4_C)	R/W
	0x008A	Port C Pin 5 Configuration (CONFIG5_C)	R/W
	0x008C	Port C Pin 6 Configuration (CONFIG6_C)	R/W
	0x008E	Port C Pin 7 Configuration (CONFIG7_C)	R/W
	0x0090	Port C Pin 8 Configuration (CONFIG8_C)	R/W
	0x0092	Port C Pin 9 Configuration (CONFIG9_C)	R/W
	0x0094	Port C Pin 10 Configuration (CONFIG10_C)	R/W
	0x0096	Port C Pin 11 Configuration (CONFIG11_C)	R/W
	0x0098	Port C Pin 12 Configuration (CONFIG12_C)	R/W
	0x009A	Port C Pin 13 Configuration (CONFIG13_C)	R/W
	0x009C	Port C Pin 14 Configuration (CONFIG14_C)	R/W
	0x009E	Port C Pin 15 Configuration (CONFIG15_C)	R/W
	0x00A0	Port C Interrupt Flag (PORTIFR_C)	R/W
	0x00A2	Reserved	—
	0x00A4	Port C Data (PORTDATA_C)	R/W
	0x00A6	Port C Input (PORTIR_C)	R
	0x00A8	Port C Pin 0 Data (PINDATA0_C)	R/W
	0x00A9	Port C Pin 1 Data (PINDATA1_C)	R/W
	0x00AA	Port C Pin 2 Data (PINDATA2_C)	R/W
	0x00AB	Port C Pin 3 Data (PINDATA3_C)	R/W
	0x00AC	Port C Pin 4 Data (PINDATA4_C)	R/W
	0x00AD	Port C Pin 5 Data (PINDATA5_C)	R/W
	0x00AE	Port C Pin 6 Data (PINDATA6_C)	R/W
	0x00AF	Port C Pin 7 Data (PINDATA7_C)	R/W
	0x00B0	Port C Pin 8 Data (PINDATA8_C)	R/W
	0x00B1	Port C Pin 9 Data (PINDATA9_C)	R/W
	0x00B2	Port C Pin 10 Data (PINDATA10_C)	R/W
	0x00B3	Port C Pin 11 Data (PINDATA11_C)	R/W
	0x00B4	Port C Pin 12 Data (PINDATA12_C)	R/W
	0x00B5	Port C Pin 13 Data (PINDATA13_C)	R/W
	0x00B6	Port C Pin 14 Data (PINDATA14_C)	R/W
	0x00B7	Port C Pin 15 Data (PINDATA15_C)	R/W
	0x00B8 – 0x00BF	Reserved	—

Table 18-4. PIM Memory Map — Port *x* Registers (continued)

	PIM Offset	Register Description	Access
Port D	0x00C0	Port D Pin 0 Configuration (CONFIG0_D)	R/W
	0x00C2	Port D Pin 1 Configuration (CONFIG1_D)	R/W
	0x00C4	Port D Pin 2 Configuration (CONFIG2_D) [1]	R/W
	0x00C6	Port D Pin 3 Configuration (CONFIG3_D)	R/W
	0x00C8	Port D Pin 4 Configuration (CONFIG4_D)	R/W
	0x00CA	Port D Pin 5 Configuration (CONFIG5_D)	R/W
	0x00CC	Port D Pin 6 Configuration (CONFIG6_D)	R/W
	0x00CE	Port D Pin 7 Configuration (CONFIG7_D)	R/W
	0x00D0	Port D Pin 8 Configuration (CONFIG8_D)	R/W
	0x00D2	Port D Pin 9 Configuration (CONFIG9_D)	R/W
	0x00D4	Port D Pin 10 Configuration (CONFIG10_D)	R/W
	0x00D6	Port D Pin 11 Configuration (CONFIG11_D)	R/W
	0x00D8	Port D Pin 12 Configuration (CONFIG12_D)	R/W
	0x00DA	Port D Pin 13 Configuration (CONFIG13_D)	R/W
	0x00DC	Port D Pin 14 Configuration (CONFIG14_D)	R/W
	0x00DE	Port D Pin 15 Configuration (CONFIG15_D)	R/W
	0x00E0	Port D Interrupt Flag (PORTIFR_D)	R/W
	0x00E2	Reserved	—
	0x00E4	Port D Data (PORTDATA_D)	R/W
	0x00E6	Port D Input (PORTIR_D)	R
	0x00E8	Port D Pin 0 Data (PINDATA0_D)	R/W
	0x00E9	Port D Pin 1 Data (PINDATA1_D)	R/W
	0x00EA	Port D Pin 2 Data (PINDATA2_D) [1]	R [1]
	0x00EB	Port D Pin 3 Data (PINDATA3_D)	R/W
	0x00EC	Port D Pin 4 Data (PINDATA4_D)	R/W
	0x00ED	Port D Pin 5 Data (PINDATA5_D)	R/W
	0x00EE	Port D Pin 6 Data (PINDATA6_D)	R/W
	0x00EF	Port D Pin 7 Data (PINDATA7_D)	R/W
	0x00F0	Port D Pin 8 Data (PINDATA8_D)	R/W
	0x00F1	Port D Pin 9 Data (PINDATA9_D)	R/W
	0x00F2	Port D Pin 10 Data (PINDATA10_D)	R/W
	0x00F3	Port D Pin 11 Data (PINDATA11_D)	R/W
	0x00F4	Port D Pin 12 Data (PINDATA12_D)	R/W
	0x00F5	Port D Pin 13 Data (PINDATA13_D)	R/W
	0x00F6	Port D Pin 14 Data (PINDATA14_D)	R/W
	0x00F7	Port D Pin 15 Data (PINDATA15_D)	R/W
	0x00F8 to 0x00FF	Reserved	—

1. The PD2 GPI function is not available on mask set L49P devices, and this offset must be treated as reserved.

Table 18-4. PIM Memory Map — Port *x* Registers (continued)

	PIM Offset	Register Description	Access
Port E	0x0100	Port E Pin 0 Configuration (CONFIG0_E)	R/W
	0x0102	Port E Pin 1 Configuration (CONFIG1_E)	R/W
	0x0104	Port E Pin 2 Configuration (CONFIG2_E)	R/W
	0x0106	Port E Pin 3 Configuration (CONFIG3_E)	R/W
	0x0108	Port E Pin 4 Configuration (CONFIG4_E)	R/W
	0x010A	Port E Pin 5 Configuration (CONFIG5_E)	R/W
	0x010C	Port E Pin 6 Configuration (CONFIG6_E)	R/W
	0x010E	Port E Pin 7 Configuration (CONFIG7_E)	R/W
	0x0110	Port E Pin 8 Configuration (CONFIG8_E)	R/W
	0x0112	Port E Pin 9 Configuration (CONFIG9_E)	R/W
	0x0114	Port E Pin 10 Configuration (CONFIG10_E)	R/W
	0x0116	Port E Pin 11 Configuration (CONFIG11_E)	R/W
	0x0118	Port E Pin 12 Configuration (CONFIG12_E)	R/W
	0x011A	Port E Pin 13 Configuration (CONFIG13_E)	R/W
	0x011C	Port E Pin 14 Configuration (CONFIG14_E)	R/W
	0x011E	Port E Pin 15 Configuration (CONFIG15_E)	R/W
	0x0120	Port E Interrupt Flag (PORTIFR_E)	R/W
	0x0122	Reserved	—
	0x0124	Port E Data (PORTDATA_E)	R/W
	0x0126	Port E Input (PORTIR_E)	R
	0x0128	Port E Pin 0 Data (PINDATA0_E)	R/W
	0x0129	Port E Pin 1 Data (PINDATA1_E)	R/W
	0x012A	Port E Pin 2 Data (PINDATA2_E)	R/W
	0x012B	Port E Pin 3 Data (PINDATA3_E)	R/W
	0x012C	Port E Pin 4 Data (PINDATA4_E)	R/W
	0x012D	Port E Pin 5 Data (PINDATA5_E)	R/W
	0x012E	Port E Pin 6 Data (PINDATA6_E)	R/W
	0x012F	Port E Pin 7 Data (PINDATA7_E)	R/W
	0x0130	Port E Pin 8 Data (PINDATA8_E)	R/W
	0x0131	Port E Pin 9 Data (PINDATA9_E)	R/W
	0x0132	Port E Pin 10 Data (PINDATA10_E)	R/W
	0x0133	Port E Pin 11 Data (PINDATA11_E)	R/W
	0x0134	Port E Pin 12 Data (PINDATA12_E)	R/W
	0x0135	Port E Pin 13 Data (PINDATA13_E)	R/W
	0x0136	Port E Pin 14 Data (PINDATA14_E)	R/W
	0x0137	Port E Pin 15 Data (PINDATA15_E)	R/W
	0x0138 – 0x013F	Reserved	—

Table 18-4. PIM Memory Map — Port *x* Registers (continued)

	PIM Offset	Register Description	Access
Port F	0x0140	Port F Pin 0 Configuration (CONFIG0_F)	R/W
	0x0142	Port F Pin 1 Configuration (CONFIG1_F)	R/W
	0x0144	Port F Pin 2 Configuration (CONFIG2_F)	R/W
	0x0146	Port F Pin 3 Configuration (CONFIG3_F)	R/W
	0x0148	Port F Pin 4 Configuration (CONFIG4_F)	R/W
	0x014A	Port F Pin 5 Configuration (CONFIG5_F)	R/W
	0x014C	Port F Pin 6 Configuration (CONFIG6_F)	R/W
	0x014E	Port F Pin 7 Configuration (CONFIG7_F)	R/W
	0x0150	Port F Pin 8 Configuration (CONFIG8_F)	R/W
	0x0152	Port F Pin 9 Configuration (CONFIG9_F)	R/W
	0x0154	Port F Pin 10 Configuration (CONFIG10_F)	R/W
	0x0156	Port F Pin 11 Configuration (CONFIG11_F)	R/W
	0x0158	Port F Pin 12 Configuration (CONFIG12_F)	R/W
	0x015A	Port F Pin 13 Configuration (CONFIG13_F)	R/W
	0x015C	Port F Pin 14 Configuration (CONFIG14_F)	R/W
	0x015E	Port F Pin 15 Configuration (CONFIG15_F)	R/W
	0x0160	Port F Interrupt Flag (PORTIFR_F)	R/W
	0x0162	Reserved	—
	0x0164	Port F Data (PORTDATA_F)	R/W
	0x0166	Port F Input (PORTIR_F)	R
	0x0168	Port F Pin 0 Data (PINDATA0_F)	R/W
	0x0169	Port F Pin 1 Data (PINDATA1_F)	R/W
	0x016A	Port F Pin 2 Data (PINDATA2_F)	R/W
	0x016B	Port F Pin 3 Data (PINDATA3_F)	R/W
	0x016C	Port F Pin 4 Data (PINDATA4_F)	R/W
	0x016D	Port F Pin 5 Data (PINDATA5_F)	R/W
	0x016E	Port F Pin 6 Data (PINDATA6_F)	R/W
	0x016F	Port F Pin 7 Data (PINDATA7_F)	R/W
	0x0170	Port F Pin 8 Data (PINDATA8_F)	R/W
	0x0171	Port F Pin 9 Data (PINDATA9_F)	R/W
	0x0172	Port F Pin 10 Data (PINDATA10_F)	R/W
	0x0173	Port F Pin 11 Data (PINDATA11_F)	R/W
	0x0174	Port F Pin 12 Data (PINDATA12_F)	R/W
	0x0175	Port F Pin 13 Data (PINDATA13_F)	R/W
	0x0176	Port F Pin 14 Data (PINDATA14_F)	R/W
	0x0177	Port F Pin 15 Data (PINDATA15_F)	R/W
	0x0178 – 0x017F	Reserved	—

Table 18-4. PIM Memory Map — Port *x* Registers (continued)

	PIM Offset	Register Description	Access
Port G	0x0180	Port G Pin 0 Configuration (CONFIG0_G)	R/W
	0x0182	Port G Pin 1 Configuration (CONFIG1_G)	R/W
	0x0184	Port G Pin 2 Configuration (CONFIG2_G)	R/W
	0x0186	Port G Pin 3 Configuration (CONFIG3_G)	R/W
	0x0188	Port G Pin 4 Configuration (CONFIG4_G)	R/W
	0x018A	Port G Pin 5 Configuration (CONFIG5_G)	R/W
	0x018C	Port G Pin 6 Configuration (CONFIG6_G)	R/W
	0x018E	Port G Pin 7 Configuration (CONFIG7_G)	R/W
	0x0190	Port G Pin 8 Configuration (CONFIG8_G)	R/W
	0x0192	Port G Pin 9 Configuration (CONFIG9_G)	R/W
	0x0194	Port G Pin 10 Configuration (CONFIG10_G)	R/W
	0x0196	Port G Pin 11 Configuration (CONFIG11_G)	R/W
	0x0198	Port G Pin 12 Configuration (CONFIG12_G)	R/W
	0x019A	Port G Pin 13 Configuration (CONFIG13_G)	R/W
	0x019C	Port G Pin 14 Configuration (CONFIG14_G)	R/W
	0x019E	Port G Pin 15 Configuration (CONFIG15_G)	R/W
	0x01A0	Port G Interrupt Flag (PORTIFR_G)	R/W
	0x01A2	Reserved	—
	0x01A4	Port G Data (PORTDATA_G)	R/W
	0x01A6	Port G Input (PORTIR_G)	R
	0x01A8	Port G Pin 0 Data (PINDATA0_G)	R/W
	0x01A9	Port G Pin 1 Data (PINDATA1_G)	R/W
	0x01AA	Port G Pin 2 Data (PINDATA2_G)	R/W
	0x01AB	Port G Pin 3 Data (PINDATA3_G)	R/W
	0x01AC	Port G Pin 4 Data (PINDATA4_G)	R/W
	0x01AD	Port G Pin 5 Data (PINDATA5_G)	R/W
	0x01AE	Port G Pin 6 Data (PINDATA6_G)	R/W
	0x01AF	Port G Pin 7 Data (PINDATA7_G)	R/W
	0x01B0	Port G Pin 8 Data (PINDATA8_G)	R/W
	0x01B1	Port G Pin 9 Data (PINDATA9_G)	R/W
	0x01B2	Port G Pin 10 Data (PINDATA10_G)	R/W
	0x01B3	Port G Pin 11 Data (PINDATA11_G)	R/W
	0x01B4	Port G Pin 12 Data (PINDATA12_G)	R/W
	0x01B5	Port G Pin 13 Data (PINDATA13_G)	R/W
	0x01B6	Port G Pin 14 Data (PINDATA14_G)	R/W
	0x01B7	Port G Pin 15 Data (PINDATA15_G)	R/W
	0x01B8 – 0x01BF	Reserved	—

Table 18-4. PIM Memory Map — Port *x* Registers (continued)

	PIM Offset	Register Description	Access
Port H	0x01C0	Port H Pin 0 Configuration (CONFIG0_H)	R/W
	0x01C2	Port H Pin 1 Configuration (CONFIG1_H)	R/W
	0x01C4	Port H Pin 2 Configuration (CONFIG2_H)	R/W
	0x01C6	Port H Pin 3 Configuration (CONFIG3_H)	R/W
	0x01C8	Port H Pin 4 Configuration (CONFIG4_H)	R/W
	0x01CA	Port H Pin 5 Configuration (CONFIG5_H)	R/W
	0x01CC	Port H Pin 6 Configuration (CONFIG6_H)	R/W
	0x01CE	Port H Pin 7 Configuration (CONFIG7_H)	R/W
	0x01D0	Port H Pin 8 Configuration (CONFIG8_H)	R/W
	0x01D2	Port H Pin 9 Configuration (CONFIG9_H)	R/W
	0x01D4	Port H Pin 10 Configuration (CONFIG10_H)	R/W
	0x01D6	Port H Pin 11 Configuration (CONFIG11_H)	R/W
	0x01D8	Port H Pin 12 Configuration (CONFIG12_H)	R/W
	0x01DA	Port H Pin 13 Configuration (CONFIG13_H)	R/W
	0x01DC	Port H Pin 14 Configuration (CONFIG14_H)	R/W
	0x01DE	Port H Pin 15 Configuration (CONFIG15_H)	R/W
	0x01E0	Port H Interrupt Flag (PORTIFR_H)	R/W
	0x01E2	Reserved	—
	0x01E4	Port H Data (PORTDATA_H)	R/W
	0x01E6	Port H Input (PORTIR_H)	R
	0x01E8	Port H Pin 0 Data (PINDATA0_H)	R/W
	0x01E9	Port H Pin 1 Data (PINDATA1_H)	R/W
	0x01EA	Port H Pin 2 Data (PINDATA2_H)	R/W
	0x01EB	Port H Pin 3 Data (PINDATA3_H)	R/W
	0x01EC	Port H Pin 4 Data (PINDATA4_H)	R/W
	0x01ED	Port H Pin 5 Data (PINDATA5_H)	R/W
	0x01EE	Port H Pin 6 Data (PINDATA6_H)	R/W
	0x01EF	Port H Pin 7 Data (PINDATA7_H)	R/W
	0x01F0	Port H Pin 8 Data (PINDATA8_H)	R/W
	0x01F1	Port H Pin 9 Data (PINDATA9_H)	R/W
	0x01F2	Port H Pin 10 Data (PINDATA10_H)	R/W
	0x01F3	Port H Pin 11 Data (PINDATA11_H)	R/W
	0x01F4	Port H Pin 12 Data (PINDATA12_H)	R/W
	0x01F5	Port H Pin 13 Data (PINDATA13_H)	R/W
	0x01F6	Port H Pin 14 Data (PINDATA14_H)	R/W
	0x01F7	Port H Pin 15 Data (PINDATA15_H)	R/W
	0x01F8 – 0x01FF	Reserved	—

Table 18-4. PIM Memory Map — Port *x* Registers (continued)

	PIM Offset	Register Description	Access
Port I [2]	0x0200	Port I Pin 0 Configuration (CONFIG0_I)	R/W
	0x0202	Port I Pin 1 Configuration (CONFIG1_I)	R/W
	0x0204	Port I Pin 2 Configuration (CONFIG2_I)	R/W
	0x0206	Port I Pin 3 Configuration (CONFIG3_I)	R/W
	0x0208	Port I Pin 4 Configuration (CONFIG4_I)	R/W
	0x020A	Port I Pin 5 Configuration (CONFIG5_I)	R/W
	0x020C	Port I Pin 6 Configuration (CONFIG6_I)	R/W
	0x020E	Port I Pin 7 Configuration (CONFIG7_I)	R/W
	0x0210	Port I Pin 8 Configuration (CONFIG8_I) [3]	R/W
	0x0212	Port I Pin 9 Configuration (CONFIG9_I) [3]	R/W
	0x0214	Port I Pin 10 Configuration (CONFIG10_I) [3]	R/W
	0x0216	Port I Pin 11 Configuration (CONFIG11_I) [3]	R/W
	0x0218	Port I Pin 12 Configuration (CONFIG12_I) [3]	R/W
	0x021A	Port I Pin 13 Configuration (CONFIG13_I) [3]	R/W
	0x021C	Port I Pin 14 Configuration (CONFIG14_I) [3]	R/W
	0x021E	Port I Pin 15 Configuration (CONFIG15_I) [3]	R/W
	0x0220	Port I Interrupt Flag (PORTIFR_I)	R/W
	0x0222	Reserved	—
	0x0224	Port I Data (PORTDATA_I)	R/W
	0x0226	Port I Input (PORTIR_I)	R
	0x0228	Port I Pin 0 Data (PINDATA0_I)	R/W
	0x0229	Port I Pin 1 Data (PINDATA1_I)	R/W
	0x022A	Port I Pin 2 Data (PINDATA2_I)	R/W
	0x022B	Port I Pin 3 Data (PINDATA3_I)	R/W
	0x022C	Port I Pin 4 Data (PINDATA4_I)	R/W
	0x022D	Port I Pin 5 Data (PINDATA5_I)	R/W
	0x022E	Port I Pin 6 Data (PINDATA6_I)	R/W
	0x022F	Port I Pin 7 Data (PINDATA7_I)	R/W
	0x0230	Port I Pin 8 Data (PINDATA8_I)	R/W
	0x0231	Port I Pin 9 Data (PINDATA9_I)	R/W
	0x0232	Port I Pin 10 Data (PINDATA10_I)	R/W
	0x0233	Port I Pin 11 Data (PINDATA11_I)	R/W
	0x0234	Port I Pin 12 Data (PINDATA12_I)	R/W
	0x0235	Port I Pin 13 Data (PINDATA13_I)	R/W
	0x0236	Port I Pin 14 Data (PINDATA14_I)	R/W
	0x0237	Port I Pin 15 Data (PINDATA15_I)	R/W
	0x02F8 to 0x03BF	Reserved	—

2. These offsets must be treated as reserved on devices other than mask set L38Y.
3. Since there is no peripheral function associated with PI[15:8], CONFIG[15:8]_I[MODE] must be 0.

18.5.1 Register Descriptions

The following sections describe the PIM registers in detail. Since all 9 ports (Ports A–I) are identical in their memory map, the following descriptions are generic.

18.5.1.1 PIM Port *x* Pin Configuration Registers (CONFIG*n_x*)

Each pin (16 per port) may be independently configured to select either peripheral or GPIO mode, as well as the pin characteristics when GPIO mode is selected.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	MODE	DD	ODE	RDS	PULL		PIE	PIF
W																
Reset	—	—	—	—	—	—	—	—	0/1 [1]	0	0	0	0	0	0	0
Reg Addr	See Table 18-4															

[1] When booted into any single-chip mode, all MODE bits are cleared to 0. When booted into an expanded mode, the MODE bit is set to 1 for PA[15:0], C[15:0] and D[15:5, 2:0] and is 0 for all other pins.

Figure 18-2. PIM Port *x* Pin Configuration Registers (CONFIG*n_x*)

Table 18-5. CONFIG*n_x* Field Descriptions

Bits	Name	Description
15–8	—	Reserved.
7	MODE	This bit switches the associated pin between peripheral and GPIO modes 0 Pin is in GPIO mode 1 Pin is in peripheral mode [1]
6	DD	Data direction. This bit switches the pin between output and input when the pin is in GPIO mode (MODE=0). 0 Pin is an input 1 Pin is an output [2]
5	ODE	Open drain enable. This bit configures the pin to be open drain when the pin is in GPIO mode (MODE=0). 0 Open drain is disabled 1 Open drain is enabled
4	RDS	Reduced drive strength. This bit selects the drive strength of the pin, and is active in both peripheral and GPIO modes. 0 Pin drive strength is full 1 Pin drive strength is reduced to 1/3 of full strength

Table 18-5. CONFIG*n*_*x* Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>3–2</td><td>PULL[1:0]</td><td>Pull-up/down select. These bits serve the dual purpose of selecting the polarity of the active interrupt edge and selecting a pull-up or pull-down device. Pull-up / pull-down may be enabled or disabled when the pin is in either peripheral or GPIO mode. [3]
<table>
<tr><th></th><th>Pull-up/down</th><th>External Interrupt Edge</th></tr>
<tr><td>00</td><td>Disabled</td><td>Rising Edge</td></tr>
<tr><td>01</td><td>Disabled</td><td>Falling Edge</td></tr>
<tr><td>10</td><td>Pull-down Enabled</td><td>Rising Edge</td></tr>
<tr><td>11</td><td>Pull-up Enabled</td><td>Falling Edge</td></tr>
</table></td></tr>
<tr><td>1</td><td>PIE</td><td>Pin interrupt enable. This bit masks the interrupt associated with a particular Pin Interrupt Flag bit. If this bit is set, an interrupt will occur if the corresponding PORTIFR bit is set. If this bit is cleared, then no interrupt will occur. Note that this bit does not affect the setting/clearing of the PORTIFR bit, it only determines whether a system interrupt will be generated or not.
0 Interrupt is masked
1 Interrupt is generated when PORTIFR bit is set and an active edge occurs</td></tr>
<tr><td>0</td><td>PIF</td><td>Pin interrupt flag. This bit indicates that an active edge (i.e., external interrupt) has been detected. Writing a 1 to this bit clears the pending interrupt. Writing 0 has no effect. Interrupts are enabled on the pin only when the pin is in GPIO mode and is configured to be an input (MODE=0, DDR=0).
0 No active edge pending.
1 An active edge has been detected on the corresponding pin. If the PIE bit is also set, an interrupt will occur.</td></tr>
</table>

1 Since there is no peripheral function associated with PI[15:8], CONFIG[15:8]_I[MODE] must be 0.

2 PD2 cannot be used as a general purpose output. Refer to Section 18.7.3, “PD2 / CLKOUT Configuration.”

3 On mask set L49P devices, pull-up/down control is not available in peripheral mode.

18.5.1.2 PIM Port *x* Interrupt Flag Register (PORTIFR_*x*)

This 16-bit register allows the reading and clearing of the PIF (pin interrupt flag) bits for all pins in a single bus access. This is useful for determining the exact source of an interrupt/wakeup, and for clearing multiple interrupts simultaneously.

A read of this register returns the value of the CONFIG*n*_*x*[PIF] bit for each pin in the port. Writing a 1 to any bit position clears the pending interrupt for the corresponding pin. Writing 0 to any bit position has no effect.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIF15	PIF14	PIF13	PIF12	PIF11	PIF10	PIF9	PIF8	PIF7	PIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	See Table 18-4															

Figure 18-3. PIM Port *x* Interrupt Flag Register (PORTIFR_*x*)

Table 18-6. PORTIFR_*x* Field Descriptions

Bits	Name	Description
15–0	PIF*n*	Interrupt flag. A read returns the value of the CONFIG*n*[PIF] bit for each pin in the port. Writing a 1 to any bit position clears the pending interrupt for the corresponding pin. Writing 0 to any bit position has no effect. Reading/writing from/to bit *n* of this register is functionally identical to reading/writing the PIF bit of the corresponding CONFIG*n* register.

18.5.1.3 PIM Port *x* Data Register (PORTDATA_*x*)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	See Table 18-4															

Figure 18-4. PIM Port *x* Data Register (PORTDATA_*x*)

Table 18-7. PORTDATA_*x* Field Descriptions

Bits	Name	Description
15–0	DATA	16-bit write/read data. This 16-bit field is used to drive and sample values to and from the associated pins. It has the following functionality (on a pin-by-pin basis): • In peripheral mode, the corresponding bit of the register is not used • In GPIO mode when a pin is configured as an input (MODE = 0, DDR = 0), the corresponding bit in the PORTDATA register reflects the value(s) driven onto the pin(s) by external hardware, not the value previously written to the PORTDATA. Note that there is a 2 clock cycle delay between changing of the level driven into the pin and the reflection of the new value in the PORTDATA register. • In GPIO mode when the pin is configured as an output (MODE = 0, DDR = 1), the corresponding bit in the PORTDATA register is used to specify the value to drive onto the pin.

18.5.1.4 PIM Port *x* Input Register (PORTIR_*x*)

This 16-bit read-only register returns the value on the associated pins, and can be used to detect overload or short circuit conditions on output pins. Functionally, it is related to the PORTDATA register in the following manner:

- When a pin is configured as an input (DDR = 0), reading the PORTDATA and PORTIR registers will return the same result in the corresponding bit position.
- When the pin is configured as an output (DDR = 1), reading the PORTIR register will return the value on the pin, while reading the PORTDATA register will yield the value in the PORTDATA register.
- If the pin is in peripheral mode (MODE = 1), then the value in the PORTDATA register may not be the same as the value driven onto the pin by the peripheral.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA															
W																
Reset	—[1]	—[1]	—[1]	—[1]	—[1]	—[1]	—[1]	—[1]	—[1]	—[1]	—[1]	—[1]	—[1]	—[1]	—[1]	—[1]
Reg Addr	See Table 18-4															

[1] The reset value is the level on the pin at the time of the first register read after $\overline{\text{RESET}}$.

Figure 18-5. PIM Port *x* Input Register (PORTIR_*x*)

Table 18-8. PORTIR_*x* Field Descriptions

Bits	Name	Description
15–0	DATA	16-bit read data register.

18.5.1.5 PIM Port *x* Pin Data Registers (PINDATA*n*_*x*)

These 1-bit registers provide a means to drive and sample individual pins without the need for mask and shift operations in software, and are particularly useful for using the GPIO functionality of a pin under eDMA control. Writing to or reading from these register is functionally equivalent to writing/reading the corresponding bit in a PORTDATA*n*, but without the need to mask and shift the value. (That is, reading from PINDATA3 is equivalent to reading PORTDATA, ANDing the result with 0x0008, and shifting the result right by 3). See Section 18.7.2.3.3, “Driving/Sampling Individual Pins,” for examples using this feature.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	DATA
W								
Reset	—	—	—	—	—	—	—	0
Reg Addr	See Table 18-4							

Figure 18-6. PIM Port *x* Pin Data Registers (PINDATA*n*_*x*)

Table 18-9. PINDATA*n*_*x* Field Descriptions

Bits	Name	Description
7–1	—	Reserved.
0	DATA	1-bit read/write data.

18.5.1.6 PIM Global Interrupt Status Register (GLBINT)

This 16-bit register allows software to read the PIF (pin interrupt flag) status for all ports in a single bus access. This is useful for quickly determining which port is the source of an interrupt/wakeup without the need to read the PORTIFR register of each port.

A read of this register returns the logical OR of the PORTIFR[15:0] bits for each register in the PIM. Writes are ignored.

NOTE

This register, and the function it provides, is not implemented on L49P mask set devices.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	GPIFH	GPIFG	GPIFF	GPIFE	GPIFD	GPIFC	GPIFB	GPIFA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	See Table 18-3															

Figure 18-7. PIM Global Interrupt Status Register (GLBINT)

Table 18-10. GLBINT Field Descriptions

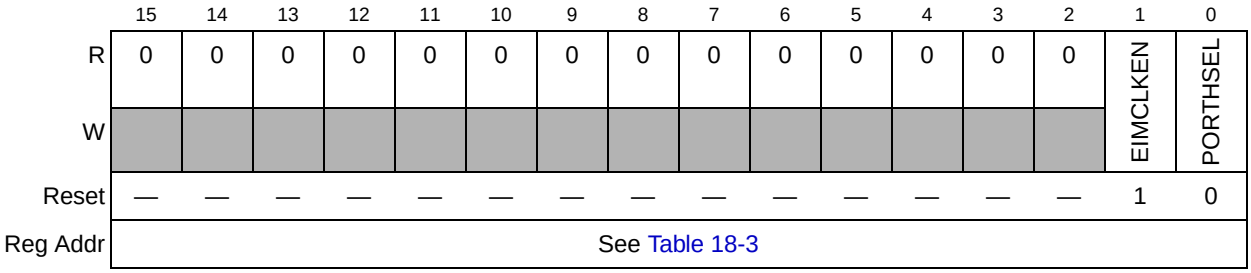
Bits	Name	Description
15–8	—	Reserved.
7–0	GPIF*x*	Global Port Interrupt Flag *x*. The logical OR of the PORTIFR[15:0] bits for port *x*. 0 No active edge pending in port *x*. 1 An active edge has been detected by at least one pin in port *x*.

18.5.1.7 PIM Global Configuration Register (PIMCONFIG)

This register allows software to switch Port C and Port H in the memory map in order to maintain software compatibility between device variations (in particular, the MAC7101, MAC7106, MAC7111 and MAC7116). It also allows the EIM clock to be disabled in order to reduce chip power consumption when the external bus is not in use.

NOTE

This register is not implemented on L49P or L61W mask set devices. The Port C/H mapping function is handled by the SSM Port Select Register (PORTSEL) register on L49P devices (L61W devices do not implement Port H). The EIM clock disable function is not available on L49P devices (L61W devices do not implement the EIM).



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EIMCLKEN	PORTHSEL
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	0
Reg Addr	See Table 18-3															

Figure 18-8. PIM Global Configuration Register (PIMCONFIG)

Table 18-11. PIMCONFIG Field Descriptions

Bits	Name	Description
15–2	—	Reserved.
1	EIMCLKEN	EIM Clock Enable. This bit controls the clock to the EIM module. If the external bus is not in use, clearing this bit reduces power consumption of the device. 0 EIM module clock disabled 1 EIM module clock enabled
0	PORTHSEL	Port H Select. [1] This bit configures the locations of Ports C and H in the PIM memory map. Refer to Table 18-4 for the addresses of these two ports. For MAC71x2 devices, which do not implement Port H, this bit has no effect. 0 Ports C and H occupy default locations in the PIM memory map. 1 Ports C and H swap locations in the PIM memory map.

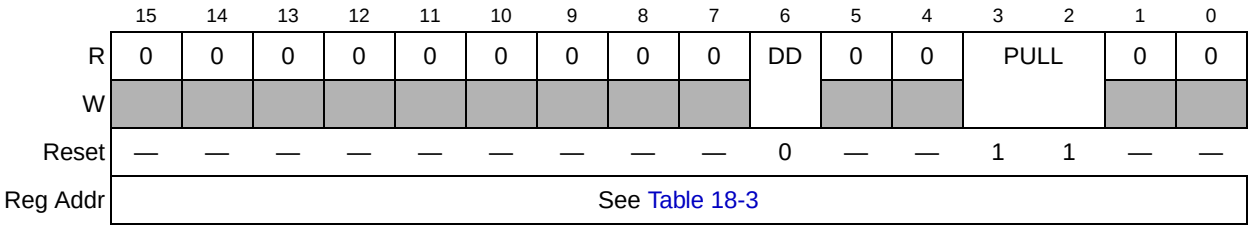
[1] On devices that do not implement Port H, this bit must be written as zero.

18.5.1.8 PIM Configure TDI Pin Register (CONFIG_TDI)

This register is used to enable or disable TDI functionality, as well as control the pull-up/pull-down configuration of the pin.

NOTE

This register, and the function it provides, is not implemented on L49P mask set devices.



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	DD	0	0	PULL		0	0
W																
Reset	—	—	—	—	—	—	—	—	—	0	—	—	1	1	—	—
Reg Addr	See Table 18-3															

Figure 18-9. PIM Configure TDI Pin Register (CONFIG_TDI)

Table 18-12. CONFIG_TDI Field Descriptions

Bits	Name	Description
15–7	—	Reserved.
6	DD	Data direction. This bit switches the pin between output and input. 0 Pin is an input, and TDI functionality is enabled. 1 Pin is an output, TDI functionality is disabled and the pin is driven low.
5–4	—	Reserved.
3–2	PULL[1:0]	Pull-up/down select. These bits select a pull-up or pull-down device. Pull-up / pull-down is enabled only when the pin is selected as an input. 00 Disabled 01 Disabled 10 Pull-down Enabled 11 Pull-up Enabled
1–0	—	Reserved.

18.5.1.9 PIM Configure TDO Pin Register (CONFIG_TDO)

This register is used to enable or disable TDO functionality, as well as control the drive strength configuration of the pin.

NOTE

This register, and the function it provides, is not implemented on L49P mask set devices.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	DD	0	RDS	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Reg Addr	See Table 18-3															

Figure 18-10. PIM Configure TDO Pin Register (CONFIG_TDO)

Table 18-13. CONFIG_TDO Field Descriptions

Bits	Name	Description
15–7	—	Reserved.
6	DD	Data direction. This bit switches the pin between output and input. 0 Pin is an input, and TDO functionality is enabled. 1 Pin is an output, TDO functionality is disabled, and changes on the pin are ignored.
5	—	Reserved.

Table 18-13. CONFIG_TDO Field Descriptions (continued)

Bits	Name	Description
4	RDS	Reduced drive strength. This bit selects the drive strength of the pin. Drive strength control is enabled only when the pin is selected as an output. 0 Pin drive strength is full 1 Pin drive strength is reduced to 1/3 of full strength
3–0	—	Reserved.

18.5.1.10 PIM Configure TMS Pin Register (CONFIG_TMS)

This register is used to enable or disable TMS functionality, as well as control the pull-up/pull-down configuration for the pin.

NOTE

This register, and the function it provides, is not implemented on L49P mask set devices.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	DD	0	0	PULL		0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
Reg Addr	See Table 18-3															

Figure 18-11. PIM Configure TMS Pin Register (CONFIG_TMS)

Table 18-14. CONFIG_TMS Field Descriptions

Bits	Name	Description
15–7	—	Reserved.
6	DD	Data direction. This bit switches the pin between output and input. 0 Pin is an input, and TMS functionality is enabled. 1 Pin is an output, TMS functionality is disabled and the pin is driven low.
5–4	—	Reserved.
3–2	PULL[1:0]	Pull-up/down select. These bits select a pull-up or pull-down device. Pull-up / pull-down is enabled only when the pin is selected as an input. 00 Disabled 01 Disabled 10 Pull-down Enabled 11 Pull-up Enabled
1–0	—	Reserved.

18.5.1.11 PIM Configure TCK Pin Register (CONFIG_TCK)

This register is used to enable or disable TCK functionality, as well as control the pull-up/pull-down configuration for the pin.

NOTE

This register, and the function it provides, is not implemented on L49P mask set devices.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	DD	0	0	PULL		0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Reg Addr	See Table 18-3															

Figure 18-12. PIM Configure TCK Pin Register (CONFIG_TCK)

Table 18-15. CONFIG_TCK Field Descriptions

Bits	Name	Description
15–7	—	Reserved.
6	DD	Data direction. This bit switches the pin between output and input. 0 Pin is an input, and TCK functionality is enabled. 1 Pin is an output, TCK functionality is disabled and the pin is driven low.
5–4	—	Reserved.
3–2	PULL[1:0]	Pull-up/down select. These bits select a pull-up or pull-down device. Pull-up / pull-down is enabled only when the pin is selected as an input. 00 Disabled 01 Disabled 10 Pull-down Enabled 11 Pull-up Enabled
1–0	—	Reserved.

18.5.1.12 PIM Configure $\overline{TA}$ / $\overline{AS}$ Pin Register (CONFIG_TA)

This register is used to switch between $\overline{TA}$ and $\overline{AS}$ functionality, as well as control the drive type/strength and pull-up/pull-down configuration for the pin.

NOTE

This register, and the function it provides, is not implemented on L49P and L61W mask set devices, as the $\overline{AS}$ and EIM functions are not available on the respective mask sets.

$\overline{TA}$ / $\overline{AS}$ are EIM signals, so writing to this register is only valid for MAC7111, MAC7116, MAC7131 and MAC7136 devices.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	DD	ODE	RDS	PULL		0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
Reg Addr	See Table 18-3															

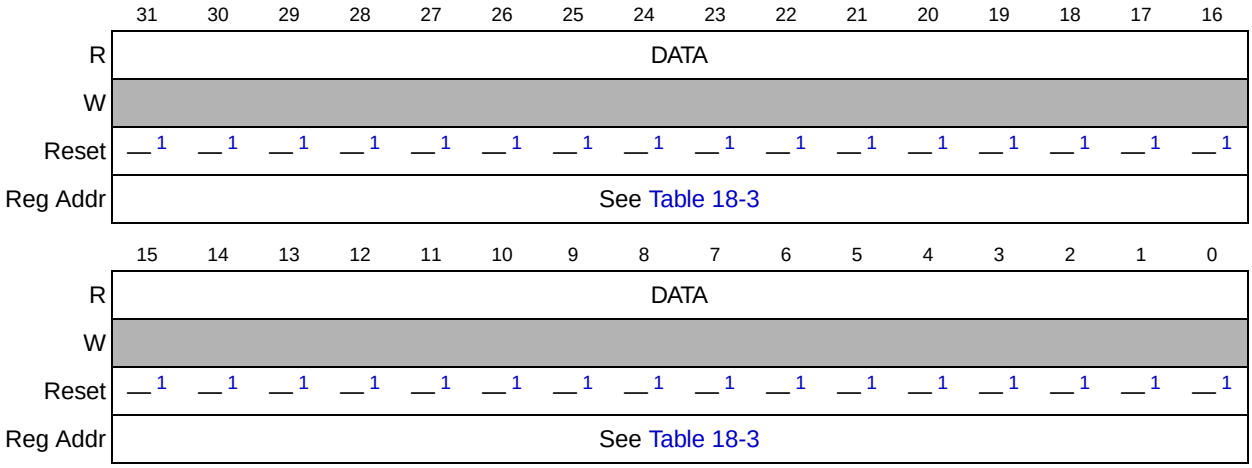
Figure 18-13. PIM Configure $\overline{TA}$ / $\overline{AS}$ Pin Register (CONFIG_TA)

Table 18-16. CONFIG_TA Field Descriptions

Bits	Name	Description
15–7	—	Reserved.
6	DD	Data direction. This bit switches the pin between $\overline{TA}$ and $\overline{AS}$ functionality. 0 Pin is an input, with $\overline{TA}$ functionality enabled 1 Pin is an output, with $\overline{AS}$ functionality enabled
5	ODE	Open drain enable. This bit selects the drive type of the pin in $\overline{AS}$ mode. 0 Open drain is disabled 1 Open drain is enabled
4	RDS	Reduced drive strength. This bit selects the drive strength of the pin in $\overline{AS}$ mode. 0 Pin drive strength is full 1 Pin drive strength is reduced to 1/3 of full strength
3–2	PULL[1:0]	Pull-up/down select. These bits select a pull-up or pull-down device for the pin. Pull-up / pull-down is enabled on the pin only when in $\overline{TA}$ mode. 00 Disabled 01 Disabled 10 Pull-down Enabled 11 Pull-up Enabled
1–0	—	Reserved.

18.5.1.13 PIM Port *x/x* 32-bit Input Registers (PORT32IR_*xx*)

These 32-bit read-only registers concatenate pairs of PORTIR_x registers, allowing high-speed input of bulk data via port pins. Functional characteristics are identical to the individual port input registers, as described in Section 18.5.1.4, “PIM Port x Input Register (PORTIR_x).” Refer to Table 18-3 for address offsets for the eight available register pair concatenations.



1 The reset value is the level on the pin at the time of the first register read after $\overline{\text{RESET}}$.

Figure 18-14. PIM Port *x*/*x* 32-bit Input Register (PORT32IR_*xx*)

Table 18-17. PORT32IR_*xx* Field Descriptions

Bits	Name	Description
32–0	DATA	32-bit read data register.

18.6 Functional Description

The Port Integration Module provides the means to utilize pins that are not used for peripheral or external bus functions as general purpose inputs/outputs (GPIO), with the following configurable capabilities:

- 5V output drive with two selectable drive strengths
- Pull-up or pull-down devices
- Open drain for wired-or connections
- Interrupt capability (with glitch filtering and interrupt mask)

18.6.1 Reset

After reset, all ports (Port A–Port I) are configured as general purpose inputs with open drain disabled and no pull-up/pull-down (see Section 18.6.3, “General Purpose Input Mode”) unless an expanded chip mode is selected or the Nexus debug capability is enabled. In those cases, the pins associated with the EIM and/or the debug interface are assigned the appropriate peripheral function as described in Table 18-1.

The $\overline{\text{TA}}$ / $\overline{\text{AS}}$ and E-ICE JTAG pins are configured as shown in Table 18-2.

18.6.2 Peripheral Mode

In peripheral mode, the peripheral functionality associated with the pin(s) are under the control of the applicable peripheral module.

Peripheral mode is enabled for a particular pin by setting the MODE bit in the corresponding CONFIG*n_x* register (see Section 18.7.1, “Using a Pin in Peripheral Mode,” for an example).

In peripheral mode, the following bits of the CONFIG*n_x* are unused:

- DDR (CONFIG*n_x*[6])
- ODE (CONFIG*n_x*[5])
- PIE (CONFIG*n_x*[1])
- PIF (CONFIG*n_x*[0])

Even though the above bits are unused in peripheral mode, it is possible to write to them at any time. If a pin is switched to GPIO mode, the last values written to them will take effect.

The PIF bits corresponding to the pin (both the CONFIG*n_x* bit 0 and the PORTIFR_*x* bit *n*) will not be changed while the pin is in peripheral mode. While in peripheral mode, the data registers PORTDATA_*x*, PORTIR_*x* and PINDATA*n_x* may be used as follows:

Table 18-18. PIM Register Behavior in Peripheral Mode

Register	Write	Read
PORTDATA_*x*	Writing to the corresponding bit in the PORTDATA_*x* register will not drive the value onto the pin, but it will set/clear the bit in the register. When the pin is switched to general purpose output mode, the new value will be driven onto the pin.	The value read from the PORTDATA_*x* register in peripheral mode depends on the current setting of the Data Direction bit. • DD = 1 (Output) – Returns the current value in the PORTDATA_*x* register. • DD = 0 (Input) – Returns the value driven into the pin, synchronized to the system clock. This is identical to reading the corresponding bit in the PORTIR_*x* register.
PORTIR_*x*	Writing to the PORTIR_*x* register will have no effect, as it is a read-only register in all modes.	Reading from the PORTIR_*x* register returns the value on the pin. This value is synchronized to the system clock, and there is a two clock cycle delay between a change of the value on the pin and the update of the PORTIR_*x* register.
PINDATA*n_x*	Writing to the PINDATA*n_x* register is identical to writing the corresponding bit in the PORTDATA_*x* register.	Reading from the PINDATA*n_x* register is identical to reading the corresponding bit from the PORTDATA_*x* register.

Figure 18-15 illustrates the logical structure of a single PIM pad cell in peripheral mode.

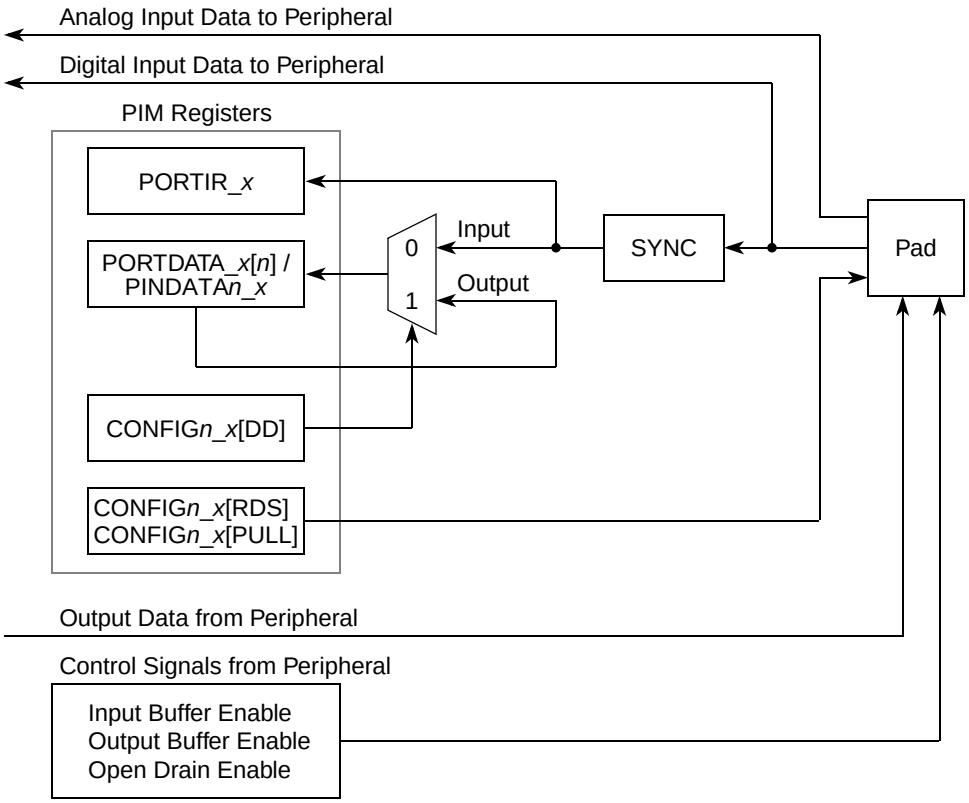


Figure 18-15. PIM Pad in Peripheral Mode Logical Diagram

18.6.3 General Purpose Input Mode

In general purpose input (GPI) mode the pin is configured as an input, with full software control of the pad and external interrupt capability.

GPI mode is set on a particular pin by clearing the MODE and DD bits in the appropriate CONFIG*n_x* register (see Section 18.7.2, “Using a Pin in GPIO Mode,” for an example). In GPI mode, the following CONFIG*n_x* bit is unused:

- RDS (CONFIG*n_x*[4])

Even though the above bit is unused in this mode, it may be written to at any time. If a pin is switched to GPO mode, the last value written to it will take effect. While in GPI mode, the data registers PORTDATA_*x*, PORTIR_*x* and PINDATA*n_x* may be used as follows:

Table 18-19. PIM Register Behavior in GPI Mode

Register	Write	Read
PORTDATA_*x*	Writing to the corresponding bit in PORTDATA_*x* will not drive the value onto the pin, but it will set/clear the bit in the register. If the pin is switched to general purpose output mode, the new value will be driven onto the pin.	Reading from the PORTDATA_*x* register will return the value driven on the pin, synchronized to the system clock. This is identical to reading the corresponding bit in the PORTIR_*x* register.
PORTIR_*x*	Writing to the PORTIR_*x* register will have no effect, as it is a read-only register in all modes.	Reading from the PORTIR_*x* register returns the value on the pin. This value is internally synchronized to the system clock, and there is a two clock cycle delay between a change of the value on the pin and the update of the PORTIR_*x* register.
PINDATA*n*_*x*	Writing to the PINDATA*n*_*x* register is identical to writing the corresponding bit in the PORTDATA_*x* register.	Reading from the PINDATA*n*_*x* register is identical to reading the corresponding bit from the PORTDATA_*x* register.

Figure 18-16 illustrates the logical structure of a single PIM pad cell in GPI mode.

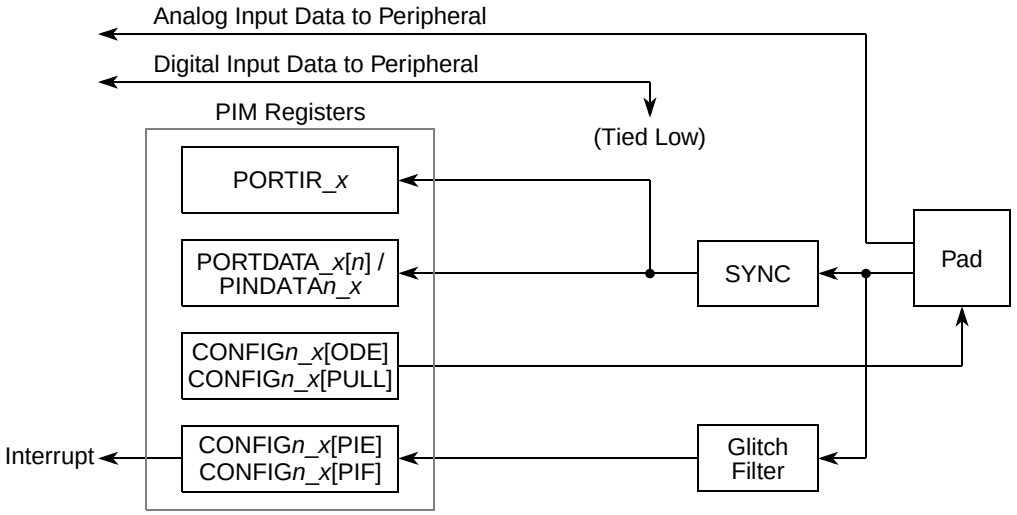


Figure 18-16. PIM Pad in General Purpose Input Mode

18.6.3.1 Interrupts

When configured as a general purpose input (GPI), a pin may also be used to detect transitions on the external signal and generate an interrupt. A pin may be used as an external interrupt by configuring the PULL bits as shown in the table below and setting the PIE bit in the appropriate CONFIG*n*_*x* register.

Table 18-20. PIM GPI Interrupt Polarity Configuration

PULL	Interrupt Type
00	Active High / Rising Edge
10	Active High / Rising Edge with pull-down
01	Active Low / Falling Edge
11	Active Low / Falling Edge with pull-up

Detection of an external signal transition may be done in one of two ways:

- Polling – Detect an external signal transition by continuously reading the CONFIG*n_x* register until the PIF bit is set. Alternatively, the GLBLINT or PORTIR_*x* register may be read to detect all external transitions for the entire PIM or port, respectively. When polling is used, the PIE bit in the CONFIG*n_x* register should be cleared to prevent the generation of an interrupt.
- Interrupt Driven – By setting the PIE bit in the appropriate CONFIG*n_x*, a system interrupt will be requested when an external signal transition occurs (with a minimum delay as described in Table 18-21). Depending on the configuration of the INTC (see Chapter 10, "Interrupt Controller Module (INTC)"), the interrupt request can force the processor to take an exception, where the PIF*n* may be serviced and cleared.

In either case, once an external signal transition has been recognized, the appropriate PIF bit must be cleared before a new external signal transition on that pin will be recognized, as described in Table 18-5 and Table 18-6).

In order to minimize interrupt service routine latency, the following strategies may be used:

- For up to 16 interrupt sources, assign all signals that are monitored for interrupt generation to a single port. This allows a single read of the appropriate PORTIR_*x* register to determine the source.
- For up to 8 interrupt sources, assign each signal that is monitored for interrupt generation to a unique port. This allows a single read of the GLBLINT register to determine the source.

All GPIO pins utilize a glitch filter, driven by f_{SYS}, to prevent spurious interrupt signals. When the system is in low power modes or during clock failure, f_{SYS} may driven by the slower on-chip oscillator or PLL in self-clocking mode. Therefore, filter and interrupt latencies will differ depending on the source of the clock signal in use by the system when the external signal transition occurs. Refer to Chapter 4, "System Clocks Module (OSC and CRG)," for more information. Table 18-21 shows the minimum pulse width for an external interrupt/wakeup signal.

Table 18-21. PIM Input Glitch Filter Characteristics

System Mode	Minimum time negated before active edge (t_{setup})	Minimum time asserted after active edge (t_{hold})
Run (Normal)	4 x t_{SYS}	4 x t_{SYS}
Stop (Low Power)	4 x 2.4µs = 9.6µs	4 x 2.4µs = 9.6µs

For example, in normal mode with a 20 Mhz bus clock (50 ns period), t_{setup} = 200 ns and t_{hold} = 200 ns. Figure 18-17 illustrates the timing parameters t_{setup} and t_{hold}. Six (6) bus clock cycles after the active edge (assuming no filtered glitches), the PIF bit will be set in the corresponding CONFIG*n_x* register.

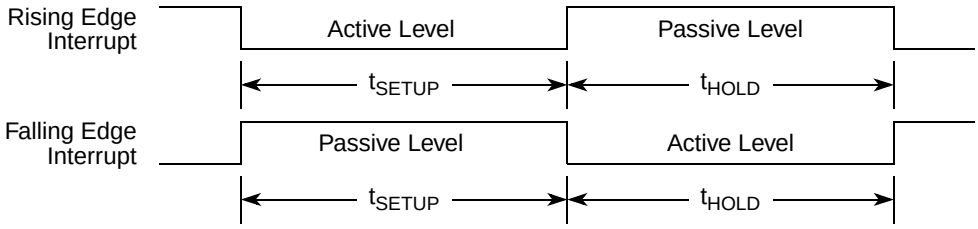


Figure 18-17. PIM External Interrupt Detection Requirements

NOTE

The above illustration is for timing purposes only. Level changes during the setup or hold period(s) that are short in duration will be ignored by the glitch filter.

For further discussion of interrupt usage, refer to Section 18.7.2.4, "Using Port Interrupts."

18.6.4 General Purpose Output Mode

In general purpose output (GPO) mode, the pin is configured as an output, with full software control of the pad (with the exception of PD2 / CLKOUT, as shown below).

GPO mode is enabled for a pin by clearing the MODE bit and setting the DD bit in the appropriate CONFIG*n_x* register (See Section 18.7.2, "Using a Pin in GPIO Mode" for an example).

In GPO mode, the following register bits are unused:

- PIE (CONFIG*n_x*[1])
- PIF (CONFIG*n_x*[0])

Even though the above bits are unused in this mode, they may be written at any time. If the pin is switched to GPI mode the last value written to the bits will take effect. While in GPO mode, the data registers PORTDATA_*x*, PORTIR_*x* and PINDATA*n_x* may be used as follows:

Figure 18-18 and Figure 18-19 illustrate the logical structure of a single PIM pad in GPO mode.

Table 18-22. PIM Register Behavior in GPO Mode

Register	Write	Read
PORTDATA_*x*	Writing to the corresponding bit in the PORTDATA_*x* register will drive that value onto the pin.	Reading the PORTDATA_*x* register returns the value in the PORTDATA_*x* register, which is also driven onto the pin. Comparing the value read from a bit in PORTDATA_*x* and PORTIR_*x* is useful for detecting shorts or driver overload conditions.

Table 18-22. PIM Register Behavior in GPO Mode

Register	Write	Read
PORTIR_*x*	Writing to the PORTIR_*x* register will have no effect, as it is a read-only register in all modes.	Reading from the PORTIR_*x* register returns the value on the pin. That this value is internally synchronized to the system clock, and there is a two clock cycle delay between changing the value on the pin and the update of the PORTIR_*x*.
PINDATA*n*_*x*	Writing to the PINDATA*n*_*x* register is identical to writing the corresponding bit *n* in the PORTDATA_*x* register.	Reading from the PINDATA*n*_*x* register is identical to reading the corresponding bit *n* from the PORTDATA_*x* register. **Note:** It is not possible to read the value driven onto PD2 when in GPO mode (see Section 18.7.3).

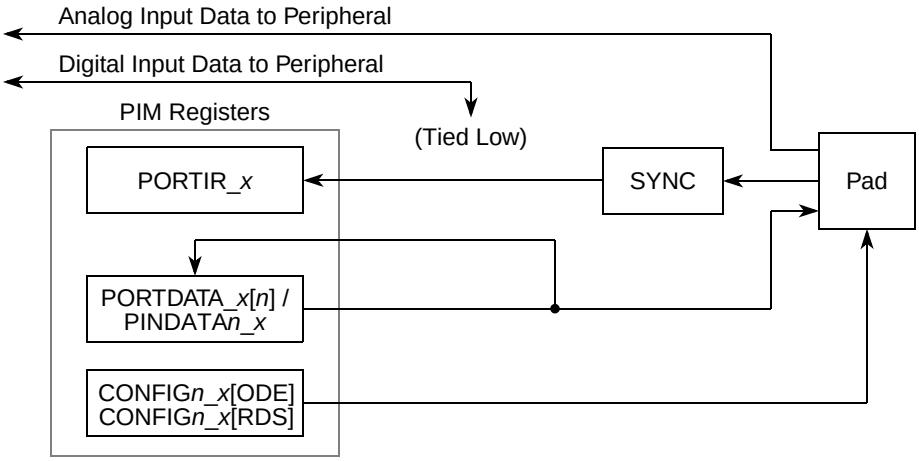


Figure 18-18. PIM Pad in GPO Mode (except PD2)

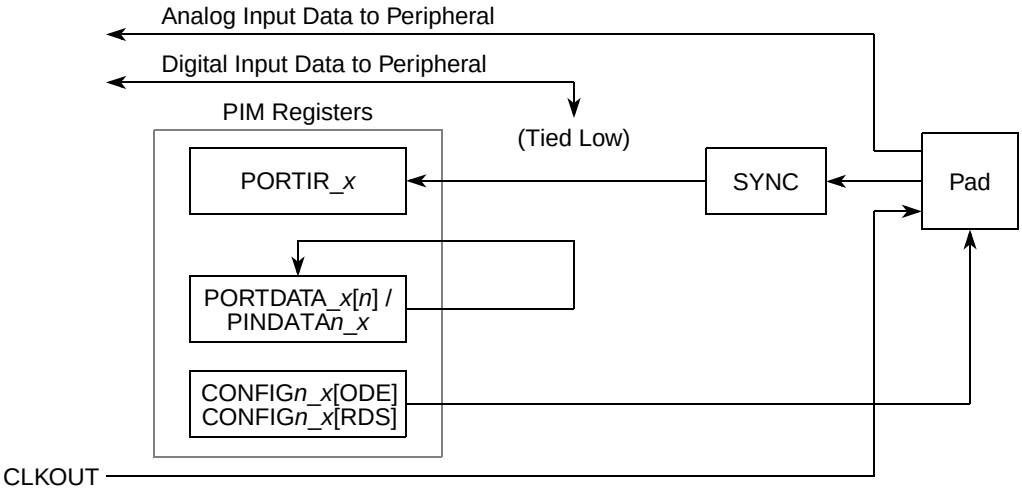


Figure 18-19. PIM Pad in GPO Mode (PD2 only)

18.7 Initialization / Application Information

18.7.1 Using a Pin in Peripheral Mode

To use a pin in peripheral mode, follow these steps:

1. Using Table 18-2, determine the port number(s) for the desired peripheral pin(s). Note that, in most cases, all pins associated with a peripheral should be switched to peripheral mode for proper operation of the peripheral.
2. Using Table 18-4, determine the appropriate CONFIG*n_x* register(s) for the port signals that will be used by the peripheral.
3. Determine whether a pull-up or pull-down is required.
4. Write 0x80, 0x88 or 0x8C to all of the required CONFIG*n_x* register(s). Note that in peripheral mode the values of all bits except the MODE and PULL[1:0] are unused.
5. If required, enable the peripheral. Refer to the applicable section for the peripheral module to determine the proper procedure to enable the peripheral.

18.7.1.1 PIM Example — Enable the I^2C module

1. For the I^2C module, it is necessary to set the pins associated with both the SDA and SCL signals to peripheral mode. Using Table 18-2, it is determined that the I^2C signals are multiplexed with the Port B0 and B1 signals, respectively.
2. Using Table 18-4, it is determined that the addresses for Port B0 and B1 pin configuration registers is ‘PIM base + 0x0040’ and ‘PIM Base + 0x0042,’ respectively.
3. Write 0x80 to ‘PIM Base + 0x0040’ and 0x80 to ‘PIM Base + 0x0042’ as the code example below illustrates.
4. The I^2C module may then be enabled by writing to the appropriate register in the I^2C register map.

```
In File registers.h:
#define PIM_BASE            0xFC0E8000          /* Example only! */
#define PORTB               0x040
/* Following example assumes short is 16-bits */
volatile unsigned short *CONFIG0_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x00);
volatile unsigned short *CONFIG1_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x02);
volatile unsigned short *CONFIG2_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x04);

In File main.c:
#include "registers.h"
        .
        .
        .
*CONFIG0_B = 0x0080;// No pull-up or pull-down
*CONFIG1_B = 0x0088;// Pull-down enabled
*CONFIG2_B = 0x008C;// Pull-up enabled
```

18.7.2 Using a Pin in GPIO Mode

18.7.2.1 GPIO Mode Initialization

To use a pin in GPIO mode, follow these steps:

1. Determine which peripheral functions will not be used. The pins corresponding to the unused peripheral(s) may be used for GPIO. Although it is possible to dynamically switch between peripheral and GPIO modes, there is a 2 cycle latency that must be accounted for, and it is generally recommended that switching is done only during initialization of the part (i.e.-statically).
2. If required, disable the peripheral. Refer to the applicable section for the peripheral module to determine the proper procedure to disable the peripheral.
3. Using Table 18-2, determine the port number(s) for the desired peripheral pin(s). Note that, in most cases, all pins associated with a peripheral should be switched to GPIO mode to avoid any possible spurious inputs to the peripheral.
4. Using Table 18-4, determine the applicable CONFIG*n_x* register(s) for the selected GPIO signals.
5. Determine the proper configuration for the pin(s):

Output Pin	Clear the MODE bit Set the DD bit Set the ODE bit as required Set the PULL bits as required Set the RDS bit as required
Input Pin	Clear the MODE bit Clear the DD bit Set the PULL bits as required Set the PIER/PIFR bits as required

6. Write the proper configuration to the selected CONFIG*n_x* register(s). Note that in GPIO mode, all unreserved bits in the CONFIG*n_x* register are used and should written as zero.

18.7.2.2 PIM Example — Use PB[1:0] in GPO/GPI Mode

For this example it is assumed that the I^2C module is unused and thus the two pins assigned to I^2C signals are available for GPIO use, one of which is used to implement an output control signal and one as an external interrupt/wakeup source.

1. For the I^2C module, it is necessary to set the pins associated with both the SDA and SCL signals to GPIO mode. Using Table 18-2, it is determined that the I^2C signals are multiplexed with the Port B0 and B1 signals, respectively.
2. The I^2C module is disabled by writing to the appropriate register in the I^2C register map.
3. Using Table 18-4, it is determined that the addresses for CONFIG0_B and CONFIG1_B are ‘PIM Base + 0x0040’ and ‘PIM Base + 0x0042’, respectively.
4. Port B0 is to be configured as an output control signal with the following characteristics:
   — No open drain
   — Full drive strength

The correct CONFIG*n_x* register value to set this configuration is 0x0040

5. Port B1 is to be configured as an input interrupt/wakeup signal with the following characteristics:

   — Active high interrupt with pull-down

   The correct CONFIG*n_x* register value to set this configuration is 0x000B (‘1’ is written to the PIF bit to ensure that any pending interrupts are cleared)

6. Write 0x0040 to ‘PIM Base + 0x0040’ and 0x000B to ‘PIM Base + 0x0042’. The following example code accomplishes the desired configuration:

```
In File registers.h:
#define PIM_BASE             0xFC0E8000          /* Example only! */
#define PORTB                0x040
/* Following example assumes short is 16-bits */
volatile unsigned short *CONFIG0_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x00);
volatile unsigned short *CONFIG1_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x02);

In File main.c:
#include "registers.h"
         .
         .
         .
*CONFIG0_B = 0x0040;
*CONFIG1_B = 0x000B;
         .
         .
         .
```

18.7.2.3 Accessing Data

For maximum flexibility, the value of signals may be driven or sampled both on a pin basis and a port (16-pin) basis by the processor core or via an eDMA channel. This section details how to drive and sample pins, as well as several examples.

18.7.2.3.1 Driving/Sampling an Entire Port

To drive or sample and entire port (16 pins), the pins must first be configured, following the steps outlined in Section 18.7.2.1, “GPIO Mode Initialization.” The signals may be driven onto the pins (output mode) or values sampled from the pins (input mode) by writing or reading the appropriate PORTDATA_*x* register. Note that writing to pins that are specified as inputs will have no effect. Conversely, reading from pins that are specified as outputs will simply return the last value written into the PORTDATA_*x* register. In this manner, it is possible to use the PORTDATA_*x* register to drive or sample less than the full 16 pins in a port. The following example illustrates this point.

18.7.2.3.2 PIM Example — Configure Port B For Mixed Peripheral/GPIO Use

In this example, it is desired to configure the PIM, I^2C and DSPI Modules to interface to an Inter-Integrated Circuit bus, 10 general purpose outputs and 4 general purpose inputs. The 4 inputs are then sampled and a data pattern driven onto the outputs.

In File **registers.h:**

```
#define PIM_BASE             0xFC0E8000          /* Example only! */
#define PORTB                0x040
/* Following example assumes short is 16-bits */
volatile unsigned short *CONFIG0_B  = (volatile unsigned short *) (PIM_BASE+PORTB+0x00);
volatile unsigned short *CONFIG1_B  = (volatile unsigned short *) (PIM_BASE+PORTB+0x02);
volatile unsigned short *CONFIG2_B  = (volatile unsigned short *) (PIM_BASE+PORTB+0x04);
volatile unsigned short *CONFIG3_B  = (volatile unsigned short *) (PIM_BASE+PORTB+0x06);
volatile unsigned short *CONFIG4_B  = (volatile unsigned short *) (PIM_BASE+PORTB+0x08);
volatile unsigned short *CONFIG5_B  = (volatile unsigned short *) (PIM_BASE+PORTB+0x0A);
volatile unsigned short *CONFIG6_B  = (volatile unsigned short *) (PIM_BASE+PORTB+0x0C);
volatile unsigned short *CONFIG7_B  = (volatile unsigned short *) (PIM_BASE+PORTB+0x0E);
volatile unsigned short *CONFIG8_B  = (volatile unsigned short *) (PIM_BASE+PORTB+0x10);
volatile unsigned short *CONFIG9_B  = (volatile unsigned short *) (PIM_BASE+PORTB+0x12);
volatile unsigned short *CONFIG10_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x14);
volatile unsigned short *CONFIG11_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x16);
volatile unsigned short *CONFIG12_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x18);
volatile unsigned short *CONFIG13_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x1A);
volatile unsigned short *CONFIG14_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x1C);
volatile unsigned short *CONFIG15_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x1E);
volatile unsigned short *PORTDATA_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x24);

In File main.c:
#include "registers.h"
        .
        .
        .
/* Initialize Port B */
*CONFIG0_B  = 0x0080;  /* Peripheral Mode */
*CONFIG1_B  = 0x0080;  /* Peripheral Mode */
*CONFIG2_B  = 0x0040;  /* GPIO Mode: Output */
*CONFIG3_B  = 0x0040;  /* GPIO Mode: Output */
*CONFIG4_B  = 0x0040;  /* GPIO Mode: Output */
*CONFIG5_B  = 0x0040;  /* GPIO Mode: Output */
*CONFIG6_B  = 0x0040;  /* GPIO Mode: Output */
*CONFIG7_B  = 0x0040;  /* GPIO Mode: Output */
*CONFIG8_B  = 0x0040;  /* GPIO Mode: Output */
*CONFIG9_B  = 0x0040;  /* GPIO Mode: Output */
*CONFIG10_B = 0x0040;  /* GPIO Mode: Output */
*CONFIG11_B = 0x0040;  /* GPIO Mode: Output */

/* The reset value of all CONFIGn_B registers is 0x0000,
   So the following equates may not be needed, but are
   included or illustration purposes. */
*CONFIG12_B = 0x0000;  /* GPIO Mode: Input, No interrupt */
*CONFIG13_B = 0x0000;  /* GPIO Mode: Input, No interrupt */
*CONFIG14_B = 0x0000;  /* GPIO Mode: Input, No interrupt */
*CONFIG15_B = 0x0000;  /* GPIO Mode: Input, No interrupt */
```

```
/* Sample the 4 input pins */
unsigned int value;
/* Mask out bits 0-11 and shift into bit positions 0-4.
   It is necessary to mask bits 0-11 because a read from
   PORTDATA_B on output ports will return the value of those
   bits in the PORTDATA_B register */
value = (*PORTDATA_B & 0xF000) >> 11;

/* drive the 10 output pins */
/* Shift our desired value '0x3FF' into bit positions 2-11.
   It is not necessary to mask out any other bits (i.e.-bits
   0-1 and 12-15) because writes to pins in GPIO Input or
   Peripheral Mode have no effect. However, these writes do
   change the value in the PORTDATA_B register, and therefore
   will be reflected in the next read of the PORTDATA_B register. */
*PORTDATA_B = 0x3FF < 2;
        .
        .
        .
```

18.7.2.3.3 Driving/Sampling Individual Pins

As illustrated in the above example, when using the PORTDATA_*x* register, it is necessary to perform mask and shift operations when sampling or driving data. For smaller numbers of pins, this can be inefficient, and can not be programmed into an eDMA channel easily. The example below illustrates using the pin data registers (PINDATA*n*_*x*) to overcome these two disadvantages.

NOTE

The PINDATA*n*_*x* register is simply a "mirror" of the corresponding bit *n* in the PORTDATA_*x* register. Therefore writes to the PORTDATA_*x* register will change the read value of a PINDATA*n*_*x* register, and vice-versa.

18.7.2.3.4 PIM Example — Use PB[1:0] as General Purpose Inputs

```
In File registers.h:
#define PIM_BASE            0xFC0E8000          /* Example only! */
#define PORTB               0x040
/* Following example assumes short is 16-bits */
volatile unsigned short *CONFIG0_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x00);
volatile unsigned short *CONFIG1_B = (volatile unsigned short *) (PIM_BASE+PORTB+0x02);
        .
        .
        .
volatile unsigned char *PINDATA0_B = (volatile unsigned char *)  (PIM_BASE+PORTB+0x28);
volatile unsigned char *PINDATA1_B = (volatile unsigned char *)  (PIM_BASE+PORTB+0x29);

In File main.c:
```

```
#include "registers.h"
         .
         .
         .
/* Initialize Port B */
*CONFIG0_B = 0x0000;    /* GPIO Mode: Input, No interrupt */
*CONFIG1_B = 0x0000;    /* GPIO Mode: Input, No interrupt */

/* Sample Ports B0 & B1 */
unsigned char value_B0;
unsigned char value_B1;
value_B0 = *PINDATA0_B; /* Equivalent to (*PORTDATA_B & 0xF001) >> 0 */
value_B1 = *PINDATA1_B; /* Equivalent to (*PORTDATA_B & 0x0002) >> 1 */
         .
         .
         .
```

18.7.2.3.5 Using the eDMA

Like the PORTDATA_*x* register, writes to the PINDATA*n_x* registers for input pins have no effect (other than to set the corresponding bit in the PORTDATA_*x* register). Therefore, it is possible to use an eDMA channel to drive multiple pins, even though the pins may not be contiguous (i.e., pins B0–B3 and B5–B8). Conversely, reads from the PINDATA*n_x* registers for output pins will simply return the value of the corresponding bit in the PORTDATA_*x* register. This section illustrates how to use an eDMA channel and a PIT timer along with the DMAMux to implement periodic protocols (such as PWM) on the GPIO ports.

As shown in the previous section, it is relatively easy to drive or sample multiple pins without the need for mask or shift operations, using the PINDATA*n_x* registers. This allows the use of an eDMA channel to copy entire tables of data from/to several pins at a time. In addition to standard DMA operations, the PIM, PIT and DMAMux interoperate to generate eDMA triggers to perform periodic copies of data. For details on how to setup and configure eDMA triggers in the DMAMux, refer to Section 17.5.1, “eDMA Channels 0 to 7,” on page 17-263. For details on how to setup and configure DMA triggers in the PIT, refer to the Section 25.4.1.5, “PIT Interrupt/DMA Select Register (PITINTSEL),” on page 25-555. In general, the procedure is as follows:

1. Configure the selected PIM port
2. Configure an available eDMA channel 0 through 7 to copy data tables from memory to the PIM port
3. Configure the appropriate PIT Timer 1 through 8 to generate periodic DMA requests
4. Configure the DMAMux to associate an “always enabled” source to the selected eDMA channel / PIT timer combination in periodic trigger mode (refer to Table 17-2 on page 17-261)

18.7.2.3.6 PIM Example — Drive Port Pins With eDMA Channel

The following procedure configures an eDMA channel to drive PG[2:0] with data from memory, updated every 3µs for 129µs.

1. Disable eSCI_A and eSCI_B
2. Configure PG0, PG1 and PG2 as outputs (CONFIG*n*_G = 0x0040)
3. Configure an available eDMA channel (0 through 8) to copy 3-byte “samples” from memory. This is done by using a minor loop count of 3 and a major loop count equal to the number of samples required for the complete waveform. In order to drive data for 129µs with a 3µs delay between each signal transition, a total of 44 samples of data are required. Therefore the major counter is 44.
4. Configure the PIT timer associated with the selected eDMA channel for a period of 3µs.
5. Configure the DMAMux to connect the selected eDMA channel to an “always on” request source and enable periodic trigger mode for that channel.

18.7.2.4 Using Port Interrupts

To use a pin to detect external signal transitions and request an interrupt, follow these steps:

1. Determine which peripheral(s) will not be used. The corresponding pins of the selected peripheral(s) may be used to trigger interrupts based on external signal transitions. Although it is possible to dynamically switch between peripheral and GPIO modes, there is a 2 cycle latency that must be accounted for, and it is generally recommended that switching is done only during initialization of the part (i.e. statically).
2. If required, disable the peripheral.
3. Using Table 18-2, determine the port signal(s) corresponding to the unused peripheral pin(s). Note that, in most cases, all pins associated with a peripheral should be switched to GPIO mode to avoid spurious inputs to the peripheral.
4. Using Table 18-4, determine the correct CONFIG*n_x* register(s) for the selected port signal(s).
5. Determine the proper configuration for the pin(s)
   - — Clear the MODE bit (pin is in GPIO mode)
   - — Clear the DD bit (pin is an input)
   - — Set the PIE bit (if an interrupt request is to be generated)
   - — Configure the PULL bits for proper polarity (see Table 18-5 or Table 18-20)
6. Write the proper configuration to the selected CONFIG*n_x* register(s). Note that, in GPIO mode, all unreserved bits in the CONFIG*n_x* register are used, and should be properly configured.
7. When a signal transition occurs on any pin configured for edge detection with the corresponding PIE bit set, an interrupt request is sent to the INTC module and the processor will receive an interrupt according to the configuration of that module. The interrupt service routine (ISR) should determine the exact source of the interrupt as follows:
   - — Read the INTC IPRH/IPRL registers to determine the IRQ source; if IPRH[IPR61] (bit 29) is set, then the PIM caused the interrupt.
   - — Read the GLBLINT register to determine which ports have pending interrupts. [1]

— Read the PORTIFR_*x* register for each port that has a pending interrupt request to identify which pins have pending interrupts. Note that there may be multiple interrupts pending at any given time.

8. After the exact sources of the interrupt have been determined, the ISR should write the appropriate value (0x10 for pin 8 of a port, for example) to the PORTIFR_*x* registers to clear only the CONFIG*n*_*x*[PIF] bits being serviced by the current iteration of the ISR. Because the MAC7100 IPS bus does not support an indivisible read-modify-write operation, care must be taken to avoid missing interrupts, particularly on pins configured as edge-sensitive inputs.

18.7.2.4.1 PIM Example — Use Port E As Interrupt Inputs

The example below disables the ATD_A module and uses all Port E signals as external interrupts.

1. Using Table 18-2, determine that the ATD_A signals are located on Port E.
2. Disable the ATD module by writing to the appropriate register in the ATD register map.
3. Using Table 18-4, determine that the addresses for PE[15:0] CONFIG*n*_*x* registers are ‘PIM Base + 0x0100’ to ‘PIM Base + 0x011E.’
4. Configure all PE[15:0] signals as interrupt/wakeup inputs with the characteristics:
   — Active low interrupt with pull-up (CONFIG*n*_E = 0x000F)
5. Clear all pending Port E interrupts by writing 0x00FF to the PORTIFR_E register.
6. Write 0x000F to ‘PIM Base + 0x0100’ through ‘PIM Base + 0x011E’.

The following example code configures Port E as desired:

```
In File registers.h:
#define PIM_BASE             0xFC0E8000          /* Example only! */
#define PORTE                0x100
/* Following example assumes short is 16-bits */
volatile unsigned short *CONFIG0_E  = (volatile unsigned short *) (PIM_BASE+PORTE+0x00);
volatile unsigned short *CONFIG1_E  = (volatile unsigned short *) (PIM_BASE+PORTE+0x02);
volatile unsigned short *CONFIG2_E  = (volatile unsigned short *) (PIM_BASE+PORTE+0x04);
volatile unsigned short *CONFIG3_E  = (volatile unsigned short *) (PIM_BASE+PORTE+0x06);
volatile unsigned short *CONFIG4_E  = (volatile unsigned short *) (PIM_BASE+PORTE+0x08);
volatile unsigned short *CONFIG5_E  = (volatile unsigned short *) (PIM_BASE+PORTE+0x0A);
volatile unsigned short *CONFIG6_E  = (volatile unsigned short *) (PIM_BASE+PORTE+0x0C);
volatile unsigned short *CONFIG7_E  = (volatile unsigned short *) (PIM_BASE+PORTE+0x0E);
volatile unsigned short *CONFIG8_E  = (volatile unsigned short *) (PIM_BASE+PORTE+0x10);
volatile unsigned short *CONFIG9_E  = (volatile unsigned short *) (PIM_BASE+PORTE+0x12);
volatile unsigned short *CONFIG10_E = (volatile unsigned short *) (PIM_BASE+PORTE+0x14);
volatile unsigned short *CONFIG11_E = (volatile unsigned short *) (PIM_BASE+PORTE+0x16);
volatile unsigned short *CONFIG12_E = (volatile unsigned short *) (PIM_BASE+PORTE+0x18);
volatile unsigned short *CONFIG13_E = (volatile unsigned short *) (PIM_BASE+PORTE+0x1A);
volatile unsigned short *CONFIG14_E = (volatile unsigned short *) (PIM_BASE+PORTE+0x1C);
volatile unsigned short *CONFIG15_E = (volatile unsigned short *) (PIM_BASE+PORTE+0x1E);
```

1. Note that the GLBLINT register is not available on L49P mask devices, therefore the PORTIFR_x registers of all ports configured to generate interrupts must be polled to determine the source of the interrupt(s).

```
In File main.c:
#include "registers.h"
         .
         .
         .
*CONFIG0_E  = 0x000F;
*CONFIG1_E  = 0x000F;
*CONFIG2_E  = 0x000F;
*CONFIG3_E  = 0x000F;
*CONFIG4_E  = 0x000F;
*CONFIG5_E  = 0x000F;
*CONFIG6_E  = 0x000F;
*CONFIG7_E  = 0x000F;
*CONFIG8_E  = 0x000F;
*CONFIG9_E  = 0x000F;
*CONFIG10_E = 0x000F;
*CONFIG11_E = 0x000F;
*CONFIG12_E = 0x000F;
*CONFIG13_E = 0x000F;
*CONFIG14_E = 0x000F;
*CONFIG15_E = 0x000F;
         .
         .
         .
```

When an interrupt request is asserted by the PIM and the INTC determines that it is the highest priority unmasked interrupt, vector number 0x003D will be returned during the IACK cycle (refer to Table 6-2 on page 6-85 and Section 10.6.3, “Vector Generation During IACK,” on page 10-117). The ISR that is executed may then read the GLBLINT and PORTIFR_*x* registers to determine which pins have pending interrupts, clear the appropriate CONFIG*n_x*[PIF] bits and execute the corresponding service functions for each pin with a pending interrupt. The following example code illustrates this, assuming all pins on all ports are used for external interrupts:

```
In File registers.h:
#define PIM_BASE             0xFC0E8000          /* Example only! */
/* Following example assumes short is 16-bits */
volatile unsigned short *GLBLINT   = (volatile unsigned short *) (PIM_BASE+0x03C0);
volatile unsigned short *PORTIFR_A = (volatile unsigned short *) (PIM_BASE+0x0020);
volatile unsigned short *PORTIFR_B = (volatile unsigned short *) (PIM_BASE+0x0060);
volatile unsigned short *PORTIFR_C = (volatile unsigned short *) (PIM_BASE+0x00A0);
volatile unsigned short *PORTIFR_D = (volatile unsigned short *) (PIM_BASE+0x00E0);
volatile unsigned short *PORTIFR_E = (volatile unsigned short *) (PIM_BASE+0x0120);
volatile unsigned short *PORTIFR_F = (volatile unsigned short *) (PIM_BASE+0x0160);
volatile unsigned short *PORTIFR_G = (volatile unsigned short *) (PIM_BASE+0x01A0);
volatile unsigned short *PORTIFR_H = (volatile unsigned short *) (PIM_BASE+0x01E0);
volatile unsigned short *PORTIFR_I = (volatile unsigned short *) (PIM_BASE+0x0200);
```

```
In File main.c:
#include "registers.h"
         .
         .
         .
unsigned short ifr_global;
unsigned short ifr_port_a;
unsigned short ifr_port_b;
unsigned short ifr_port_c;
unsigned short ifr_port_d;
unsigned short ifr_port_e;
unsigned short ifr_port_f;
unsigned short ifr_port_g;
unsigned short ifr_port_h;
unsigned short ifr_port_i;

ifr_global = *GLBLINT;    /* Determine which ports have pending interrupts */

ifr_port_a = *PORTIFR_A;  /* Determine which interrupt sources are pending on Port A */
*PORTIFR_A = 0xFFFF;      /* Clear all pending interrupts on Port A */

ifr_port_b = *PORTIFR_B;  /* Determine which interrupt sources are pending on Port B */
*PORTIFR_B = 0xFFFF;      /* Clear all pending interrupts on Port B */

ifr_port_c = *PORTIFRC;   /* Determine which interrupt sources are pending on Port C */
*PORTIFR_C = 0xFFFF;      /* Clear all pending interrupts on Port C */

ifr_port_d = *PORTIFR_D;  /* Determine which interrupt sources are pending on Port D */
*PORTIFR_D = 0xFFFF;      /* Clear all pending interrupts on Port D */

ifr_port_e = *PORTIFR_E;  /* Determine which interrupt sources are pending on Port E */
*PORTIFR_E = 0xFFFF;      /* Clear all pending interrupts on Port E */

ifr_port_f = *PORTIFR_F;  /* Determine which interrupt sources are pending on Port F */
*PORTIFR_F = 0xFFFF;      /* Clear all pending interrupts on Port F */

ifr_port_g = *PORTIFR_G;  /* Determine which interrupt sources are pending on Port G */
*PORTIFR_G = 0xFFFF;      /* Clear all pending interrupts on Port G */

ifr_port_h = *PORTIFR_H;  /* Determine which interrupt sources are pending on Port H */
*PORTIFR_H = 0xFFFF;      /* Clear all pending interrupts on Port H */

ifr_port_h = *PORTIFR_I;  /* Determine which interrupt sources are pending on Port I */
*PORTIFR_I = 0xFFFF;      /* Clear all pending interrupts on Port I */
```

18.7.3 PD2 / CLKOUT Configuration

The peripheral functionality associated with port pins other than PD2 is controlled by the appropriate peripheral module, as shown in Figure 18-1. The PD2 / CLKOUT signal multiplexing is controlled exclusively within the PIM.

Peripheral and general purpose output modes do not operate on PD2 / CLKOUT in the same manner as other port pins. When the CONFIG2_D[DD] bit is set, the CLKOUT signal is always driven on the pin regardless of the CONFIG2_D[MODE] bit setting, while the MODE bit determines how the pin drive configuration may be controlled as show below.

Table 18-23. PD2 / CLKOUT Mode Selection

CONFIG2_D[MODE]	CONFIG2_D[DD]	PD2 / CLKOUT Mode
0	0	PD2 General Purpose Input
0	1	CLKOUT with ODE / RDS control
1	x	CLKOUT with no ODE / RDS control

NOTE

For L49P mask set devices: 1) PD2 functionality is not implemented and the pin always operates in the CLKOUT mode, 2) CLKOUT is disabled by changing PD0 to an input (writing 0x00 to CONFIG0_A), 3) CLKOUT drive strength is controlled by the CONFIG0_A[RDR] bit.

When the external bus is enabled following reset (only available on the MAC7111, MAC7116, MAC7131 and MAC7136), the CLKOUT function is automatically enabled (CONFIG2_D[MODE] = 1]). When the external bus is disabled following reset, the PD2 general purpose input function is automatically enabled (CONFIG2_D[MODE, DD] = 0b00]).

18.7.3.1 Using PD2 GPI Functionality

Independent of the chip operating mode and EIM module status, the GPI function may be selected to “disable” the CLKOUT signal, and the pin may be optionally read as a normal general purpose input. To select the GPI mode, clear the CONFIG2_D[MODE, DD] bits and configure the remaining register bits appropriately as described in Section 18.6.3, “General Purpose Input Mode.”

18.7.3.2 Using CLKOUT Functionality

Independent of the chip operating mode and EIM module status, the CLKOUT signal may be driven on the external pin. The drive strength and type may also be selected as needed, as described in Section 18.6.4, “General Purpose Output Mode.”

NOTE

Even though L61W mask set devices do not implement the EIM, the CLKOUT function is available, if required.

18.7.4 $\overline{TA}$ / $\overline{AS}$ Configuration

While the functionality of the $\overline{TA}$ / $\overline{AS}$ signal is associated with the EIM (refer to Chapter 13, “External Interface Module (EIM)”), multiplexing is controlled via the PIM.

When the external bus is enabled following reset (only available on the MAC7111, MAC7116, MAC7131 and MAC7136), the $\overline{TA}$ function (with pull-up enabled) is selected by default. To change the $\overline{TA}$ configuration or select the $\overline{AS}$ function, program the CONFIG_TA register bits appropriately:

$\overline{AS}$ Enabled	Set the DD bit Configure the ODE bit as desired Configure the RDS bit as desired
$\overline{TA}$ Enabled	Clear the DD bit Configure the PULL bits as desired (do not leave floating)

When the external bus is disabled following reset, the $\overline{TA}$ function is automatically selected (but the input is ignored) and the pull-up device is enabled.

NOTE

$\overline{AS}$ is not implemented on L49P mask set devices, and the pin always functions in the $\overline{TA}$ mode. $\overline{TA}$ / $\overline{AS}$ is only available on the MAC7111, MAC7116, MAC7131 and MAC7136.

Since L61W devices do not implement the EIM, the $\overline{TA}$ / $\overline{AS}$ functionality is not present, and the CONFIG_TA register memory location must be treated as reserved.

18.7.5 E-ICE JTAG Port Configuration

While the functionality of the E-ICE JTAG port signals are associated with the debug interface (refer to Appendix A, “Debug Interface”), configuration is controlled via the PIM. Refer to Section 18.5.1.8 through Section 18.5.1.11 for detailed configuration options.

NOTE

This functionality is not implemented on L49P mask set devices.

18.7.6 Using the PIMCONFIG Register

The PIMCONFIG register provides two control bits to provide simplified software porting between various members of the MAC7100 Family, and to reduce power consumption in cases where the external bus is not used.

NOTE

The PIMCONFIG register is not implemented on mask set L49P and L61W devices.

18.7.6.1 Using the PORTHSEL Bit

NOTE

This feature is supported via the SSM PORTSEL register on L49P mask set devices. Refer to Section 26.6.6 on page 26-576.

Bits [15:2] of the PIMCONFIG register are reserved for future use, and bit [1] is used to disable the EIM clock; therefore these bits should be masked out when reading this register, as shown in the following code example:

```
In File registers.h:
#define PIM_BASE             0xFC0E8000          /* Example only! */
/* Following example assumes short is 16-bits, int is 32-bits */
volatile unsigned short *PIMCONFIG = (volatile unsigned short *) (PIM_BASE+0x03C2);
         typedef union {
                  unsigned short regval;
                  struct {
                           unsigned int porthsel :1;
                           } bitval;
                  } PIMCONFIG;
         volatile PIMCONFIG *pimconfig_reg = (volatile PIMCONFIG *);

In File main.c:
         #include "registers.h"
                  :
                  :
         /* Swap port assignments... */
         /* Set only the Port H select bit! */
         pimconfig_reg->bitval.porthsel = 1;
```

18.7.6.2 Using the EIMCLKEN Bit

Add appropriate description for disabling the EIM clock.

18.7.7 Minimizing Power Consumption

As described in Chapter 2, “Signal Description,” in packages smaller than 208 pins all non-bonded out pins should be configured as an output after reset in order to avoid current draw from floating inputs. Table 2-1 on page 2-13 details which pins should be configured in this manner for each device.

Chapter 19
Analog-to-Digital Converter Module (ATD)

19.1 Overview

The Analog-to-Digital (ATD) Converter Module is a 16-channel multiplexed input successive approximation analog-to-digital converter with a programmable resolution of 8 or10 bits. MAC7100 Family devices implement up to two Analog-to-Digital (ATD) Converter modules; refer to Table 1-1 on page 1-3 for the ATD configuration of a specific device. This section describes a single module using register names and addresses that should be fully qualified in software source code (i.e., append appropriate module designations, _A or _B, to register names and specify the base address defined in Chapter 8, “Device Memory Map,” for each module).

The ATD modules may be serviced by the eDMA in order to improve overall system performance. Two DMA request channels may be used by each ATD, one channel to move conversion results out of each ATD result register, and one channel to move conversion command words to each ATD. The conversion command word is used to define parameters such as the mode of conversion, the channel to be converted and the length of the conversion. By defining a number of conversion words in the MCU system memory, it is possible to build a predefined sequence of conversions which will be executed without the intervention of the CPU. It is also possible for the CPU to write the conversion command word and to read the conversion results directly without the need to use the eDMA.

MAC7100 Family ATD modules include the ability to trigger a conversion sequence based on either an external signal, or by one of two internal signal lines, SYSTRIG0 or SYSTRIG1. In order to use an external trigger source to initiate a conversion, any one of the analog channels can be used as an off-chip trigger. The incoming trigger signal is synchronized to the system clock (which introduces a delay of 2 clock cycles before the conversion is triggered).

The internal trigger signal lines are connected to the Programmable Interrupt Timer (PIT) which provides two dedicated programmable 24-bit timers to trigger the ATD. The counter associated with the SYSTRIG*n* signals can be programed with the desired conversion trigger periods. This counter will count down from the pre-loaded value to zero at the rate defined by the system clock frequency. When the counter reaches zero the trigger signal is asserted to the ATD, then it is reloaded and the count down continues.

When it is necessary to perform synchronous conversion by both ATD modules, ATD_A and ATD_B can use the same external source, with one channel from each module assigned as the input for this trigger. Alternatively both modules can define the same internal system trigger SYSTRIG*n* as the source.

Each of the ATD modules can be independently disabled by writing to the MDIS bit in the ATDMODE register. Disabling the module will turn off the clock and shut down the analog circuits, although all of the module registers remain available to be accessed by the core.

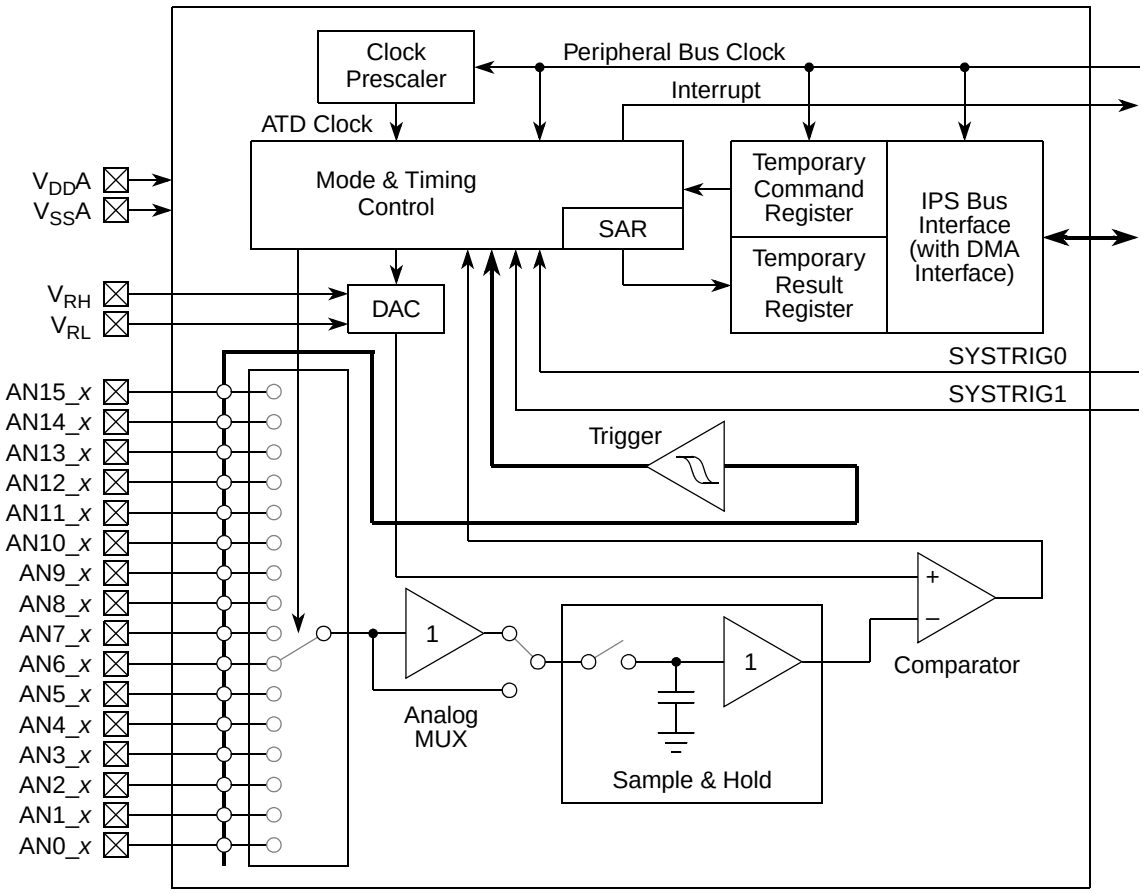


Figure 19-1. ATD Block Diagram

19.2 Features

- 8-bit/10-bit resolution
- 7 μsec, 10-bit single conversion time
- Sample buffer amplifier
- Analog input multiplexer for 16 analog input channels per ATD
- Programmable sample time
- Left/right justified, signed/unsigned result data
- Conversion completion interrupt generation
- DMA request interface
- Programmable channel sampling order
- Command words for conversion are stored in on-chip or external memory
- Continuous conversion, convert and pause, and convert on trigger sampling modes supported
- Flexible trigger control
  - — Configurable external trigger functionality on any ATD channel
  - — Two selectable on-chip triggers (PIT Timers 9 and 10)

The ATD is an 16-channel multiplexed input successive approximation analog-to-digital converter (ATD) with a programmable resolution of 8 or 10 bits (refer to *MAC7100 Microcontroller Family Hardware Specifications*, MAC7100EC, for accuracy specifications).

The ATD module may be configured to use an eDMA channel to move conversion results from the result register to system memory without the intervention by the core. The ATD controls conversions using a command word which determines aspects of the conversion such as the channel to be converted, the resolution of the conversion and the trigger source, if any, used to start the conversion. An eDMA channel may also be used to provide a stream of conversion commands to the ATD in order to queue up conversion sequences without requiring CPU intervention.

Conversions can be started either under the direct control of the ATD module or by a trigger source. The trigger source can be external to the device, using one of the analog channel pins as its input, or from one of two on-board programmable timers.

19.3 Modes of Operation

The ATD has 5 modes of operation:

- Normal mode
- Doze mode for medium power saving
- Stop mode for maximum power saving
- Disabled mode for maximum power saving
- Debug mode for error tracking

Four of these operating modes correspond to those described in Chapter 7, “Modes of Operation,” and Appendix A, “Debug Interface.” The Disabled mode is specific to the ATD module. Refer to Section 19.6.4, “ATD Operating Mode Details,” for details.

19.4 Signal Description

The ATD module utilizes 20 external signals. Note that in order to use the basic functionality of the ATD, it is only necessary to clear the MDIS bit. Since all analog inputs on this device are routed directly through the Port Integration Module (PIM) to peripherals (if applicable), it is not necessary to set up the PIM in order to use the ATD to sample channels (refer to Section 18.6.2, “Peripheral Mode,” on page 18-296). In this case, a pin can be used as both an analog input for the ATD and as digital general purpose input. It is not recommended to put any pins into general purpose output mode if they are also being used as ATD channels. The ATD signals are summarized in Table 19-1 and described in more detail in the following sub-sections. Refer to Chapter 2, “Signal Description,” for more complete descriptions.

Table 19-1. ATD Signal Properties

Signal Name [1]	Direction	Description	Reset state
$V_{DD}A$	Input	Secondary power supply	—
$V_{SS}A$	Input	Ground for secondary power supply	—
V_{RH}	Input	High reference voltage for ATD conversions	—
V_{RL}	Input	Low reference voltage for ATD conversions	—

Table 19-1. ATD Signal Properties (continued)

Signal Name [1]	Direction	Description	Reset state
AN0_*x*	Input	Analog input channel 0	—
AN1_*x*	Input	Analog input channel 1	—
AN2_*x*	Input	Analog input channel 2	—
AN3_*x*	Input	Analog input channel 3	—
AN4_*x*	Input	Analog input channel 4	—
AN5_*x*	Input	Analog input channel 5	—
AN6_*x*	Input	Analog input channel 6	—
AN7_*x*	Input	Analog input channel 7	—
AN8_*x*	Input	Analog input channel 8	—
AN9_*x*	Input	Analog input channel 9	—
AN10_*x*	Input	Analog input channel 10	—
AN11_*x*	Input	Analog input channel 11	—
AN12_*x*	Input	Analog input channel 12	—
AN13_*x*	Input	Analog input channel 13	—
AN14_*x*	Input	Analog input channel 14	—
AN15_*x*	Input	Analog input channel 15	—

1 "*x*" designates the specific module name, A or B, as listed in Table 2-2 on page 2-22.

19.4.1 AN*n*_*x*

Analog input channels, which can be configured as an external trigger for the ATD conversion.

19.4.2 V_{RH} / V_{RL}

High and low reference voltages for ATD conversions.

19.4.3 $V_{DD}A$ / $V_{SS}A$

Quiet power supply for analog section of the ATD.

19.5 Memory Map / Register Definition

Table 19-2. ATD Memory Map

ATD *x* Offset [1]	Register Description	Access
0x0000	ATD Trigger Control Register (ATDTRIGCTL)	R/W
0x0001	ATD External Trigger Channel Register (ATDETRIGCH)	R/W
0x0002	ATD Prescaler Register (ATDPRE)	R/W
0x0003	ATD Operating Mode Register (ATDMODE)	R/W

Table 19-2. ATD Memory Map (continued)

0x0004 to 0x000D	Reserved	–
0x000E	ATD Interrupt Register (ATDINT)	R/W
0x000F	ATD Flag Register (ATDFLAG)	R/W
0x0010	ATD Command Word Register (ATDCW) high	R/W
0x0012	ATD Command Word Register (ATDCW) low	R
0x0014	ATD Result Register (ATDRR) high	R
0x0016	ATD Result Register (ATDRR) low	R

[1] Register address = **ATD** base address + offset, where the base address is specified in Chapter 8, "Device Memory Map."

19.5.1 Register Descriptions

19.5.1.1 ATD Trigger Control Register (ATDTRIGCTL)

This register defines how both external and on-chip triggers are used, and is used to configure which trigger (if any) is used to start a conversion. Writing to this register will stop the current conversion and cancel any active DMA request for fetching a command word or saving a result. A new conversion can be started only by writing a command word to the command register. Note that care must be taken when writing to this register, as switching the trigger channel can cause an unintended trigger.

If a conversion is in progress when a new edge is detected, it is ignored and the external trigger overrun bit (ETO) in the ATDFLAG register is set. Edge sensitive triggers can never interrupt a conversion, and can start new conversions only as long as the CWCM bits in the new command words are set to wait for trigger (see Section 19.5.1.7, "ATD Command Word Register (ATDCW)").

If a trigger is level sensitive and the according trigger level is asserted, a conversion can start. While the conversion is running, the trigger must not be de-asserted or the conversion will immediately stop and cancel any active DMA request. No DMA request will be able to be asserted by the ATD until the processor writes a new command word to the ATDCW register.

If a conversion has finished and the CWCM bits of a new command word are set to wait for trigger, a new conversion is started if the trigger level is still asserted.

	7	6	5	4	3	2	1	0
R	0	0	0	0	TRIGSEL		TRIGP	TRIGLE
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	ATD *x* Base + 0x0000							

Figure 19-2. ATD Trigger Control Register (ATDTRIGCTL)

Table 19-3. ATDTRIGCTL Field Descriptions

Bits	Name	Description
7–4	—	Reserved.
3–2	TRIGSEL[1:0]	Trigger source select. These bits determine the trigger source. An external trigger is always synchronized internally to the peripheral bus clock, resulting in a delay of 4 peripheral bus clock cycles between the trigger and a conversion start. When using SYSTRIG0 or SYSTRIG1, there is always a delay of 2 peripheral bus clock cycles between the trigger and a conversion start. 00 No external trigger used 01 Use on-chip SYSTRIG0 10 Use on-chip SYSTRIG1 11 Use analog input channel specified by ATDETRIGCH[ETRIGCH] **Note:** Avoid sampling a channel that is used as a trigger, as this causes increased current consumption on that pin.
1	TRIGP	Trigger polarity. This bit determines the polarity of the level or edge trigger event. 0 Trigger is low level/falling edge sensitive 1 Trigger is high level/rising edge sensitive
0	TRIGLE	Trigger level/edge sensitivity. This bit determines whether the trigger is edge or level sensitive. When the conversion mode for the current command word is set to “wait for trigger” the conversion will start only if the appropriate trigger is asserted. If edge sensitivity is selected, then the value of the TRIGP bit will determine if it is rising edge sensitive or falling edge sensitive. If edge sensitivity is selected, each edge of the selected polarity can start a conversion. 0 Trigger is edge sensitive 1 Trigger is level sensitive

19.5.1.2 ATD External Trigger Channel Register (ATDETRIGCH)

This register defines which of the 16 analog input channels will be used as an external trigger if the ATDTRIGCTL register is set to use external triggering. Writing to this register will stop the current conversion and cancel any active DMA request for fetching a command word or saving a result. A new conversion can only be started by writing a command word to the command register. An external trigger is always synchronized internally to the peripheral bus clock, resulting in a delay of 4 peripheral bus clock cycles between the trigger and a conversion start. Note that care must be taken when writing to this register, as switching the trigger channel can cause an unintended trigger. Also note that the external trigger signal must operate as a 0V to 5V digital signal, as it is routed to the ATD module via the PIM digital circuitry. The PIM must be configured to place the desired pin in peripheral mode; refer to Section 18.7.1, “Using a Pin in Peripheral Mode,” for more information.

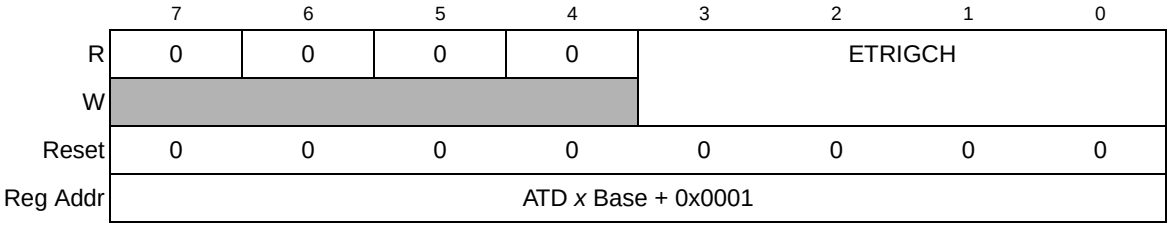


Figure 19-3. ATD External Trigger Channel Register (ATDETRIGCH)

Table 19-4. ATDETRIGCH Field Descriptions

Bits	Name	Description
7–4	—	Reserved.
3–0	ETRIGCH[3:0]	External trigger channel. These bits define which analog input channel will be used as the input trigger source. 0000 Channel 0 1000 Channel 8 0001 Channel 1 1001 Channel 9 0010 Channel 2 1010 Channel 10 0011 Channel 3 1011 Channel 11 0100 Channel 4 1100 Channel 12 0101 Channel 5 1101 Channel 13 0110 Channel 6 1110 Channel 14 0111 Channel 7 1111 Channel 15

19.5.1.3 ATD Prescaler Register (ATDPRE)

This register defines the divider for the clock used by the successive approximation state machine. As the analog circuit requires a clock (ATD clock) between 0.5 MHz and 2 MHz, the peripheral bus clock must be divided so that it falls within this range. The clock divider can be set between 2 and 128 in steps of one. Writing to this register will stop the current conversion and cancel any active DMA request for fetching a command word or saving a result. A new conversion can only be started only by writing a command word to the command register.

The following formula defines how the ATD clock is derived from the peripheral bus clock:

$$f_{ATDCLK} = f_{IPSCLK} \div \text{Prescaler Value} \qquad \textit{Eqn. 19-1}$$

For example, assume the peripheral bus clock is 48 MHz. Since the ATD clock must be between 0.5 MHz and 2.0 MHz, the above formula can be used to calculate the minimum and maximum prescaler values. The minimum clock prescaler is calculated to be 48 MHz ÷ 2 MHz = 24. The PRES value for a prescaler ratio of 24, according to the formula in Table 19-5 below, is $23_{10} = 0010111_b$. The maximum clock prescaler is 48 MHz ÷ 0.5 MHz = 96. The PRES value for a prescaler ratio of 96 is $95_{10} = 1011111_b$. Therefore, the PRES value may be selected between 23 and 95.

	7	6	5	4	3	2	1	0
R	0	PRES						
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	ATD *x* Base + 0x0002							

Figure 19-4. ATD Prescaler Register (ATDPRE)

Table 19-5. ATDPRE Field Descriptions

Bits	Name	Description
7	—	Reserved.
6–0	PRES[6:0]	The value of this field defines the division ratio used to derive the ATD module clock from the IPS bus clock. PRES can range from 0000000_b to 1111111_b to divide the module clock by 2 to 128 according to this formula: Prescaler Value = Max(2, PRES + 1) ***Eqn. 19-2*** **Note:** Executing conversions with an ATD clock frequency outside the range of 0.5 to 2.0 MHz will lead to unreliable conversion results.

19.5.1.4 ATD Operating Mode Register (ATDMODE)

This register provides control bits for four of the five ATD operating modes. Note that stop mode is a system mode and can not be enabled or disabled in the ATD. Refer to Section 19.3, “Modes of Operation,” and Section 19.6.4, “ATD Operating Mode Details,” for details pertaining to all ATD modes.

Writing to this register will stop the current conversion and cancel any active DMA request for fetching a command word or saving a result. A new conversion can only be started by writing a command word to the command register.

	7	6	5	4	3	2	1	0
R	MDIS	DOZE	DEBUG		0	0	0	0
W								
Reset	1	0	0	0	0	0	0	0
Reg Addr	ATD *x* Base + 0x0003							

Figure 19-5. ATD Operating Mode Register (ATDMODE)

Table 19-6. ATDMODE Field Descriptions

Bits	Name	Description
7	MDIS	Module disable. Setting this bit causes all ATD clocks to halt and the analog circuit to shut down (DAC, comparator, buffer etc.). 0 Module enabled 1 Module disabled: digital and analog circuits of the ATD will stop operating. Any conversion in progress will be stopped **Note:** Enabling and disabling of the ATD via the MDIS bit is intended for use primarily during the startup of the device. Care should be taken to ensure that the ATD is in an idle state, with no conversions pending, before setting or clearing the MDIS bit.
6	DOZE	Doze mode support. Setting this bit enables ATD Doze mode. 0 Do not support Doze mode: ATD will continue operating in ATD normal mode when the MCU enters Doze mode 1 Support Doze mode: ATD analog circuit is shut down when MCU enters Doze mode

Table 19-6. ATDMODE Field Descriptions (continued)

Bits	Name	Description
5–4	DEBUG[1:0]	Debug mode support. These bits allow the user to halt during or after a conversion when the system enters debug mode. This is useful for debugging real-time systems, where continuation of the conversion process after the processor core has been halted could result in abnormal system behavior. There are four different behaviors taken by the ATD when debug mode is entered: 00 Continue conversion (no influence on the ATD behavior) 01 Continue conversion with debug features active 10 Finish current conversion, then freeze 11 Freeze Immediately [1] When “continue conversion” is selected, the behavior of the ATD is not affected by the MCU entering debug mode. If ATD debug mode is enabled and the MCU enters debug mode, the analog circuit is not powered down.
3–0	—	Reserved.

[1] A frozen conversion can be resumed only by causing the system to exit debug mode (i.e., clearing the DEBUG bits result in the conversion being lost). Once the conversion is resumed, it will continue from the point at which it was frozen. The result of a frozen conversion depends on how long the conversion was stopped, as the leakage of the sample-hold capacitor and the comparator reference capacitors will have an influence on the conversion process.

19.5.1.5 ATD Interrupt Register (ATDINT)

This register contains the interrupt enables corresponding to the flags in the ATDFLAG register. If a flag is set and the interrupt enable bit for this flag is also set, an interrupt will be asserted to the INTC. This interrupt can be cleared only by reading the flag register ATDFLAG (see Section 19.5.1.6, “ATD Flag Register (ATDFLAG)”).

	7	6	5	4	3	2	1	0
R	0	0	CQFIE	CQEIE	CRLIE	ETOIE	CPIE	CCIE
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	ATD *x* Base + 0x000E							

Figure 19-6. ATD Interrupt Register (ATDINT)

Table 19-7. ATDINT Field Descriptions

Bits	Name	Description
7–6	—	Reserved.
5	CQFIE	Command queue full (CQF) interrupt enable 1 Interrupt for CQF is enabled. 0 Interrupt for CQF is disabled.
4	CQEIE	Command queue empty (CQE) interrupt enable 1 Interrupt for CQE is enabled. 0 Interrupt for CQE is disabled.

Table 19-7. ATDINT Field Descriptions (continued)

Bits	Name	Description
3	CRLIE	Conversion result lost (CRL) interrupt enable 1 Interrupt for CRL is enabled. 0 Interrupt for CRL is disabled.
2	ETOIE	External trigger overrun (ETO) interrupt enable 1 Interrupt for ETO is enabled. 0 Interrupt for ETO is disabled.
1	CPIE	Conversion paused (CP) interrupt enable 1 Interrupt for CP is enabled. 0 Interrupt for CP is disabled.
0	CCIE	Conversion complete (CC) interrupt enable 1 Interrupt for CC is enabled. 0 Interrupt for CC is disabled.

19.5.1.6 ATD Flag Register (ATDFLAG)

This register contains the flags for the ATD module. When an event occurs (e.g. a conversion finished), the appropriate flag bit is set. If the interrupt enable bit corresponding to the flag is also set, an interrupt will be asserted to the system interrupt controller.

Performing a read to this register will clear all interrupt flags when the ATD is in normal mode. In debug mode (except for DEBUG = 00), a read from this register will not clear any interrupt flag. Conversely, performing a write to this register will have no effect on the flag bits in normal mode or debug mode with DEBUG = 00. However, writing ‘1’ to a flag bit in this register in debug mode with DEBUG ≠ 00 will clear the flag. In doze mode, the flags cannot be cleared.

The CQF and CQE flags are dynamic, which means that their value can change without reading/clearing them as they represent the current state. The flags CRL, ETO, CP and CC remain set when they are set by the ATD. These flags must be cleared by the processor.

	7	6	5	4	3	2	1	0
R	0	0	CQF	CQE	CRL	ETO	CP	CC
W					See Note 1			
Reset	0	0	0	1	0	0	0	0
Reg Addr	ATD *x* Base + 0x000F							

1. Writes to these bits are enabled only in debug mode.

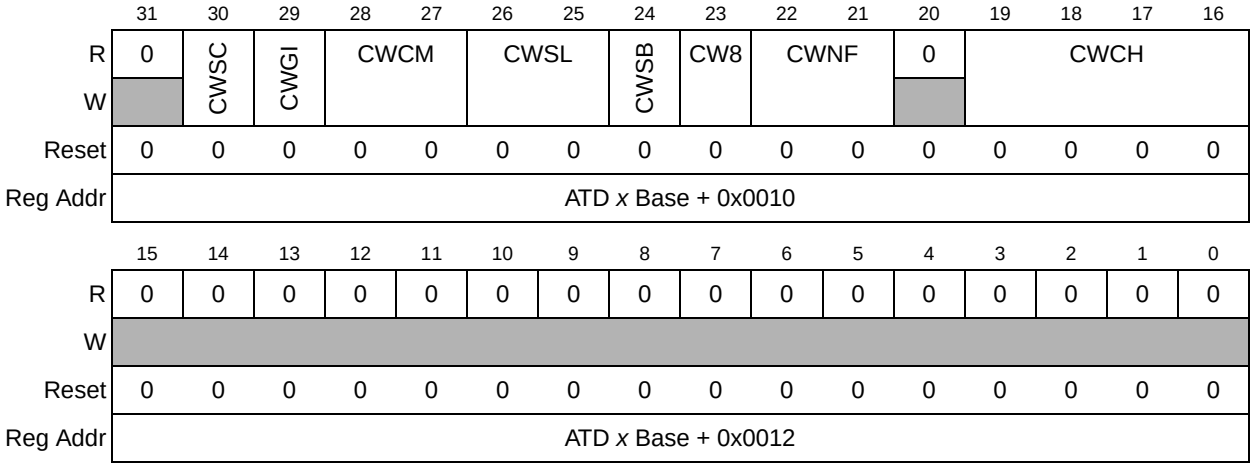
Figure 19-7. ATD Flag Register (ATDFLAG)

Table 19-8. ATDFLAG Field Descriptions

Bits	Name	Description
7–6	—	Reserved.
5	CQF	Command queue full flag. This bit indicates that the queue for command words is full. When CQF is set, any write access to the ATDCW register will overwrite the command that was previously in the command register, and the current conversion is not affected. 0 Command queue not full 1 Command queue is full
4	CQE	Command queue empty flag. This bit indicates that the queue for command words is empty. It also indicates that no conversion is currently running. 0 Command queue not empty 1 Command queue is empty
3	CRL	Conversion result lost flag. This bit indicates that a conversion result was overwritten before it could be read by the processor or stored in memory. 0 Conversion results have always been stored/read in time 1 A conversion result was overwritten **Note:** The CRL bit is useful to detect possible problems with the availability of the eDMA channels. Therefore it is useful to enable this interrupt (CRLIE = 1) during software debugging.
2	ETO	External trigger overrun flag. This flag indicates if additional edges have been recognized while a conversion was executing, if the conversion was started by a trigger event (CWCM= 10, wait for trigger). 0 No additional edges have been detected 1 One or more additional edges have been detected **Note:** ETO only recognizes additional edges. If a trigger is level sensitive, ETO is never set.
1	CP	Conversion paused flag. This bit indicates whether a conversion with CWCM=01 (convert then pause) has finished or is still being executed. When such a conversion finishes no new command word is executed until a new command word is written to the ATDCW register. 0 A conversion with CWCM=01 is being executed 1 A conversion with CWCM=01 finished
0	CC	Conversion complete flag. When a conversion finishes, this flag will be set (and remain set) if the CWGI bit of the command word is also set. 0 No conversion finished where CWGI=1 1 A conversion finished where CWGI=1

19.5.1.7 ATD Command Word Register (ATDCW)

This register contains the command word for the next conversion to be executed after the current one.



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	CWSC	CWGI	CWCM		CWSL		CWSB	CW8	CWNF		0	CWCH			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	ATD *x* Base + 0x0010															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	ATD *x* Base + 0x0012															

Figure 19-8. ATD Command Word Register (ATDCW)

Table 19-9. ATDCW Field Descriptions

Bits	Name	Description
31	—	Reserved.
30	CWSC	Special channel. When this bit is set the ATD samples an internal reference voltage rather than an external signal. The available references are listed below. Refer to Section 19.7.11, “Measuring Internal Reference Voltages,” for more information.
29	CWGI	Generate Interrupt. This bit defines whether the CC flag bit in the ATDFLAG register should be set after a conversion finishes. 0 Do not set CC flag 1 Set CC flag

CWSC	CWCH[3:0]	Analog Input Channel
0	xxxx	Channel selected by CWCH
1	00xx	Reserved
1	0100	V_{RH}
1	0101	V_{RL}
1	0110	$(V_{RH} + V_{RL}) \div 2$
1	0111	Reserved
1	1xxx	Reserved

Table 19-9. ATDCW Field Descriptions (continued)

Bits	Name	Description
28–27	CWCM	Conversion mode. These bits define the manner in which channels are sampled. 00 Conversion reset: stop any current conversion. After the command word has been written to the command register, any conversion is stopped immediately. A DMA request to fetch a command word is cancelled, while a DMA request to save a result remains asserted. To start a new conversion, the processor must write a new command word to the ATDCW register. When CWCM=00 the values of the remaining bits in the command word are ignored. 01 Convert and pause: stop after the conversion finishes and set the CP bit in the ATDFLAG register. No DMA request to fetch a command word will be asserted. Only a DMA request for saving the result in the memory will be asserted. To start a new conversion the processor must write a new command word to the command register. 10 Wait for trigger: start converting only after a valid trigger has been asserted. Refer to Section 19.5.1.1, “ATD Trigger Control Register (ATDTRIGCTL),” and Section 19.5.1.2, “ATD External Trigger Channel Register (ATDETRIGCH),” for trigger source selection. 11 Convert continuously: start a new conversion after the current conversion has finished. The start/behavior of the new conversion depends on the new command word. If no new command word is available, the ATD will continue to assert a DMA request. When a new command word has been received from the eDMA or the processor, a new conversion will start according to the new command word.
26–25	CWSL	Sample length. These bits select the length of the second phase of the sample time measured in ATD clock cycles. 00 2 ATD clock periods 01 4 ATD clock periods 10 8 ATD clock periods 11 16 ATD clock periods **Note:** The ATD conversion clock period is a function of the prescaler value (PRES). The sample time consists of two phases. The first phase is two ATD clock cycles long (when CWSB = 1) and transfers the sample quickly via the buffer amplifier onto the ATD storage node. The second phase attaches the external analog signal directly to the storage node for final charging and high accuracy.
24	CWSB	Sample amplifier bypass. This bit determines whether the sample amplifier is bypassed during the sample phase. Bypassing saves 2 ATD clock cycles. 0 Sample amplifier is used 1 Sample amplifier is bypassed **Note:** Bypassing the sample amplifier is not recommended, as accuracy is significantly diminished and device may be overstressed. If absolutely necessary, it should be used only for low impedance sources. **Note:** Due to internal RC time constants, the use of a sample time of two ATD clock cycles when the sample amplifier is bypassed is not recommended. A minimum of four ATD clock cycles sample time must be allowed for all conversions of external channels. Conversions of the internal voltages V_{RL}, V_{RH}, or $(V_{RH} + V_{RL}) \div 2$ require only two ATD clock cycles sample time. The sample amplifier cannot be used in this case (the CWSC bit is ignored).

Table 19-9. ATDCW Field Descriptions (continued)

Bits	Name	Description
23	CW8	8-bit resolution. This bit defines whether a conversion result will have a resolution of 8 or 10 bits. When 8-bit resolution is set, the third phase of a conversion will only last 8 cycles instead of 10 cycles for 10-bit resolution. 0 Conversion resolution is 10 bits 1 Conversion resolution is 8 bits
22–21	CWNF	Numeric format. These bits determine in which of the four available numeric formats the conversion result is stored. For detailed information on the numeric formats see Figure 19-10 to Figure 19-13 and Figure 19-11. 00 Right justified, unsigned 01 Right justified, signed 10 Left justified, unsigned 11 Left justified, unsigned
20	—	Reserved.
19–16	CWCH	Channel select. These bits determine the input channel that will be sampled when the CWSC bit is cleared. 0000 Channel 0 1000 Channel 8 0001 Channel 1 1001 Channel 9 0010 Channel 2 1010 Channel 10 0011 Channel 3 1011 Channel 11 0100 Channel 4 1100 Channel 12 0101 Channel 5 1101 Channel 13 0110 Channel 6 1110 Channel 14 0111 Channel 7 1111 Channel 15
15–0	—	Reserved.

19.5.1.8 ATD Result Register (ATDRR)

This read-only register contains the result for the last conversion that has been executed. After a conversion, the ATD will assert a DMA request to transfer the result to on-chip or external memory. Alternatively, the result can be read by the processor. When the processor reads the result, the DMA request (for saving the result in the memory) will be de-asserted. After either the processor or the eDMA has read the result, a new result can be stored in the result register.

If a result has not been read by the processor or eDMA before a new conversion result needs to be saved, a result loss will occur; the new conversion result will overwrite the previous result and the CRL (conversion result loss) bit in the flag register (ATDFLAG) will be set to ‘1’. Following the result loss, the ATD will continue executing conversions.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	RRSC	RRGI	RRCM		RRSL		RRSB	RR8	RRNF		0	RRCH			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	ATD *x* Base + 0x0014															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RRCR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	ATD *x* Base + 0x0016															

Figure 19-9. ATD Result Register (ATDRR)

Table 19-10. ATDRR Field Descriptions

Bits	Name	Description
31–16	RRCH, RRNF, RR8, RRSB, RRSL, RRCM, RRGI, RRSC	Command word bits — After a conversion, ATDCW[31:16] are copied to ATDRR[31:16] without any changes, to assist identification of the result. Each of the ATDRR field names is prefixed with RR rather than the CW. For example, the RRCH bits have the same definition as the CWCH bits. Refer to Section Table 19-9., “ATDCW Field Descriptions,” for the function of each field.
15–0	RRCR	Conversion Result — Available to be read only after a conversion has finished. This field contains the result of the conversion in the numeric format specified by the CWNF bit of the command word. For detailed information about the numeric formats see Figure 19-10 to Figure 19-13 and Figure 19-11.

The figures below show the 8-bit and 10-bit right-justified unsigned conversion results in the RRCR. In the table, ‘Bit*n*’ indicates the bits generated by the successive approximation state machine.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10-bit	0	0	0	0	0	0	Bit9 MSB	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 LSB
8-bit	0	0	0	0	0	0	0	0	Bit9 MSB	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2 LSB

Figure 19-10. ATD Right Justified Unsigned 8/10-Bit Result Format

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10-bit	Bit9 MSB	Bit9 MSB	Bit9 MSB	Bit9 MSB	Bit9 MSB	Bit9 MSB	Bit9 MSB	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 LSB
8-bit	Bit9 MSB	Bit9 MSB	Bit9 MSB	Bit9 MSB	Bit9 MSB	Bit9 MSB	Bit9 MSB	Bit9 MSB	Bit9 MSB	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2 LSB

Figure 19-11. ATD Right Justified Signed 8/10-Bit Result Format

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10-bit	Bit9 MSB	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 LSB	0	0	0	0	0	0
8-bit	Bit9 MSB	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2 LSB	0	0	0	0	0	0	0	0

Figure 19-12. ATD Left Justified Unsigned 8/10-Bit Result Format

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10-bit	Bit9 MSB	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 LSB	0	0	0	0	0	0
8-bit	Bit9 MSB	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2 LSB	0	0	0	0	0	0	0	0

Figure 19-13. ATD Left Justified Signed 8/10-Bit Result Format

The following table explains in more detail how conversion results are returned by giving examples with bit values. The conversion resolution has been chosen as 8-bit. The examples use two different reference voltage configurations:

- Reference voltage #1: V_{RH} = +5.120 V, V_{RL} = 0.000 V; 1 LSB = 20mV
- Reference voltage #2: V_{RH} = +2.560 V, V_{RL} = –2.560 V; 1 LSB = 20mV

The plain text bits represent the bits used to fill an 8-bit value to a 16-bit value. The bits shown in **bold** represent the bits determined by the successive approximation algorithm.

Table 19-11. Numeric Examples of Result Values

Input Voltage #1	Input Voltage #2	Right Justified Unsigned 8-bit	Right Justified Signed 8-bit	Left Justified Unsigned 8-bit	Left Justified Signed 8-bit
5.090..5.120	2.530..2.560	00000000 **11111111**	00000000 **01111111**	**11111111** 00000000	**01111111** 00000000
5.070..5.090	2.510..2.530	00000000 **11111110**	00000000 **01111110**	**11111110** 00000000	**01111110** 00000000
5.050..5.070	2.490..2.510	00000000 **11111101**	00000000 **01111101**	**11111101** 00000000	**01111101** 00000000
5.030..5.050	2.470..2.490	00000000 **11111100**	00000000 **01111100**	**11111100** 00000000	**01111100** 00000000
⋮	⋮	⋮	⋮	⋮	⋮
2.590..2.610	0.030..0.050	00000000 **10000010**	00000000 **00000010**	**10000010** 00000000	**00000010** 00000000
2.570..2.590	0.010..0.030	00000000 **10000001**	00000000 **00000001**	**10000001** 00000000	**00000001** 00000000
2.550..2.570	-0.010..0.010	00000000 **10000000**	00000000 **00000000**	**10000000** 00000000	**00000000** 00000000
2.530..2.550	-0.030..-0.010	00000000 **01111111**	11111111 **11111111**	**01111111** 00000000	**11111111** 00000000
2.510..2.530	-0.050..-0.030	00000000 **01111110**	11111111 **11111110**	**01111110** 00000000	**11111110** 00000000
2.490..2.510	-0.070..-0.050	00000000 **01111101**	11111111 **11111101**	**01111101** 00000000	**11111101** 00000000
⋮	⋮	⋮	⋮	⋮	⋮
0.050..0.070	-2.510..-2.490	00000000 **00000011**	11111111 **10000011**	**00000011** 00000000	**10000011** 00000000
0.030..0.050	-2.530..-2.510	00000000 **00000010**	11111111 **10000010**	**00000010** 00000000	**10000010** 00000000
0.010..0.030	-2.550..-2.530	00000000 **00000001**	11111111 **10000001**	**00000001** 00000000	**10000001** 00000000
0.000..0.010	-2.560..-2.550	00000000 **00000000**	11111111 **10000000**	**00000000** 00000000	**10000000** 00000000

19.6 Functional Description

19.6.1 General

The ATD performs analog-to-digital conversions using a successive approximation architecture. It functions by comparing the stored analog sample potential with a series of digitally generated analog potentials. By following a binary search algorithm, the ATD identifies the approximating potential that is nearest to the sampled potential. The resolution is selectable as either 8 bits or 10 bits.

NOTE

Only analog input signals within the potential range of V_{RL} to V_{RH} (ATD reference potentials) will result in a non-railed digital output code.

19.6.2 Analog Sub-Module

The analog sub-module contains the analog electronics required to perform a single conversion step, including a multiplexor, comparator and a reference DAC. A separate power supply, $V_{DD}A$ / $V_{SS}A$, allow isolation of other MCU circuitry noise from the analog module.

NOTE

When the ATD is initially powered up, the analog circuit requires a recovery time t_{REC} to elapse before a reliable conversion result can be obtained. Any conversion started in that time should be ignored. [1]

19.6.2.1 Analog Input Multiplexer

The analog input multiplexer connects one of the 16 analog input channels to the sample amplifier and the Sample and Hold Machine.

19.6.2.2 Sample Buffer Amplifier

The sample buffer amplifier is used to buffer the analog input signal (from the analog input multiplexer) so that the storage node can be quickly charged to the sample potential. This amplifier can be bypassed, in which case the analog input signal is connected directly to the Sample and Hold Machine. Bypassing the amplifier is not recommended, as it significantly reduces accuracy and may overstress the device, and should be used only with a low impedance source if absolutely necessary.

Due to internal RC time constants, it is not recommended to use a sample time of two ATD clock cycles when the sample amplifier is bypassed. A minimum time of four ATD clock cycle samples must be allowed for all conversions of external channels. Internal conversions of V_{RL}, V_{RH}, or $(V_{RH} + V_{RL}) \div 2$ require only two ATD clock cycles of sample time.

1. Refer to *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC) for recovery time t_{REC}.

19.6.2.3 Sample and Hold Machine

The Sample and Hold Machine accepts one analog signal either from the analog input multiplexer (bypassing sample buffer amplifier) or from the sample buffer amplifier. It stores the signal level as a capacitor charge on a storage node.

The sample process uses a two stage approach. In the first stage, the sample amplifier is used to quickly charge the storage node. The second stage connects the input directly to the storage node to complete the sample for higher accuracy.

19.6.2.4 DAC

The DAC generates a reference voltage that is compared with the sampled voltage. By using a successive approximation scheme, the reference voltage from the DAC is adjusted step-wise to the sampled voltage until a best fit is determined.

19.6.2.5 Comparator

The comparator compares the signal from the Sample and Hold Machine with the reference voltage generated by the DAC. After each approximation step, the output signal of the comparator is saved in the SAR (successive approximation register).

19.6.2.6 Schmitt Trigger

One of the analog inputs can be used as a trigger input (to start a conversion). A Schmitt-Trigger is used to improve system stability.

19.6.3 Digital Sub-Module

The digital sub-module controls the analog sub-module. It contains the conversion control state machine, the result of each step for the successive approximation (SAR) and it handles bus accesses. The digital sub-module also selects the trigger input channel if an external trigger source is used to start a conversion.

19.6.3.1 Mode / Timing Control

The Timing Control sub-module handles conversions by selecting the channel to be converted, when the conversion should start, etc. As described in Section 19.6.4, “ATD Operating Mode Details,” this sub-module affects the behavior of the converter according to the ATD operating mode. The result from the comparator is analyzed and stored in the SAR (successive approximation register). The generation of the reference voltage in the DAC, transfer of the command/result, and interrupt generation is also handled by this sub-module.

19.6.3.2 Clock Prescaler

The speed at which the analog circuitry operates is 0.5 MHz to 2 MHz. The higher peripheral bus clock is therefore divided down by the Clock Prescaler to fall within this range.

19.6.3.3 IPS Bus Interface

The IPS Bus interface allows read/write transfers of command words and results by the processor or eDMA. The current command word and the current conversion result are also stored in the bus interface.

19.6.3.4 SYSTRIG0, SYSTRIG1, External Trigger Input

Triggers can be used to initiate conversions that allow the synchronization of conversions to the external environment, rather than relying on software to signal to the ATD module when conversions should take place. The trigger source can be one of the analog inputs or the on-chip SYSTRIG0 / SYSTRIG1 signals (refer to Chapter 25, “Periodic Interrupt Timer Module (PIT)”).

The trigger is programmable to be edge or level sensitive with polarity control. Refer to Section 19.5.1.1, “ATD Trigger Control Register (ATDTRIGCTL),” and Section 19.5.1.2, “ATD External Trigger Channel Register (ATDETRIGCH),” for more information.

19.6.4 ATD Operating Mode Details

The ATD can be configured to operate in five distinct modes. The ATD must be operating in either normal or debug modes in order to execute a conversion. The Disabled, Doze and Stop modes provide reduced power consumption as needed.

19.6.4.1 ATD Normal Mode

To perform conversions, the ATD must be operating in normal mode or one of the debug modes described below. If the MCU is in Run mode, the ATD is in normal mode unless specifically disabled as described in Section 19.6.4.3, “ATD Disabled Mode.”

19.6.4.2 ATD Debug Mode

If the MCU enters debug mode, the contents of the ATDMODE register determine whether the ATD continues to operate in normal mode or enters debug mode. If DEBUG = 00 the ATD will continue in normal mode. If DEBUG ≠ 00 the ATD will enter debug mode. In this mode, all clocks are running, all registers are accessible and conversions can be executed; thus, this mode is not intended for power saving, but for use during software debug. In debug mode a conversion can be frozen either immediately upon mode entry or at the end of the conversion execution.

When the MCU enters debug mode, the values of the DEBUG bits in the ATDMODE register determine the ATD mode:

Table 19-12. ATD Debug Modes

ATDMODE DEBUG Bits	ATD Behavior
00	Continue conversion — This mode has no influence on the behavior of the ATD, as the module continues to operate in normal mode.
01	Continue conversion with debug features active — The behavior of the ATD is the same as “continue conversion” with the exception that the ATDFLAG register interrupt flags CRL, ETO, CP and CC are not cleared when read.

Table 19-12. ATD Debug Modes (continued)

ATDMODE DEBUG Bits	ATD Behavior
01	Finish current conversion, then freeze — The ATD will stop converting after the current conversion has finished. No new conversion will be started until debug mode is exited or the processor writes to ATDCW. If no conversion is running when debug mode is entered, the ATD will freeze immediately. The ATDFLAG register interrupt flags CRL, ETO, CP and CC are not cleared when read.
11	Freeze immediately — The ATD will immediately halt the conversion that is currently running.

To exit ATD debug mode, the MCU must exit debug mode. Simply clearing the debug bits in the ATDMODE register will result in the conversion being lost. After leaving debug mode, any conversion that was frozen upon debug mode entry will resume at the point where it was frozen. The accuracy of the conversion result depends on the duration for which the debug mode has been set: the longer the conversion has been frozen, the greater the leakage from the sample-hold capacitor and the comparator reference capacitors.

19.6.4.3 ATD Disabled Mode

A mode that is independent of the MCU mode is the ATD disabled mode. At any time the MDIS bit in the ATDMODE register may be set to disable the ATD. This mode causes all ATD clocks to halt and the analog circuit to shut down (DAC, comparator, buffer etc.). All registers remain accessible. This mode causes the ATD to draw minimal power while other peripheral modules may continue to operate normally.

The ADT Disabled mode should not be used to temporarily disable the ATD, and is recommended for use only when the ATD is not used in the application. To temporarily disable the ATD, Stop mode should be used as described below.

19.6.4.4 ATD Doze Mode

If the MCU enters Doze mode, the contents of the ATDMODE register determine whether the ATD continues to operate in normal mode or enters Doze mode. If the DOZE bit is cleared, the ATD will remain in normal mode. If the DOZE bit is set, the ATD will enter Doze mode. ATD Doze mode shuts down the analog circuit (DAC, comparator, buffer etc.) but leaves the registers accessible, thus offering power savings over operation in normal mode.

Entering Doze mode terminates any conversion being performed. All DMA requests are cancelled and the ATD registers remain accessible.

When the MCU exits Doze mode or the DOZE bit is cleared, the ATD clock is turned on again. Any conversion stopped upon ATD Doze entry will remain stopped. To start a new conversion, the processor must write a new command word to the ATDCW register.

When recovering from Doze mode a minimum recovery time, t_{REC}, period must elapse before initiating a new ATD conversion sequence. [1] All conversions starting/ending within the recovery time will deliver unreliable results.

1. Refer to *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC) for recovery time t_{REC}

19.6.4.5 ATD Stop Mode

If the MCU enters STOP mode, all clocks stop and therefore all modules stop. This mode causes all ATD clocks to halt and the analog circuit to shut down (DAC, comparator, buffer etc.), and thus offers maximum power saving.

Entering Stop mode terminates any conversion being performed. All DMA requests are cancelled and the ATD registers cannot be accessed.

After exiting Stop mode, the clocks are turned on again. Any conversion stopped upon ATD Stop entry will remain stopped. To start a new conversion, the processor must write a new command word to the ATDCW register.

When recovering from Stop mode (the CPU exits Stop mode) a minimum recovery time, t_{REC}, period must elapse before initiating a new ATD conversion sequence. [1] All conversions starting/ending within the recovery time will deliver unreliable results. There is no recovery time required before accessing the module registers, therefore they can be used immediately after exiting Stop mode.

19.6.5 Conversion process

To perform a conversion, two elements must be provided: a command word and a trigger. The command word contains the information on how a conversion will start, how it will run and what is to be done after a conversion is finished. The trigger indicates when a conversion will start. The following paragraphs describe various ways in which these two elements may be provided.

After a system reset, the ATD is disabled so the module must be enabled at the system level by setting the ATDMODE register MDIS bit. Once enabled, the ATD will be in normal mode, ready to receive command words before starting any conversion.

To start a conversion, a command word must be written to the ATDCW register. This command word includes the CWCM bits, which tell the ATD how to perform the conversion and what action to take after the conversion has been completed. There are 4 different combinations, but only 3 ways for the ATD to react:

1. CWCM= ‘00’ (conversion reset): Writing a command word into the command register ATDCW with CWCM set to 00 immediately stops any conversion that is currently active. This will put the ATD into the same state as after a system Reset.
2. CWCM= ‘01’ (convert and pause): The ATD starts a conversion immediately after the command word has been written to the command register (ATDCW). The write to ATDCW register acts as the conversion trigger.
3. CWCM= ‘10’ (wait for trigger): The ATD will start a conversion only when an external trigger has been asserted as described below.
4. CWCM= ‘11’ (convert continuously): The ATD starts a conversion immediately after the command word has been written to the ATDCW register. The write to the ATDCW register acts as the conversion trigger.

On completion of a conversion the value of the CWCM bits will determine what action the ATD will take:

1. CWCM= ‘00’ (conversion reset): Not possible, as writing a command word with this value immediately stops any conversion.
2. CWCM= ‘01’ (convert and pause): The ATD will wait for a new command word. A new conversion will not be started.
3. CWCM= ‘10’ (wait for trigger): The ATD will start a new conversion if a new command word is available. If a new command word is ready a conversion may start immediately (according to the new command word). If not, the ATD will continue waiting for a new command.
4. CWCM= ‘11’ (convert continuously): The ATD will wait for a new command word in the ATDCW register to start another conversion.

When a conversion is finished, the ATD always attempts to have the result of the conversion stored to memory by asserting a DMA request. This action is independent of the value of CWCM, and executes in parallel with the conversion process (see Figure 19-15a). Once the ATDRR register has been read (by either the CPU or the eDMA), the DMA request is de-asserted.

While a conversion is running, the ATD will attempt to have a new command word fetched by asserting a DMA request. When a new command word is written to the ATDCW register (by either the CPU or the eDMA) the DMA request is de-asserted. This scheme allows a new command word to be available before the current conversion finishes.

The following flow charts illustrate the conversion process in more detail. As previously mentioned, fetching commands (see **Figure 19-15**b), saving results (see Figure 19-15a) and performing conversions (see Figure 19-14) are autonomous processes that execute in parallel, and are therefore illustrated by three separate diagrams.

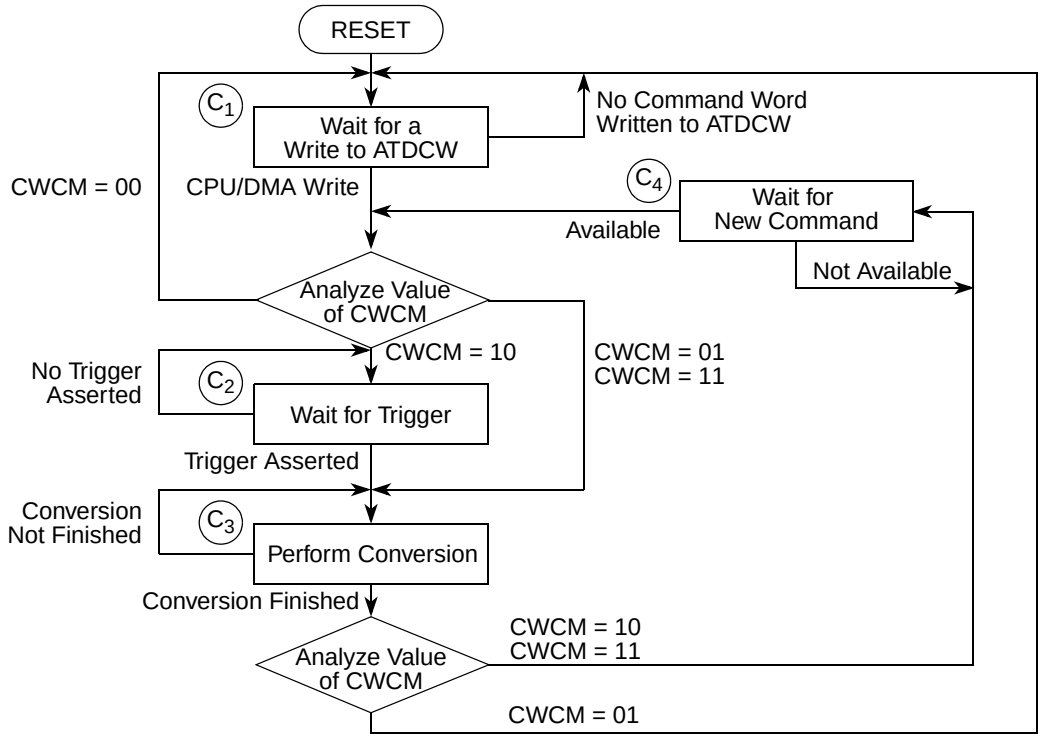


Figure 19-14. ATD Command Processing Flow Diagram

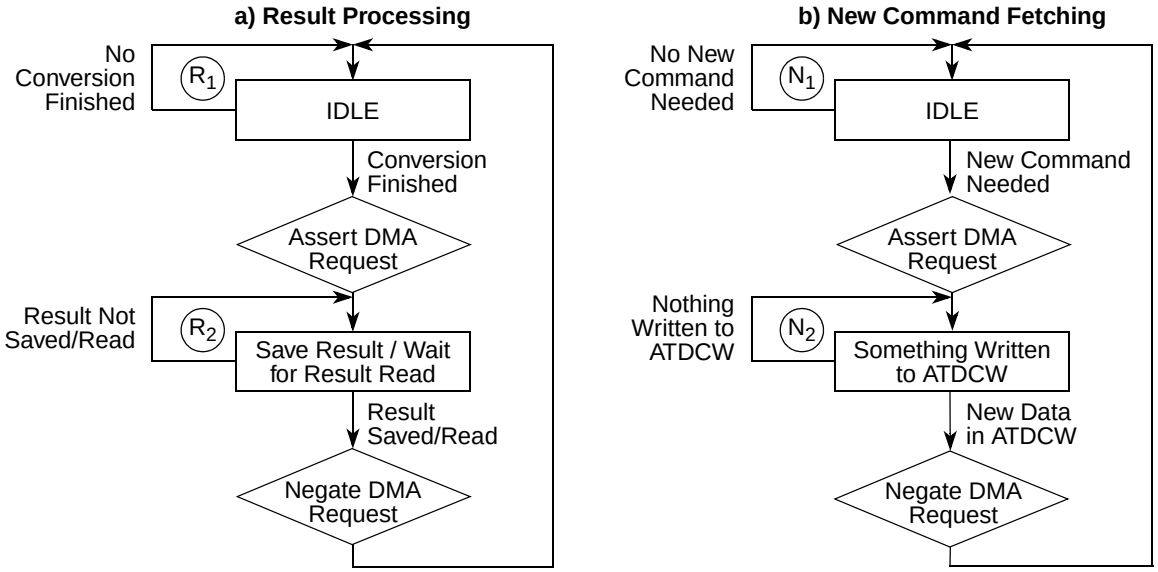


Figure 19-15. ATD Result Processing and Command Fetching Flow Diagrams

In order to enhance performance, the ATD utilizes a command word queue such that the command being used to control the current conversion is stored in a latched register (which is not readable/writable), which allows a new command word to be fetched and stored into the ATDCW while the conversion is executed.

When a conversion finishes, the current command word is transferred to the ATDRR register, where the conversion result (located in the SAR) is also stored. At the same time, the new command is transferred from the ATDCW register to the current command register.

The ATD module cannot distinguish between an access by the eDMA and an access by the CPU. Only one of the bus masters should access to either the ATDCW or the ATDRR as part of the ATD driver routines.

The conversion state machine (CSM) always acts on the contents of the current command. Figure 19-16 illustrates the registers used by the conversion process:

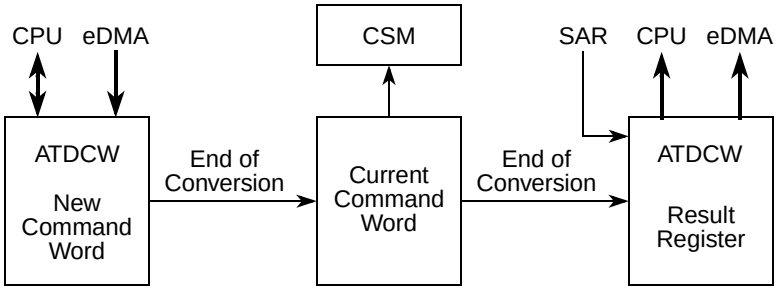


Figure 19-16. ATD Command Word Buffering

19.6.6 Reset

At reset the ATD is in the disabled state. The reset state of individual bits is listed in Section 19.5.1, “Register Descriptions.”

19.6.7 Interrupts

The following table provides an overview of the 6 possible ATD interrupt sources. All interrupt flags are located in the ATDFLAG register, with their enable bits located in the ATDINT register.

Table 19-13. ATD Interrupt Sources

INT#	Interrupt Source	Interrupt flag	Enable Bit	Other enables
1	Conversion complete	CC	CCIE	CWGI
2	Conversion paused	CP	CPIE	–
3	External trigger overrun	ETO	ETOIE	–
4	Conversion result lost	CRL	CRLIE	–
5	Command queue empty	CQE	CQEIE	–
6	Command queue full	CQF	CQFIE	–

Interrupt #1 is the only source that has an additional enable bit. Both the CWGI bit in the command word and the CCIE bit in the ATDINT register must be set in order for the CC flag to generate an interrupt on the completion of a conversion.

With the ATD module operating in normal mode (DEBUG = 00), the CC, CP, ETO and CRL flags will be cleared when the ATDFLAG register is read. The CQF and CQE flags will not be cleared as these represent the current state of the command word.

- Interrupt #1: generated when a conversion finishes and CWGI = ‘1’
- Interrupt #2: generated when a conversion finishes and ATDCW[CWCM] = ‘01’ (convert then pause)
- Interrupt #3: generated when the command word currently executing has CWCM = ‘10’ (wait for trigger), the trigger is edge-sensitive and new edge is detected prior to conversion completion
- Interrupt #4: generated when a conversion result was not read by the processor or eDMA before the next conversion completed
- Interrupt #5: generated when the current conversion completes and the ATDCW register does not contain a new command word
- Interrupt #6: generated when there is a conversion running and ATDCW also contains a command word

All interrupts share a single interrupt signal, therefore the flag register must be read to identify the source of an ATD interrupt.

19.7 Initialization / Application Information

In the following paragraphs, the initialization of the ATD module is described. Some conversion examples are also explained. The focus in these examples is on the CWCM bits.

19.7.1 ATD Initialization Sequence

After reset, the ATD module is disabled. To enable the ATD, clear the MDIS bit in the ATDMODE register. Once enabled, the ATD will wait, ready to receive command words before starting any conversion. Due to the recovery time, t_{REC}, of the analog circuit, no conversion should be started before t_{REC} has elapsed. [1]

Note that in order to use the basic functionality of the ATD, it is only necessary to clear the MDIS bit. Since all analog inputs on this device are routed directly through the Port Integration Module (PIM) to peripherals (if applicable), it is not necessary to set up the PIM in order to use the ATD to sample channels (refer to Section 18.6.2, “Peripheral Mode,” on page 18-296). In this case, a pin can be used as both an analog input for the ATD and as digital general purpose input. It is not recommended to put any pins into general purpose output mode if they are also being used as ATD channels.

19.7.2 ATD Example 1 — Simple Conversion

For this example we assume that the ATD has been enabled and t_{REC} has elapsed. [2] At a certain point, the processor writes a command word to the ATDCW register. It is assumed that CWCM = convert and pause. The CSM will immediately go to state C_3 (Figure 19-14) and execute the conversion. When the conversion is finished, the CSM will store the result in the ATDRR register and go back to C_1. As the conversion method was “convert and pause” and the conversion has finished, the CP bit in the ATDFLAG register will be set.

On completion of the conversion, the result stored in the ATDRR register must be saved. The Result State Machine (RSM) will transition from R_1 to R_2 (Figure 19-15a) and will assert a DMA request to store the

1. Refer to *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC) for recovery time t_{REC}
2. Refer to *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC) for recovery time t_{REC}

result into memory. During the time the DMA request is asserted, the CPU can read the result register. If it does so, the DMA request is cancelled.

If the CPU does not read the result, the DMA request remains asserted until the result is stored in memory. After the result has been read/stored, the RSM goes back to the idle state R_1.

At the end of this single conversion, the CP flag in the ATDFLAG register is set as the conversion method was “convert and pause”. If the CPIE bit is set, this generates an interrupt.

19.7.3 ATD Example 2 — Three Consecutive Conversions

For this example three conversions will be performed in a row with the CSM assumed to be starting from C_1 (Figure 19-14). The three command words have the following CWCM bit values:

1. CWCM= “11” (convert continuously)
2. CWCM= “11” (convert continuously)
3. CWCM= “01” (convert and pause)

Writing the first command to the ATDCW register will bring the CSM from state C_1 to C_3 and execute the conversion. While a conversion is in progress, the ATD will assert a DMA request for a new command word. The state machine that fetches the new command (the NSM), transitions from state N_1 to N_2. The command word can be written to the ATDCW register by either the eDMA or the processor without affecting the current conversion. It is assumed that the eDMA delivers the new command word and therefore, the NSM goes back from state N_2 to N_1.

When the first conversion has been completed (the result has been stored in ATDRR) the CSM will transition into state C_4. As the second command word is now available, the current command word will be loaded with the new command from the ATDCW and the CSM will move into state C_3 enabling the next conversion to be executed. If no new command word is available, the CSM will stay in state C_4 until the command word has been received.

At the end of the first conversion, the RSM tries to have the result from ATDRR stored into memory (by asserting a DMA request) or waits for the CPU to read the result. For this example, it is assumed that the result has been successfully stored in memory.

When the second conversion finishes, the CSM will again go from C_3 to C_4 and wait for a new command, while the second result is stored by the RSM. If the third command is available, the CSM will go to C_3 and execute the conversion.

If the result of the first conversion was not saved before the second conversion was completed, a conversion result loss would occur: the result of the first conversion is overwritten by the result of the second conversion. To indicate that a result has been lost, the CRL flag in the ATDFLAG register will be set. For this example, it is assumed that all results are saved in time.

After the third conversion has finished, the CSM will move from state C_3 to state C_1. This is because the CWCM bits of the last command word are set to “convert and pause”. The ATD will not start any further conversions, but will try to save the result of the third conversion.

19.7.4 ATD Example 3 — Interrupted Continuous Conversion

This example (which is similar to the second one) will show how a continuous conversion can be interrupted.

For a continuous conversion, the CWCM bits in the command words must be either set to ‘10’ (wait for trigger) or ‘11’ (convert continuously). This example assumes that all but the last command word have CWCM set to ‘11’.

- All but the last command word: CWCM = 11 (continuous conversion)
- Last command word: CWCM = 00 (conversion reset)

When the first conversion is started, according to example 2 (CSM goes from C_1 to C_3 and finally to C_4) the ATD will keep switching between the states C_3 and C_4 as long as new command words are written to ATDCW and they have CWCM=11 set. The result of each conversion is saved in the same way as described in example 2.

When the last command word with CWCM = 00 is written to the ATDCW register, the CSM will return to state C_1 and stop converting immediately after the write access. This is independent of the current CSM state. This behavior has not been shown in Figure 19-14 in order to simplify the figure.

If a command word with CWCM = 00 is written when the NSM is in state N_2 (a new command word is requested) and therefore a DMA request is asserted, this DMA request is cancelled and the NSM will go to state N_1.

19.7.5 ATD Example 4 — Edge Triggered Conversion

This example shows how a conversion can be started by a trigger other than a write to the ATDCW. The example uses a rising-edge sensitive trigger from analog channel 7. To set up a trigger, the following must be done:

- ATDTRIGCTL[TRIGSEL] = ‘11’ (use analog input channel as trigger)
- ATDTRIGCTL[TRIGP] = ‘1’ (trigger is rising-edge sensitive)
- ATDTRIGCTL[TRIGLE] = ‘0’ (trigger is edge sensitive)
- ATDETRIGCH[ETRIGCH] = ‘0111’ (analog channel 7 is trigger)

Each write to either the ATDTRIGCTL or ATDETRIGCH register will stop any conversion that may be executing. For this example it is assumed that both registers have been configured previously and no conversions were being executed.

When a command word with CWCM = 10 (wait for trigger) is written to the ATDCW register, the CSM will move from state C1 to C2, where it will wait for a rising edge on the analog input channel 7. When this trigger arrives, the CSM will go from state C2 to C3 and the conversion will continue (as described in Section 19.7.2, “ATD Example 1 — Simple Conversion,” to Section 19.7.4, “ATD Example 3 — Interrupted Continuous Conversion”).

A conversion waiting for a trigger can be aborted in the same way as other operations by writing a command to ATDCW with CWCM = 00 (as described in Section 19.7.4, “ATD Example 3 — Interrupted Continuous Conversion”).

Figure 19-17 illustrates some examples of valid edge-sensitive triggers. The minimum trigger pulse length must be one peripheral bus clock cycle; longer trigger pulses are allowed. The minimum trigger length is independent of the ATD clock period. Due to synchronization and internal processing, the conversion start (symbolized by the arrows in Figure 19-17) will be delayed by 4 peripheral bus clock cycles.

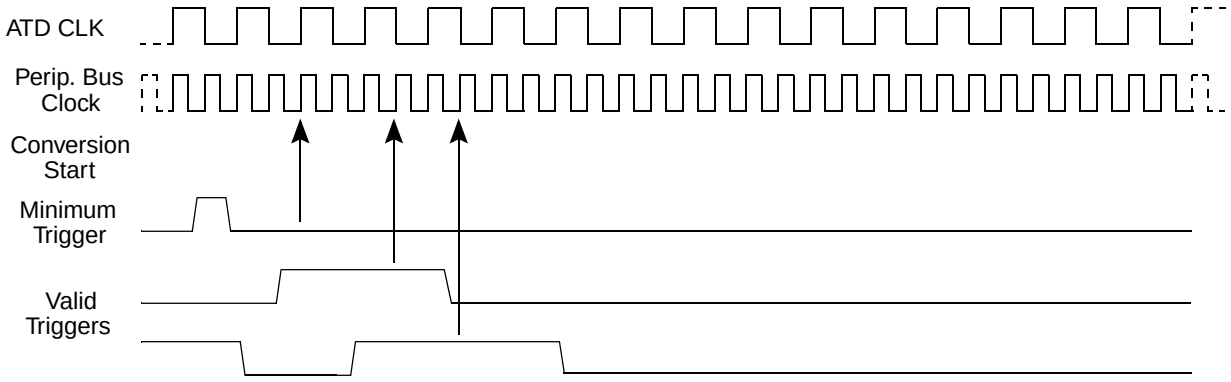


Figure 19-17. ATD Edge-Based Trigger Example

If another trigger pulse is asserted between the first trigger and the end of the conversion, The ETO bit of the ATDFLAG register will be set. The unexpected trigger will be ignored and therefore the current conversion is not affected.

19.7.6 ATD Example 5 — Level Triggered Conversion

This example shows a conversion started by a high-level sensitive trigger asserted at the analog input channel 7, as shown in Figure 19-18. The trigger is set up by:

- ATDTRIGCTL[TRIGSEL] = ‘11’ (use analog input channel as trigger)
- ATDTRIGCTL[TRIGP] = ‘1’ (trigger is high-level sensitive)
- ATDTRIGCTL[TRIGLE] = ‘1’ (trigger is level sensitive)
- ATDETRIGCH[ETRIGCH] = ‘0111’ (analog channel 7 is trigger)

Each write to either the ATDTRIGCTL or ATDETRIGCH register will stop a conversion that may be executing. For this example it is assumed that both registers have been configured previously and no conversions were being executed.

When a command word with CWCM = 10 (wait for trigger) is written to the ATDCW register, the CSM will move from state C1 to C2, where it will wait for a high level on the analog input channel 7. When this trigger arrives the CSM will go from state C2 to C3 and the conversion will continue (as described in Section 19.7.2, “ATD Example 1 — Simple Conversion,” to Section 19.7.4, “ATD Example 3 — Interrupted Continuous Conversion”).

After the conversion has finished (C_3 to C_4) the ATD will wait for a new command word. For this example, it is assumed that another command word with CWCM = 10 is written to the ATDCW register (by the eDMA or the CPU) and that the high level is still asserted at the analog input channel 7, causing a new conversion to be started.

If the trigger input level switches to the inactive state during the execution of a conversion, the conversion will continue until completed. If there is a new command word waiting for a trigger, the ATD will remain in state C_2 until the trigger is asserted or the CSM is reset.

While the trigger is asserted, new “wait for trigger” conversions can start. Due to synchronization and internal processing, the conversion start is delayed by 4 IPS bus clock cycles.

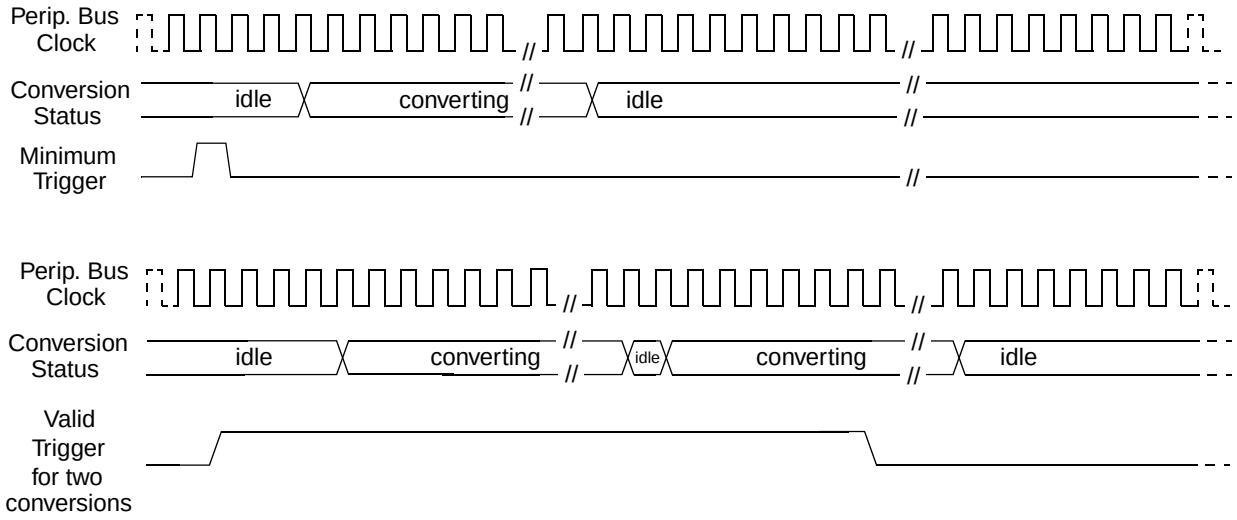


Figure 19-18. ATD Level-Based Trigger Examples

The minimum trigger length for a level-sensitive trigger is one peripheral bus clock cycle, although it is not recommended to use a level-sensitive trigger with such a short trigger pulse. Generally, a level-sensitive trigger is used to control how long conversions should be executed.

The delay between the end of the previous level-triggered conversion and the start of the next conversion will be one peripheral bus clock cycle if the trigger is asserted the entire time (as shown in the lower half of Figure 19-18).

19.7.7 ATD Example 6 — Using External Triggers

In order to use the external triggering capability of the ATD, the following steps should be followed:

1. Configure the appropriate pin(s) to peripheral mode, via PIM configuration registers
2. Clear the MDIS bit in the ATD
3. Write the ATDETRIGCH register in the ATD to set the desired channel number
4. Write 0b11 to the TRIGSEL bits in the ATDTRIGCTL register in the ATD

19.7.8 ATD Example 7 — Using System Triggers

If the ATD should start conversions synchronous to SYSTRIG0 or SYSTRIG1, the following steps must be performed in the PIT (refer to Chapter 25, “Periodic Interrupt Timer Module (PIT),” for details):

1. Clear MDIS bit
2. Specify timer reload value for selected SYSTRIG*n*

3. Enable timer

The timer reload value can be calculated with the following formula:

$$\frac{\text{trigger period}}{\text{clock period}} - 1 = \text{timer reload value} \qquad \textit{Eqn. 19-3}$$

For example, assume the system clock is 50 Mhz (t_{fSYS} = 20ns) and thus the peripheral bus clock period, T_{fIPS}, is 40 ns. If the desired trigger period is 25 µs, the reload value is calculated as:

$$\frac{25\mu s}{40ns} - 1 = 625 - 1 = 624 = 0x0000\_0270 \qquad \textit{Eqn. 19-4}$$

If SYSTRIG0 is selected as the trigger, write the reload value to TLVAL10 and enable Timer 10.

If SYSTRIG1 should trigger a conversion, write the reload value to TLVAL9 and enable timer 9.

In the ATD the following steps must be performed (it is assumed that all other registers in the ATD are already configured):

1. Clear MDIS bit
2. Write 8b0000_0110 to the ATDTRIGCTL register. This selects rising edge sensitivity for SYSTRIG0.

   -or-

   Write 8b0000_1010 to the ATDTRIGCTL register. This selects rising edge sensitivity for SYSTRIG1.
3. Write a command word to the ATDCW register that has the bits CWCM set to ‘10’ (wait for trigger)

After the command word is written to the ATDCW register, the ATD will start the conversion when the trigger is asserted.

NOTE

A new command word must be provided before the next trigger is asserted. Otherwise the ATDFLAG[ETO] flag will be set.

19.7.9 Conversion Mechanism — CWCH, CWNF, CWGI and CWSC Bits

Section 19.6.5, “Conversion process,” through Section 19.7.1, “ATD Initialization Sequence,” describe how conversions are started and what is done after a conversion finishes. This section and Section 19.7.10, “Conversion Mechanism — CWSL, CWSB and CW8 Bits,” describe the ATD conversion sequence as well as the remaining bits in a command word that have not previously been described. For the complete list of bit names with brief descriptions, refer to Table 19-9

Each conversion must define which analog channel will be sampled. The CWCH bits in the command word should be programmed with a value from 0 to 15 to identify the appropriate channel number to be sampled (see Section 19.5.1.7, “ATD Command Word Register (ATDCW)”). The special channel bit (CWSC) must also be cleared.

If CWSC is set, the CWCH bits are not used to select an external analog signal, but rather to select one of the special internal reference voltages. These channels provide an input voltage to the comparator equal to

V_{RL}, V_{RH} or (V_{RH} + V_{RL}) ÷ 2. Refer to Section 19.7.11, “Measuring Internal Reference Voltages,” for more information.

When a conversion has been completed, the result is saved using one of four available numeric formats: right-justified signed, right-justified unsigned, left-justified signed and left justified unsigned. The default format is right-justified unsigned. By changing the CWNF bits, any of the four formats can be specified (see Section 19.5.1.7, “ATD Command Word Register (ATDCW)”).

The use of the CWSL, CWSB and CW8 bits are explained below in Section 19.7.10, “Conversion Mechanism — CWSL, CWSB and CW8 Bits.”

The use of the CWCM bits are explained above in Section 19.6.5, “Conversion process,” and Section 19.7.2, “ATD Example 1 — Simple Conversion,” through Section 19.7.6, “ATD Example 5 — Level Triggered Conversion.”

The value of the CWGI bit determines if the Conversion Complete flag (CC) in the ATDFLAG register will be set on completing the conversion. Setting the CWGI bit in a command word will cause CC to be set. Clearing the CWGI bit in a command word will cause no change to the CC bit. This can be used to mark the end of a sequence of conversions e.g. after 8 conversions there should be an interrupt,. Therefore the CWGI bit in the first seven command words would be 0 and in the last command word it would be 1.

If the Conversion Complete Interrupt Enable (CCIE) bit in the ATDINT register is set, an interrupt will be generated after the eighth conversion is completed.

19.7.10 Conversion Mechanism — CWSL, CWSB and CW8 Bits

A conversion sequence consists of 2 phases:

1. A sampling phase where the input voltage at an analog input pin is stored in a capacitor
2. The successive approximation of an internal generated reference voltage to match the stored input voltage.

Furthermore, the sampling phase can be divided into the following periods:

1. Sample buffer time (CWSB): 0 or 2 ATD clock cycles (first part of sample phase)
2. Sample time (CWSL): 2, 4, 8 or 16 ATD clock cycles (second part of sample phase)
3. Conversion time (CW8): 8 or 10 ATD clock cycles (corresponds to 8- or 10-bit resolution)

During the first step of the sample phase (sample buffer time) the input voltage of an analog input is sampled through an operational amplifier (op-amp) to quickly reproduce the voltage level at the sample capacitor and minimize the load on the analog input signal. The amplifier can be used for 2 ATD clock cycles (CWSB = ‘0’) or bypassed (CWSB = ‘1’). When the amplifier is bypassed, [1] a conversion starts with the second step of the sample phase (sample time).

During the sample time, the op-amp is bypassed and the analog input directly charges the sample capacitor for improved accuracy. The minimum sample time is 2 ATD clock cycles and the maximum is 16 ATD clock cycles.

1. Note that bypassing the sample amplifier is not recommended.

During the conversion time, the voltage stored by the sample capacitor is compared with an internally generated reference voltage (DAC) using a successive approximation scheme. This can last 8 ATD clock cycles for 8-bit resolution or 10 ATD clock cycles for 10-bit resolution.

The following example illustrates the different steps during a conversion sequence. The sample buffer time is assumed to be 2 ATD clock cycles, the sample time is 4 ATD clock cycles and the resolution for the conversion is 10 bits.

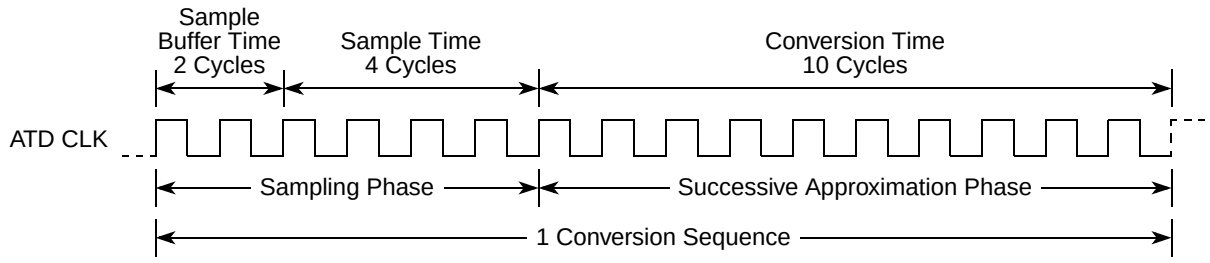


Figure 19-19. ATD 10-bit Conversion Timing Example

The next example shows a conversion sequence without the use of the sample amplifier. [1] The sample time is 4 ATD clock cycles and the resolution for the conversion is 8 bits.

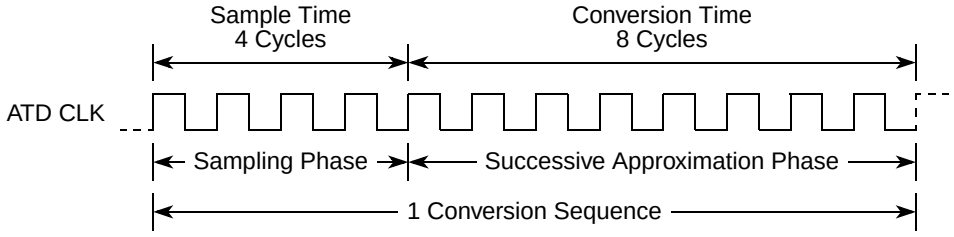


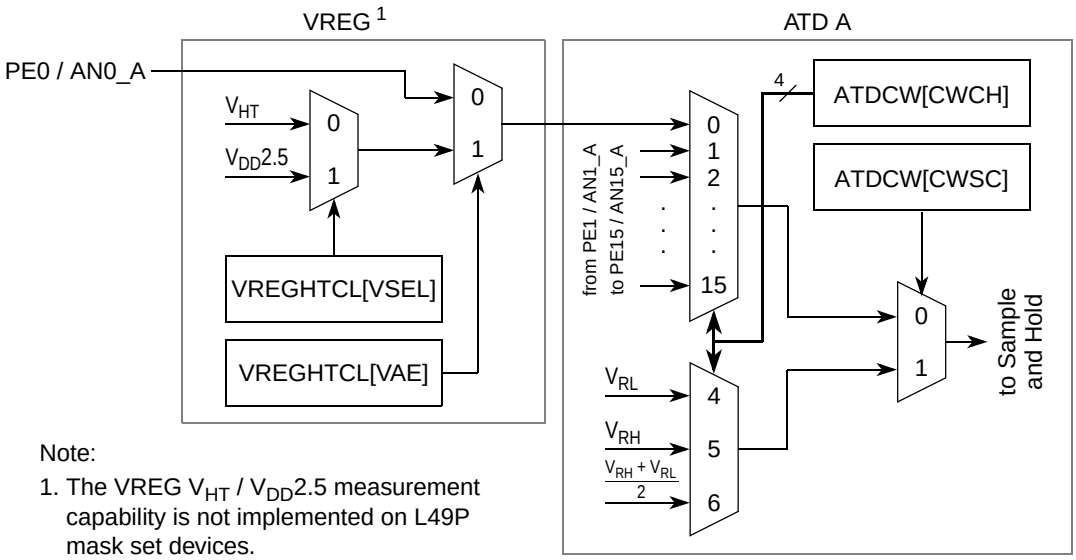
Figure 19-20. ATD 8-bit Conversion Timing, Amplifier Bypass Example

Depending on the length of the sample buffer time and sample time, the impedance of the circuitry providing the ATD analog channel inputs will differ. Bypass the sample amplifier should only be done for low impedance sources. [1] Because of internal RC time constants, it is not recommended to use a sample time of two ATD clock cycles when the sample amplifier is bypassed. [1] A minimum sample time of four ATD clock cycles must be allowed for all conversions of external channels. Internal conversions of V_{RL}, V_{RH}, or $(V_{RH} + V_{RL}) \div 2$ only require a total sample time of two ATD clock cycles.

19.7.11 Measuring Internal Reference Voltages

Three special ATD internal reference voltages are available for calibration, as described in Section 19.7.9, “Conversion Mechanism — CWCH, CWNF, CWGI and CWSC Bits.” On mask sets later than L49P, two additional internal voltages are available for system monitoring via the VREG module.

The figure below illustrates the logical connection of the reference voltages via multiplexors to the sample and hold circuit for ATD A on later mask set devices. For ATD B (on those devices that implement it), the logical connections are identical with the exception of the optional VREG override on channel 0, which is not present on ATD B.



Note:
1. The VREG V_{HT} / $V_{DD}2.5$ measurement capability is not implemented on L49P mask set devices.

Figure 19-21. ATD Internal Reference Voltage Selection Logical Diagram

Refer to Section 3.5.1.1, "VREG High Temperature Control Register (VREGHTCL)," on page 3-33 for more information on using the VREG reference voltages. As the V_{HT} and $V_{DD}2.5$ monitoring feature is external to the ATD, it is subject to the same minimum timing constraints (four ATD clocks) as inputs to external pins.

Note that the reference voltages used by the comparator are always V_{RH}/V_{RL}, which has important implications for using the reference voltages. Since the references are all positive voltages with respect to $V_{SS}A$, but V_{RL} is not required to be tied to system ground (see the *MAC7100 Microcontroller Family Hardware Characteristics*, MAC7100EC), any offset between $V_{SS}A$ and V_{RL} must be accounted for in measurement algorithms. Likewise, since V_{RH} is allowed to be below $V_{DD}2.5$, and possibly the maximum level of V_{HT}, measurement of the VREG reference voltages may be invalid if $V_{RH} < V_{DD}2.5$.

Chapter 20
Enhanced Modular I/O Subsystem Module (eMIOS)

20.1 Overview

The Enhanced Modular Timer Subsystem (eMIOS) provides functionality to generate or measure time events. The eMIOS module implemented on MAC7100 Family devices has 16 unified timer channels (UCs), with all channel counters being 16-bits wide. Each UC is identical and can be configured to provide a wide range of timer functions. The module implements 3 counter busses: Counter bus A, Counter bus B and Counter bus C. Counter bus A is driven by UC15 and can be shared across all UCs. Counter bus B is driven by UC0 and can be shared by UC0 to UC7. Counter bus C is driven by UC8 and can be shared by UC8 to UC15.

The eMIOS has an interface to the peripheral bus for communication with timer channel submodules via a local inter-module bus, with the submodules providing the timer and counter functions needed by the applications. Common time bases can be shared between submodules using counter busses in order to offer synchronous operation.

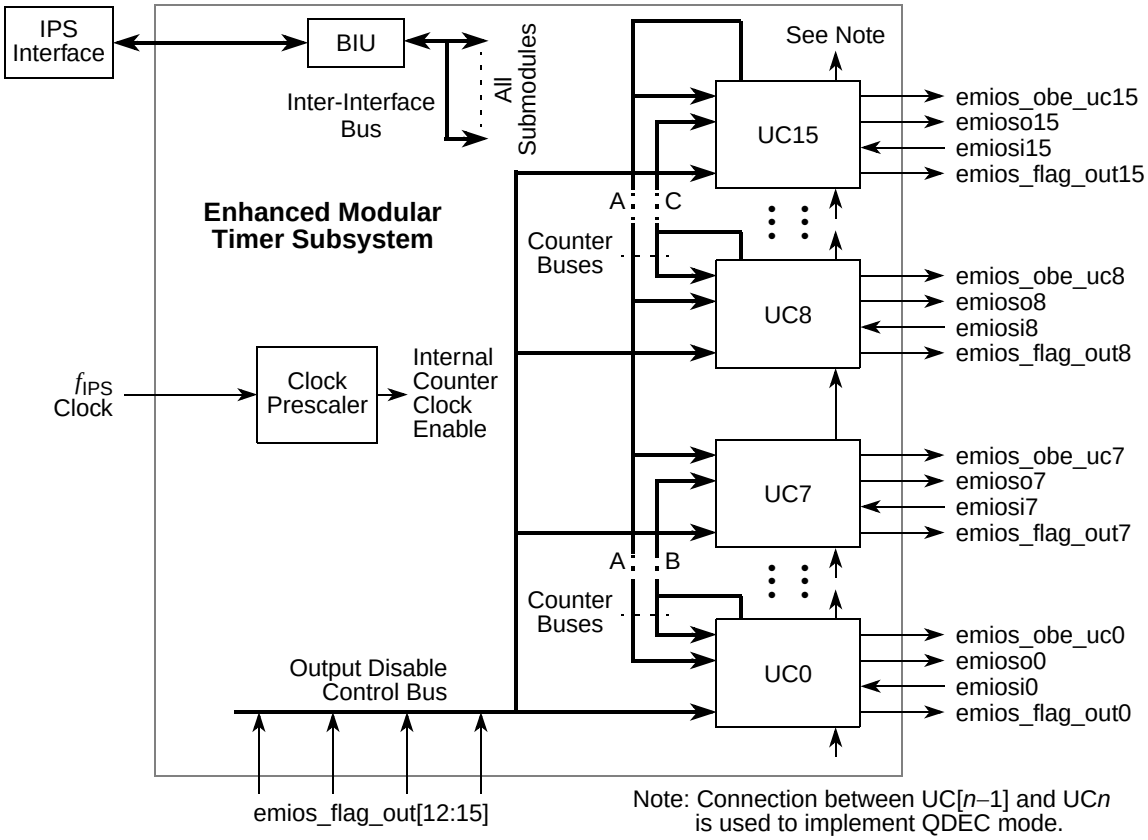


Figure 20-1. eMIOS Block Diagram

Each of the UCs has a single input/output signal associated with it, resulting in a module with 16 external signals available for the user. The eMIOS module can be independently disabled by writing to the MDIS

bit in the MCR register. Disabling the module turns off the clock to the module, although some of the module registers (MCR, OUDIS and UCDIS) remain available to be accessed by the core via the peripheral bus. The MDIS bit is intended to be used when the module is not required in the application.

Figure 20-1 shows the block diagram of the eMIOS. Note that names of on-chip only signals are denoted in lower case. See Section 20.4, "Signal Description," for details on the relationship between internal the signals emiosi*n*, emioso*n* and emios_obe_uc*n* and external signals. The emios_flag_out*n* signals are inputs to the INTC Module used to generate interrupt requests and corresponding exception vectors as described in Chapter 10, "Interrupt Controller Module (INTC)," and Chapter 6, "Exceptions." The emios_flag_out*n* signals of UC12 through UC15 are also available for use as an output disable for all UC*n*s.

20.2 Features

The basic features of the eMIOS are the following:

- 16 Unified Channels
- 16-bit counters and data registers
- Independent time base available for each channel, in addition to the shared counter busses
- Shared timebases through the counter busses A, B and C
- Counter Bus A can be driven by UC15
- Counter Busses B and C can be driven by UC0 and UC8 respectively
- Synchronization among timebases
- One Global prescaler
- One Prescaler per channel
- Control and Status bits grouped in a single register
- Shadow FLAG register
- State of the Unified Channel can be frozen for debug purposes
- Motor control capability

20.3 Modes of Operation

Channels can be configured to operate in the following modes (see Section 20.6, "Functional Description"):

- General Purpose Input/Output
- Single Action Input Capture
- Single Action Output Compare
- Input Pulse Width Measurement
- Input Period Measurement
- Double Action Output Compare
- Pulse/Edge Accumulation
- Pulse/Edge Counting
- Quadrature Decode
- Windowed Programmable Time Accumulation
- Modulus Counter, Normal or Buffered
- Output Pulse Width and Frequency Modulation, Normal or Buffered
- Center Aligned Output Pulse Width Modulation, Normal or Buffered
- Output Pulse Width Modulation, Normal or Buffered

20.4 Signal Description

While each eMIOS Unified Channel has one input and one output signal, MAC7100 Family devices connect both signals to a single bidirectional pin (see Chapter 18, "Port Integration Module (PIM)"). Note that the PIM must be configured to enable the peripheral function of the appropriate pins (refer to Section 18.6.2, "Peripheral Mode") prior to configuring an eMIOS UC.

Table 20-1. eMIOS Signal Properties

Signal	Direction	Function	Reset State	Pull up
emios*i*	input	eMIOS Unified Channel *n* Input	—	see Chapter 18
emioso*n*	output	eMIOS Unified Channel *n* Output	0 / Hi-Z [1]	see Chapter 18

[1] Value "0" refers to the reset value of the UCCR*n*.UCOUT signal that will be driven on the pin if enabled. Hi-Z refers to the state of the external pin as controlled by the emios_obe_uc*n* signal.

20.4.1 emiosi*n* — eMIOS Unified Channel *n* Input Signal

The internal emiosi*n* signal from the pad (see Figure 18-15 on page 18-298) is synchronized and filtered by the input programmable filter (IPF). The IPF output is then used by the channel logic and is available to be read by the MCU through the UCIN bit of the UCCR*n* register.

20.4.2 emioso*n* — eMIOS Unified Channel *n* Output Signal

The internal emioso*n* signal reflects the value of the UCOUT bit of the UCCR*n* register, and is used to drive the pin (see Figure 18-15 on page 18-298) if the corresponding emios_obe_uc*n* signal is also asserted in order to enable the pin output.

20.5 Memory Map / Register Definition

The module address map and UC detail are shown in Table 20-2 and Table 20-3, respectively. Attempts to access addresses that are masked reserved will result in a bus abort exception.

Table 20-2. eMIOS Memory Map

eMIOS Offset	Register Description
0x0000	Module Configuration Register (MCR)
0x0004	Global Flag Register (GFLAG)
0x0008	Output Update Disable Register (OUDIS)
0x000C	Disable Channel Register (UCDIS)
0x0010	reserved
0x0020	Unified Channel 0 (UC0)
0x0040	Unified Channel 1 (UC1)
0x0060	Unified Channel 2 (UC2)
0x0080	Unified Channel 3 (UC3)
0x00A0	Unified Channel 4 (UC4)
0x00C0	Unified Channel 5 (UC5)
0x00E0	Unified Channel 6 (UC6)
0x0100	Unified Channel 7 (UC7)

Table 20-2. eMIOS Memory Map (continued)

0x0120	Unified Channel 8 (UC8)
0x0140	Unified Channel 9 (UC9)
0x0160	Unified Channel 10 (UC10)
0x0180	Unified Channel 11 (UC11)
0x01A0	Unified Channel 12 (UC12)
0x01C0	Unified Channel 13 (UC13)
0x01E0	Unified Channel 14 (UC14)
0x0200	Unified Channel 15 (UC15)
0x0204 to 0x031F	reserved

Table 20-3. eMIOS UC*n* Memory Map Detail

eMIOS Offset	Register Description
eMIOS Base + UC Offset + 0x00	Channel A Data Register (UCA*n*)
eMIOS Base + UC Offset + 0x04	Channel B Data Register (UCB*n*)
eMIOS Base + UC Offset + 0x08	UC Counter Register (UCCNT*n*)
eMIOS Base + UC Offset + 0x0C	Channel Control Register (UCCR*n*)
eMIOS Base + UC Offset + 0x10	UC Status Register (UCSR*n*)
eMIOS Base + UC Offset + 0x14 to 0x1F	reserved

20.5.1 Register Descriptions

All registers are 32-bits, even though the eMIOS uses 16 UCs and 16-bit counters. Thus some register fields have reserved bits that may be implemented in eMIOS configurations on other Freescale devices.

20.5.1.1 eMIOS Module Configuration Register (MCR)

The MCR contains global control bits for the eMIOS module.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DOZEEN	MDIS	FRZ	0	0	GPREN	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eMIOS Base + 0x0000															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPRE								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eMIOS Base + 0x0000															

Figure 20-2. eMIOS Module Configuration Register (MCR)

Table 20-4. MCR Field Descriptions

Bits	Name	Description
31	DOZEEN	Doze enable. Refer to Section 20.6.1 for additional information. 0 Doze Mode disabled 1 Doze Mode enabled
30	MDIS	Module disable. Refer to Section 20.6.1 for additional information. 0 Clock is running 1 Enter low power mode
29	FRZ	Freeze enable. Refer to Section 20.6.1 for additional information. 0 Freeze mode disabled 1 Freeze enabled for Debug mode
28–27	—	Reserved.
26	GPREN	Global prescaler enable. Enables the prescaler counter. 0 Prescaler disabled (no clock) and prescaler counter is cleared 1 Prescaler enabled
25–16	—	Reserved.
15–8	GPRE	Global Prescaler. Selects the clock divider value as shown: GPRE / Divide Ratio 00000000 / 1 00000001 / 2 . . . / . . . 11111111 / 256
7–0	—	Reserved.

20.5.1.2 eMIOS Global Flag Register (GFLAG)

The GFLAG is a read-only register that groups the FLAG bits from all channels in order to improve interrupt handling. These bits are mirrors of the FLAG bits of each channel register (UCSR*n*).

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eMIOS Base + 0x0004															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	F15	F14	F13	F12	F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	F0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eMIOS Base + 0x0004															

Figure 20-3. eMIOS Global Flag Register (GFLAG)

20.5.1.3 eMIOS Output Update Disable Register (OUDIS)

The OUDIS is a global register that controls the manner in which all channels utilize the channel registers A1, A2, B1 and B2.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eMIOS Base + 0x0008															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OU15	OU14	OU13	OU12	OU11	OU10	OU9	OU8	OU7	OU6	OU5	OU4	OU3	OU2	OU1	OU0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eMIOS Base + 0x0008															

Figure 20-4. eMIOS Output Update Disable Register (OUDIS)

Table 20-5. OUDIS Field Descriptions

Bits	Name	Description
31–16	—	Reserved.
15–0	OU*n*	Unified Channel *n* output update disable. When running MC mode or an output mode, values are written to registers A2 and B2. OU*n* bits are used to disable transfers from registers A2 to A1 and B2 to B1. Each bit controls one channel. 0 Transfer enabled. Depending on the operation mode, transfer may occur immediately or in the next period. Unless stated otherwise, transfer occurs immediately. 1 Transfers disabled

20.5.1.4 eMIOS Channel Disable Register (UCDIS)

The UCDIS is a global register that is used to enable/disable all Unified Channels.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eMIOS Base + 0x000C															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UCDIS15	UCDIS14	UCDIS13	UCDIS12	UCDIS11	UCDIS10	UCDIS9	UCDIS8	UCDIS7	UCDIS6	UCDIS5	UCDIS4	UCDIS3	UCDIS2	UCDIS1	UCDIS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	eMIOS Base + 0x000C															

Figure 20-5. eMIOS Channel Disable Register (UCDIS)

Table 20-6. UCDIS Field Descriptions

Bits	Name	Description
31–16	—	Reserved.
15–0	UCDIS*n*	Enable Channel *n.* Used to disable any UC*n* by stopping the respective clock. 0 UC*n* enabled 1 UC*n* disabled

20.5.1.5 eMIOS Channel A Data Registers (UCA*n*)

Each UC*n* contains two internal registers A1 and A2, used for matches and captures. Depending on the mode of operation, A1 or A2 can be assigned to be accessed at the UCA*n*.offset within the UC*n* register map. Both A1 and A2 are cleared by reset. Figure 20-7 summarizes the UCA*n* read and write accesses for all modes. For more information see section Section 20.6.7, “UC Modes of Operation.”

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x00															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UCA*n*															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x00															

Figure 20-6. eMIOS Channel A Data Registers (UCA*n*)

20.5.1.6 eMIOS Channel B Data Registers (UCB*n*)

Each UC*n* contains two internal registers B1 and B2. Depending on the mode of operation, internal registers B1 or B2 can be assigned to be accessed at the UCB*n* offset within the UC*n* register map. Both B1 and B2 are cleared by reset. Table 20-7 summarizes the UCB*n* read and write accesses for all modes. For more information see section Section 20.6.7, “UC Modes of Operation.”

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x04															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UCB*n*															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x04															

Figure 20-7. eMIOS Channel B Data Registers (UCB*n*)

Table 20-7. UCA*n* and UCB*n* Access Assignment

Operation Mode	Register Access			
	UCA*n*		UCB*n*	
	Read	Write	Read	Write
GPIO	A1	A1, A2	B1	B1, B2
SAIC [1]	A2	—	B2	B2
SAOC [1]	A1	A2	B2	B2
IPWM	A2	—	B1	—
IPM	A2	—	B1	—
DAOC	A1	A2	B1	B2
PEA	A2	A1	B1	—
PEC	A1	A1	B1	B1
QDEC [1]	A1	A1	B2	B2
WPTA	A1	A1	B1	B1
MC [1]	A1	A2	B2	B2
OPWFM	A1	A2	B1	B2
OPWMC	A1	A2	B1	B2
OPWM	A1	A2	B1	B2

1 In these modes, the UCB*n* register is not used, but B2 can be accessed.

20.5.1.7 eMIOS Channel Counter Registers (UCCNT*n*)

Each UC*n* has a UCCNT*n* register that contains the value of the internal counter. When GPIO mode is selected or the channel is frozen, the UCCNT*n* register is read/write. For all others modes, the UCCNT*n* is read-only. When entering some operation modes, this register is automatically cleared (refer to Section 20.6.7, "UC Modes of Operation," for details).

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W[1]																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x08															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Counter															
W[1]																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x08															

1 In GPIO mode or Freeze action, this register is writable.

Figure 20-8. eMIOS Channel Counter Registers (UCCNT*n*)

20.5.1.8 eMIOS Channel Control Registers (UCCR*n*)

UCCR*n* gathers bits reflecting the status of the UC input/output signals and the overflow condition of the internal counter, as well as several read/write control bits.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FREN	ODIS	ODISSL		UCPRE		UCPREN	DMA	0	IF				FCK	FEN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x0C															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	BSL		EDSEL	EDPOL	MODE						
W			FORCMA	FORCMB												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x0C															

Figure 20-9. eMIOS Channel Control Registers (UCCR*n*)

Table 20-8. UCCR*n* Register Field Descriptions

Bits	Name	Description
31	FREN	Freeze enable. 1 Freeze registers when MCU enters debug mode 0 Normal operation
30	ODIS	Output disable. Allows output disable in any output mode except GPIO. 0 The output pin operates normally 1 If the selected Output Disable signal is asserted, the output signal goes to EDPOL for OPWFMB and OPWMB modes and to the complement of EDPOL for all other modes, and the UC*n* continues to operate normally. If the selected Output Disable signal is negated, the output signal operates normally
29–28	ODISSL	Output disable select. Selects the output disable source channel. 00 UC12 FLAG used for Output Disable 01 UC13 FLAG used for Output Disable 10 UC14 FLAG used for Output Disable 11 UC15 FLAG used for Output Disable
27–26	UCPRE	Prescaler. Selects the clock divider value for the UC prescaler. 00 divide by 1 01 divide by 2 10 divide by 3 11 divide by 4
25	UCPREN	Prescaler enable. Enables the prescaler counter. 0 Prescaler disabled, prescaler counter loaded with UCPRE value 1 Prescaler enabled
24	DMA	Direct memory access. 0 FLAG assigned to interrupt request 1 FLAG assigned to DMA request
23	—	Reserved.
22–19	IF [1]	Input filter. Selects the minimum input pulse width that is passed through the filter. For output modes, these bits are ignored. 0000 Bypassed (signal is synchronized, filter is not applied) 0001 2 FCK periods 0010 4 FCK periods 0100 8 FCK periods 1000 16 FCK periods other Reserved
18	FCK	Filter clock select. Selects the clock source for the input filter. 0 Prescaled clock 1 Main clock
17	FEN	FLAG enable. 0 Disable (FLAG is ignored) 1 Enable (FLAG generates an interrupt or DMA request)
16–14	—	Reserved.
13	FORCMA	Force match A. For output modes which use comparator A, setting the FORCMA bit is equivalent to a match on comparator A (except that the FLAG bit is not set). For all other modes, the bit is ignored. 0 No effect 1 Force a match at comparator A

Table 20-8. UCCR*n* Register Field Descriptions (continued)

Bits	Name	Description
12	FORCMB	Force match B. For output modes which use comparator B, setting the FORCMB bit is equivalent to a match on comparator B (except that the FLAG bit is not set). For all other modes, bit is ignored. 0 No effect 1 Force a match at comparator B
11	—	Reserved.
10–9	BSL	Bus select. Selects one of the counter buses or the internal counter. 00 All channels:counter bus A 01 UC0 to UC7:counter bus B UC8 to UC15:counter bus C 10 Reserved 11 All channels:internal counter
8	EDSEL	Edge selection. Interpretation depends on MODE and EDPOL field settings listed below, ignored for all other modes. Input modes: EDSEL selects whether internal counter is triggered by both edges of a pulse or by single edge defined by EDPOL bit. 0 Single edge triggering defined by the EDPOL bit 1 Both edge triggering GPIO Input mode: EDSEL selects FLAG generation. 0 FLAG generated as defined by the EDPOL bit 1 FLAG not generated SAOC mode: 0 EDPOL value transferred to output flip-flop on each match 1 Output flip-flop toggled on each match
7	EDPOL	Edge polarity. Interpretation depends on the selected mode: Input modes except QDEC: selects which edge triggers internal counter, input capture or FLAG. 0 Trigger on a falling edge 1 Trigger on a rising edge QDEC Count & Direction mode: [2] selects count direction according to direction signal (UC*n* input). 0 Counts down when UC*n* is asserted 1 Counts up when UC*n* is asserted QDEC Phase A & B mode: [3] selects count direction according to the phase difference. 0 Counts down if Phase_A is ahead Phase_B 1 Counts up if Phase_A is ahead Phase_B Output modes: selects the logic level on the output pin. 0 Comparator A match clears output flip-flop, Comparator B match sets output flip-flop 1 Comparator A match sets output flip-flop, Comparator B match clears output flip-flop
6–0	MODE[6:0]	Mode selection. Refer to Table 20-9 for definitions.

1 Filter latency is 3 clock edges.

2 The EDPOL bit of UC[*n*–1] selects which edge clocks the internal counter of UC*n* in this mode:
0 Clock on falling edge
1 Clock on rising edge

3 The EDPOL bit of UC[*n*–1] is ignored in this mode.

Table 20-9. UCCR*n* MODE Field Definitions

MODE	Mode of Operation [1]
0000000	General Purpose Input
0000001	General Purpose Output
0000010	Single Action Input Capture
0000011	Single Action Output Compare
0000100	Input Pulse Width Measurement
0000101	Input Period Measurement
0000110	Double Action Output Compare, FLAG Set On Second Match
0000111	Double Action Output Compare, FLAG Set On Both Matches
0001000	Pulse/Edge Accumulation, Continuous
0001001	Pulse/Edge Accumulation, Single Shot
0001010	Pulse/Edge Counting, Continuous
0001011	Pulse/Edge Counting, Single Shot
0001100	Quadrature Decode, Count and Direction Encoders
0001101	Quadrature Decode, Phase_A and Phase_B Encoders
0001110	Windowed Programmable Time Accumulation
0001111	Reserved
0010000	Modulus Up Counter, Internal Clock
0010001	Modulus Up Counter, External Clock
0010010	Reserved
0010011	
0010100	Modulus Up/Down Counter, FLAG Set On One Event, Internal Clock
0010101	Modulus Up/Down Counter, FLAG Set On One Event, External Clock
0010110	Modulus Up/Down Counter, FLAG Set On Both Events, Internal Clock
0010111	Modulus Up/Down Counter, FLAG Set On Both Events, External Clock
0011000	Output Pulse Width and Frequency Modulation, FLAG Set On Second Match, Immediate Update
0011001	Output Pulse Width and Frequency Modulation, FLAG Set On Second Match, Next Period Update
0011010	Output Pulse Width and Frequency Modulation, FLAG Set On Both Matches, Immediate Update
0011011	Output Pulse Width and Frequency Modulation, FLAG Set On Both Matches, Next Period Update
0011100	Center Aligned Output Pulse Width Modulation, FLAG Set On Trailing Edge, Trailing Edge Dead-Time
0011101	Center Aligned Output Pulse Width Modulation, FLAG Set On Trailing Edge, Leading Edge Dead-Time
0011110	Center Aligned Output Pulse Width Modulation, FLAG Set On Both Edges, Trailing Edge Dead-Time
0011111	Center Aligned Output Pulse Width Modulation, FLAG Set On Both Edges, Leading Edge Dead-Time
0100000	Output Pulse Width Modulation, FLAG Set On Second Match, Immediate Update
0100001	Output Pulse Width Modulation, FLAG Set On Second Match, Next Period Update
0100010	Output Pulse Width Modulation, FLAG Set On Both Matches, Immediate Update
0100011	Output Pulse Width Modulation, FLAG Set On Both Matches, Next Period Update
1010000	Modulus Up Counter, Buffered, Internal Clock [2]
1010001	Modulus Up Counter, Buffered, External Clock [2]
1010010	Reserved
1010011	

Table 20-9. UCCR*n* MODE Field Definitions *(continued)*

MODE	Mode of Operation [1]
1010100	Modulus Up/Down Counter, Buffered, FLAG Set On One Event, Internal Clock [2]
1010101	Modulus Up/Down Counter, Buffered, FLAG Set On One Event, External Clock [2]
1010110	Modulus Up/Down Counter, Buffered, FLAG Set On Both Events, Internal Clock [2]
1010111	Modulus Up/Down Counter, Buffered, FLAG Set On Both Events, External Clock [2]
1011000	Output Pulse Width and Frequency Modulation, Buffered, FLAG Set On Second Match [2]
1011001	Reserved
1011010	Output Pulse Width and Frequency Modulation, Buffered, FLAG Set On Both Matches [2]
1011011	Reserved
1011100	Center Aligned Output Pulse Width Modulation, Buffered, FLAG Set On Trailing Edge, Trailing Edge Dead-Time [2]
1011101	Center Aligned Output Pulse Width Modulation, Buffered, FLAG Set On Trailing Edge, Leading Edge Dead-Time [2]
1011110	Center Aligned Output Pulse Width Modulation, Buffered, FLAG Set On Both Edges, Trailing Edge Dead-Time [2]
1011111	Center Aligned Output Pulse Width Modulation, Buffered, FLAG Set On Both Edges, Leading Edge Dead-Time [2]
1100000	Output Pulse Width Modulation, Buffered, FLAG Set On Second Match [2]
1100001	Reserved
1100010	Output Pulse Width Modulation, Buffered, FLAG Set On Both Matches [2]
1100011 to 1111111	Reserved

1 Refer to Section 20.6.7, “UC Modes of Operation,” for details.

2 Not implemented on mask set L49P and L47W mask set devices.

20.5.1.9 eMIOS Channel Status Registers (UCSR*n*)

The bits in the UCSR*n* register may be read at any time to determine the cause of an interrupt request or the current state of the external pin associated with the UC. The OVR, OVFL and FLAG status bits may be cleared by writing a ‘1’ to the corresponding bit position; writing a ‘0’ is ignored.

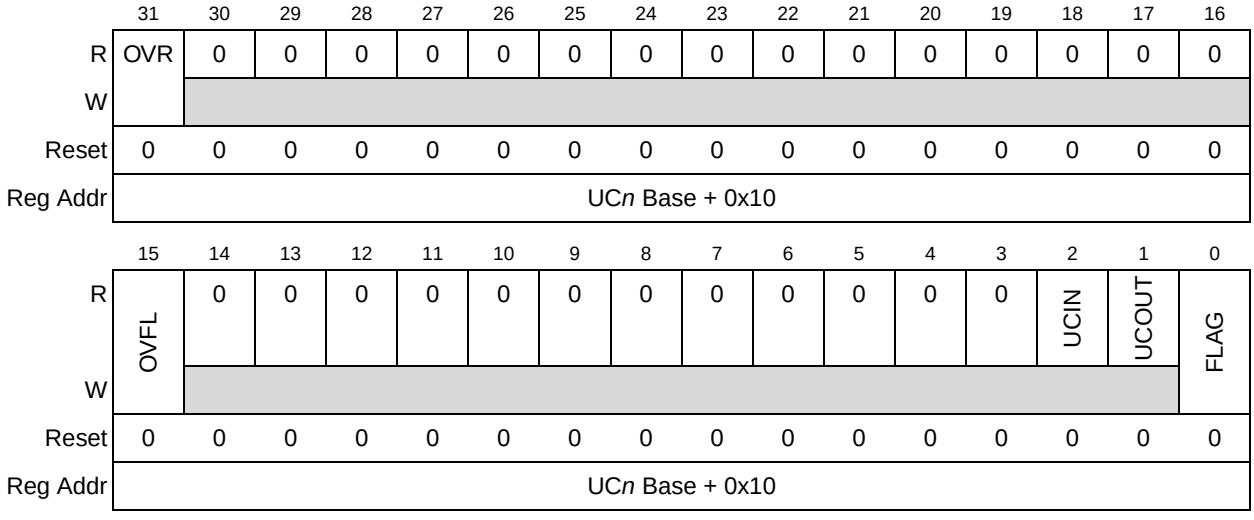


Figure 20-10. eMIOS Channel Status Registers (UCSR*n*)

Table 20-10. UCSR*n* Field Descriptions

Bits	Name	Description
31	OVR	Overrun. FLAG generation occurred with FLAG already set. Clearing the FLAG bit causes this bit to be cleared. 0 Overrun has not occurred 1 Overrun has occurred
30–16	—	Reserved.
15	OVFL	Overflow. Indicates that an internal counter overflow has occurred. Clearing the FLAG does not clear this bit. 0 No overflow 1 An overflow had occurred
14–3	—	Reserved.
2	UCIN	UC input pin. Reflects the input pin state after being filtered and synchronized.
1	UCOUT	UC output pin. Reflects the output pin state.
0	FLAG [1]	Flag. Set when an input capture or a match event in the comparators occurs. 0 FLAG set event has not occurred 1 FLAG set event has occurred

[1] emios_flag_out_*n* reflects the FLAG bit value. When the DMA bit is set, the FLAG bit can be cleared by the eDMA controller.

20.6 Functional Description

The eMIOS provides independent unified channels (UC) that can be configured and accessed by a host MCU. Up to three time bases can be shared by the channels through three counter busses, and each UC can generate its own time base.

When the MCU is reset, all registers are cleared. The eMIOS module is reset on the first positive edge of the clock following the negation of the $\overline{\text{RESET}}$ signal (synchronous reset).

20.6.1 eMIOS Operating Mode Details

The eMIOS operates in one of four modes as determined by the MCU operating mode, plus one module-specific mode. The module must be in either Normal or Debug modes to execute time-based operations. The Disabled, Doze and Stop modes provide reduced power consumption as needed.

20.6.1.1 eMIOS Normal Mode

To perform conversions, the eMIOS must be operating in Normal mode or one of the Debug modes described below. If the MCU is in Run mode, the eMIOS is in Normal mode unless specifically disabled as described in Section 20.6.1.3, “eMIOS Disabled Mode.”

20.6.1.2 eMIOS Debug Mode

If the MCU enters Debug mode, the contents of the MCR and UCCR*n* determine whether the eMIOS UCs continue to operate in Normal mode or enter Debug mode. If MCR[FRZ] = 0 or any UCCR*n*[FREN] = 0 the UC*n* will continue in Normal mode. If the MCU enters Debug mode with MCR[FRZ] = 1 and any UCCR*n*[FREN] = 1, the registers of the associated UC*n*(s) will be frozen and the UC(s) enter Debug Mode. In this mode, all clocks are running and all registers are accessible; thus, this mode is not intended for power saving, but for use during software debugging.

The eMIOS (or an individual UC*n*) exits Debug mode when the MCU exits Debug mode or if the MCR[FRZ] or UCCR*n*[FREN] bits are cleared. After leaving Debug mode, all counters that were frozen upon Debug mode entry will resume at the point where they were frozen.

20.6.1.3 eMIOS Disabled Mode

A mode that is independent of the MCU mode is the eMIOS Disabled mode. At any time, the MDIS bit in the MCR may be set to disable the eMIOS. This mode causes all eMIOS clocks to halt, which causes the module to draw minimal power while all other MCU peripheral modules may continue to operate normally. The MCR, OUDIS and UCDIS registers remain available to be accessed by the core via the peripheral bus. The MDIS bit is intended to be used when the module is not required in the application.

20.6.1.4 eMIOS Doze Mode

If the MCU enters Doze mode, the contents of the MCR[DOZEEN] bit determines whether the eMIOS continues to operate in Normal mode or enters Doze mode. If the DOZEEN bit is cleared, the eMIOS will remain in Normal mode. If the DOZEEN bit is set, the eMIOS will enter Doze mode. eMIOS Doze mode stops the module clocks but leaves the registers accessible, thus offering power savings over operation in Normal mode.

When the MCU exits Doze mode or the DOZEEN bit is cleared, the eMIOS clock is turned on again. All frozen channel actions continue from where they were stopped.

20.6.1.5 eMIOS Stop Mode

If the MCU enters STOP mode, all clocks stop and therefore all modules stop. This mode causes all eMIOS clocks to halt, and thus offers maximum power saving.

After exiting Stop mode, the clocks are turned on again. All frozen channel actions continue from where they were stopped.

20.6.2 IP Bus Interface Unit (BIU)

The BIU provides the interface between the internal interface bus (IIB) and the Peripheral Bus, allowing communication among all submodules and this IP interface.

The BIU allows 8-, 16- and 32-bit accesses. They are performed over a 32-bit data bus in a single cycle clock.

20.6.2.1 Effect of Debug Mode on the BIU

When the FRZ bit in the MCR is set and the module is in debug mode, the operation of BIU is not affected.

20.6.3 Global Clock Prescaler (GCP) Submodule

The Global Clock Prescaler (GCP) divides the f_{IPS} clock to generate a clock for the Clock Prescalers (CPs) of the Unified Channels. It is a programmable 8-bit up counter. The main clock signal is prescaled by the value defined in Table 20-4 according to the GPRE bits in MCR. The output is clocked every time the counter overflows. Counting is enabled by setting the GPREN bit in the MCR. The counter can be stopped at any time by clearing this bit, thereby stopping the internal counter in all Unified Channels. Refer to Section 20.7.3, “Time Base Generation,” for more details.

20.6.3.1 Effect of Debug Mode on the GCP

When the FRZ bit in the MCR register is set and the module is in debug mode, the operation of GCP submodule is not affected, i.e., there is no freeze function in this submodule.

20.6.4 Unified Channel (UC)

Figure 20-11 shows a logical block diagram of a Unified Channel. Each UC*n* consists of:

- A counter bus selector, which sets the time base used by each UC for all timing functions
- A programmable clock prescaler
- Two double buffered data registers, A and B, that allow up to two input capture and/or output compare events to occur without software intervention
- Two match comparators, to compare the selected counter bus with data register values
- An internal counter, which can be used as a local time base or to count input events
- A programmable input filter, which ensures that only valid pin transitions are acted upon
- A programmable input edge detector, which detects rising, falling or both edges
- An output flip-flop, which holds the logic level to be applied to the output pin
- Status and Control registers
- A selector to use the emios_flag_out*n* signal of UC12 to UC15 as the UC*n* output disable
- A Control state machine (FSM)

NOTE

The internal prescalers must be set up before enabling the global prescaler. If the UC*n* prescalers are set after enabling the global prescaler, the internal counters will increment at the same rate, but on different clock cycles. Refer to Section 20.7.3, “Time Base Generation.”

20.6.4.1 Effect of Debug Mode on Unified Channels

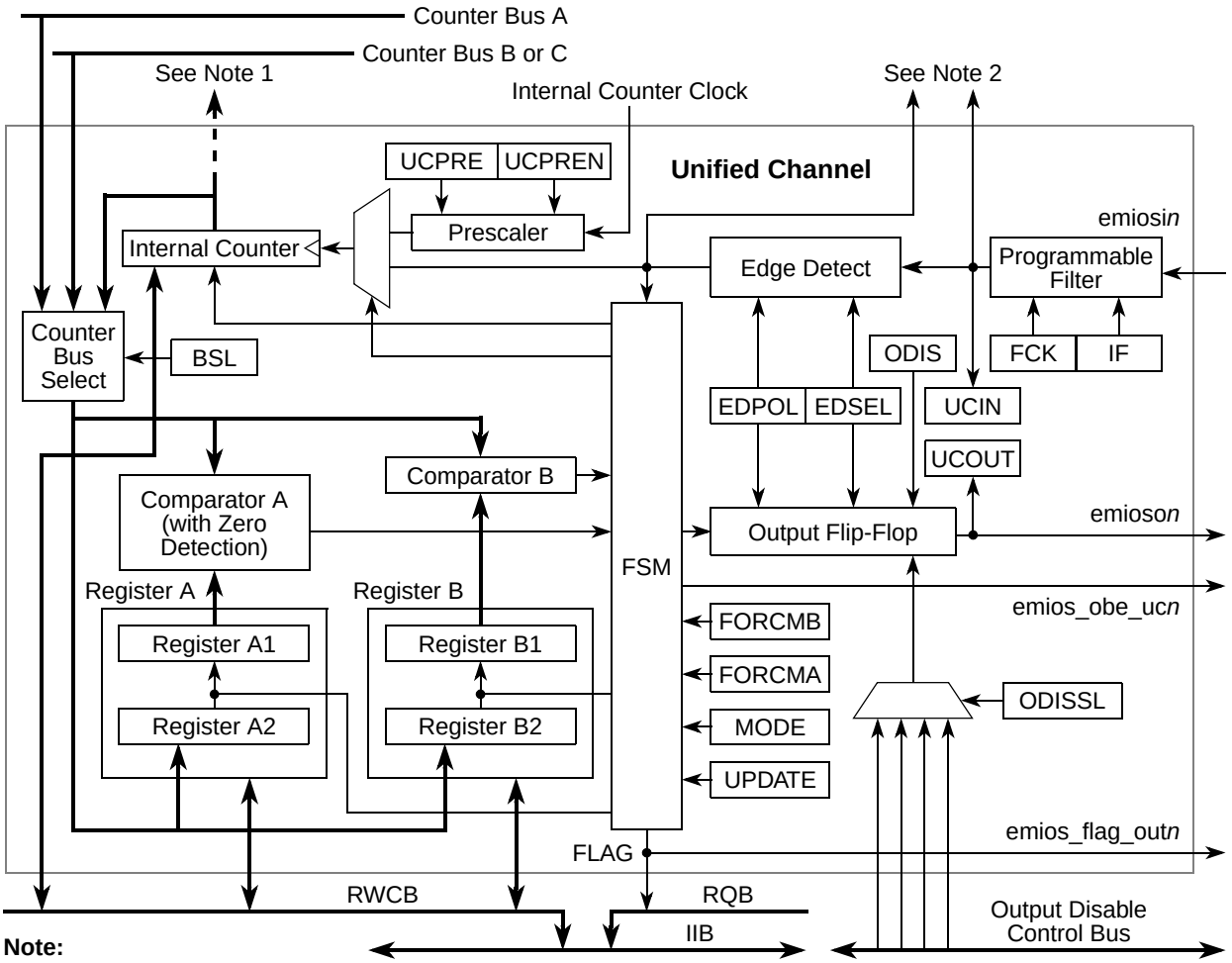
When in debug mode, if both the FRZ bit in the MCR and the FREN bit in the UCCR*n* are set, the internal counter and UC*n* capture and compare functions are halted and the FSM is frozen in the current state.

During freeze, all registers are accessible. When a UC*n* is operating in an output mode, the force match functions remain available, allowing software to force the output to the desired level. For input modes, any input events that may occur while the channel is frozen are ignored.

When exiting debug mode or if either freeze enable bit is cleared (FRZ in the MCR or FREN in the UCCR*n*) the channel actions resume.

20.6.5 Clock Prescaler (CP)

The CP divides the GCP output signal to generate a clock for the internal counter of the Unified Channels. It is a programmable 2-bit down counter. The GCP output signal is prescaled by the value defined in Figure 20-9 according to the UCPRE bits in UCCR*n*. The CP output is toggled every time the counter reaches zero. Counting is enabled by setting the UCPREN bit in the UCCR*n*. The counter can be stopped at any time by clearing this bit, thereby stopping the internal counter in the Unified Channel. Refer to Section 20.7.3, “Time Base Generation,” for more details.



Note:

1 See Figure 20-2 for information on counter bus interconnection.

2 Connects to the FSM of the UC[*n*–1]. These signals are used for QDEC modes.

Figure 20-11. eMIOS Unified Channel Block Diagram

20.6.6 Input Programmable Filter (IPF)

As shown in Figure 20-12, the IPF is a 5-bit programmable up counter that is incremented by the clock source selected by the IF bits in the UCCR*n*.The IPF function ensures that only valid input pin transitions are received by the UC*n* edge detector.

The input signal is synchronized to the f_{IPS} clock. When the synchronizer output changes states, the 5-bit counter starts counting up. As long as the new state is stable on the pin, the counter increments. If a counter overflow occurs, the new pin value is validated. In this case, it is transmitted as a pulse edge to the edge detector. If an opposite edge appears on the pin before validation (overflow), the counter is reset. At the next synchronized pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the edge detector. Figure 20-13 shows an example of input filter operation.

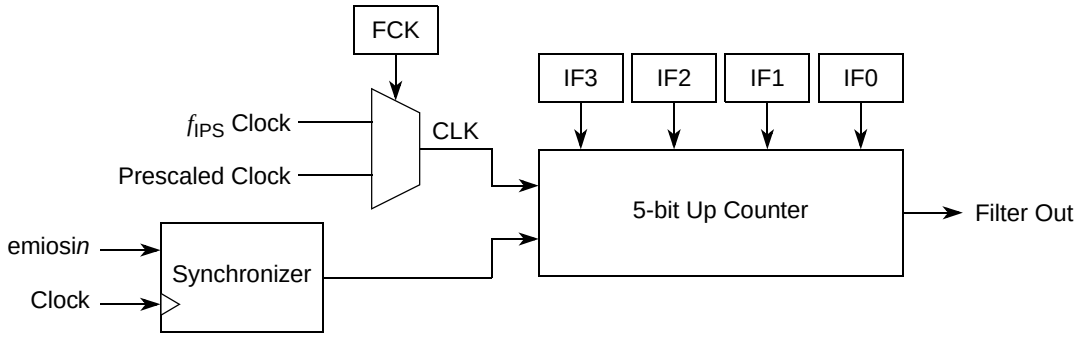


Figure 20-12. eMIOS Input Programmable Filter Submodule Diagram

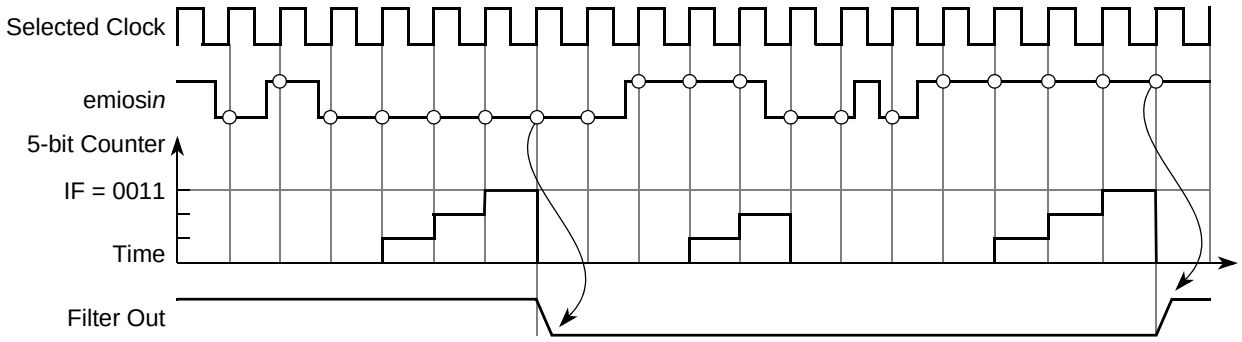


Figure 20-13. eMIOS Input Programmable Filter Example

20.6.7 UC Modes of Operation

The mode of operation of the Unified Channel is determined by the MODE select bits in the UCCR*n* (see Table 20-9 for details). When entering an output mode (except for GPIO mode), the output flip-flop is set to the complement of the EDPOL bit in the UCCR*n*. When changing the MODE field, it is required that the channel is first switched to GPIO mode to reset the internal functions of the UC properly. Failure to do this may lead to invalid and unexpected output compare or input capture results or the FLAG bit being set incorrectly. As the internal counter UCCNT*n* continues to run in all modes (except for GPIO mode), it is possible to use this as a time base if the resource is not used in the current mode.

In order to provide smooth waveform generation while allowing A and B registers to be changed on the fly, the MCB, OPWFMB, OPWMB and OPWMCB modes are provided. In these modes the A and B registers are double buffered. Descriptions of the double-buffered modes are presented separately, since there are several basic differences from the single-buffered MC, OPWFM, OPWM and OPWMC modes.

20.6.7.1 General Purpose Input/Output (GPIO) Mode

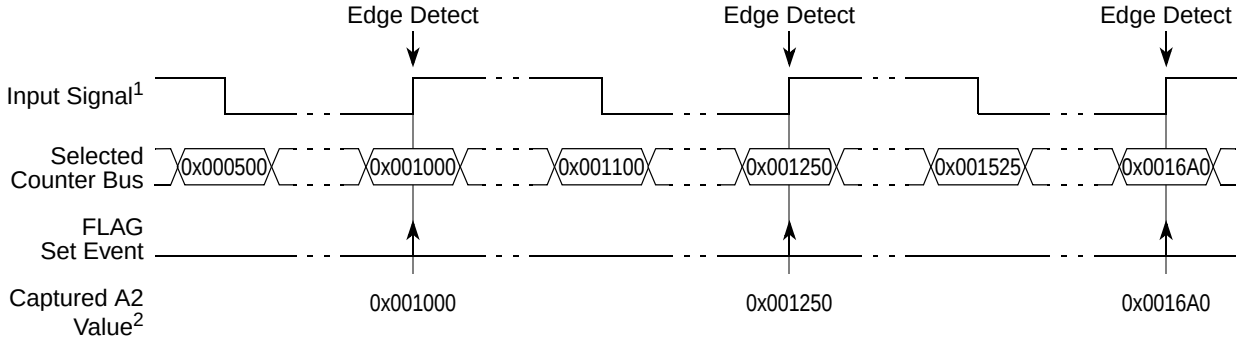
In GPIO mode, all input capture and output compare functions of the UC are disabled, the internal counter (UCCNT*n*) is cleared and disabled. All control bits remain accessible. In order to prepare the UC for a new operation mode, writing to registers UCA*n* or UCB*n* stores the same value in registers A1/A2 or B1/B2, respectively. The MODE[0] bit selects between input (MODE[0] = 0) and output (MODE[0] = 1) modes.

In GPIO input mode, the FLAG generation is determined according to EDPOL and EDSEL bits and the input pin status can be determined by reading the UCIN bit.

In GPIO output mode, the UC*n* is used as a single output port pin and the value of the EDPOL bit is transferred to the output flip-flop when the UCCR*n* is written.

20.6.7.2 Single Action Input Capture (SAIC) Mode

In SAIC mode, when a triggering event occurs on the input pin, the value on the selected time base is captured into register A2. At the same time, the FLAG bit is set to indicate that an input capture has occurred. Reading register UCA*n* returns the value of register A2 The input capture is triggered by a rising, falling or either edge in the input pin, as configured by EDPOL and EDSEL bits in UCCR*n*. Figure 20-14 shows how the Unified Channel can be used for input capture.



Notes: 1. After input filter.
2. Reading UCA*n* returns the value of A2.

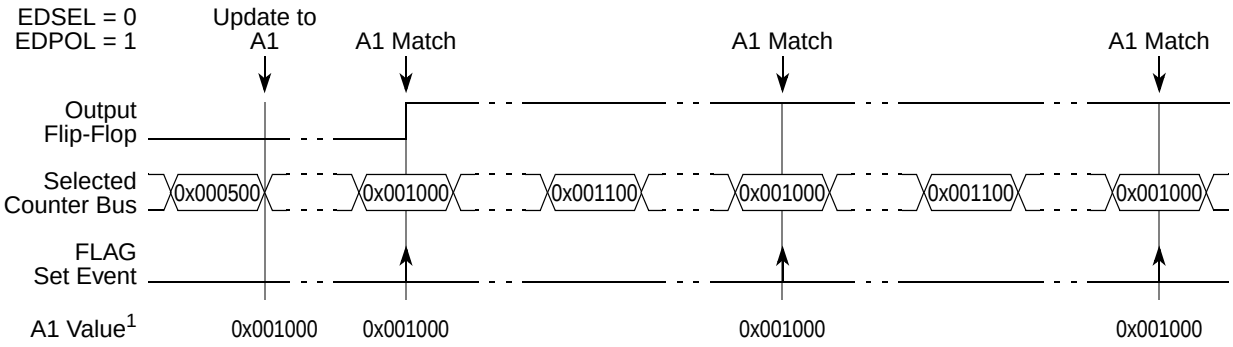
Figure 20-14. eMIOS SAIC Mode Example

20.6.7.3 Single Action Output Compare (SAOC) Mode

In SAOC mode a match value is loaded in register A2 and then transferred to register A1 to be compared with the selected time base. When a match occurs, the EDSEL bit determines if the output flip-flop is toggled or if the value in EDPOL is transferred to the output. At the same time, the FLAG bit is set to indicate that the output compare match has occurred. Writing to register UCA*n* stores the value in register A2 and reading to register UCA*n* returns the value of register A1.

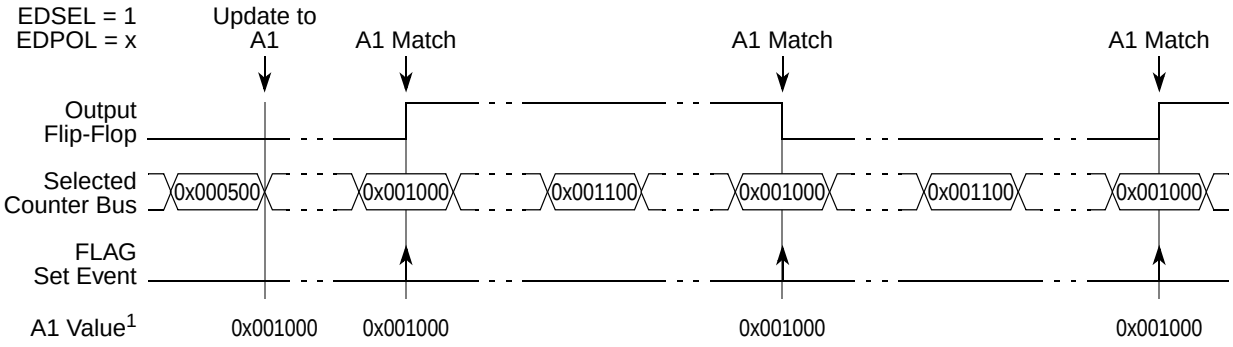
An output compare match can be simulated in software by setting the FORCMA bit in UCCR*n*. In this case, the FLAG bit is not set.

Figure 20-15 and Figure 20-16 show how a Unified Channel can be used to perform a single output compare with EDPOL value being transferred to the output flip-flop and toggling the output flip-flop at each match, respectively.



Notes: 1. Writing UCA*n* loads A2.
A2 value transferred to A1 according to OU*n* bit of the OUDIS register.

Figure 20-15. eMIOS SAOC Mode Example — EDPOL Transferred to Output



Notes: 1. Writing UCA*n* loads register A2.
A2 value transferred to A1 according to OUDIS.OU*n* bit.

Figure 20-16. eMIOS SAOC Mode Example — Toggle Output

20.6.7.4 Input Pulse Width Measurement (IPWM) Mode

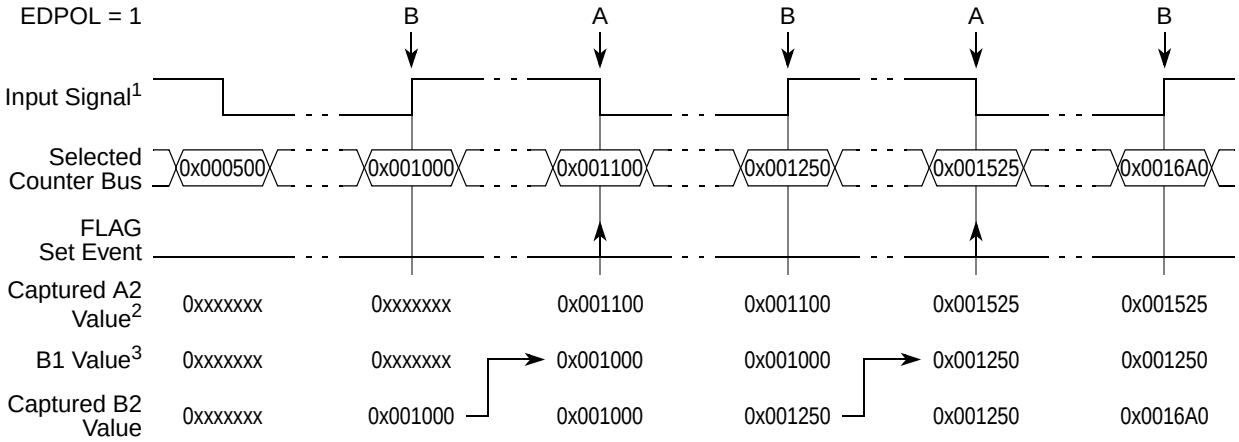
The IPWM mode allows the measurement of the width of a positive or negative pulse by capturing the leading edge on register B1 and the trailing edge on register A2. The input pulse width is calculated by subtracting the value in B1 from A2. Successive captures are done on consecutive edges of opposite polarity. The leading edge sensitivity (i.e., pulse polarity) is selected by EDPOL bit in the UCCR*n*. Registers UCA*n* and UCB*n* return the values in register A2 and B1, respectively.

The capture function of register A2 remains disabled until the first leading edge triggers the first input capture on register B2. When the leading edge is detected, the count value of the selected time base is latched into register B2; the FLAG bit is not set. When the trailing edge is detected, the count value of the selected time base is latched into register A2, the FLAG bit is set and the content of register B2 is transferred to register B1.

If subsequent input capture events occur while the corresponding FLAG bit is set, registers A2 and B1 will be updated with the latest captured values and the FLAG will remain set. Reading registers UCA*n* and UCB*n* return the value in registers A2 and B1, respectively.

In order to guarantee coherent access, reading UCB*n* disables transfers between B2 and B1 until a read of the UCB*n* register has occurred. After the UCB*n* read, transfer is re-enabled.

Figure 20-17 shows how a Unified Channel can be used for input pulse width measurement.



Notes: 1. Output of the input filter.
2. Reading UCA*n* returns the value of A2.
3. Reading UCB*n* returns the value of B1.

Figure 20-17. eMIOS PWM Mode Example

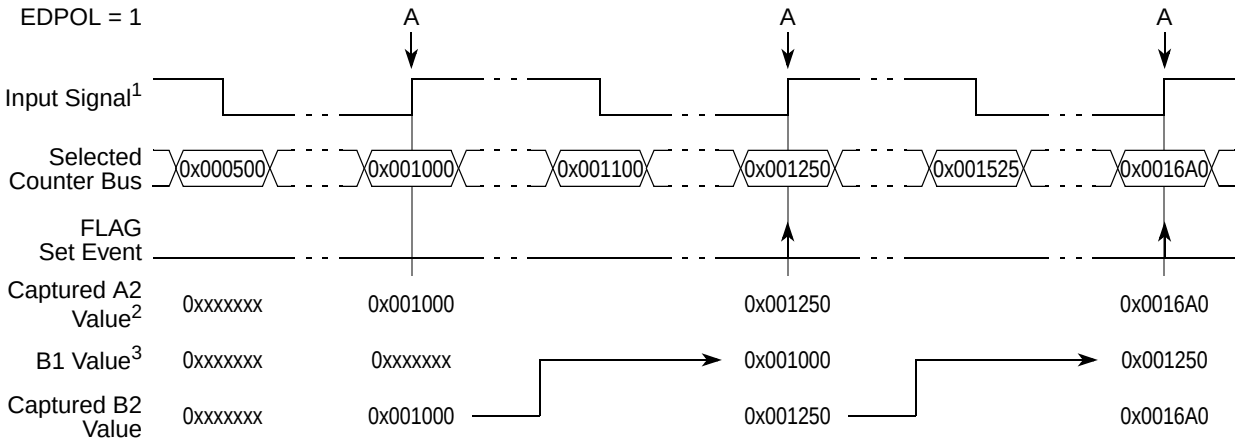
20.6.7.5 Input Period Measurement (IPM) Mode

The IPM mode allows the measurement of the period of an input signal by capturing two consecutive rising edges or two consecutive falling edges. Successive input captures are done on consecutive edges of the same polarity. The edge polarity is defined by the EDPOL bit in the UCCR*n*.

When the first edge of selected polarity is detected, the selected time base is latched into the registers A2 and B2, and the data previously held in register B2 is transferred to register B1. On this first capture the FLAG bit is not set, and the value in register B1 is meaningless.When the second edge of the selected polarity is detected, the counter bus value is latched into registers A2 and B2, the data previously held in register B2 is transferred to data register B1, and the FLAG bit is set to indicate the start and end points of a complete period have been captured. This sequence of events is repeated for each subsequent capture. Registers UCA*n* and UCB*n* return the values in register A2 and B1, respectively. The input pulse period is calculated by subtracting the value in B1 from A2.

In order to guarantee coherent access, reading UCA*n* disables transfers between B2 and B1 until a read of the UCB*n* register has occurred; then any pending transfer is executed.

Figure 20-18 shows how a Unified Channel can be used for input period measurement.



Notes: 1. Output of the input filter.
2. Reading UCA*n* returns the value of A2.
3. Reading UCB*n* returns the value of B1.

Figure 20-18. eMIOS IPM Mode Example

20.6.7.6 Double Action Output Compare (DAOC) Mode

In the DAOC mode the leading and trailing edges of the variable pulse width output are generated by matches occurring on comparators A and B, respectively.

When the DAOC mode is initially selected both comparators are disabled. Comparators A and B are enabled by updating registers A1 and B1, respectively, and remain enabled until a match occurs on that comparator, when it is disabled again. In order to update registers A1 and B1, a write to A2 and B2 must occur and the OUDIS*n* bit must be cleared.

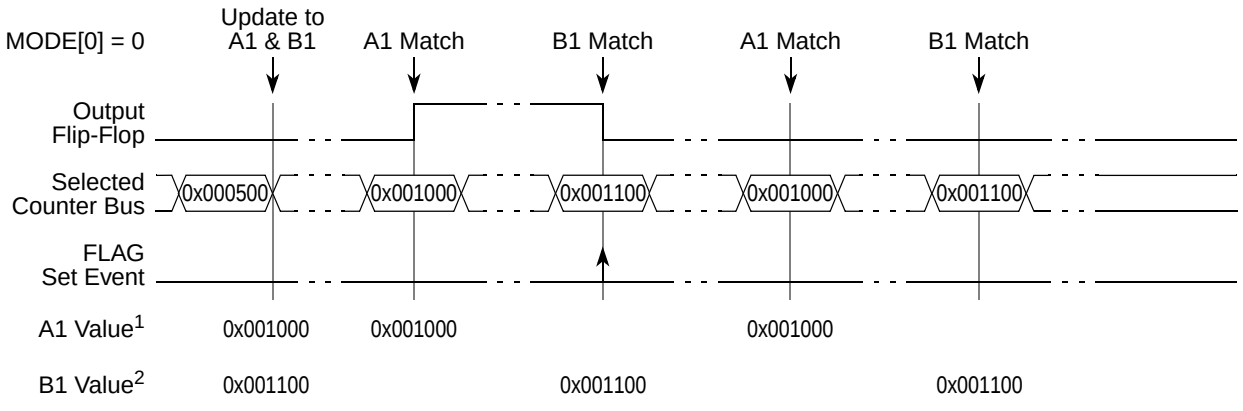
The output flip-flop is set to the value of EDPOL when a match occurs on comparator A and to the complement of EDPOL when a match occurs on comparator B.

MODE[0] determines if the FLAG is set on both matches or just on the second match (see Table 20-9 for details). If subsequent enabled output matches occur on registers A1 and B1, pulses will continue to be generated, regardless of the state of the FLAG bit.

If both registers A1 and B1 are loaded with the same value and the time base reaches that value, the Unified Channel behaves as if a single match on comparator B has occurred (i.e., the output pin will be set to the complement of EDPOL bit and the FLAG bit is set).

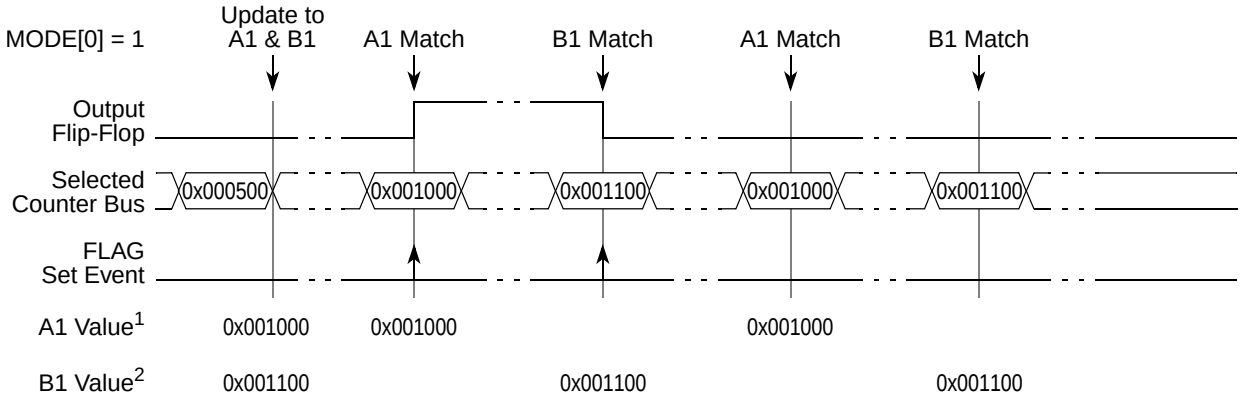
At any time, the FORCMA and FORCMB bits allow software to force the output flip-flop to the level corresponding to a comparison event in comparator A or B, respectively. Note that the FLAG bit is not affected by forced operations.

Figure 20-19 and Figure 20-20 show how the Unified Channel can be used to generate a single output pulse with FLAG bit being set on the second match or on both matches, respectively.



Notes: 1. Writing UCA*n* loads A1. A2 transferred to A1 according to OU*n* bit in the OUDIS register.
2. Writing UCB*n* loads B1. B2 transferred to B1 according to OU*n* bit in the OUDIS register.

Figure 20-19. eMIOS DAOC Mode Example — FLAG Set On Second Match



Notes: 1. Writing UCA*n* loads A1. A2 transferred to A1 according to OU*n* bit in the OUDIS register.
2. Writing UCB*n* loads B1. B2 transferred to B1 according to OU*n* bit in the OUDIS register.

Figure 20-20. eMIOS DAOC Mode Example — FLAG Set On Both Matches

20.6.7.7 Pulse/Edge Accumulation (PEA) Mode

The PEA mode returns the time interval needed to detect a desired number of input events. The MODE[0] bit selects between continuos or single shot operation.

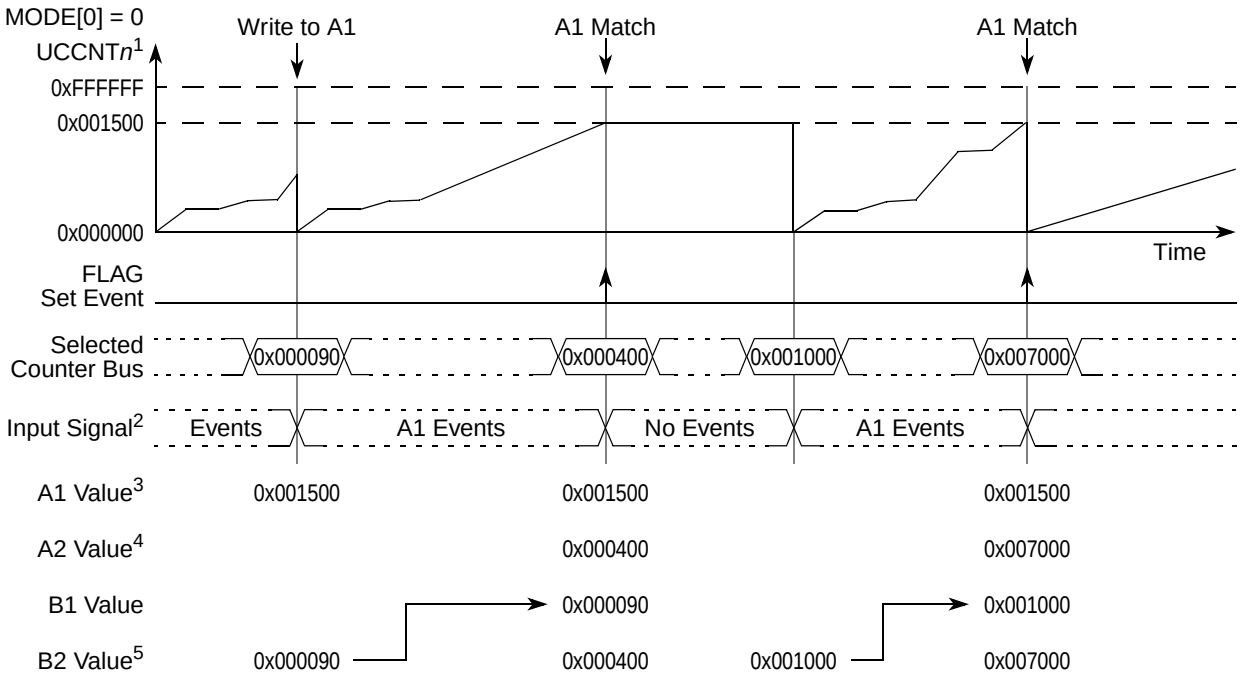
After writing to register A1, the internal counter is cleared on the first input event, ready to start counting input events, and the selected timebase is latched into register B2. On the match between the internal counter and register A1, a counter bus capture is triggered to register A2 and B2. The data previously held in register B2 is transferred to register B1 and the FLAG bit is set to indicate that an event has occurred. Reading registers UCA*n* and UCB*n* return the values in register A2 and B1, respectively. The measured time interval can be determined by subtracting register B1 from A2.

In order to guarantee coherent access, reading UCA*n* disables transfers between B2 and B1 until a read of the UCB*n* register has occurred, then any pending transfer is executed.

Triggering of the counter clock (input event) occurs on rising, falling or both edges of the input pin. The polarity of the triggering edge is selected by the EDSEL and EDPOL bits in UCCR*n*.

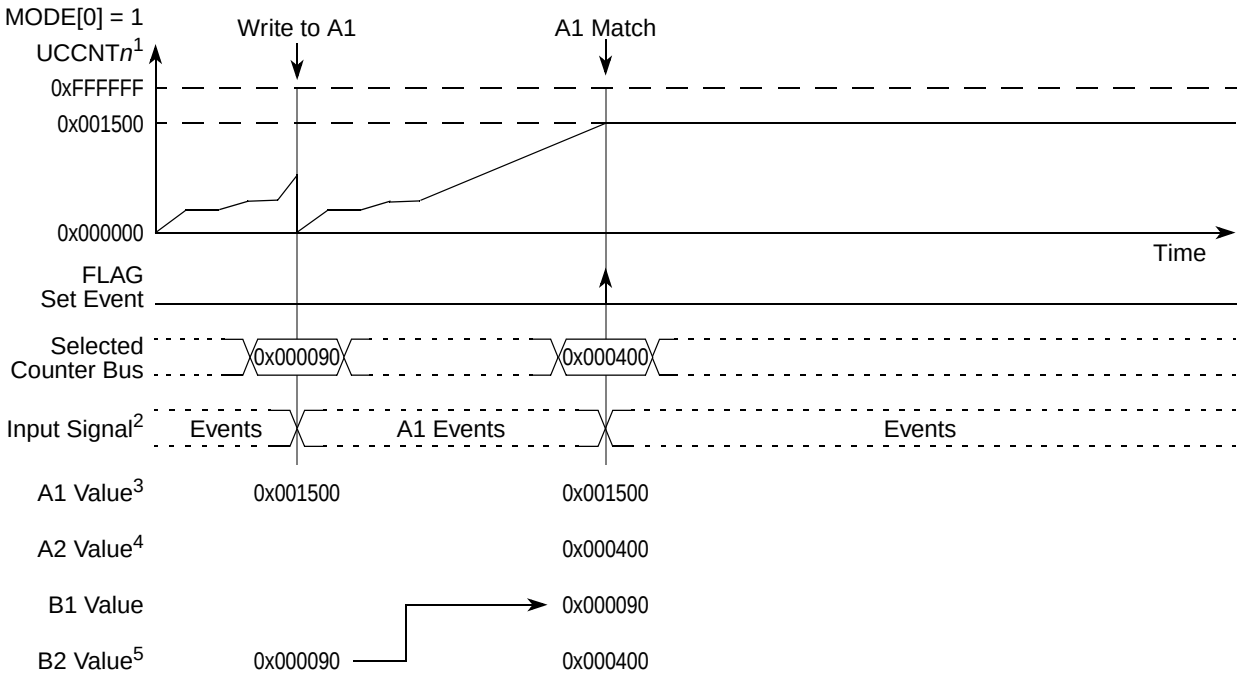
For continuous operation mode (MODE[0] cleared), the counter is cleared on the next input event after a FLAG generation and continues to operate as described above. For single shot operation (MODE[0] set), the counter is not cleared or incremented after a FLAG generation until a new write to register A occurs.

Figure 20-21 and Figure 20-22 shows how a Unified Channel can be used for continuos and single shot pulse/edge accumulation.



Notes: 1. Cleared on first input event after writing register A1.
2. Output of the input filter.
3. Writing UCA*n* loads A1.
4. Reading UCA*n* returns the value of A2.
5. Reading UCB*n* returns the value of B1.

Figure 20-21. eMIOS PEA Mode Example — Continuous Operation



Notes: 1. Cleared on first input event after writing register A1.
2. Output of the input filter.
3. Writing UCA*n* loads A1.
4. Reading UCA*n* returns the value of A2.
5. Reading UCB*n* returns the value of B1.

Figure 20-22. eMIOS PEA Mode Example — Single Shot Operation

20.6.7.8 Pulse/Edge Counting (PEC) Mode

The PEC mode returns the number of pulses or edges detected on the input during time window. MODE[0] bit selects between continuos or single shot operation.

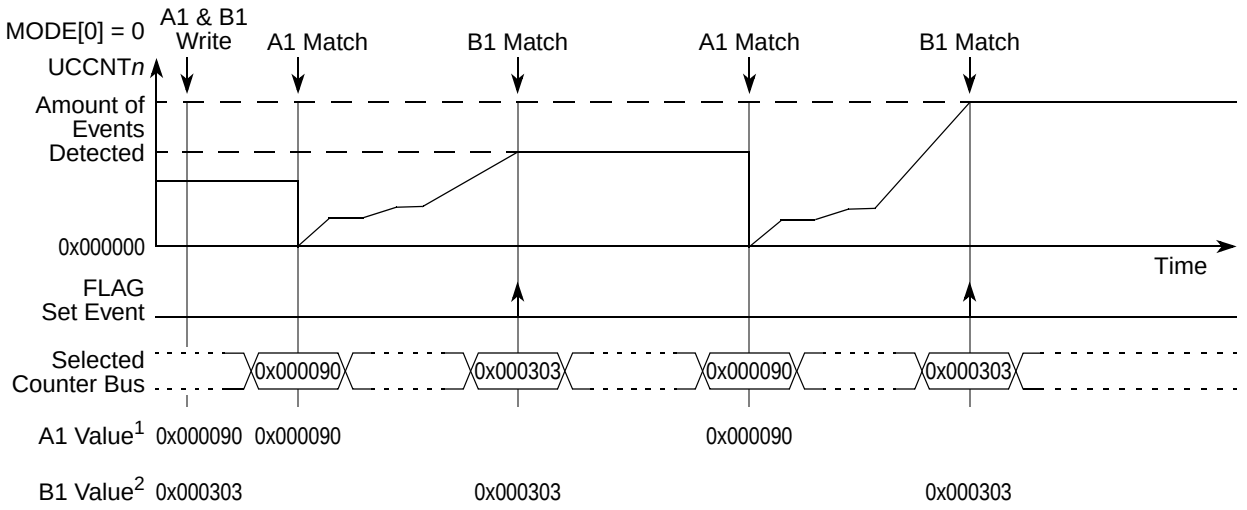
Triggering of the internal counter occurs on rising, falling or both edges on the input signal. The polarity and triggering edge is selected by EDSEL and EDPOL bits in UCCR*n*.

Register A1 holds the start timebase value and register B1 holds the stop timebase value for the period window. After writing to register A1, when a match occur between comparator A and the selected timebase, the internal counter is cleared and it is ready to start counting input events. When the time base matches comparator B, the internal counter is disabled and the FLAG bit is set. Reading the UCCNT*n* returns the number of pulses detected during the window.

For continuous operation (MODE[0] cleared), the next match between comparator A and the selected time base clears the internal counter and counting is enabled again. In order to guarantee the accuracy when reading UCCNT*n* after the FLAG bit is set, software must verify that the time base value is outside of the time interval defined by registers A1 and B1.

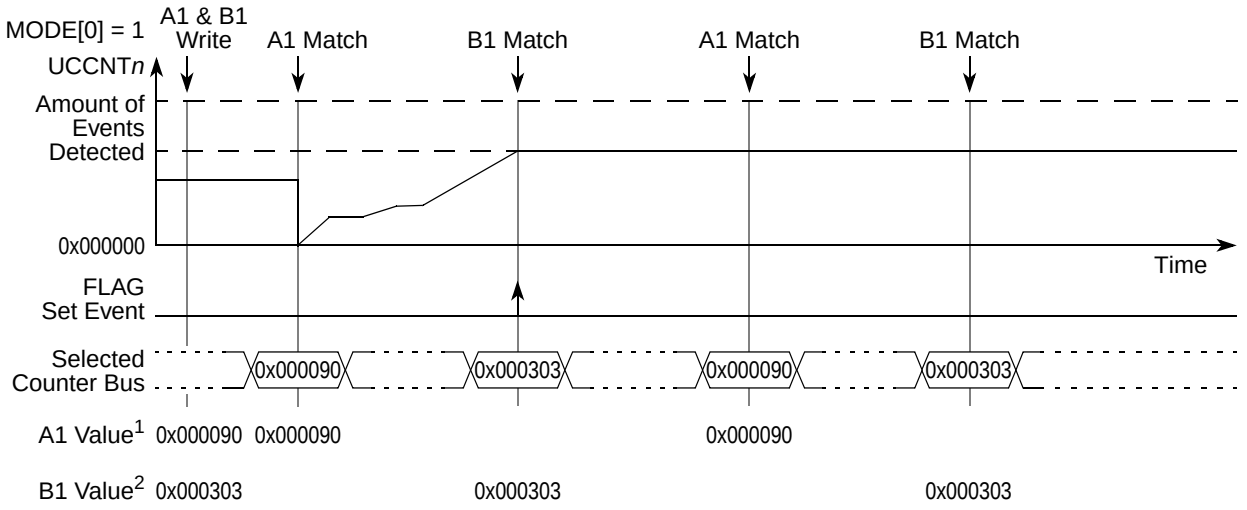
For single shot operation (MODE[0] set), a match between comparator A and the selected time base has no effect until a new write to register A is performed.

Figure 20-23 and Figure 20-24 shows how a Unified Channel can be used for continuous or single shot pulse/edge counting mode.



Notes: 1. Writing UCA*n* loads A1.
2. Writing UCB*n* loads B1.

Figure 20-23. eMIOS PEC Mode Example — Continuous Operation



Notes: 1. Writing UCA*n* loads A1.
2. Writing UCB*n* loads B1.

Figure 20-24. eMIOS PEC Mode Example — Single Shot Operation

20.6.7.9 Quadrature Decode (QDEC) Mode

Quadrature decode mode requires two external signals for either count and direction or phase A/B. When UC*n* is operating in QDEC mode, the input programmable filter (IPF) from UC[*n*–1] is used for the second signal. Note that UC[*n*–1] can be configured for simultaneous operation in a mode that does not use I/O pins, such as modulus counter (MC) mode. The connection among the UCs is circular, i.e., when UC0 is operating in QDEC mode, the UC15 input programmable filter is used.

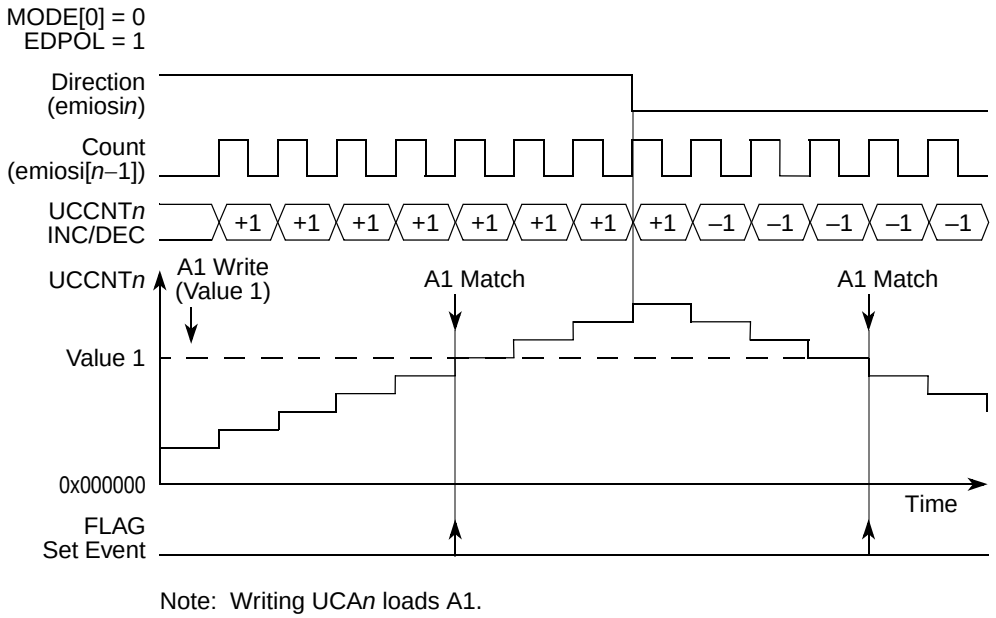
This mode generates a FLAG every time the internal counter matches A1 register. The internal counter is automatically selected and is not cleared when entering this mode.

The MODE[0] bit selects which type of encoder will be used: count & direction encoder or Phase_A & Phase_B encoders.

When operating with a count & direction encoder (MODE[0] cleared), the UC*n* input must be connected to the direction signal and the UC[*n*–1] input must be connected to the count signal of the quadrature encoder. The UC*n* EDPOL bit selects the polarity of the direction signal and the UC[*n*–1] EDPOL bit selects the edge polarity of the count signal used to clock the internal counter.

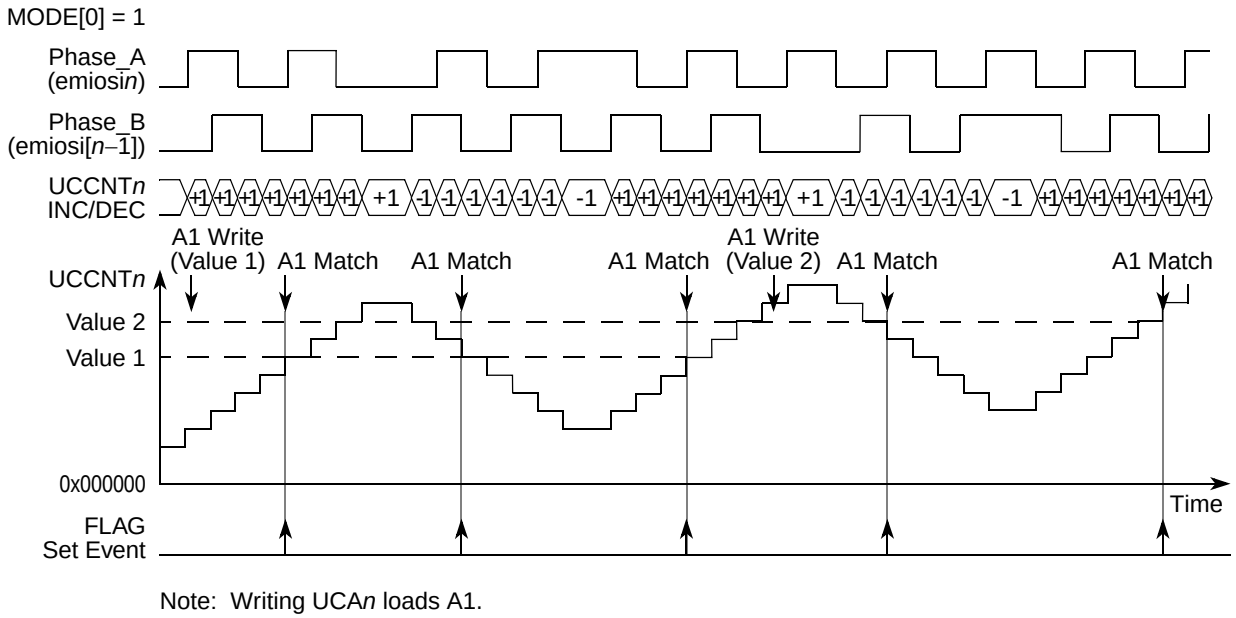
When operating with a Phase_A & Phase_B encoder (MODE[0] set), the UC*n* input must be connected to the Phase_A signal and the UC[*n*–1] input must be connected to the Phase_B signal of the quadrature encoder. The UC*n* EDPOL bit selects the count direction according to the phase difference between Phase_A & Phase_B signals. The UC[*n*–1] EDPOL bit is ignored in this mode.

Figure 20-25 and Figure 20-26 show two Unified Channels configured for quadrature decode mode for count & direction encoder and Phase_A & Phase_B encoders, respectively.



Note: Writing UCA*n* loads A1.

Figure 20-25. eMIOS QDEC Mode Example — Count & Direction Encoder



Note: Writing UCA*n* loads A1.

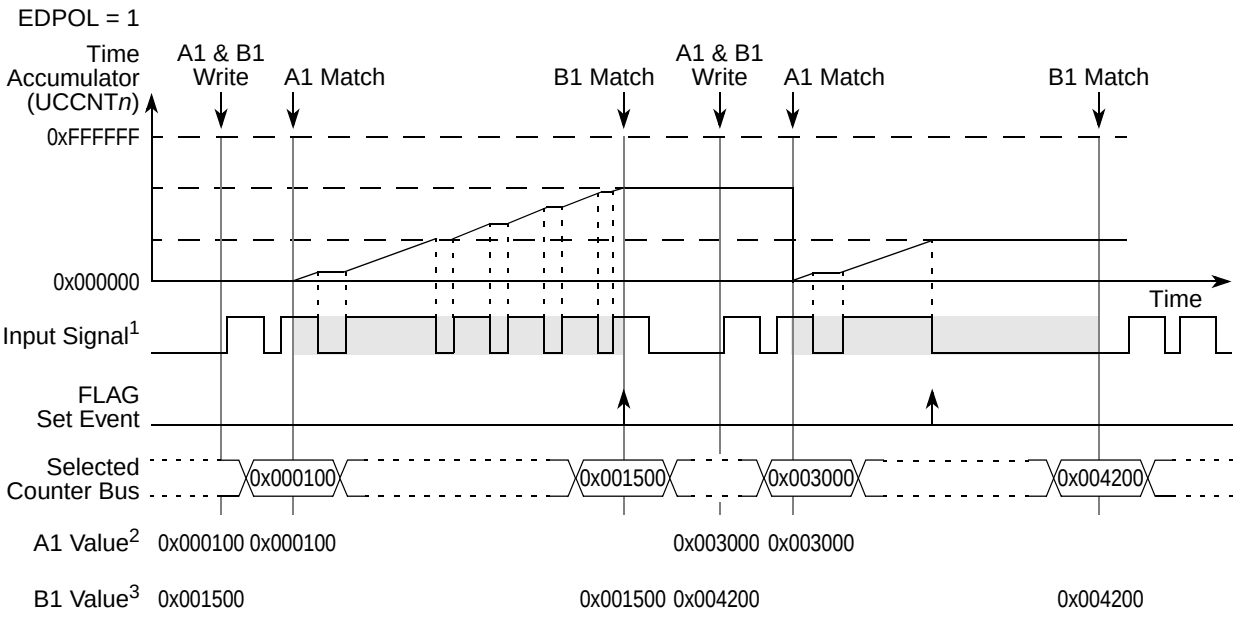
Figure 20-26. eMIOS QDEC Mode Example — Phase_A & Phase_B Encoder

20.6.7.10 Windowed Programmable Time Accumulation (WPTA) Mode

The WPTA mode accumulates the sum of the total high or low time of an input signal during an interval.

The prescaler bits UCPRE of the UCCR*n* define the increment rate of the internal counter. Register A1 holds the start time and register B1 holds the stop time of the time interval. When a match occurs between register A and the selected timebase, the internal counter is cleared and it is ready to start counting. The internal counter is used as a time accumulator, i.e., it counts up when the input signal has the polarity defined by the EDPOL bit in UCCR*n* and does not count otherwise. When a match occurs in comparator B, the internal counter is disabled, regardless of the input signal polarity, and the FLAG bit is set. Reading UCCNT*n* returns the high or low time of the input signal.

Figure 20-27 shows how a Unified Channel can be used to accumulate high time of an input signal.



Notes: 1. Output of the input filter.
2. Writing UCA*n* loads A1.
3. Writing UCB*n* loads B1.

Figure 20-27. eMIOS WPTA Mode Example

20.6.7.11 Modulus Counter (MC) Mode

The modulus counter mode is used to provide a time base for a counter bus or as a general purpose timer.

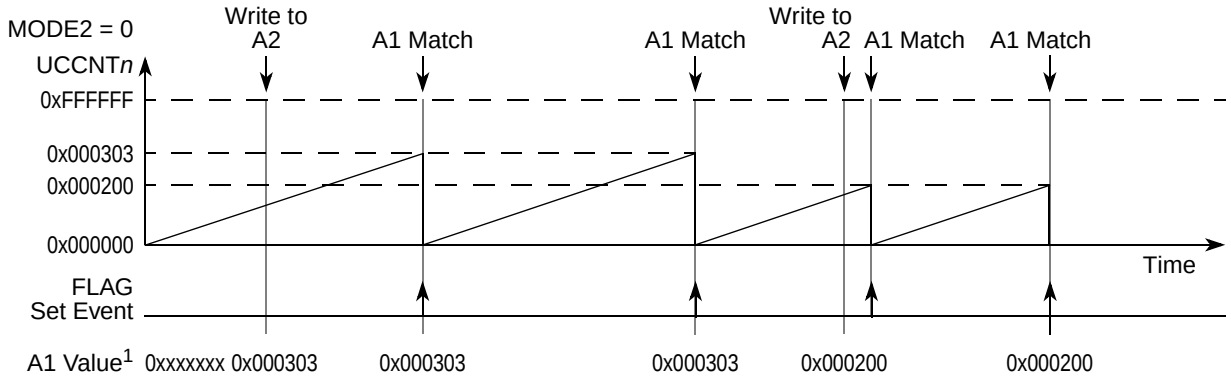
The MODE[0] bit selects an internal or external clock source when cleared or set, respectively. When an external clock is selected, the input signal is used as the source and the triggering polarity edge is selected by the EDPOL and EDSEL bits in the UCCR*n*.

The internal counter counts up from the current value until it matches the value in register A1. Register B1 is cleared and cannot be written, as it is used as the zero match value. The MODE2 bit selects up or up/down mode when cleared or set, respectively.

When in count up mode, a match between the internal counter and register A1 sets the FLAG and clears the internal counter.

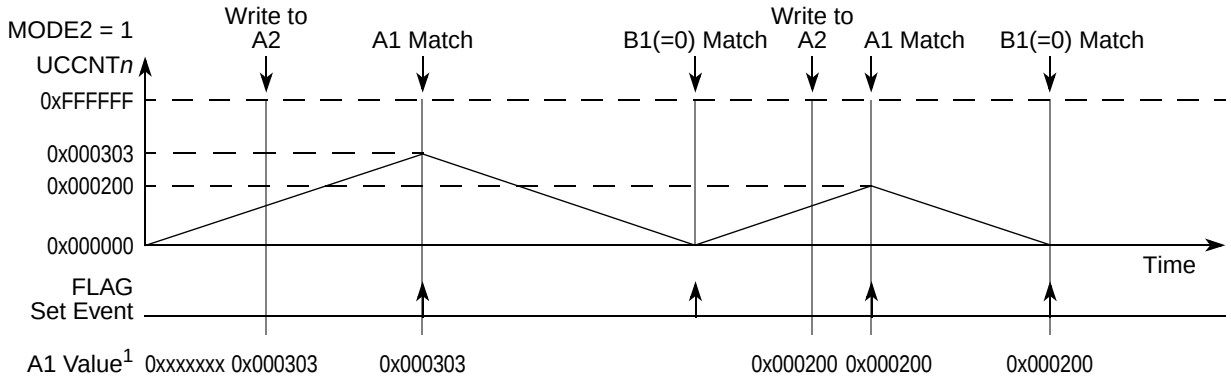
When in count up/down mode, a match between the internal counter and register A1 sets the FLAG and changes the counter direction from increment to decrement. A match between register B1 (always zero) and the internal counter changes the counter direction from decrement to increment. If the MODE[1] bit is set, the FLAG is set on a B1 match.

Figure 20-28 and Figure 20-29 shows how a Unified Channel can be used as modulus counter in up mode and up/down mode, respectively.



Notes: 1. Writing UCA*n* loads A1.
A2 transferred to A1 according to the OU*n* bit of the OUDIS register.

Figure 20-28. eMIOS MC Mode Example — Up Operation



Notes: 1. Writing UCA*n* loads A1.
A2 transferred to A1 according to the OU*n* bit of the OUDIS register.

Figure 20-29. eMIOS MC Mode Example — Up/Down Operation

20.6.7.12 Output Pulse Width and Frequency Modulation (OPWFM) Mode

In this mode, the duty cycle of the output signal is defined by the value in register A1 plus one, and the value in register B1 plus one sets the period. [1] The MODE[0] bit controls the transfer from register B2 to B1, which can be done either immediately to provide the fastest change in the duty cycle (MODE[0] cleared), or at every match of register A1 (MODE[0] set).

The active state of the output flip-flop is the complement of the EDPOL bit. The output flip-flow is active during the duty cycle (from the start of the cycle until a match occurs in comparator A). After a match in comparator A, the output flip-flop is in the inactive state (the value of the EDPOL bit) until the next cycle starts.

The internal counter is automatically selected as the time base, therefore the BSL field in the UCCR*n* is ignored. When a match on comparator A occurs, the output flip-flop is set to the value of the EDPOL bit.

1. Mask set L49P devices operate differently, in that the values that should be loaded into registers A1 and B1 depend on the prescaler value. If prescaler = 1, the value in register B1 + 1 sets the period and the value in register A1 + 1 sets the duty cycle. If prescaler > 1, the value in register B1 sets the period and the value in register A1 sets the duty cycle.

When a match occurs on comparator B, the output flip-flop is set to the complement of the EDPOL bit and the internal counter is cleared. A FLAG event can be generated on match B (MODE[1] cleared), or on both matches (MODE[1] set). If subsequent comparisons occur on comparators A and B, the output flip-flop is set accordingly regardless of the state of the FLAG bit.

At any time, setting the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A or B respectively. Setting FORCMB also clears the internal counter. If FORCMA and FORCMB are set simultaneously, then the action taken depends on the value in register A: if A = 0 the pin is forced to the complement of EDPOL; if A ≠ 0 the pin is forced to the value of EDPOL. The FLAG bit is not set by the FORCMA or FORCMB operations.

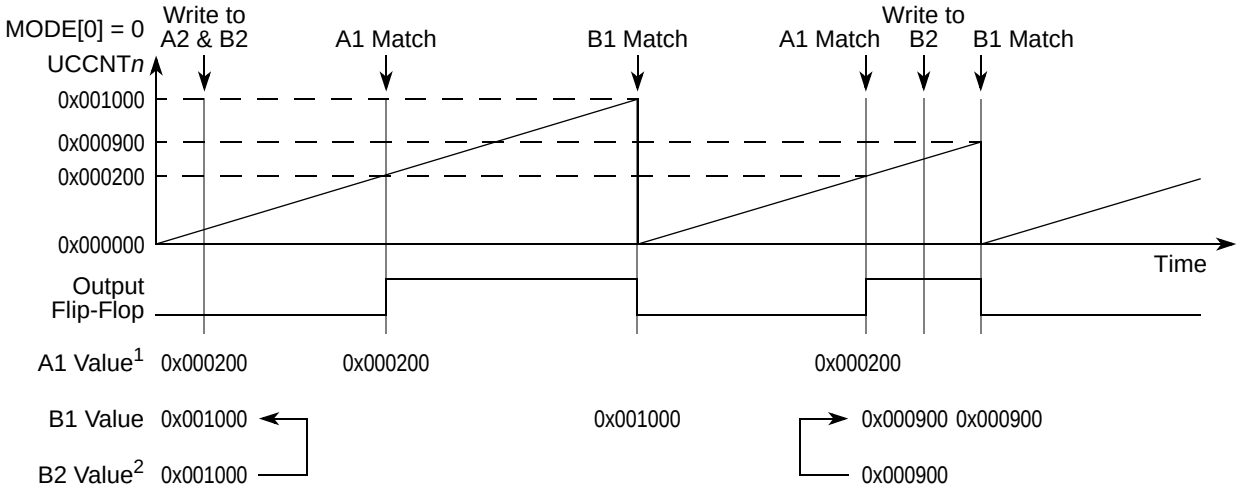
In order to achieve a 0% duty cycle, both registers A1 and B1 must be set to the same value. When a simultaneous match occurs on comparators A and B, the output flip-flop is set at every period to the value of the EDPOL bit. To temporarily change from the current duty cycle to 0% and then return to the current duty cycle, follow this sequence:

1. If not currently stored, read and save the value of register A,
2. Set register A = register B,
3. If immediate 0% duty cycle is desired, set FORCA,
4. To return to the previous duty cycle, restore register A to the saved value.

A 100% duty cycle is achieved by writing 0x000000 to register A. When a match occurs, the output flip-flop is set at every period to the complement of EDPOL bit. The transfer from register B2 to B1 is still controlled by the MODE[0] bit. To temporarily change from the current duty cycle to 100% and then return to the current duty cycle, follow this sequence:

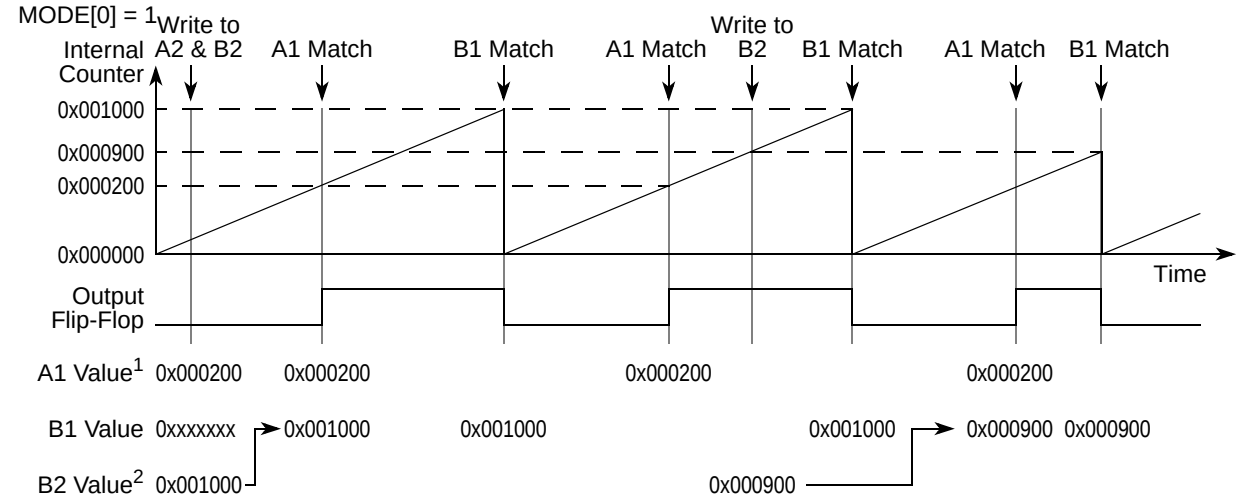
1. If not currently stored, read and save the value of register A,
2. Set register A = 0x000000,
3. If immediate 0% duty cycle is desired, set FORCMB,
4. To return to the previous duty cycle, restore register A to the saved value.

Figure 20-30 and Figure 20-31 show the Unified Channel running in OPWFM mode with immediate and next-period updates, respectively. In both figures, EDPOL = 1, so the output is low during the duty cycle. Table 20-11 shows several examples of OPWFM waveforms.



Notes: 1. Writing UCA*n* loads A1.
2. Writing UCB*n* loads B2.
A2 transferred to A1 according to the OU*n* bit of the OUDIS register.
B2 transferred to B1 according to the OU*n* bit of the OUDIS register.

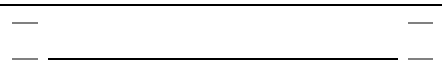
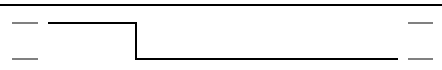
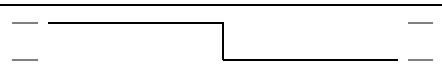
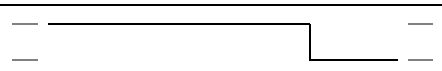
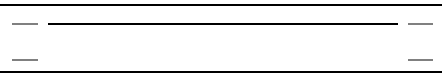
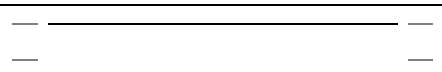
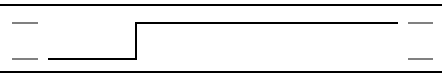
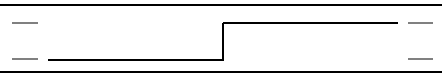
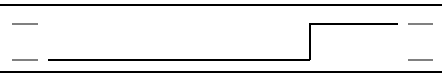
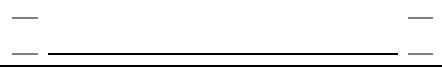
Figure 20-30. eMIOS OPWFM Mode Example — Immediate Update Operation



Notes: 1. Writing UCA*n* loads A1.
2. Writing UCB*n* loads B2.
A2 transferred to A1 according to the OU*n* bit of the OUDIS register.
B2 transferred to B1 according to the OU*n* bit of the OUDIS register.

Figure 20-31. eMIOS OPWFM Mode Example — Next-Period Update Operation

Table 20-11. eMIOS OPWFM Output Waveforms Examples

EDPOL	Duty Cycle	A (decimal)	B (decimal)	Waveform
0 (active high output)	0%	1000	1000	
	25%	250	1000	
	50%	500	1000	
	75%	750	1000	
	100%	0	1000	
1 (active low output)	0%	1000	1000	
	25%	250	1000	
	50%	500	1000	
	75%	750	1000	
	100%	0	1000	

20.6.7.13 Center Aligned Output Pulse Width Modulation (OPWMC) Mode

This mode generates a center aligned PWM with dead time insertion on the leading or trailing edge.

The counter bus selected by the BSL field must be an up/down time base, as shown in Figure 20-29. Register A1 contains the ideal duty cycle for the PWM signal and is compared with the selected time base. Register B1 contains the dead time value and is compared with the internal counter. The internal counter may be driven by the internal prescaler, while the selected up/down time base may be driven by a different prescaler ratio. Unexpected results will be produced if the dead time interval is greater than the duty cycle interval.

For leading edge dead time insertion (MODE[0] set), the output PWM duty cycle is equal to the difference between register A1 and register B1. For trailing edge dead time insertion (MODE[0] clear), the output PWM duty cycle is equal to the sum of register A1 and register B1.

When operating with leading edge dead time insertion, the first match between A1 and the selected time base clears the internal counter and switches the selected time base to the internal counter. When a match occurs between register B1 and the selected time base, the output flip-flop is set to the value of the EDPOL bit and the time base is switched to the selected counter bus. In the next match between register A1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously.

When operating with trailing edge dead time insertion, the first match between A1 and the selected time base sets the output flip-flop to the value of the EDPOL bit. In the next match between register A1 and the selected time base, the internal counter is cleared and the selected time base is switched to the internal counter. When a match occurs between register B1 and the selected time base, the output flip-flop is set to

the complement of the EDPOL bit and the time base is switched to the selected counter bus. This sequence repeats continuously.

A FLAG event can be generated on the trailing edge of the output PWM signal when MODE[1] is cleared, or on both edges when MODE[1] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

At any time, setting the FORCMA or FORCMB bits are equivalent to a match on comparator A or B with the exception that the FLAG bit is not set. When in freeze mode, the FORCMA or FORCMB bits only force the output flip-flop to the level corresponding of a match on A or B, respectively, and do not reset counters or switch the selected time base.

In order to achieve a duty cycle of 100%, both registers A1 and B1 must be set to the same value. When a simultaneous match occurs between the selected time base and registers A1 and B1, the output flip-flop is set at every period to the value of EDPOL bit and the selected time base switches to the selected counter bus, allowing a new cycle to begin at any time, as previously described. A 0% duty cycle is achieved by writing 0x000000 to both registers A1 and B1. When a match occurs, the output flip-flop is set at every period to the complement of EDPOL bit and the selected time base switches to the selected counter bus, allowing a new cycle to begin at any time, as previously described. In both cases, FLAG is generated regardless of MODE[1] bit.

Figure 20-32 and Figure 20-33 shows the Unified Channel running in OPWMC with leading and trailing dead time, respectively.

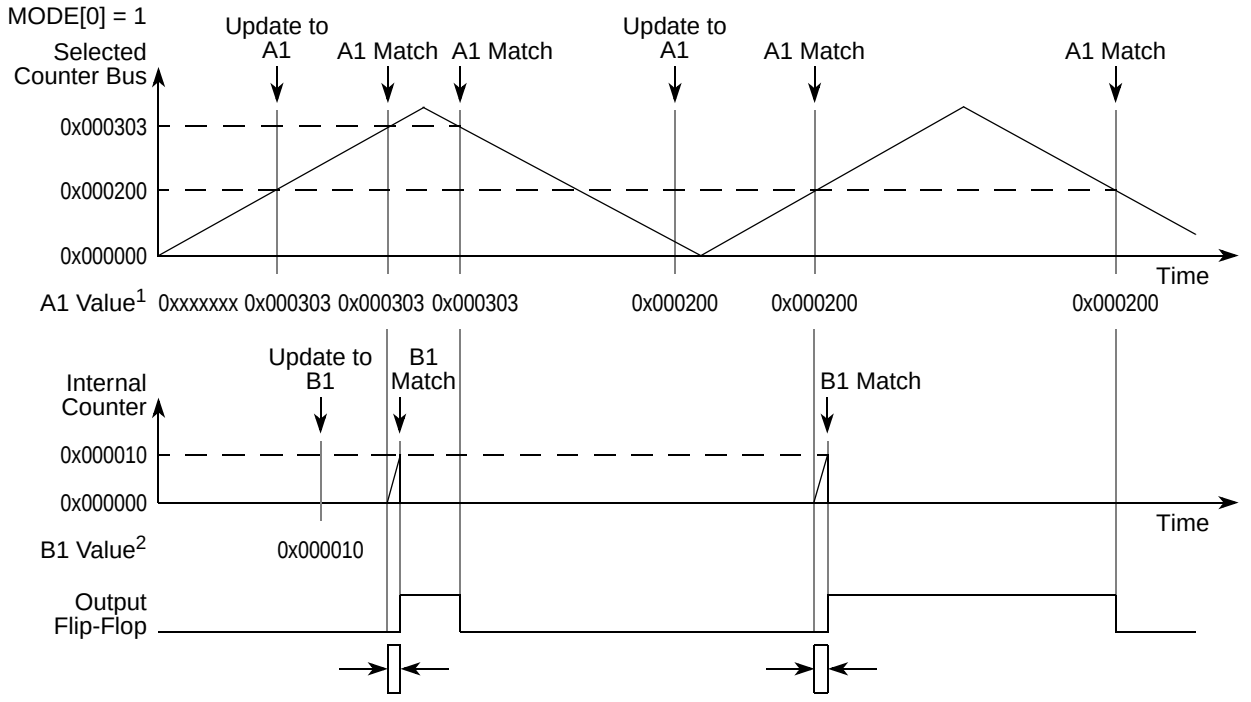
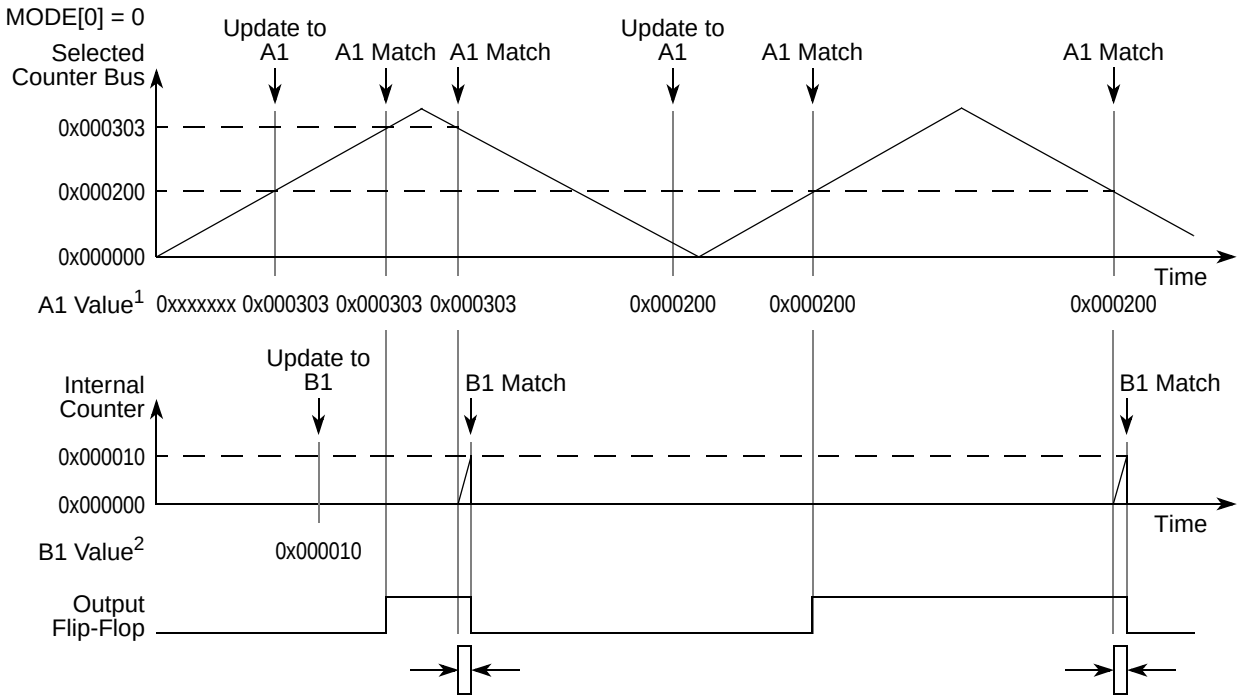


Figure 20-32. eMIOS OPWMC Example — Leading Edge Dead Time



Notes: 1. Writing UCA*n* loads A1. A2 transferred to A1 according to the OU*n* bit of the OUDIS register.
2. Writing UCB*n* loads B1. B2 transferred to B1 according to the OU*n* bit of the OUDIS register.

Figure 20-33. eMIOS OPWMC Example — Trailing Edge Dead Time

20.6.7.14 Output Pulse Width Modulation (OPWM) Mode

This mode generates a simple PWM output signal.

Registers A1 and B1 define the leading and trailing edges of the PWM output pulse, respectively. The MODE[0] bit controls the transfer from register B2 to B1, which can be done either immediately to provide the fastest change in the duty cycle (MODE[0] cleared), or at every match of register A1 (MODE[0] set).

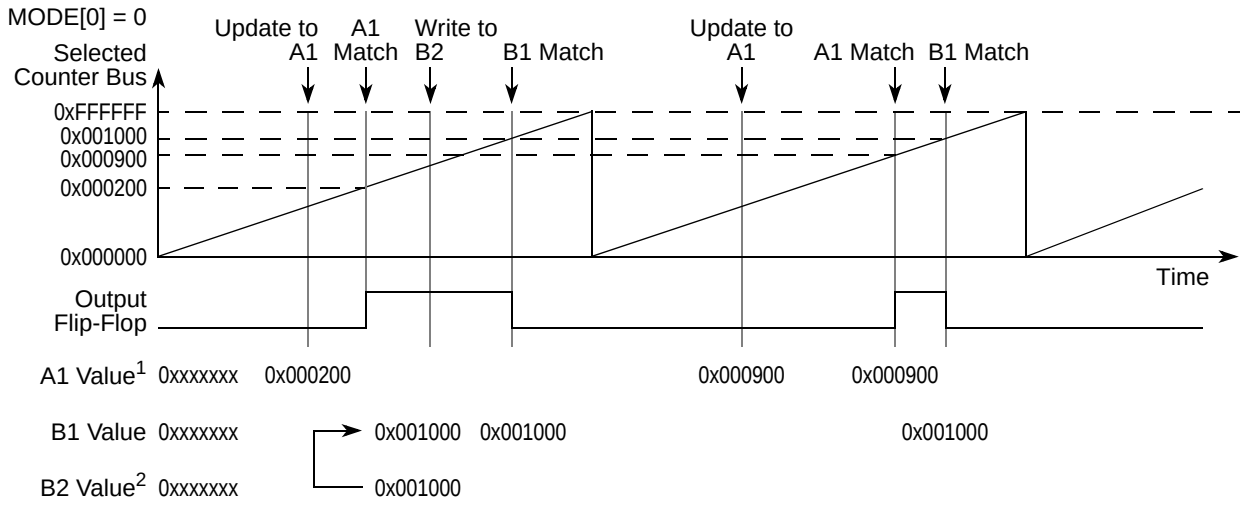
The value loaded in register A1 is compared with the value on the selected time base. When a match on comparator A occurs, the output flip-flop is set to the value of the EDPOL bit. When a match occurs on comparator B, the output flip-flop is set to the complement of the EDPOL bit.

The FLAG bit can be generated on match B (MODE[1] clear) or on both matches (MODE[1] set). If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated regardless of the state of the FLAG bit.

At any time, setting the FORCMA and FORCMB bits allow software to force the output flip-flop to the level corresponding to a match on A or B respectively. Note that the FLAG bit is not set by FORCMA and FORCMB set operations.

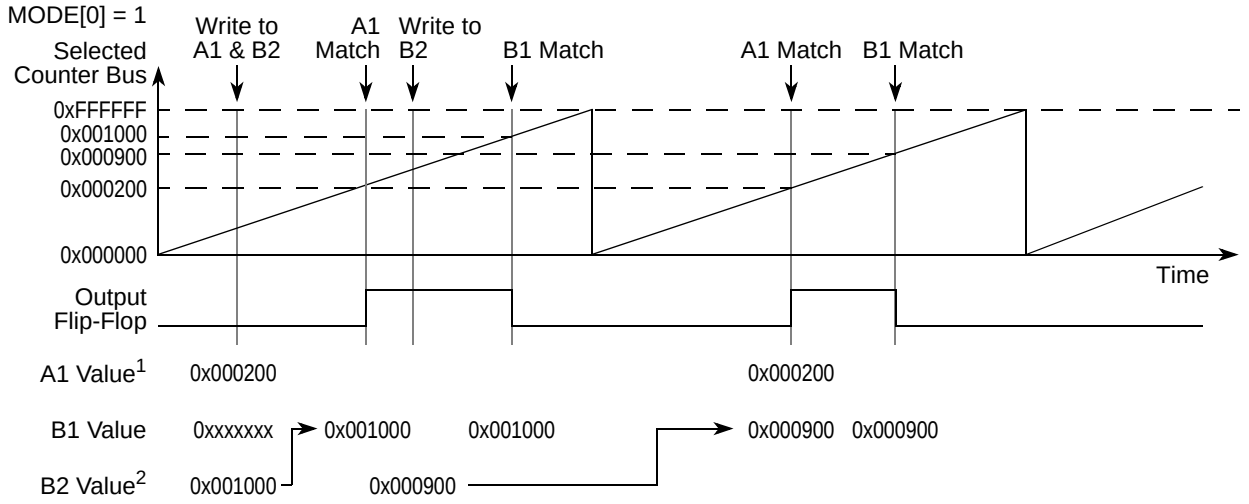
In order to achieve 100% duty cycle, both registers A1 and B1 must be set to the same value. When simultaneous matches on comparators A and B occur, the output flip-flop is set at every period to the value of EDPOL bit. A 0% duty cycle is achieved by writing 0x000000 to both registers A1 and B1. When a match occurs, the output flip-flop is set at every period to the complement of EDPOL bit. The transfer from register B2 to B1 is controlled by MODE[0] bit in all cases.

Figure 20-34 and Figure 20-35 show the Unified Channel running in OPWM with immediate update and next period update, respectively.



Notes: 1. Writing UCA*n* loads A1.
2. Writing UCB*n* loads B2.
A2 transferred to A1 according to the OU*n* bit of the OUDIS register.
B2 transferred to B1 according to the OU*n* bit of the OUDIS register.

Figure 20-34. eMIOS OPWM Mode Example — Immediate Update



Notes: 1. Writing UCA*n* loads A1.
2. Writing UCB*n* loads B2.
A2 transferred to A1 according to the OU*n* bit of the OUDIS register.
B2 transferred to B1 according to the OU*n* bit of the OUDIS register.

Figure 20-35. eMIOS OPWM Mode Example — Next Period Update

20.6.7.15 Modulus Counter, Buffered (MCB) Mode

The MCB mode provides a time base which can be shared with other channels through the internal counter buses. Register A1 is double buffered, thus allowing smooth transitions between cycles when changing the

A2 register value on the fly. The A1 register is updated at the cycle boundary, which is defined as when the internal counter reaches the value one. Note that the internal counter values are within a range from one up to register A1 value in MCB mode.

The MODE[0] bit selects the internal clock source if clear or external if set. When an external clock is selected, the channel input pin is used as the channel clock source. The active edge of this clock is defined by EDPOL and EDSEL bits in the MTSC channel register.

When entering the MCB mode, if up counter is selected (MODE[2] = 0), the internal counter starts counting up from its current value to until an A1 match occurs. On the next f_{IPS} clock cycle after an A1 match occurs, the internal counter is set to one and the counter continues counting up. If up/down mode is selected (MODE[2] = 1), the counter changes direction at the A1 match and counts down until it reaches one and is then set to count up again. In this mode B1 is set to one and cannot be changed, as it is used to generate a match to switch from down count to up count.

Note that versus the MC mode, the MCB mode counts between one and the A1 register value. The counter cycle period in up count mode is equal to the A1 value. In up/down counter mode the period is defined by the formula: (2 × A1) – 2.

Figure 20-36 illustrates the counter cycle for several A1 values. Register A1 is loaded with the A2 value at the cycle boundary. Thus any value written to A2 within cycle (n) will be updated to A1 at the next cycle boundary, and therefore will be used on cycle (n+1). The cycle boundary between cycle (n) and cycle (n+1) is defined as the first clock cycle of cycle (n+1). Note that flags are set when A1 matches occur.

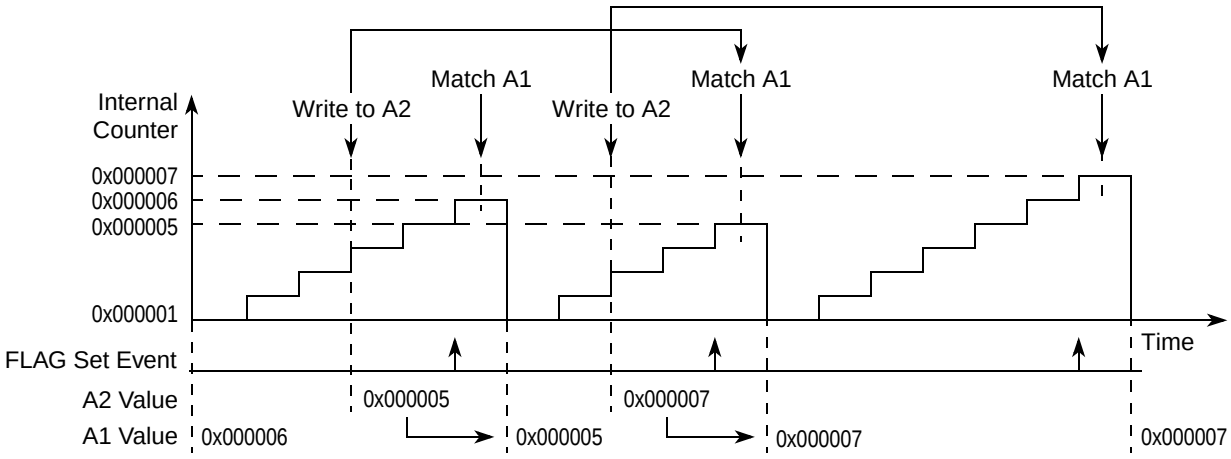


Figure 20-36. eMIOS MCB Mode Example — Up Operation

NOTE

If a prescaler greater than 1 is used, there are several f_{IPS} clock cycles between when the flag is asserted and the counter is set to one. This should be considered when the A value is changed in every cycle, since A1 is updated on the cycle boundary, which is after the flag is set.

Figure 20-37 illustrates the MCB up/down counter mode. The A1 register is updated at the cycle boundary. If A2 is written in cycle (n), this new value will be used in cycle (n+1) for the next A1 match.

Flags are generated only at an A1 match if MODE[1] is 0. If MODE[1] is 1, flags are also generated at the cycle boundary.

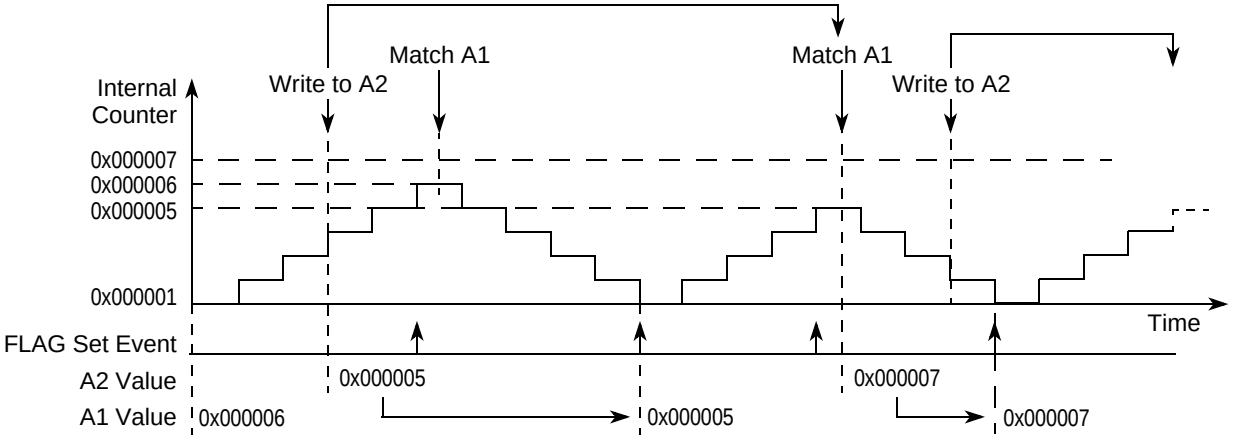


Figure 20-37. eMIOS MCB Mode Example — Up/Down Operation

Figure 20-38 provides a more detailed illustration of the A1 update process in up counter mode. The A1 load signal is generated based on the detection of the internal counter reaching one, and has the duration of one f_{IPS} clock cycle. Note that during the load pulse A1 still holds its previous value. It is actually updated at the second f_{IPS} clock cycle.

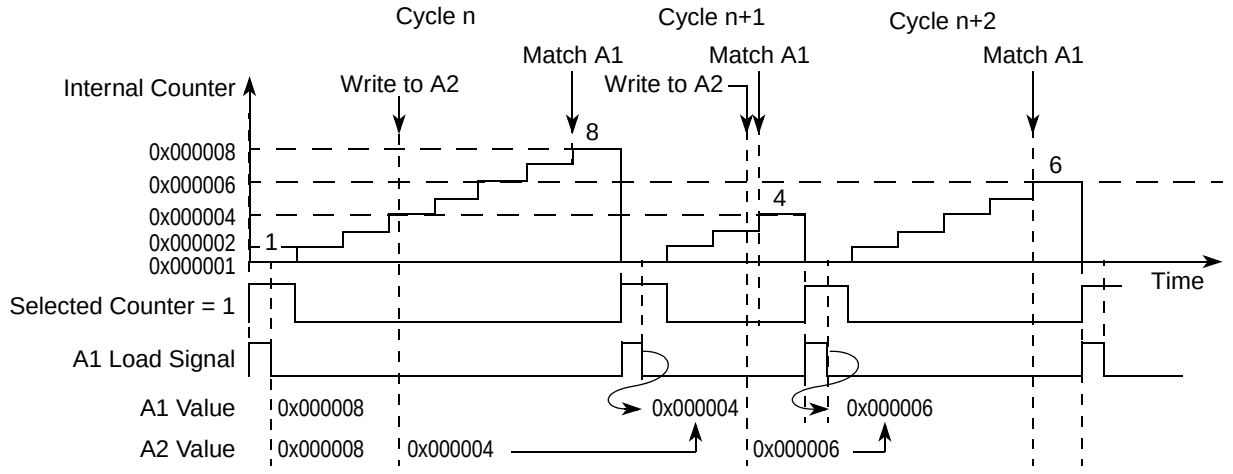


Figure 20-38. eMIOS MCB Mode Example — Up Operation A1 Register Update

Figure 20-39 illustrates the A1 register update process in up/down counter mode. Note that A2 can be written at any time within cycle (n) in order to be used in cycle (n+1). Thus A1 receives the new value at the next cycle boundary. The OUDIS[n] bits can be used to disable the update of A1 register.

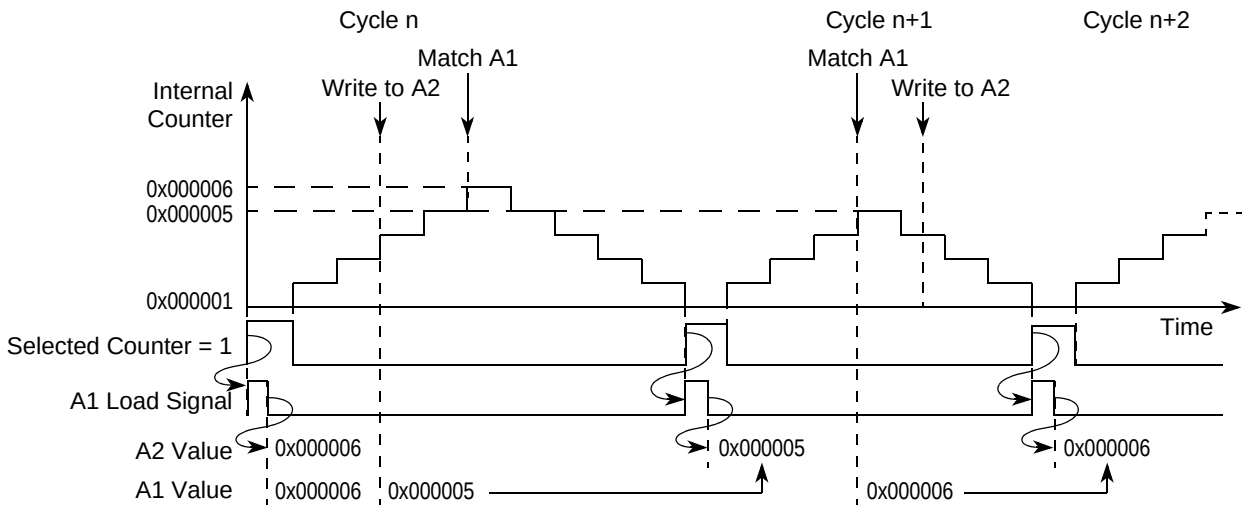


Figure 20-39. eMIOS MCB Mode Example — Up/Down Operation A1 Register Update

20.6.7.16 Output Pulse Width and Frequency Modulation, Buffered (OPWFMB) Mode

This mode generates waveforms with variable duty cycle and frequency. The internal channel counter is automatically selected as the time base, A1 sets the duty cycle and B1 determines the frequency. Both A1 and B1 are double buffered to allow smooth signal generation when changing the register values on the fly. 0% and 100% duty cycles are supported.

In order to provide smooth and consistent channel operation, this mode differs substantially from the OPWFM mode. The main differences are in how A1 and B1 are updated, the delay from the A1 match to the output pin transition, and the range of the internal counter which ranges from 1 up to B1 value.

When a match on comparator A occurs, the output register is set to the value of EDPOL. When a match on comparator B occurs, the output register is set to the complement of EDPOL. A B1 match also causes the internal counter to transition to 1, thus re-starting the counter cycle.

Figure 20-40 shows an example of OPWFMB mode operation. Note that the output pin transition occurs when the A1 or B1 match signal is negated, as detected by the negative edge of the A1 match signal. For example, if register A1 is set to 0x000004, the output pin transitions 4 counter periods after the cycle starts, plus one f_{IPS} clock cycle. Note that in the example shown in Figure 20-40 the prescaler ratio is set to two (refer to Section 20.7.3, “Time Base Generation”).

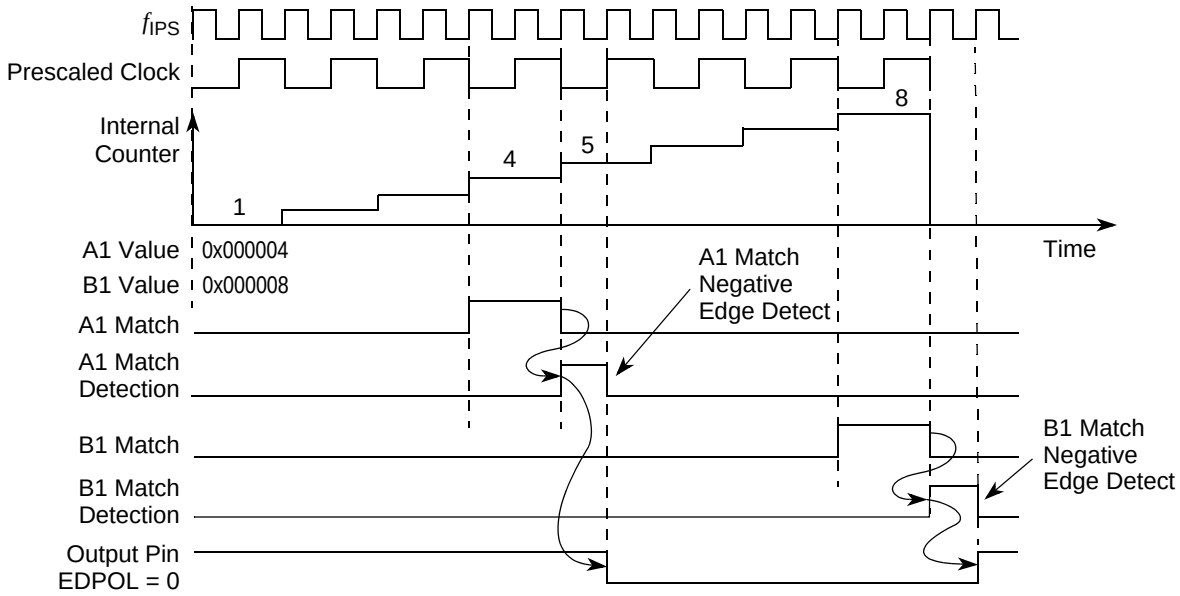


Figure 20-40. eMIOS OPWFMB Mode Example — A1/B1 Match to Output Register Delay

Figure 20-41 shows the generated output signal if A1 is 0. Since the counter does not reach zero in this mode, the channel internal logic infers a match as if A1 = 1, with the difference that in this case the positive edge of the match signal is used to trigger the output pin transition instead of the positive edge that is used when A1 = 1. Note that the A1 positive edge match signal from cycle (n+1) occurs at the same time as the B1 match negative edge from cycle (n). This allows the use of the A1 match positive edge to mask the B1 match negative edge when they occur at the same time. The result is that no transition occurs on the output flip-flop, and a 0% duty cycle is generated.

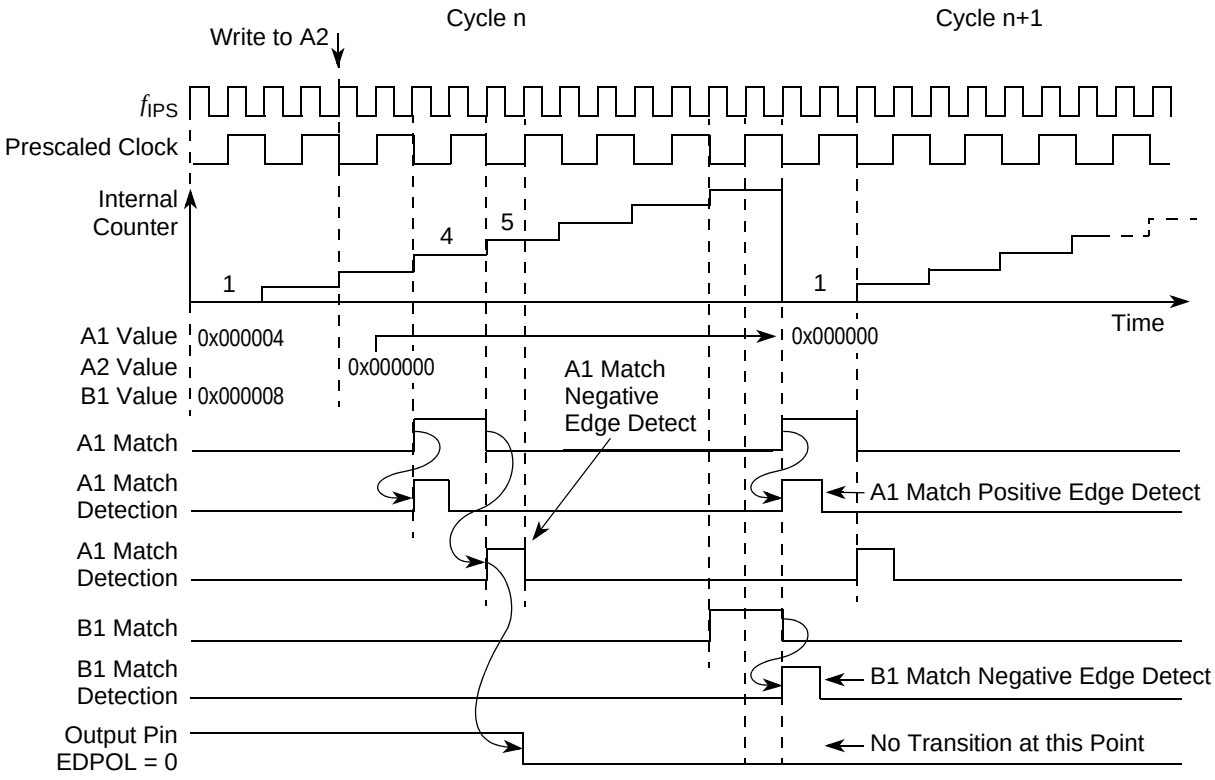


Figure 20-41. eMIOS OPWFMB Mode Example — A1 = 0 (0% Duty Cycle)

Figure 20-42 shows the timing for the A1 and B1 loading. A1 and B1 use the same signal to trigger a load, which is generated based on the selected counter reaching one. This event is defined as the cycle boundary. The load signal pulse has the duration of one f_{IPS} clock cycle and occurs at the first f_{IPS} clock period of every counter cycle. If A2 and B2 are written within cycle (n), their values are loaded into A1 and B1, respectively, at the first clock of cycle (n+1). The update disable bits, OUDIS, can be used to control the update of these registers, thus allowing the delay of A1 and B1 update for synchronization purposes.

During the load pulse A1 still holds its old value, which is updated on the following f_{IPS} clock cycle. During the A1 load pulse, an internal by-pass allows the use of A2 instead of A1 for matches if A2 is either 0 or 1, thus allowing matches to be generated even when A1 is being loaded. This approach allows a uniform channel operation for any A2 value, including 1 and 0.

In Figure 20-42 it is assumed that the channel and global prescalers are set to one, meaning that the channel internal counter transition at every f_{IPS} clock cycle. FLAGs can be generated only on B1 matches when MODE[1] is cleared, or on both A1 and B1 matches when MODE[1] is set. Since B1 FLAG occurs at the cycle boundary, this flag can be used to indicate that A2 or B2 data written on cycle (n) were loaded to A1 or B1, respectively, thus generating matches in cycle (n+1).

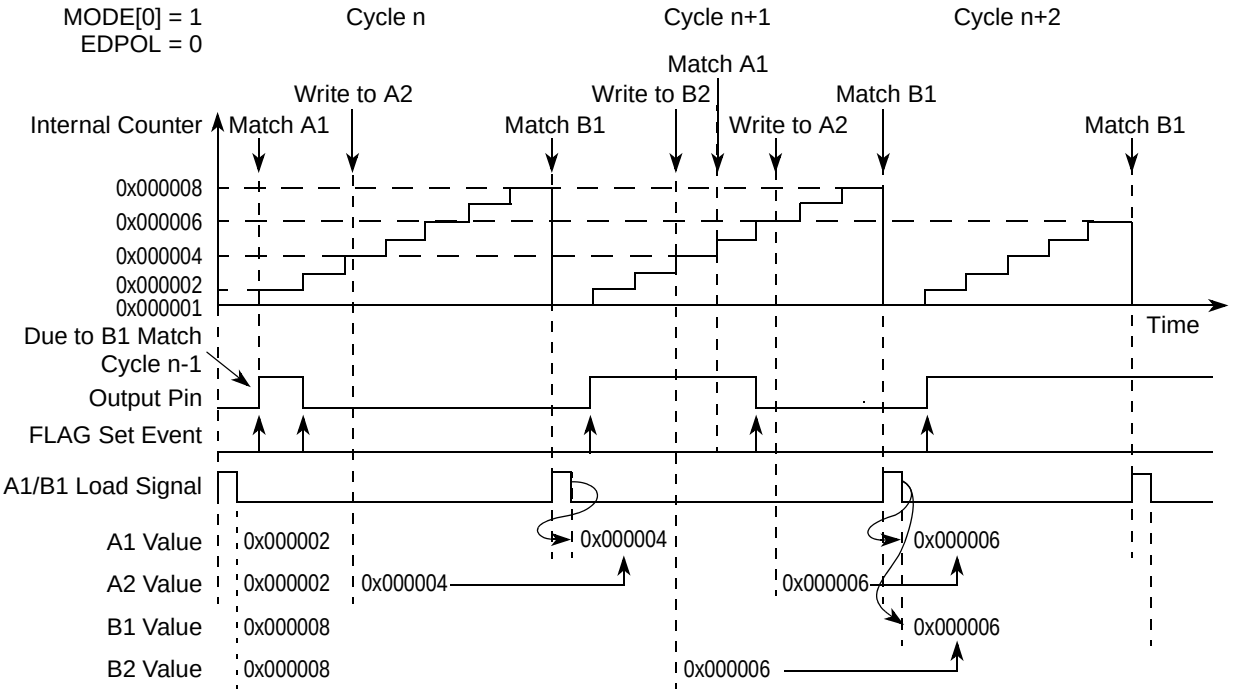


Figure 20-42. eMIOS OPWFMB Mode Example — A1/B1 Updates and Flags

Figure 20-43 shows the operation of the Output Disable feature in OPWFMB mode. Unlike OPWFM mode, the output disable forces the channel output flip-flop to the EDPOL bit value. This functionality targets applications that use active high signals and a high to low transition at A1 match. For such cases EDPOL should be 0.

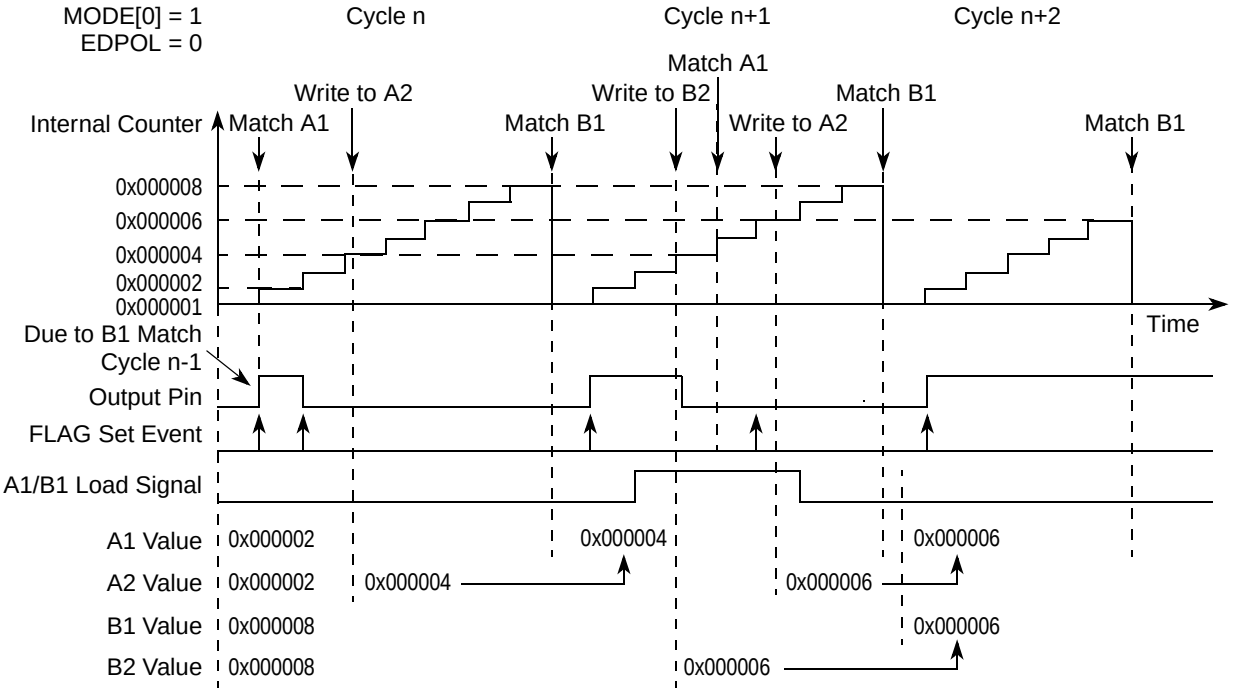


Figure 20-43. eMIOS OPWFMB Mode Example — Active Output Disable

Note that the output disable has a synchronous operation, meaning that the assertion of the Output Disable input signal causes the channel output flip-flop to transition to EDPOL at the next f_{IPS} clock cycle. If the Output Disable input is negated, the output pin transitions at the following A1 or B1 match.

In Figure 20-43 it is assumed that the Output Disable input is enabled and selected for the channel (refer to Section 20.5.1.8, “eMIOS Channel Control Registers (UCCRn),” for a detailed description of the ODIS and ODISSL bits and selection of the Output Disable inputs).

The FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on comparators A or B respectively. Similar to a B1 match, FORCMB clears the internal counter. The FLAG bit is not set when the FORCMA or FORCMB bits are set.

Figure 20-44 illustrates the generation of 100% and 0% duty cycle signals. It is assumed that EDPOL = 0 and the prescaler ratio is 1. Initially A1 = 0x000008 and B1 = 0x000008. In this case, a B1 match has precedence over an A1 match, thus the output flip-flop is set to the complement of EDPOL. This cycle corresponds to a 100% duty cycle signal. The same output signal can be generated for any A1 value greater than or equal to B1.

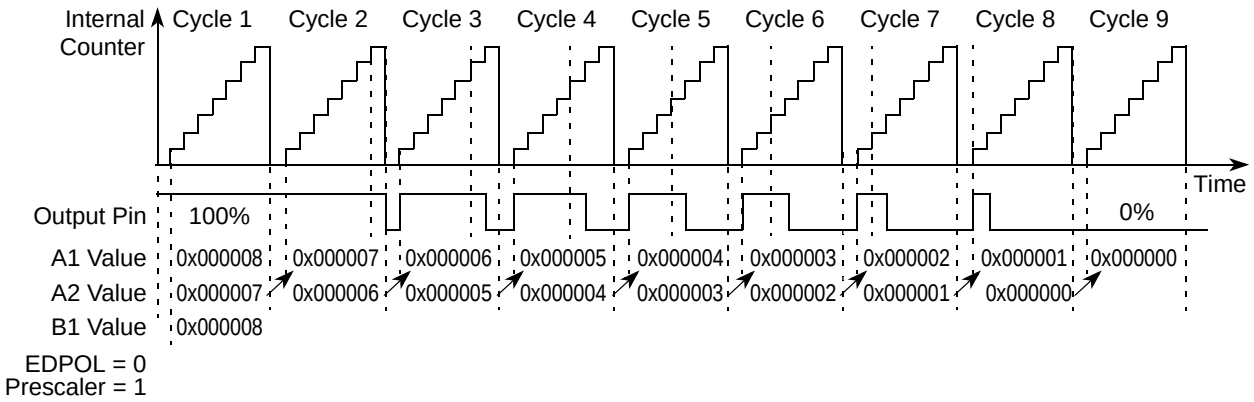


Figure 20-44. eMIOS OPWFMB Mode Example — 100% to 0% Duty Cycle

A 0% duty cycle signal is generated if A1 = 0 as shown in Figure 20-44 cycle 9. In this case the B1 = 0x000008 match from cycle 8 occurs at the same time as the A1 = 0x000000 match from cycle 9. Refer to Figure 20-41 for a description of A1 and B1 match generation for a case where A1 match has precedence over B1 match and the output signal transitions to EDPOL.

20.6.7.17 Center Aligned Output Pulse Width Modulation, Buffered (OPWMCB) Mode

This mode generates a center aligned PWM with dead time insertion on the leading or trailing edge. A1 and B1 registers are double buffered to allow smooth output signal generation when changing A2 or B2 values on the fly.

The selected counter bus for a channel configured to OPWMCB mode must be another channel running in MCB up/down counter mode (refer to Section 20.6.7.15). Register A1 contains the ideal duty cycle for the PWM signal and is compared with the selected time base. Register B1 contains the dead time value and is compared against the internal counter. For a leading edge dead time insertion, the output PWM duty cycle is equal to the difference between register A1 and register B1, and for a trailing edge dead time

insertion, the output PWM duty cycle is equal to the sum of register A1 and register B1. The MODE[0] bit selects between trailing and leading dead time insertion, respectively.

NOTE

It is recommended that the internal prescaler of the OPWMCB channel be set to the same value as the MCB channel prescaler, and the prescalers should also be synchronized. This allows the A1 and B1 registers to represent the same time scale for duty cycle and dead time insertion.

Figure 20-45 illustrates loading of the A1 and B1 registers, which occurs when the selected counter bus reaches the value one. This counter value defines the cycle boundary. Values written to A2 or B2 within cycle (n) are loaded into A1 or B1 registers and are used to generate matches in cycle (n+1).

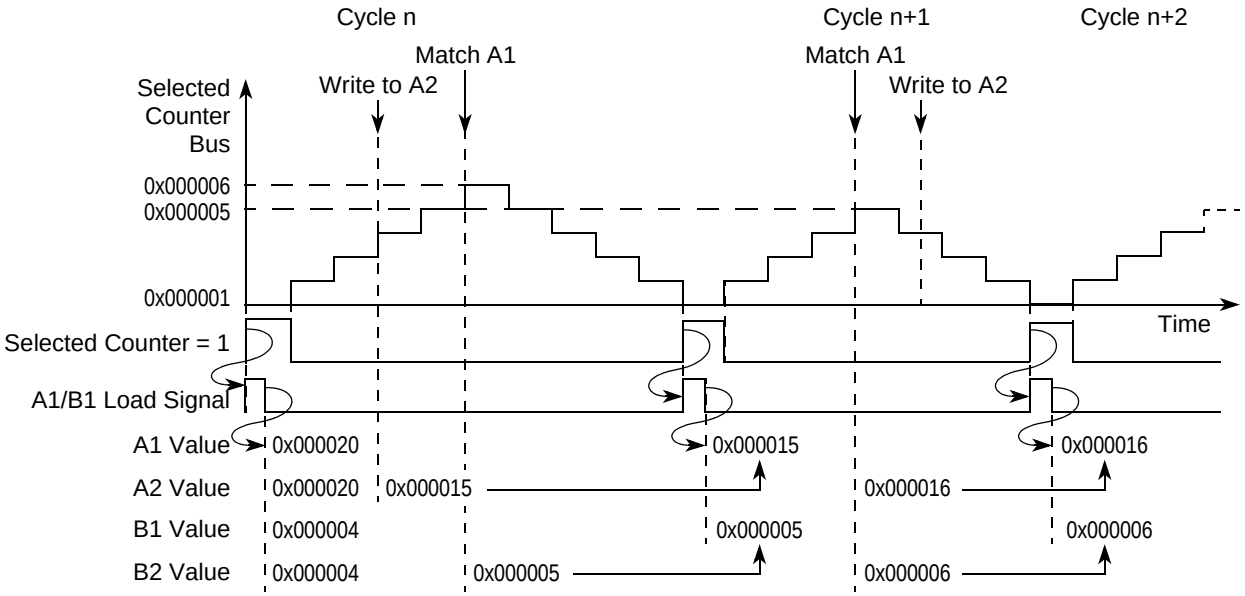


Figure 20-45. eMIOS OPWMCB Mode Example — A1/B1 Register Loading

The OUDIS[n] bit can be used to disable the A1 and B1 updates, thus allowing the loading of these registers to be synchronized with the load of A1 or B1 registers in others channels. Note that by using the update disable bit, the A1 and B1 registers can be updated in the same counter cycle.

In this mode A1 matches set the internal counter to one. When operating with leading edge dead time insertion, the first A1 match clears the internal counter. When a match occurs between register B1 and the internal time base, the output flip-flop is set to the value of the EDPOL bit. In the following match between A1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously. Figure 20-46 shows two cycles of a Center Aligned PWM signal. Note that both A1 and B1 register values are changing within the same cycle, which allows the duty cycle and dead time values to be changed at simultaneously.

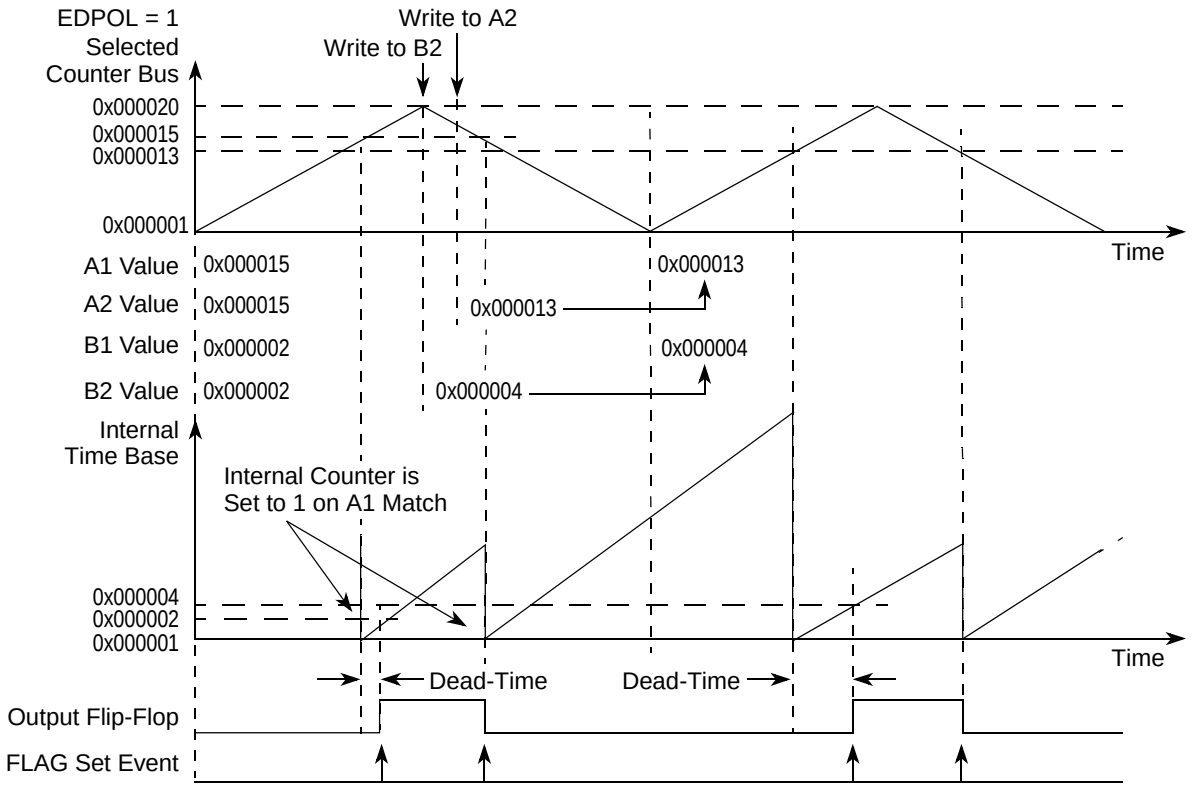


Figure 20-46. eMIOS OPWMCB Mode Example — Lead Dead Time Insertion

As shown in Figure 20-47, when operating with trailing edge dead time insertion the first match between A1 and the selected time base sets the output flip-flop to the value of the EDPOL bit and clears the internal counter. In the second match between register A1 and the selected time base, the internal counter is cleared and B1 matches are enabled. When the match between register B1 and the selected time base occurs the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously.

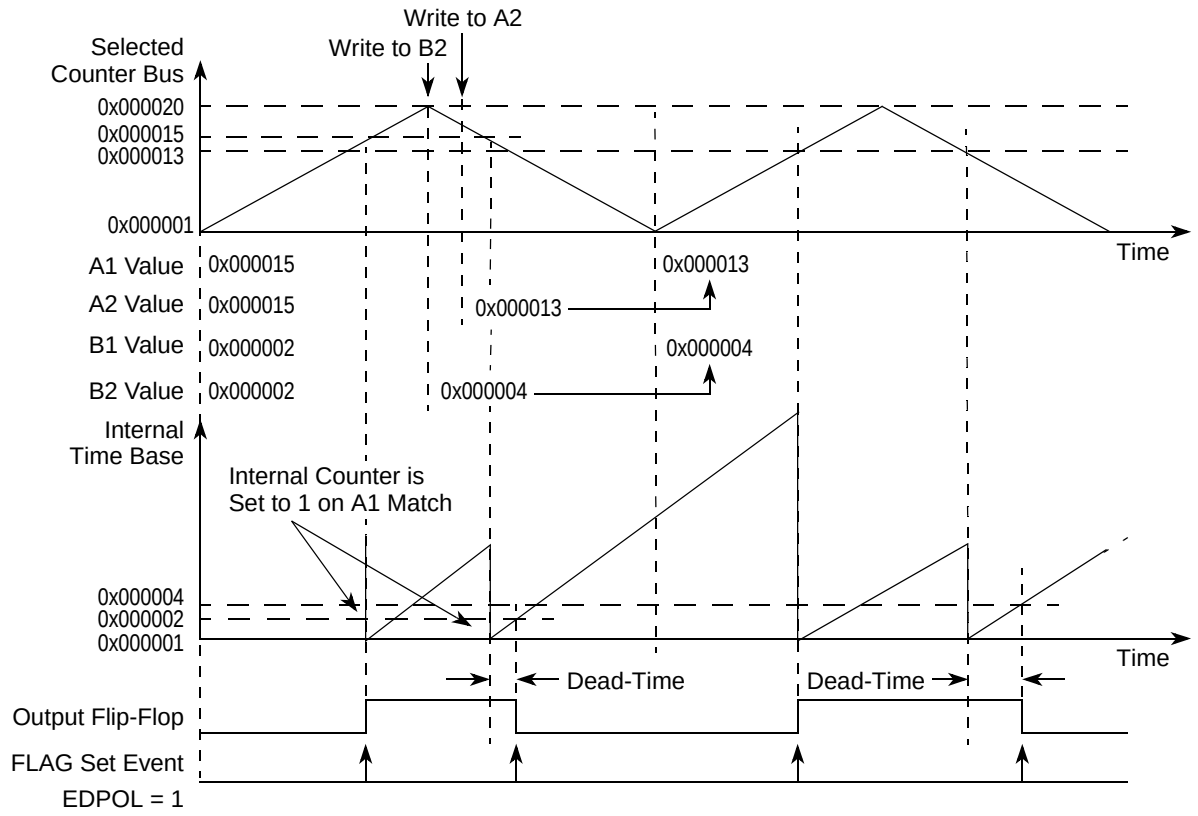


Figure 20-47. eMIOS OPWMCB Mode Example — Trailing Dead Time Insertion

FLAG can be generated in the trailing edge of the output PWM signal when MODE[1] is cleared, or on both edges when MODE[1] is set. If subsequent matches occur on A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

NOTE

In OPWMCB mode, FORCMA and FORCMB do not have the same behavior as a regular match. Instead they force the output flip-flop to a constant value which depends upon the selected dead time insertion mode, lead or trail and the value of the EDPOL bit.

FORCMA has different behaviors depending on the selected dead time insertion mode. In leading dead time insertion mode, writing one to FORCMA sets the output flip-flop to the compliment of EDPOL. In trailing dead time insertion mode, the output flip-flop is forced to the value of EDPOL.

If FORCMB is set, the output flip-flop value depends on the selected dead time insertion mode. In leading dead time insertion mode, FORCMB sets the output flip-flop to the value of EDPOL. In trailing dead time insertion mode, the output flip-flop is forced to the compliment of EDPOL.

NOTE

Setting the FORCMA bit does not reset the internal time base as a regular A1 match does. FORCMA and FORCMB have the same behavior even in Freeze or normal mode regarding the output pin transition.

The FLAG bit is not set in the case of the FORCMA, FORCMB or both bits being set at the same time.

When FORCMA and FORCMB are both set, the output flip-flop is set to the compliment of the EDPOL bit. This is equivalent to FORCMA having precedence over FORCMB when lead dead time insertion is selected and FORCMB having precedence over FORCMA when trailing dead time insertion is selected.

Duty cycles from 0% to 100% can be generated by setting appropriate A1 and B1 values relative to the period of the external time base. Setting A1 = 1 or A1 = 0 generates a 100% duty cycle waveform. If A1 > period ÷ 2, where period refers to the selected counter bus period, then a 0% duty cycle is produced. Assuming EDPOL is one and OPWMCB mode with trailing dead time insertion mode is selected, 100% duty cycle signals can be generated if B1 occurs at or after the cycle boundary (external counter = 1).

NOTE

A special case occurs when A1 is set to the external counter bus period ÷ 2, which is the maximum value of the external counter. In this case the output flip-flop is constantly set to the EDPOL bit value.

Internal channel logic prevents matches from one cycle to propagate to the next cycle. In trailing dead time insertion mode, a B1 match from cycle (n) could eventually cross the cycle boundary and occur in cycle (n+1). In this case the B1 match is masked out and does not cause the output flip-flop to transition. Therefore matches in cycle (n+1) are not affected by the late B1 matches from cycle (n).

Figure 20-48 shows a 100% duty cycle output signal generated by setting A1 = 4 and B1 = 3. In this case the trailing edge is positioned at the boundary of cycle (n+1), which is actually considered to belong to cycle (n+2) and therefore does not cause the output flip-flip to transition.

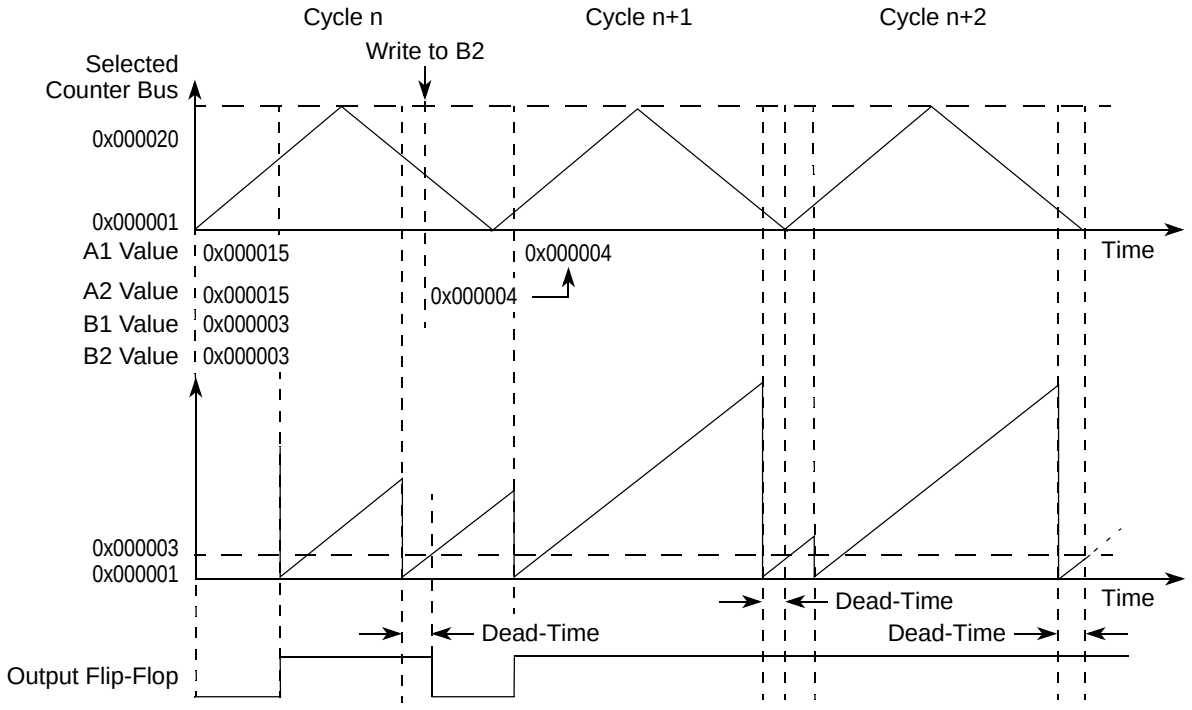


Figure 20-48. eMIOS OPWMCB Mode Example — 100% Duty Cycle (A1 = 4, B1 = 3)

The output disable input, if enabled, causes the output flip-flop to transition to the compliment of EDPOL. This allows to the channel output pin to be forced to a “safety” state. The internal channel matches continue to occur in this case, thus generating flags. When the output disable is negated, the channel output pin is again controlled by A1 and B1 matches. This process is synchronous, meaning that the output channel pin transitions only on f_{IPS} clock edges.

It is important to note that, like in OPWMB and OPWFMB modes, the match signal used to set or clear the channel output flip-flop is generated on the negation of the channel comparator output signal which compares the selected time base with A1 or B1. Refer to Figure 20-40, which illustrates the delay from matches to output flip-flop transition in OPWFMB mode.

20.6.7.18 Output Pulse Width Modulation, Buffered (OPWMB) Mode

OPWMB mode is used to generate pulses with programmable leading and trailing edge placement. An external counter is selected from one of the counter buses. The A1 register value defines the first edge and B1 defines the second edge. The output signal polarity is defined by the EDPOL bit. If EDPOL is zero, a negative edge occurs when A1 matches and a positive edge occurs when B1 matches.

The A1 and B1 registers are double buffered and updated from A2 and B2, respectively, at the cycle boundary. The load operation is similar to the OPWFMB mode. Refer to Figure 20-42 for more information on A1 and B1 register updates.

Flags are generated at B1 matches when MODE[1] is cleared, or on both A1 and B1 matches when MODE[1] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated regardless of the state of the FLAG bit.

The FORCMA and FORCMB bits allow software to force the output flip-flop to the level corresponding to a match on A1 or B1 respectively. FLAG is not set by the FORCMA and FORCMB operations.

The following rules apply to the OPWMB mode:

- B1 matches have precedence over A1 matches if they occur at the same time within the same counter cycle.
- A1 = 0 match from cycle (n) has precedence over a B1 match from cycle (n-1).
- A1 matches are masked if they occur after a B1 match within the same cycle.
- Values written to A2 or B2 on cycle (n) are loaded to A1 or B1 at the following cycle boundary (assuming OUDIS[n] is not asserted). Thus the new values will be used for A1 and B1 matches in cycle (n+1).

Figure 20-49 illustrates operation in OPWMB mode with A1/B1 matches and the transition of the channel output pin. In this example EDPOL is zero.

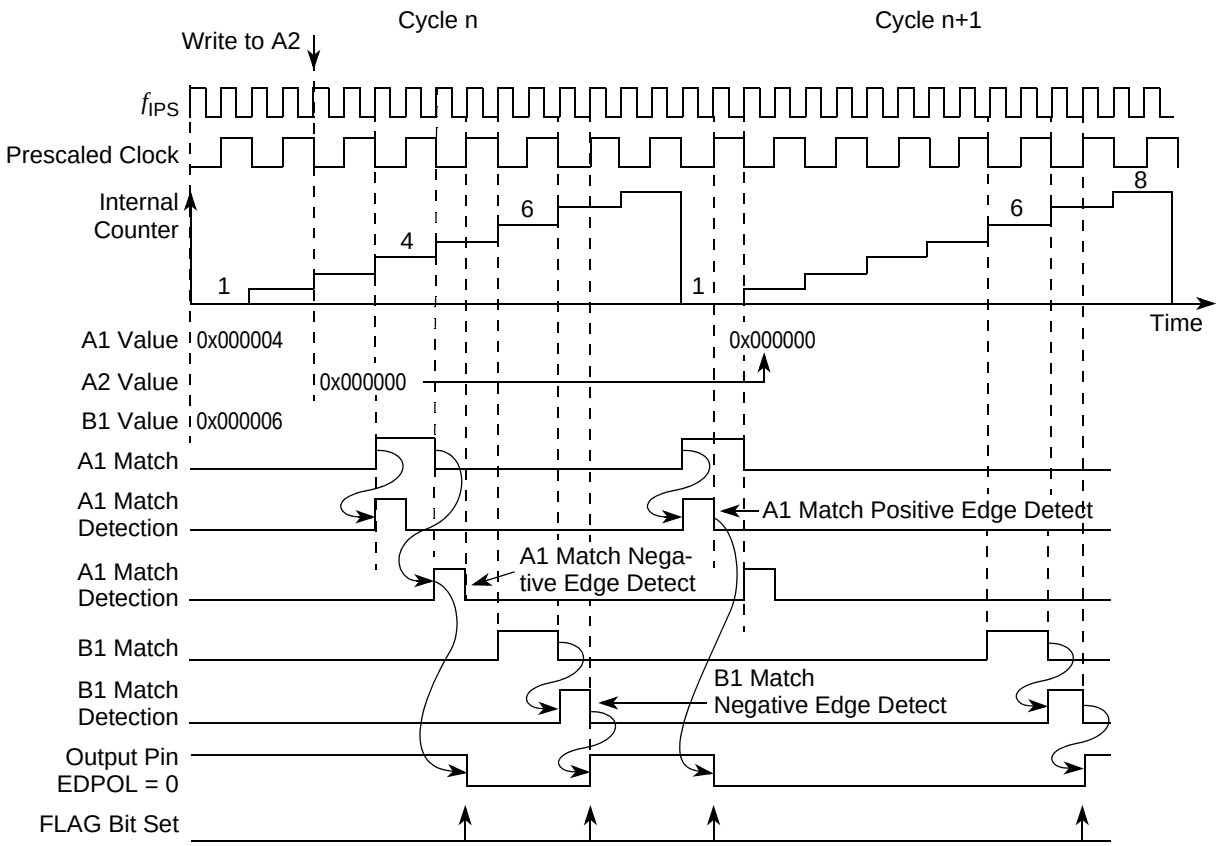


Figure 20-49. eMIOS OPWMB Mode Example — Matches and Flags

Note that the output pin transitions are based on the negative edges of the A1 and B1 match signals. Figure 20-49 shows the value of A1 being set to zero in cycle (n+1). In this case the match positive edge is used instead of the negative edge to transition the output flip-flop.

Figure 20-50 illustrates the channel operation for 0% duty cycle. Note that the A1 match signal positive edge occurs at the same time as the B1 = 8 signal negative edge. In this case the A1 match has precedence over the B1 match, causing the output pin to remain at the EDPOL value, thus generating a 0% duty cycle.

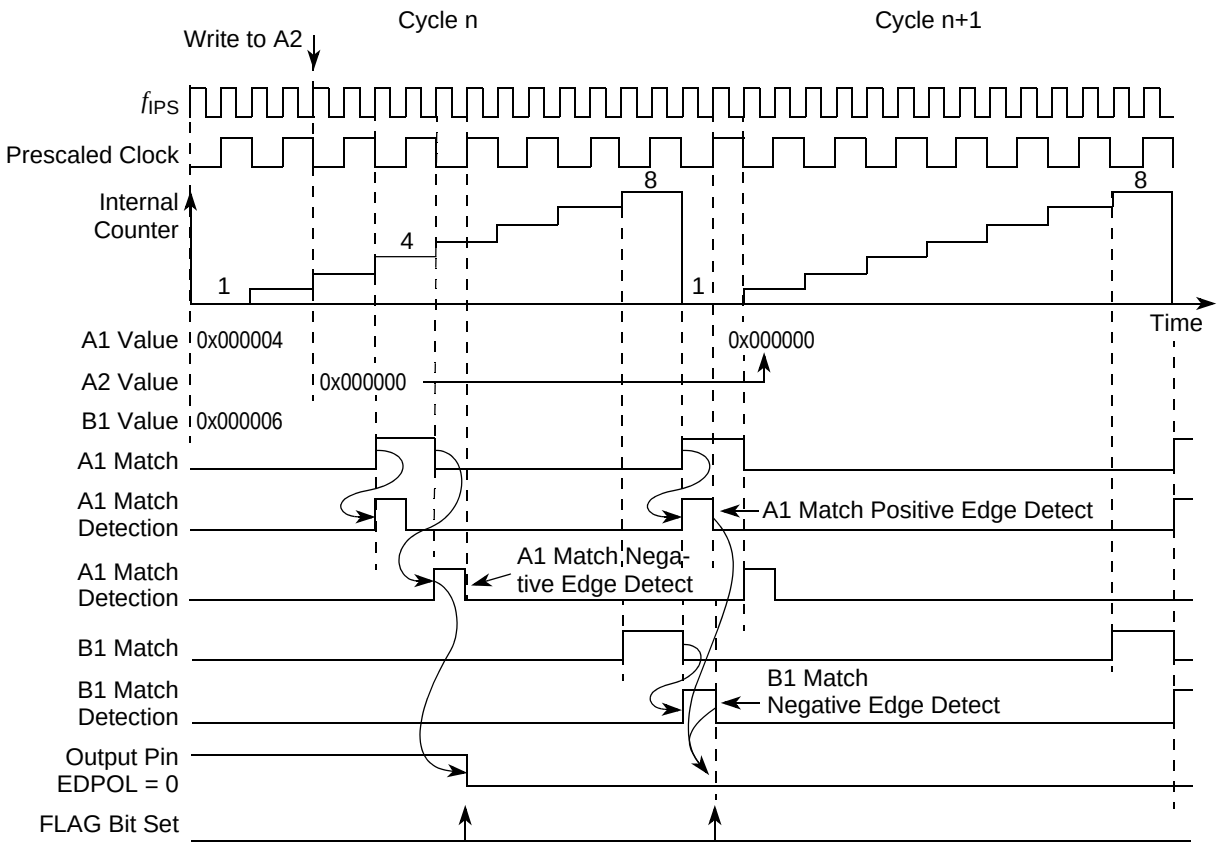


Figure 20-50. eMIOS OPWMB Mode Example — 0% Duty Cycle

Figure 20-51 shows the operation of the OPWMB mode with the Output Disable signal asserted. The output disable forces a transition in the output pin to the EDPOL bit value. After the output disable is negated, the output pin is allowed to transition at the next A1 or B1 match. The output disable does not modify the Flag bit behavior. Note that there is one f_{IPS} clock delay between the assertion of the output disable signal and the transition of the output pin.

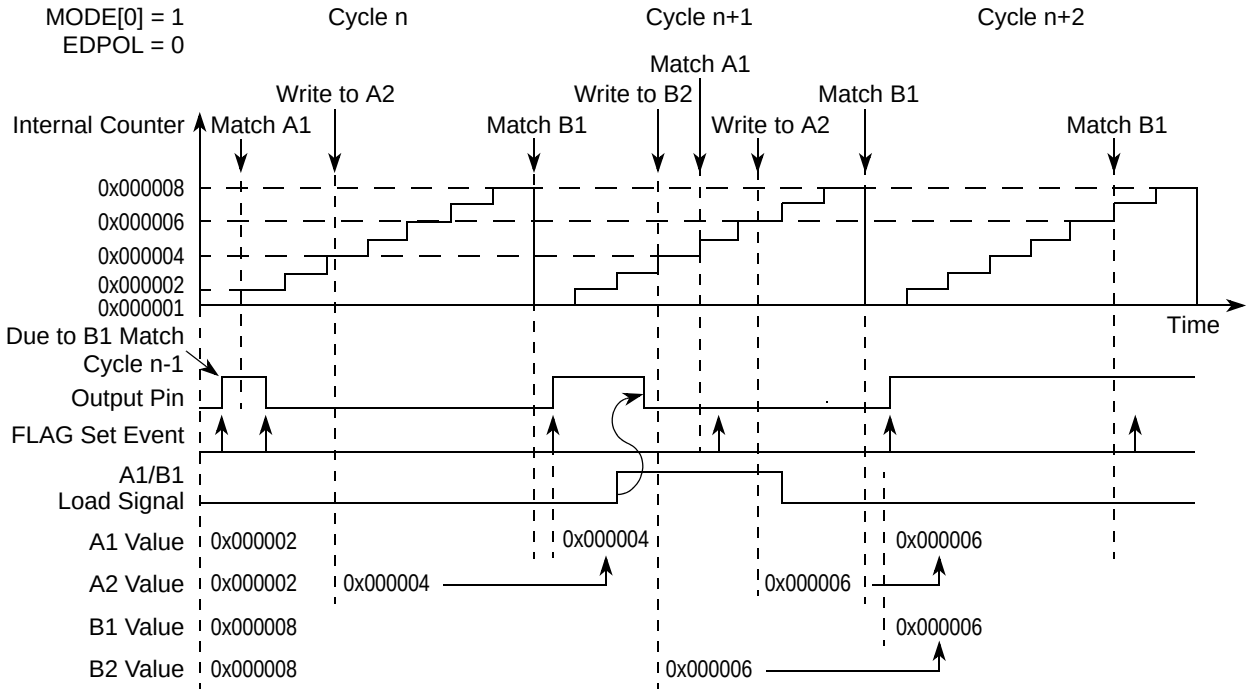


Figure 20-51. eMIOS OPWMB Mode Example — Active Output Disable

Figure 20-52 shows a waveform changing from 100% to 0% duty cycle. In this case EDPOL is zero and B1 is set to the same value as the period of the selected external time base.

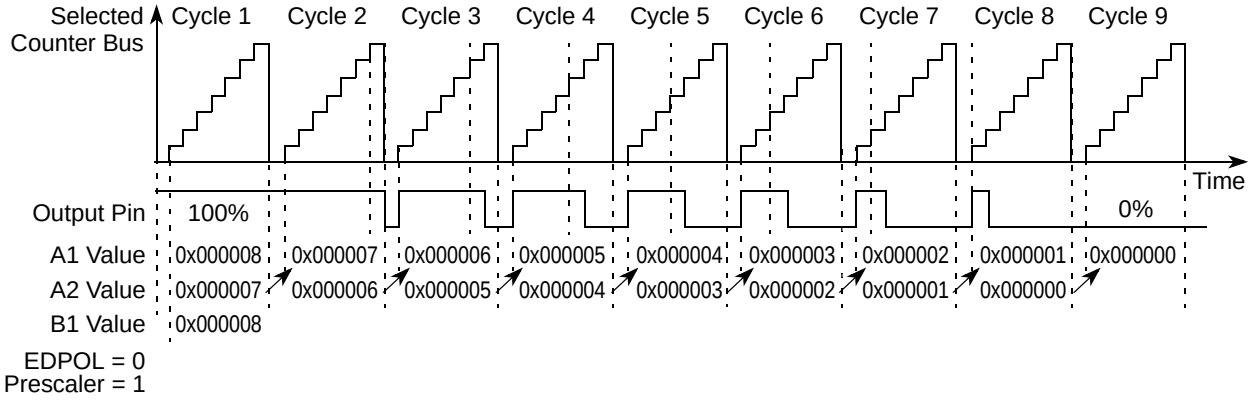


Figure 20-52. eMIOS OPWMB Mode Example — 100% to 0% Duty Cycle

In Figure 20-52 if B1 is set to a value lower than 0x000008 it is not possible to achieve 0% duty cycle by only changing A1 register value. Since B1 matches have precedence over A1 matches, the output pin transitions to the compliment of EDPOL at B1 matches. In this example, if B1 = 0x000009, a B1 match does not occur, and thus a 0% duty cycle signal is generated.

20.7 Initialization / Application Information

When the MCU is reset, all of the eMIOS Unified Channels enter GPIO input mode.

20.7.1 Changing UC Mode Considerations

Before changing an operating mode, a Unified Channel must be programmed to GPIO mode and UCA*n* and UCB*n* registers must be updated with the correct values for the next operating mode. Then the UCCR*n* can be written with the new operating mode. If a UC is changed from one mode to another without performing this procedure, the first operation cycle of the selected time base may be random, i.e., matches may occur in random time because the contents of UCA*n* or UCB*n* have not been updated with the correct value before the time base matches the previous contents of UCA*n* or UCB*n*.

When interrupts are enabled, the software must clear the FLAG bits before exiting the interrupt service routine.

20.7.2 Correlated Output Signal Generation

Correlated output signals can be generated in all output operation modes. The bits of the OUDIS register can be used to control the update of these output signals.

In order to guarantee that the internal counters of correlated channels are incremented in the same clock cycle, the internal prescalers must be set up before enabling the global prescaler. If the UC*n* prescalers are set after enabling the global prescaler, the internal counters will increment at the same rate, but on different clock cycles.

20.7.3 Time Base Generation

For all operation modes that generate a time base (MC, OPWFM, OPWM, MCB, OPWFMB and OPWMB), the clock prescaler can use several ratios calculated as

$$\text{Ratio} = (\text{GPRE} + 1) \times (\text{UCPRE} + 1) \qquad \textit{Eqn. 20-1}$$

The prescaled clocks in Figure 20-54, Figure 20-55, and Figure 20-56 illustrate this ratio. For example, if the ratio is 1, the prescaled clock is high and continuously enables the internal counter (UCCNT*n*) (Figure 20-54); if the ratio is 3, then it pulses every 3 clock cycles (Figure 20-55) and the internal counter increments every 3 clock cycles; if the ratio is 9, it pulses every 9 clock cycles, etc. This high pulse enables the UCCNT*n* to increment as long as no other conditions disable this counter. The match signal is generated by pulsing every time the internal counter matches the programmed match value. Note that for the same programmed match value, the period is shorter when using a prescaler ratio greater than one.

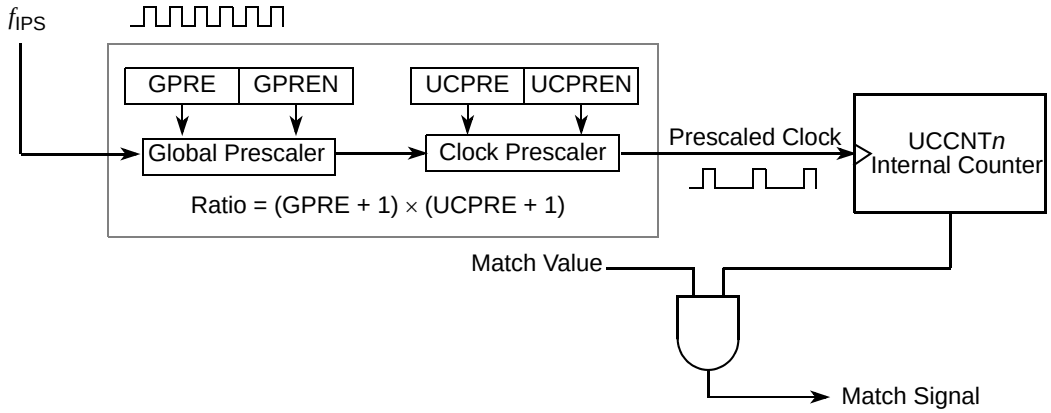
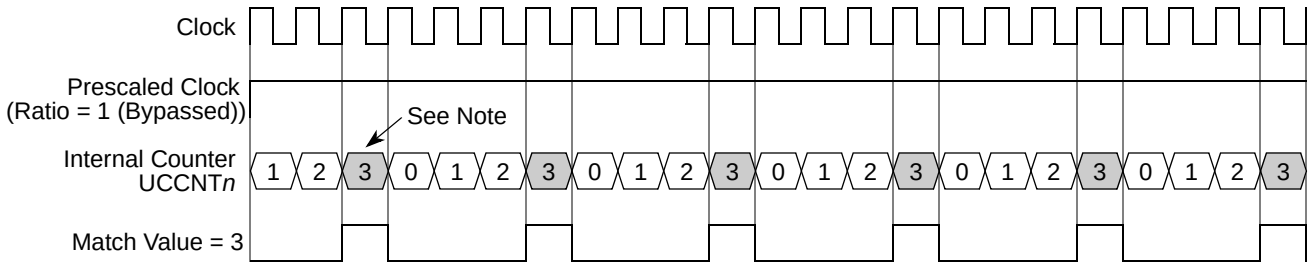
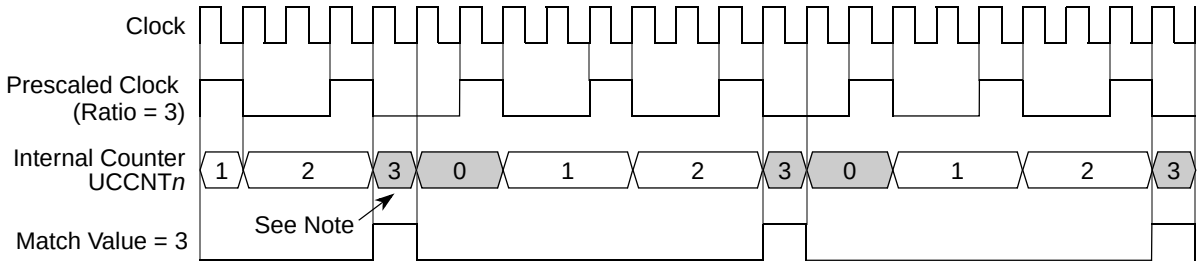


Figure 20-53. eMIOS Time Base Generation Block Diagram



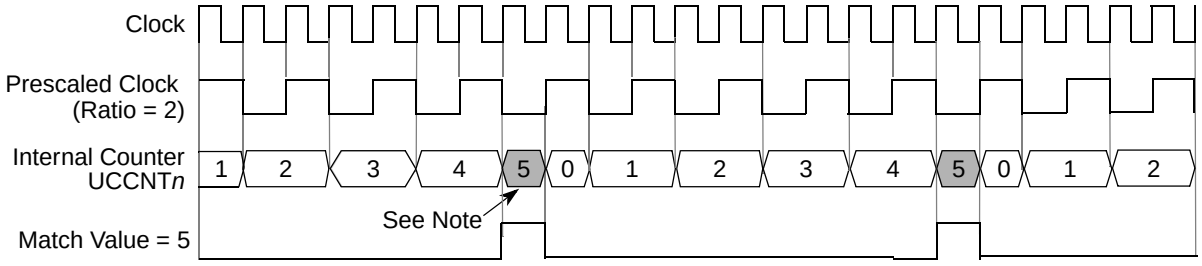
NOTE: The period of the time base includes the match value. When a match occurs, the first clock cycle is used to clear the internal counter, starting another period

Figure 20-54. eMIOS Time Base Example — Fastest Prescaler Ratio



NOTE: The period of the time base does not include the match value. When a match occurs, the first clock cycle is used to clear the internal counter, starting another period

Figure 20-55. eMIOS Time Base Example — Prescale Ratio = 3, Match Value = 3



NOTE: The period of the time base does not include the match value. When a match occurs, the first clock cycle is used to clear the internal counter, starting another period

Figure 20-56. eMIOS Time Base Example — Prescale Ratio = 2, Match Value = 5

Chapter 21
Enhanced Serial Communications Interface Module (eSCI)

21.1 Overview

There are up to four Enhanced Serial Communications Interfaces (eSCI_A, eSCI_B, eSCI_C and eSCI_D) implemented on MAC7100 Family devices. Refer to the Table 1-1 on page 1-3 for a general description of each device.

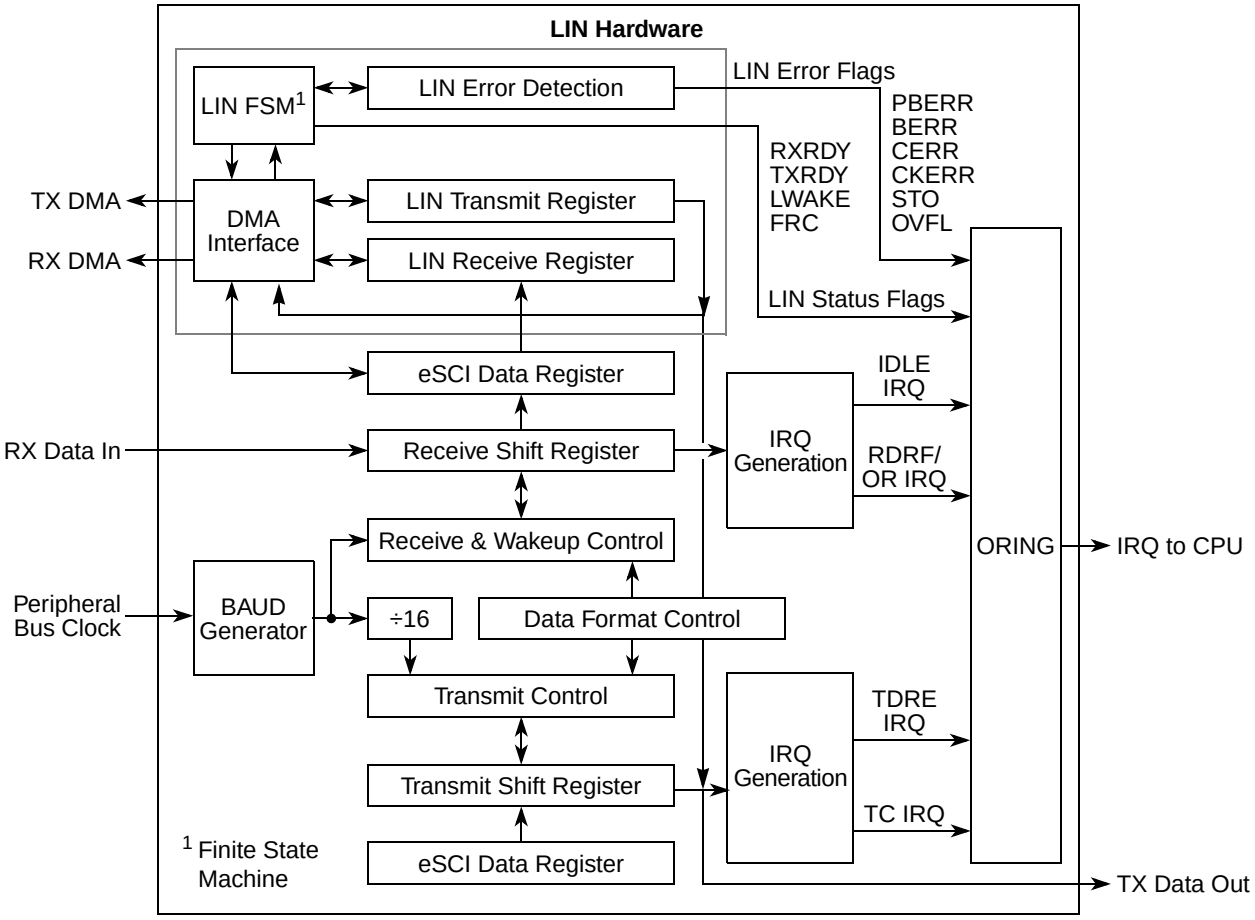


Figure 21-1. eSCI Block Diagram

The eSCI provides asynchronous serial communications with peripheral devices and other CPUs. The enhancement offered by this module over other Freescale SCIs is the inclusion of additional features which support a Local Interconnect Network (LIN) bus master, which complies to the LIN2.0 specification.

Each of the eSCI modules can be independently disabled by writing to the module disable (MDIS) bit in the module's control register 3 (ESCICR3). Disabling the module turns off the clock to the module, although the module registers may be accessed by the core via the IPS bus. The MDIS bit is intended to be used when the module is not required in the application. By default the eSCIs are disabled, so prior to use the MDIS bit must be cleared.

21.2 Features

The eSCI includes these distinctive features:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- Configurable baud rate
- Programmable 8-bit or 9-bit data format
- Separately enabled transmitter and receiver
- Programmable transmitter output parity
- Two receiver wake up methods:
- Idle line wake-up
- Address mark wake-up
- Interrupt-driven operation with eight flags:
  - Transmitter empty
  - Transmission complete
  - Receiver full
  - Idle receiver input
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit time noise detection
- 2 channel DMA request interface

21.2.1 LIN support

The eSCI provides the following LIN features:

- LIN Master functionality
- Compatible with LIN slaves from revisions 1.x and 2.0 of the LIN standard
- Detection of bit errors, physical bus errors and checksum errors
- All status bit can generate maskable interrupts
- Application layer CRC support
- Programmable CRC polynomial
- Double Stop flag insertion after bit errors
- Detection and generation of wake-up characters
- Programmable wake-up delimiter time
- Programmable slave timeout
- Can be configured to include header bits in checksum
- LIN DMA request interface

21.3 Modes of Operation

The eSCI functions the same in normal, special, and emulation modes. It has two low-power modes, doze and stop.

- Run Mode
- Doze Mode
- Stop Mode

The eSCI delays the system from going into stop mode until it has completely transmitted the current TX byte, or completely received the current RX byte. In LIN mode it will complete any frames which do not require further processor intervention (e.g. transmission of a checksum byte).

21.4 Signal Description

Please refer to Section 2.1.2, “General Purpose / Peripheral I/O,” on page 2-12 for a detailed description of the signals associated with the eSCI module. Note that the Port Integration Module (PIM) must be configured to enable the peripheral function of the appropriate pins (refer to Section 18.6.2, “Peripheral Mode,” on page 18-296) prior to configuring an eSCI channel.

21.4.1 TXD_*x* — SCI Transmit Data

This signal serves as transmit data output of the eSCI module(s).

21.4.2 RXD_*x* — SCI Receive Data

This signal serves as receive data input of the eSCI module(s).

21.5 Memory Map / Register Definition

The memory map for the eSCI module is given below in Table 21-1. The offset listed for each register is the address offset. The total address for each register is the sum of the base address for the eSCI module and the address offset for each register.

Table 21-1. eSCI Memory Map

eSCI *x* Offset	Register Description	Access
0x0000	eSCI Baud Rate Register High (ESCIBDH)	Read/Write
0x0001	eSCI Baud Rate Register Low (ESCIBDL)	Read/Write
0x0002	eSCI Control Register 1 (ESCICR1)	Read/Write
0x0003	eSCI Control Register 2 (ESCICR2)	Read/Write
0x0004	eSCI Control Register 3 (ESCICR3)	Read/Write
0x0005	eSCI Control Register 4 (ESCICR4) [1]	Read/Write
0x0006	eSCI Data Register High (ESCIDRH)	Read/Write
0x0007	eSCI Data Register Low (ESCIDRL)	Read/Write
0x0008	eSCI Status Register 1 (ESCISR1)	Read
0x0009	eSCI Status Register 2 (ESCISR2)	Read/Write

Table 21-1. eSCI Memory Map (continued)

eSCI *x* Offset	Register Description	Access
0x000A	LIN Status Register 1 (LINSTAT1)	Read/Write
0x000B	LIN Status Register 2 (LINSTAT2)	Read/Write
0x000C	LIN Control Register 1 (LINCTRL1)	Read/Write
0x000D	LIN Control Register 2 (LINCTRL2)	Read/Write
0x000E	LIN Control Register 3 (LINCTRL3)	Read/Write
0x000F	Reserved	—
0x0010	LIN TX Register (LINTX)	Read/Write
0x0011–0x0013	Reserved	—
0x0014	LIN RX Register (LINRX)	Read
0x0015-0x0017	Reserved	—
0x0018	LIN CRC Polynomial Register 1 (LINCRCP1)	Read/Write
0x0019	LIN CRC Polynomial Register 2 (LINCRCP2)	Read/Write
0x001A–0x3FFF	Reserved	—

[1] This register is not implemented on mask set L49P devices, and the offset should be treated as reserved.

21.5.1 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Writes to a reserved register location do not have any effect and reads of these locations return a zero. Details of register bit and field function follow the register diagrams, in bit order.

NOTE

Registers which are part of a single 32-bit word can be accessed together with 32-bit accesses; registers which are part of the same half-word can be accessed together with 16-bit accesses (e.g., ESCIDRH/L can be accessed with a 16-bit read or write).

21.5.1.1 eSCI Baud Rate Registers (ESCIBDH, ESCIBDH)

	7	6	5	4	3	2	1	0
R	0	0	0	SBR[12:8]				
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0000							

Figure 21-2. eSCI Baud Rate Register High (ESCIBDH)

	7	6	5	4	3	2	1	0
R W	SBR[7:0]							
RESET:	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0001							

Figure 21-3. eSCI Baud Rate Register Low (ESCIBDL)

Table 21-2. ESCIBDH Field Descriptions

Bits	Name	Description
7–5	—	Reserved, should be cleared.
4–0	SBR[12:8]	Used, along with the bits in ESCIBDL, by the counter to determine the eSCI baud rate. Refer to Section 21.6.3, “Baud Rate Generation,” for details.

Table 21-3. ESCIBDL Field Descriptions

Bits	Name	Description
7–0	SBR[7:0]	Used, along with the bits in ESCIBDH, by the counter to determine the eSCI baud rate. Refer to Section 21.6.3, “Baud Rate Generation,” for details.

NOTE

The baud rate generator is disabled until the TE bit or the RE bit is set for the first time after reset. The baud rate generator is disabled when SBR[12:0] = 0x0000.

Writing to ESCIBDH has no effect without writing to ESCIBDH, since writing to ESCIBDH puts the data in a temporary location until ESCIBDH is written to. Normally the baud rate should be written with a single 16-bit write.

21.5.1.2 eSCI Control Registers (ESCICR1 through ESCICR4)

	7	6	5	4	3	2	1	0
R	LOOPS	SCISDOZ	RSRC	M	WAKE	ILT	PE	PT
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0002							

Figure 21-4. eSCI Control Register 1 (ESCICR1)

	7	6	5	4	3	2	1	0
R	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0003							

Figure 21-5. eSCI Control Register 2 (ESCICR2)

	7	6	5	4	3	2	1	0
R	MDIS	FBR [1]	BSTP	IEBERR	RXDMA	TXDMA	BRK13	TXDIR
W								
Reset	1	0	1	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0004							

Figure 21-6. eSCI Control Register 3 (ESCICR3)

	7	6	5	4	3	2	1	0
R	BESM13 [1]	SBSTP [1]	0	0	ORIE	NFIE	FEIE	PFIE
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0005							

Figure 21-7. eSCI Control Register 4 (ESCICR4)

NOTE

ESCICR4 is not implemented on mask set L49P devices, and the offset should be treated as reserved.

Table 21-4. ESCICR1 Field Descriptions

Bits	Name	Description
7	LOOPS	Loop select. Enables loop operation. In loop operation, the RXD pin is disconnected from the eSCI and the transmitter output is internally connected to the receiver input. Both the transmitter and the receiver must be enabled to use the loop function. 1 Loop operation enabled 0 Normal operation enabled The receiver input is determined by the RSRC bit.
6	SCISDOZ	eSCI stop in doze mode. Disables the eSCI in doze mode. In Doze mode it is not possible to access all registers (e.g. it is not possible to clear interrupts). 1 eSCI disabled in Doze mode 0 eSCI enabled in Doze mode

Table 21-4. ESCICR1 Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>5</td><td>RSRC</td><td>Receiver source bit. When LOOPS = 1, the RSRC bit determines the source for the receiver shift register input.
1 Receiver input connected externally to transmitter
0 Receiver input internally connected to transmitter output
The table below shows how LOOPS and RSRC determine the Loop Function.
<table><tr><th>LOOPS</th><th>RSRC</th><th>Function</th></tr><tr><td>0</td><td>x</td><td>Normal operation</td></tr><tr><td>1</td><td>0</td><td>Loop mode with Rx input internally connected to Tx output</td></tr><tr><td>1</td><td>1</td><td>Single-wire mode with Rx input connected to TXD</td></tr></table></td></tr>
<tr><td>4</td><td>M</td><td>Data format mode bit. M determines whether data characters are eight or nine bits long.
1 One start bit, nine data bits, one stop bit
0 One start bit, eight data bits, one stop bit</td></tr>
<tr><td>3</td><td>WAKE</td><td>Wake-up condition bit. WAKE determines which condition wakes up the eSCI: a logic 1 (address mark) in the most significant bit position of a received data character or an idle condition on the RXD.
1 Address mark wake-up
0 Idle line wake-up
Note: This is not a wake-up out of a power-save mode, it refers solely to the receiver standby mode.</td></tr>
<tr><td>2</td><td>ILT</td><td>Idle line type bit. ILT determines when the receiver starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions.
1 Idle character bit count begins after stop bit
0 Idle character bit count begins after start bit</td></tr>
<tr><td>1</td><td>PE</td><td>Parity enable bit. PE enables the parity function. When enabled, the parity function inserts a parity bit in the most significant bit position of transmit data. On receive data, the parity bit in the most significant bit position will be verified; however, the received parity bit will not be modified.
1 Parity function enabled
0 Parity function disabled</td></tr>
<tr><td>0</td><td>PT</td><td>Parity type bit. PT determines whether the eSCI generates and checks for even parity or odd parity. With even parity, an even number of 1s clears the parity bit and an odd number of 1s sets the parity bit. With odd parity, an odd number of 1s clears the parity bit and an even number of 1s sets the parity bit.
1 Odd parity
0 Even parity</td></tr>
</table>

Table 21-5. ESCICR2 Field Descriptions

Bits	Name	Description
7	TIE	Transmitter interrupt enable. TIE enables the transmit data register empty flag ESCISR1[TDRE] to generate interrupt requests. [1] 1 TDRE interrupt requests enabled 0 TDRE interrupt requests disabled
6	TCIE	Transmission complete interrupt enable. TCIE enables the transmission complete flag ESCISR1[TC] to generate interrupt requests. [1] 1 TC interrupt requests enabled 0 TC interrupt requests disabled
5	RIE	Receiver full interrupt enable. RIE enables the receive data register full flag ESCISR1[RDRF] to generate interrupt requests. [1] 1 RDRF interrupt requests enabled 0 RDRF interrupt requests disabled
4	ILIE	Idle line interrupt enable. ILIE enables the IDLE line flag to generate interrupt requests. 1 IDLE interrupt requests enabled 0 IDLE interrupt requests disabled
3	TE	Transmitter enable. TE enables the eSCI transmitter. The TE bit can be used to queue an idle preamble. 1 Transmitter enabled 0 Transmitter disabled
2	RE	Receiver enable. RE enables the eSCI receiver. 1 Receiver enabled 0 Receiver disabled
1	RWU	Receiver wake-up. Standby state 1 RWU enables the wake-up function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU. 0 Normal operation.
0	SBK	Send break bit. Toggling SBK sends one break character (10 or 11 logic 0s, respectively 13 or 14 logics 0s if BRK13 is set). Toggling implies clearing the SBK bit before the break character has finished transmitting. As long as SBK is set, the transmitter continues to send complete break characters (10 or 11 bits, respectively 13 or 14 bits). See bit field description for BRK13 1 Transmit break characters 0 No break characters

[1] On mask set L49P devices, these interrupts are suppressed in RX and TX DMA mode.

Table 21-6. ESCICR3 Field Descriptions

Bits	Name	Description
7	MDIS	Module disable. After reset, the module is disabled and must be explicitly enabled before any other registers are accessed. 1 Module enabled 0 Module disabled

Table 21-6. ESCICR3 Field Descriptions (continued)

Bits	Name	Description
6	FBR [1]	Fast bit error detection. Handles bit error detection on a per-bit basis. If disabled bit errors are detected on a per-byte basis. 1 Detect bit errors on a per-bit basis 0 Detect bit errors on a per-byte basis (compatibility mode)
5	BSTP	Bit error / physical bus error stop. Causes DMA TX requests to be suppressed, as long as the bit error and physical bus error flags are not cleared. This stops further DMA writes, which would otherwise cause data bytes to be interpreted as LIN header information.
4	IEBERR	Enable bit error interrupt. Generates an interrupt, when a Bit Error is detected.
3	RXDM [2]	Activate RX DMA channel. If this bit is enabled and the eSCI has received data, it will assert a DMA RX request.
2	TXDMA [2]	Activate TX DMA channel. Whenever the eSCI is able to transmit data, it will assert a DMA TX request.
1	BRK13	Break transmit character length. This bit determines whether the transmit break character is 10 or 11 bit respectively 13 or 14 bits long. The detection of a framing error is not affected by this bit. LIN 2.0 now requires that a break character is always 13 bits long, so this Bit should always be set to 1. 1 Break character is 13 or 14 bits long 0 Break Character is 10 or 11 bits long
0	TXDIR	Transmitter pin data direction in single-wire mode. This bit determines whether the TXD pin is going to be used as an input or output, in the Single-Wire mode of operation. This bit is only relevant in the Single-Wire mode of operation. 1 TXD pin to be used as an output in Single-Wire mode 0 TXD pin to be used as an input in Single-Wire mode

1 Mask sets L49P and L47W devices do not implement this feature; this bit is reserved and should be cleared.

2 On mask set L49P devices, RXDMA and TXDMA override the RIE and TIE/TCIE bits and prevent the eSCI from generating RX or TX interrupts, respectively.

Table 21-7. ESCICR4 Field Descriptions

Bits	Name	Description
7	BESM13 [1]	Bit Error Sample Mode, Bit 13. Determines when to sample the incoming bit in order to detect a bit error. (This is only relevant when FBR is set.) 1 Sample at RT clock 13 (refer to Section 21.6.13, “Data Sampling”) 0 Sample at RT clock 9
6	SBSTP [1]	SCI Bit Error Stop. Stops the SCI when a Bit Error is asserted. This allows the system to stop driving the LIN bus quickly after a Bit Error has been detected.
5–4	—	Reserved, should be cleared.
3	ORIE	Overrun Error Interrupt Enable. Generates an interrupt when the Overrun Flag is set.
2	NFIE	Noise Flag Interrupt Enable. Generates an interrupt when the Noise Flag is set.

Table 21-7. ESCICR4 Field Descriptions (continued)

Bits	Name	Description
1	FEIE	Frame Error Interrupt Enable. Generates an interrupt when a Frame Error is detected.
0	PFIE	Parity Flag Interrupt Enable. Generates an interrupt when Parity Flag is set.

[1] Mask sets L49P and L47W devices do not implement this feature; this bit is reserved and should be cleared.

21.5.1.3 eSCI Data Registers (ESCIDRH, ESCIDRL)

Reading the eSCI data registers accesses the eSCI receive data register; writing to them accesses the eSCI transmit data register. When the eSCI is configured for LIN mode, these registers should not be written by software.

	7	6	5	4	3	2	1	0
R	R8	T8	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0006							

Figure 21-8. eSCI Data Register High (ESCIDRH)

	7	6	5	4	3	2	1	0
R	R[7:0]							
W	T[7:0]							
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0007							

Figure 21-9. eSCI Data Register Low (ESCIDRL)

Table 21-8. ESCIDRH Field Descriptions

Bits	Name	Description
7	R8	Received bit 8. R8 is the ninth data bit received when the eSCI is configured for 9-bit data format (M = 1).
6	T8	Transmit bit 8. T8 is the ninth data bit transmitted when the eSCI is configured for 9-bit data format (M = 1). If the value of T8 is the same as in the previous transmission, T8 does not have to be rewritten and the same value is transmitted for each frame until T8 is rewritten.
5–0	—	Reserved, should be cleared.

Table 21-9. ESCIDRL Field Descriptions

Bits	Name	Description
7–0	R[7:0] / T[7:0]	Received bits / Transmit bits 7–0 for 9-bit or 8-bit formats. Bits seven through zero from eSCI communication may be read from ESCIDRL (provided that eSCI communication was successful). Writing to ESCIDRL provides bits seven through zero for eSCI transmission.

NOTES

In 8-bit data format, only ESCIDRL needs to be accessed. When transmitting in 9-bit data format and using 8-bit write instructions, write first to ESCIDRH, then ESCIDRL. See Section 21.7.1, "Using the eSCI in 9-bit Data Mode," for recommended coding techniques.

These registers are not used in LIN mode, and software should not write to them.

If the ESCICR1[PE] bit is set, the parity bit is not masked out.

21.5.1.4 eSCI Status Registers (ESCISR1, ESCISR2)

The ESCISR1 and ESCISR2 registers indicate the current status. The status flags can be polled, and can also be used to generate interrupts. The flags are cleared by writing a 1 to the flag bit position. On mask set L49P devices, the flags are cleared by reading the status register followed by a read or write to the data register.

	0	1	2	3	4	5	6	7
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W								
Reset	1	0 [1]	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0008							

[1] On mask set L49P devices, the TC bit is set following reset.

Figure 21-10. eSCI Status Register 1 (ESCISR1)

	7	6	5	4	3	2	1	0
R	0	0	0	BERR	0	0	0	RAF
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0009							

Figure 21-11. eSCI Status Register 2 (ESCISR2)

Table 21-10. ESCISR1 Field Descriptions

Bits	Name	Description
7	TDRE	Transmit data register empty flag. TDRE is set when the transmit data register (SCIDRH/L) becomes empty and can receive a new value to transmit. The flag is cleared by writing a 1 to the bit. [1]
6	TC	Transmit complete flag. TC is set after a byte of data, a preamble, or a break character has been transmitted and no data is queued via the ESCIDRH/L, indicating that the TXD out signal has become idle (logic 1). The flag is cleared by writing a 1 to the bit. [1] When the eSCI is enabled (by clearing the MDIS bit) a preamble is transmitted; if no byte is written to the ESCIDRH/L, then the completion of the preamble can be monitored using the TC flag.
5	RDRF	Receive data register full flag. RDRF is set when the SCI data register is loaded with new received data. The flag is cleared by writing a 1 to the bit. [1]
4	IDLE	Idle line flag. IDLE is set when 10 consecutive logic 1s (if M=0) or 11 consecutive logic 1s (if M=1) have appeared on the receiver input. Once the IDLE flag is cleared, a valid frame must again set the RDRF flag before an idle condition can set the IDLE flag. The flag is cleared by writing a 1 to the bit. [1] **Note:** When the receiver wake-up bit (RWU) is set, an idle line condition does not set the IDLE flag.
3	OR	Overrun flag. OR is set when software fails to read the SCI data register before the receive shift register receives the next frame. The OR bit is set immediately after the stop bit has been completely received for the second frame. The data in the shift register is lost, but the data already in the SCI data registers is not affected. The flag is cleared by writing a 1 to the bit. [1]
2	NF	Noise flag. NF is set when the eSCI detects noise on the receiver input. NF is set during the same cycle as the RDRF flag, but does not get set in the case of an overrun. The flag is cleared by writing a 1 to the bit. [1]
1	FE	Framing error flag. FE is set when a logic 0 is accepted as the stop bit. FE bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. The flag is cleared by writing a 1 to the bit. [1] **Note:** In LIN mode, this bit is not set when break characters are transmitted.
0	PF	Parity error flag. PF is set when the parity enable bit, PE, is set and the parity of the received data does not match its parity bit. The flag is cleared by writing a 1 to the bit. [1]

[1] On mask set L49P devices, flags are cleared by reading the status register followed by the appropriate read or write to the data register.

Table 21-11. ESCISR2 Field Descriptions

Bits	Name	Description
7–5	—	Reserved, should be cleared.
4	BERR	Bit error. Indicates a Bit on the bus did not match the transmitted Bit. Checking happens after a complete byte has been transmitted and received again. This bit is only used for LIN mode. If an unrequested byte is received (i.e., a byte which is not part of an RX frame) which is not recognized as a wakeup flag, then this bit is also set (since the data on the RX line does not match the idle state which was assigned to the TX line). A Bit Error will cause the LIN FSM to reset. Writing 1 to this bit position clears BERR.

Table 21-11. ESCISR2 Field Descriptions (continued)

Bits	Name	Description
3–1	—	Reserved, should be cleared.
0	RAF	Receiver active flag. RAF is set when the receiver detects a logic 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects an idle character. 1 Reception in progress. 0 No reception in progress.

21.5.1.5 LIN Status Registers (LINSTAT1, LINSTAT2)

The LIN status registers can be read at anytime. Each bit is cleared by writing a 1 to that position.

	7	6	5	4	3	2	1	0
R	RXRDY	TXRDY	LWAKE	STO	PBERR	CERR	CKERR	FRC
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x000A							

Figure 21-12. LIN Status Register 1 (LINSTAT1)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	OVFL
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x000B							

Figure 21-13. LIN Status Register 2 (LINSTAT2)

Table 21-12. LINSTAT1 Field Descriptions

Bits	Name	Description
7	RXRDY	Receive data ready. The eSCI has received LIN data. Set when the LINRX register receives a byte, cleared by writing 1 to this bit position.
6	TXRDY	Transmit data ready. The LIN FSM can accept another write to LINTX. Set when the LINTX register is empty, cleared by writing 1 to this bit position.
5	LWAKE	Received LIN 1.x wake-up signal. A LIN slave has sent a LIN 1.x wake-up signal (0x80, 0x00 or 0xC0) on the bus. When this signal is detected, the LIN FSM will reset. If the setup of a frame had already started, the setup must be repeated. This flag is also set if the eSCI receives a LIN 2.0 wake-up signal. Refer to Section 21.7.2.6, “LIN Wake-up,” for guidelines on using this feature. Set when the condition is detected, cleared by writing 1 to this bit position.

Table 21-12. LINSTAT1 Field Descriptions (continued)

Bits	Name	Description
4	STO	Slave timeout. Represents a NO_RESPONSE_ERROR. This is set if a slave doesn't complete a frame within the specified maximum frame length. For LIN 1.3 the formula used is: $T_{FRAME\_MAX} = (10 \times N_{DATA} + 45) \times 1.4$ where N_{DATA} is the number of data byes in a frame. Refer to the LIN specification for details. Set when the condition is detected, cleared by writing 1 to this bit position.
3	PBERR	Physical bus error. No valid message can be generated on the bus. This is set if after the start of a byte transmission the input remains unchanged for 31 cycles. This will reset the LIN FSM. Set when the condition is detected, cleared by writing 1 to this bit position.
2	CERR	CRC error. The CRC pattern received with an extended frame was not correct. Set when the condition is detected, cleared by writing 1 to this bit position.
1	CKERR	Checksum error. Checksum Error on a received frame. Set when the condition is detected, cleared by writing 1 to this bit position.
0	FRC	Frame complete. LIN frame completely transmitted / All LIN data bytes received. Set when the condition is detected, cleared by writing 1 to this bit position.

Table 21-13. LINSTAT2 Field Descriptions

Bits	Name	Description
7–1	—	Reserved, should be cleared.
0	OVFL	RX register overflow. The RX register hasn't been read before a new data byte, CRC or checksum has been received from the LIN bus. Set when the condition is detected, cleared by writing 1 to this bit position.

21.5.1.6 LIN Control Registers (LINCTRL1, LINCTRL2, LINCTRL3)

LINCTRL*n* can be written when there are no ongoing transmissions.

	7	6	5	4	3	2	1	0
R	LRES	0	WUD1	WUD0	LDBG	DSF	PRTY	LIN
W		WU						
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x000C							

Figure 21-14. LIN Control Register 1 (LINCTRL1)

	7	6	5	4	3	2	1	0
R	RXIE	TXIE	WUIE	STIE	PBIE	CIE	CKIE	FCIE
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x000D							

Figure 21-15. LIN Control Register 2 (LINCTRL2)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	OFIE
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x000E							

Figure 21-16. LIN Control Register 3 (LINCTRL3)

Table 21-14. LINCTRL1 Field Descriptions

Bits	Name	Description
7	LRES	LIN resync. Causes the LIN protocol engine to return to start state. This happens automatically after bit errors, but software can force this behavior manually via this bit. The bit needs to be first set then cleared, so that the protocol engine is operational again. Status flags are not affected by this bit, and must be cleared manually as necessary.
6	WU	LIN bus wake-up. Generates a wake-up signal on the LIN bus. This needs to be set before a transmission, if the bus is in sleep mode, e.g. because it has been idle for 25000 bit times. This bit will auto-clear, so a read from this bit always returns 0. Refer to Section 21.7.2.6, “LIN Wake-up,” for details on using this feature.
5–4	WUD1–0	Wake-up delimiter time. Determines how long the LIN engine waits after generating a wake-up signal, before starting a new frame. The eSCI will not set TXRDY before this time expires. In addition to this delimiter time, the CPU and the eSCI will require some setup time to start a new transmission, typically there will be an additional bit time delay. The table below shows how the values for WUD1 and WUD0 affect the delimiter time.
3	LDBG	LIN debug mode. Prevents the LIN FSM from automatically resetting, after an exception (Bit Error, Physical Bus Error, Wake-up Flag) has been detected. This is for debug purposes only.
2	DSF	Double stop flags. When a bit error has been detected, this will add an additional stop flag to the byte in which the error occurred.

WUD1	WUD0	Bit Times
0	0	4
0	1	8
1	0	32
1	1	64

Table 21-14. LINCTRL1 Field Descriptions (continued)

Bits	Name	Description
1	PRTY	Activating parity generation. Generate the two Parity Bits in the LIN header.
0	LIN	LIN mode. Switch device into LIN mode. **Note:** In LIN mode data transfer should be done only via the LIN interface. The SBK bit (in ESCICR2) and the SCI data registers (ESCIDRH/L) should not be used. Similarly all data transfer handshaking should be done via the LIN interface, the TDRE, TC and RDRF flags in the ESCISR1 should not be used for that purpose.

Table 21-15. LINCTRL2 Field Descriptions

Bits	Name	Description
7	RXIE	LIN RXREG ready interrupt enable. Generates an Interrupt when new data is available in the LIN RXREG.
6	TXIE	LIN TXREG ready interrupt enable. Generates an Interrupt when new data can be written to the LIN TXREG.
5	WUIE	RX wake-up interrupt enable. Generates an Interrupt when a wake-up flag from a LIN 1.x slave has been received. Refer to Section 21.7.2.6, “LIN Wake-up,” for details on using this feature.
4	STIE	Slave timeout error interrupt enable. Generates an Interrupt when the slave response is too slow.
3	PBIE	Physical bus error interrupt enable. Generates an Interrupt when no valid message can be generated on the bus.
2	CIE	CRC error interrupt enable. Generates an Interrupt when a CRC error on a received extended frame is detected.
1	CKIE	Checksum error interrupt enable. Generates an Interrupt on a detected Checksum Error.
0	FCIE	Frame complete interrupt enable. Generates an Interrupt after complete transmission of a TX frame, or after the last byte of an RX frame is received. (The complete frame includes all header, data, CRC and checksum bytes as applicable.)

Table 21-16. LINCTRL3 Field Description

Bits	Name	Description
7–1	—	Reserved, should be cleared.
0	OFIE	Overflow interrupt enable. Generates an Interrupt when a data byte in the LINRX register hasn’t been read before the next databyte is received.

21.5.1.7 LIN TX Register (LINTX)

LINTX can be written when TXRDY is set. The first byte written to the register selects the transmit address, the second byte determines the frame length, the third and fourth byte set various frame options and determine the timeout counter. Header parity will be automatically generated if the

LINCTRL1[PRTY] bit is set. For TX frames, the fourth byte (bits T7–T0) will be skipped, since the timeout function does not apply. All following bytes are data bytes for the frame. CRC and checksum bytes will be automatically appended when the appropriate options are selected.

When a bit error is detected, an interrupt is set and the transmission aborted. The register can only be written again once the interrupt is cleared. Afterwards a new frame starts, and the first byte needs to contain a header again.

Additionally it is possible to flush the LINTX register by writing to the LRES bit.

NOTE

Not all values written to the LINTX register will generate valid LIN frames. The values must adhere to the LIN specification.

	7	6	5	4	3	2	1	0
R								
(Table 21-17) W	P[1:0]		ID[5:0]					
(Table 21-18) W	L[7:0]							
(Table 21-19) W	HDCHK	CSUM	CRC	TX	T[11:8]			
(Table 21-20) W	T[7:0]							
(Table 21-21) W	D[7:0]							
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0010							

Figure 21-17. LIN TX Register (LINTX)

Table 21-17. LINTX First Byte Field Descriptions

Bits	Name	Description
7–6	P[1:0]	Parity bits. When the LINCTRL1[PRT] bit is set, the parity bits are generated automatically. Otherwise they can be provided here.
5–0	ID[5:0]	Header bits. The LIN address, for LIN 1.x standard frames the length bits need to be set appropriately (see the table below), extended frames will be recognized by their specific patterns. **ID5 / ID4 / Data Bytes** 0 / 0 / 2 0 / 1 / 2 1 / 0 / 4 1 / 1 / 8 **Note:** The values 0x3C, 0x3D, 0x3E and 0x3F of the ID-field (ID0-5) indicate command and extended frames. See LIN Spec.

Table 21-18. LINTX Second Byte Field Descriptions

Bits	Name	Description
7–0	L[7:0]	Length bits. Defines the length of the frame—0 to 255 data bytes—this information is needed by the LIN state machine in order to insert the checksum or CRC pattern as required. LIN 1.x slaves will only accept frames with 2, 4 or 8 data bytes.

Table 21-19. LINTX Third Byte Field Descriptions

Bits	Name	Description
7	HDCHK	Header checksum enable. Include the header fields into the mod 256 checksum of the standard frames.
6	CSUM	Checksum enable. Append a checksum byte to the end of a TX frame - verify the checksum byte of a RX frame.
5	CRC	CRC enable. Append two CRC bytes to the end of a TX frame - verify the two CRC bytes of a RX frame are correct. If both CSUM and CRC bits are set, the LIN FSM (finite state machine) will first append the CRC bytes, then the checksum byte, and will expect them in this order, as well. If HDCHK is set, the CRC calculation will include header and data bytes, otherwise just the data bytes. CRC bytes are not part of the LIN standard - they are normal data bytes and belong to a higher-level protocol.
4	TX	Transmit Direction. Indicates a TX frame i.e. the eSCI will transmit data to a slave. Otherwise an RX frame will be assumed, and the eSCI will only transmit the header, the data bytes will be received from the slave. 1 TX frame 0 RX frame
3–0	T[11:8]	Timeout bits 11–8. Sets the counter to determine a NO_RESPONSE_ERROR, if the frame is a read access to a LIN slave. The counter value represents the maximum time available for a complete RX frame. According to LIN standard rev. 1.3 the value should be $(10 \times N_{DATA} + 45) \times 1.4$ where N_{DATA} is the number of data byes in a frame. Refer to the LIN specification for details. For transmissions the accessible timeout bits must be set to 0. The timeout bits 7–0 will not be written on a TX frame. So for TX frames the 4th byte written to the TX register is the first data byte, for RX it contains timeout bits 7-0. The time is specified in multiples of bit times. The timeout period starts with the transmission of the LIN break character.

Table 21-20. LINTX Rx Frame Fourth Byte Field Descriptions

Bits	Name	Description
7–0	T[7:0]	Timeout bits 7–0. Sets the counter to determine a NO_RESPONSE_ERROR, if the frame is a read access to a LIN slave - according to LIN standard rev 1.3 the value needs to be $(10 \times N_{DATA} + 44) \times 1.4$. For transmissions this counter has to be set to 0. The timeout bits 7–0 will not be written on a TX frame. So for TX frames the 4th byte written to the TX register is the first data byte, for RX it contains timeout bits 7-0. The time is specified in multiples of times. The timeout period starts with the transmission of the LIN break character.

Table 21-21. LINTX Tx Frame Fourth+ Byte / Rx Frame Fifth+ Byte Field Description

Bits	Name	Description
7–0	D[7:0]	Data bits for transmission.

21.5.1.8 LIN RX Register (LINRX)

LINRX can be read when LINSTAT1[RXRDY] is set.

	7	6	5	4	3	2	1	0
R	D[7:0]							
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	eSCI *x* Base + 0x0014							

Figure 21-18. LIN RX Register (LINRX)

Table 21-22. LINRX Field Descriptions

Bits	Name	Description
7–0	D[7:0]	Data bits. This register provides received data bytes from RX frames. **Note:** Data is only valid when the RXRDY flag is set, CRC and checksum information are not available in the RX register unless they are treated as data. CRC and checksum bytes may be used as data by deactivating the CSUM CRC control bits in LINTX; however, then CRC and CSUM checking must be performed by software. Software must ensure that LINRX is read before new bytes (data bytes, CRC or checksum bytes) are received from the LIN bus.

21.5.1.9 LIN CRC Polynomial Registers (LINCRCP1, LINCRCP2)

LINCRCP*n* can be written to when there are no ongoing transmissions.

	7	6	5	4	3	2	1	0
R	P[15:8]							
W								
Reset	1	1	0	0	0	1	0	1
Reg Addr	eSCI *x* Base + 0x0018							

Figure 21-19. LIN CRC Polynomial Register 1 (LINCRCP1)

	7	6	5	4	3	2	1	0
R	P[7:0]							
W								
Reset	1	0	0	1	1	0	0	1
Reg Addr	eSCI *x* Base + 0x0019							

Figure 21-20. LIN CRC Polynomial Register 2 (LINCRCP2)

Table 21-23. LINCRCP*n* Field Description

Bits	Name	Description
7–0 / 7–0	P[15:0]	Polynomial bit x^n. Used to define the LIN polynomial. The standard is $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$ (the Polynomial used for the CAN protocol).

21.6 Functional Description

21.6.1 Overview

This section provides a complete functional description of the eSCI module, detailing the operation of the design from the end user perspective in a number of subsections.

Figure 21-21 shows the structure of the eSCI module. The eSCI allows full duplex, asynchronous, NRZ serial communication between the CPU and remote devices, including other CPUs. The eSCI transmitter and receiver operate independently, although they use the same baud rate generator. The CPU monitors the status of the eSCI, writes the data to be transmitted, and processes received data.

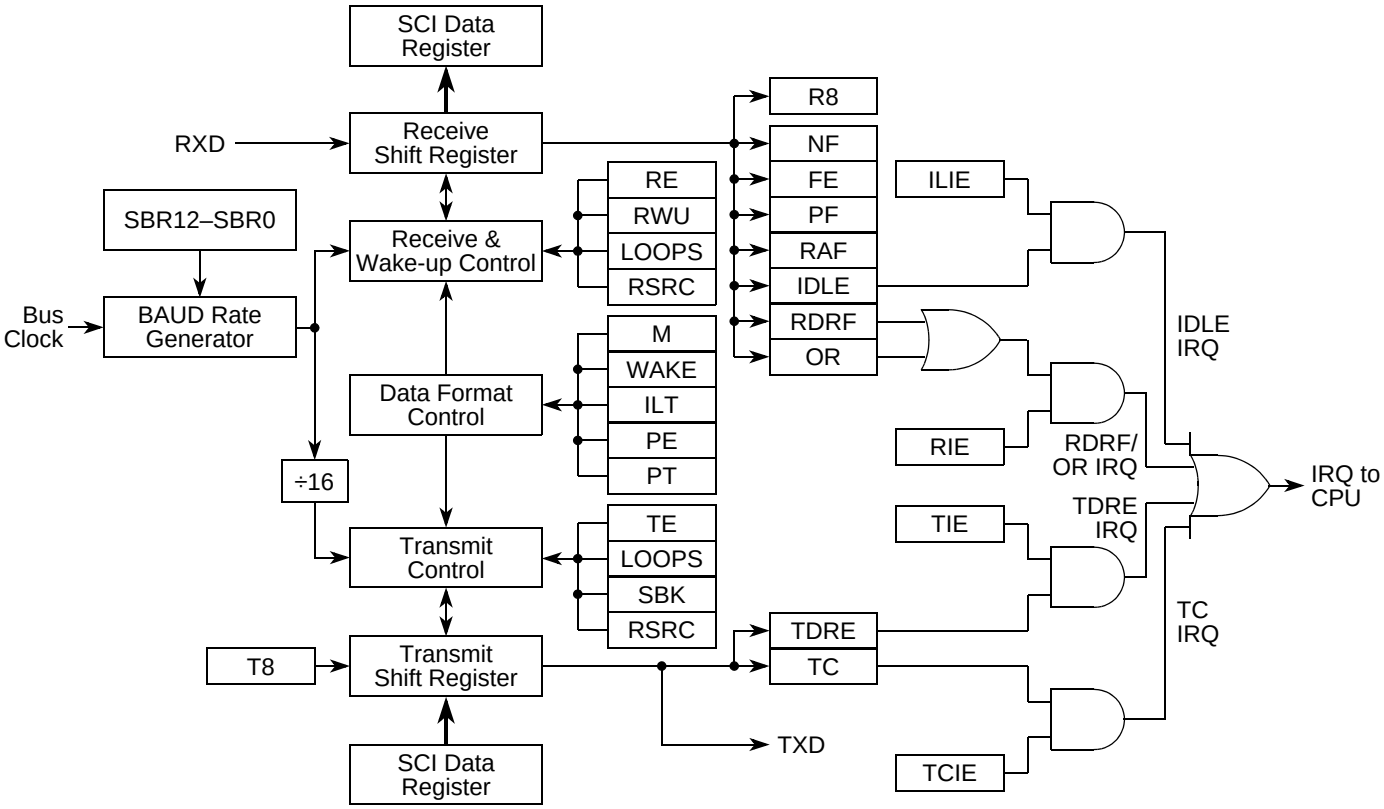


Figure 21-21. eSCI Block Diagram

21.6.2 Data Format

The eSCI uses the standard NRZ mark/space data format illustrated in Figure 21-22 below.

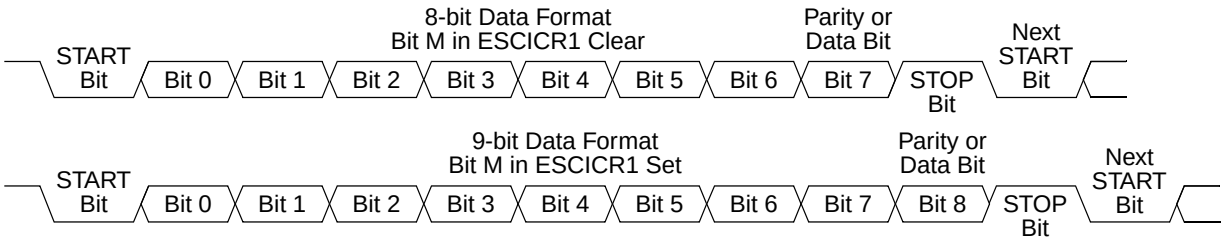


Figure 21-22. SCI Data Formats

Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing the M bit in eSCI control register 1 (ESCICR1)configures the eSCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits. Setting the M bit configures the eSCI for nine-bit data characters. A frame with nine data bits has a total of 11 bits

Table 21-24. Example of 8-bit Data Formats

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
1	8	0	0	1
1	7	0	1	1
1	7	1 [1]	0	1

[1] The address bit identifies the frame as an address character. See Section 21.6.16, “Receiver Wake-up.”

When the eSCI is configured for 9-bit data characters, the ninth data bit is the T8 bit in SCI data register high (ESCIDRH). It remains unchanged after transmission and can be used repeatedly without rewriting it. A frame with nine data bits has a total of 11 bits.

Table 21-25. Example of 9-Bit Data Formats

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
1	9	0	0	1
1	8	0	1	1
1	8	1 [1]	0	1

[1] The address bit identifies the frame as an address character. See Section 21.6.16, “Receiver Wake-up.”

21.6.3 Baud Rate Generation

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and the transmitter. The value from 0 to 8191 written to the SBR12–SBR0 bits determines the module clock divisor. The SBR bits are in the eSCI baud rate registers (ESCIBDH and ESCIBDL). The baud rate clock is synchronized with the bus clock and drives the receiver. The baud rate clock divided by 16 drives the transmitter. The receiver has an acquisition rate of 16 samples per bit time.

Baud rate generation is subject to one source of error: integer division of the module clock may not give the exact target frequency.

Table 21-26 lists some examples of achieving target baud rates with a module clock frequency of 10.2 MHz.

eSCI baud rate = eSCI module clock / (16 × ESCIBR[12:0])

Table 21-26. Baud Rates (Example: Module Clock = 10.2 Mhz)

Bits SBR[12:0]	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
17	600,000.0	37,500.0	38,400	2.3
33	309,090.9	19,318.2	19,200	.62
66	154,545.5	9659.1	9600	.62
133	76,691.7	4793.2	4800	.14
266	38,345.9	2396.6	2400	.14
531	19,209.0	1200.6	1200	.11
1062	9604.5	600.3	600	.05
2125	4800.0	300.0	300	.00
4250	2400.0	150.0	150	.00
5795	1760.1	110.0	110	.00

21.6.4 Transmitter

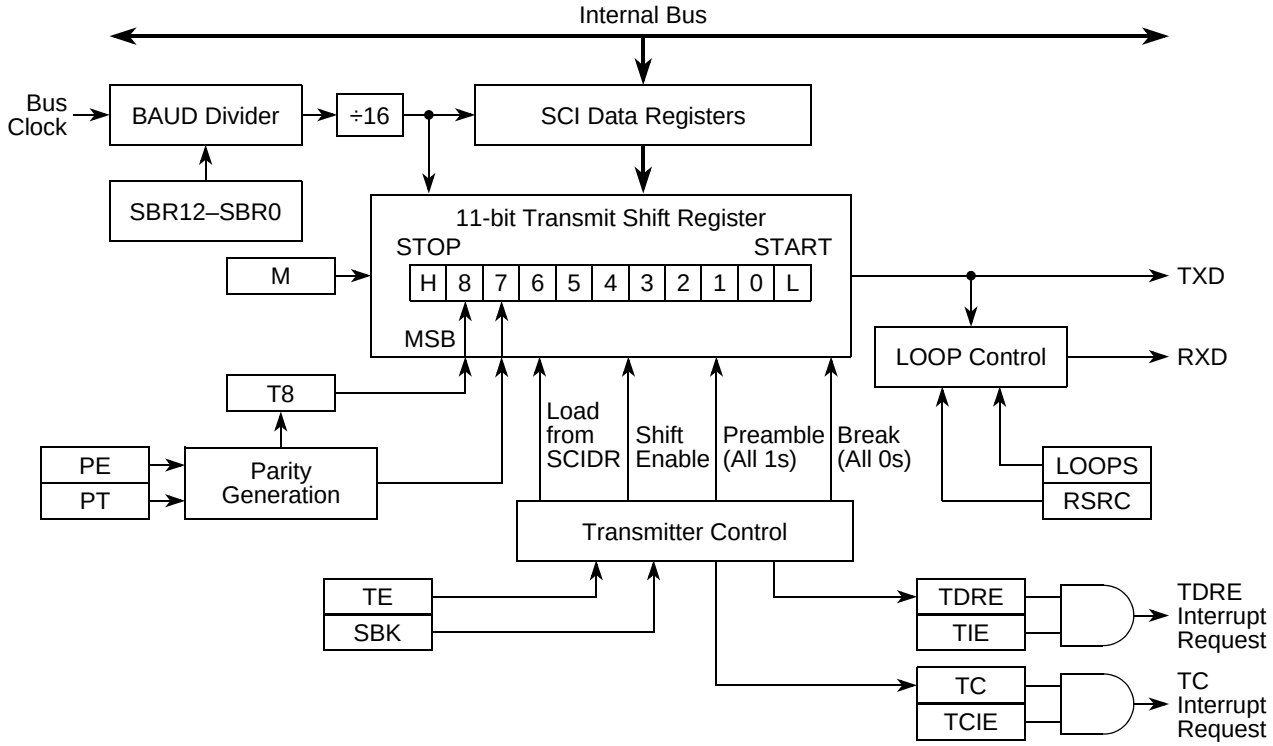


Figure 21-23. eSCI Transmitter Block Diagram

21.6.5 Transmitter Character Length

The eSCI transmitter can accommodate either 8-bit or 9-bit data characters. The state of the M bit in eSCI control register 1 (ESCICR1) determines the length of data characters. When transmitting 9-bit data, bit T8 in SCI data register high (ESCIDRH) is the ninth bit (bit 8).

21.6.6 Character Transmission

To transmit data, the MCU writes the data bits to the SCI data registers (ESCIDRH/ESCIDRL), which in turn are transferred to the transmit shift register. The transmit shift register then shifts a frame out through the Tx output signal, after it has prefaced them with a start bit and appended them with a stop bit. ESCIDRH and ESCIDRL are the write-only buffers between the internal data bus and the transmit shift register.

The eSCI also sets a flag, the transmit data register empty flag (TDRE), every time it transfers data from the buffer (ESCIDRH/ESCIDRL) to the transmit shift register. The transmit driver routine may respond to this flag by writing another byte to the transmitter buffer (ESCIDRH/ESCIDRL), while the shift register is still shifting out the first byte.

To initiate an eSCI transmission:

1. Configure the eSCI:
   a) Turn on the module by clearing ESCICR3[MDIS].
   b) Select a baud rate. Write this hex value to the eSCI baud registers (ESCIBDH/ESCIBDL) to start the baud rate generator. Remember that the baud rate generator is disabled when the baud rate is zero. Writing to the ESCIBDH has no effect without also writing to ESCIBDL.
   c) Write to ESCICR1 to configure word length, parity, and other configuration bits (LOOPS,RSRC,M,WAKE,ILT,PE,PT).
   d) Enable the transmitter, interrupts, receive, and wake up as required, by writing to the ESCICR2 register bits (TIE,TCIE,RIE,ILIE,TE,RE,RWU,SBK). A preamble or idle character will now be shifted out of the transmit shift register.
2. Transmit Procedure for Each Byte:
   a) Poll the TDRE flag by reading the ESCISR1 or responding to the TDRE interrupt. Keep in mind that the TDRE bit resets to ‘1’.
   b) If the TDRE flag is set, write the data to be transmitted to ESCIDRH/ ESCIDRL (if in 8-bit mode, ESCIDRH is not written). [1]
3. Repeat step 2 for each subsequent transmission.

NOTE

The TDRE flag is set when the shift register is loaded with the next data to be transmitted from ESCIDRH/ESCIDRL, which happens, generally speaking, a little over half-way through the stop bit of the previous frame. Specifically, this transfer occurs 9/16ths of a bit time AFTER the start of the stop bit of the previous frame.

1. On mask set L49P devices, a new transmission will not start until the TDRE flag has been cleared.

Changing the ESCICR2[TE] bit from 0 to a 1 automatically loads the transmit shift register with a preamble of 10 logic 1s (if M = 0) or 11 logic 1s (if M = 1). After the preamble shifts out, control logic transfers the data from the SCI data register into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

Hardware supports odd or even parity. When parity is enabled, the most significant bit (msb) of the data character is the parity bit.

The transmit data register empty flag, ESCISR1[TDRE], becomes set when the SCI data register transfers a byte to the transmit shift register. The TDRE flag indicates that the SCI data register can accept new data from the internal data bus. If the transmit interrupt enable bit, ESCICR2[TIE], is also set, the TDRE flag generates a transmitter interrupt request.

When the transmit shift register is not transmitting a frame, the Tx output signal goes to the idle condition, logic 1. If at any time software clears the TE bit in ESCICR2, the transmitter enable signal goes low and the transmit signal goes idle.

If software clears TE while a transmission is in progress (ESCISR1[TC] = 0), the frame in the transmit shift register continues to shift out. To avoid accidentally cutting off the last frame in a message, always wait for TDRE to go high after the last frame before clearing TE.

To separate messages with preambles with minimum idle line time, use this sequence between messages:

1. Write the last byte of the first message to ESCIDRH/ESCIDRHL.
2. Wait for the TDRE flag to go high, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting the TE bit.
4. Write the first byte of the second message to ESCIDRH/ESCIDRL.

21.6.7 Break Characters

Writing a logic 1 to the send break bit, SBK, in eSCI control register 2 (ESCICR2) loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on the M bit in the eSCI control register 1 (ESCICR1) and on the BRK13 bit in the eSCI control register 3 (ESCICR3). As long as SBK is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next frame.

The eSCI recognizes a break character when a start bit is followed by eight or nine logic 0 data bits and a logic 0 where the stop bit should be. Receiving a break character has these effects on eSCI registers:

- Sets the framing error flag, FE
- Sets the receive data register full flag, RDRF
- Clears the SCI data registers (ESCIDRH/ESCIDRL)
- May set the overrun (OR), noise (NF), parity error (PE), or receiver active (RAF) flag, (see Section 21.5.1.4, “eSCI Status Registers (ESCISR1, ESCISR2)”)

21.6.8 Idle Characters

An idle character contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on the M bit in eSCI control register 1 (ESCICR1). The preamble is a synchronizing idle character that begins the first transmission initiated after writing the TE bit from 0 to 1.

If the TE bit is cleared during a transmission, the Tx output signal becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the frame currently being transmitted.

NOTE

When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current frame shifts out through the Tx output signal. Setting TE after the stop bit appears on Tx output causes data previously written to the SCI data register to be lost. Toggle the TE bit for a queued idle character while the TDRE flag is set and immediately before writing the next byte to the SCI data register.

21.6.9 Fast Bit Error Detection

Fast bit error detection allows the flagging of LIN bit errors when they occur, rather than flagging them after a byte transmission has completed. In order to use this feature, it is assumed a physical interface connects to the LIN bus as shown in Figure 21-24.

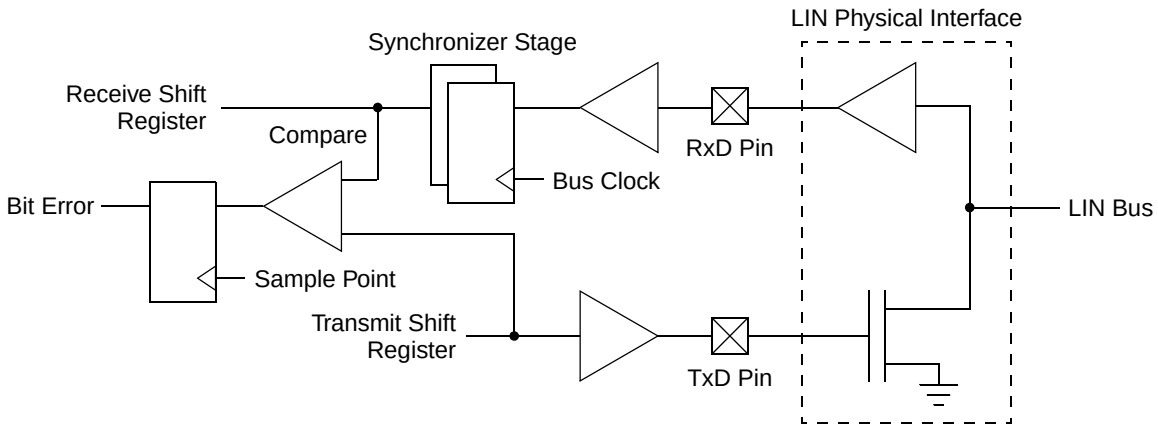


Figure 21-24. Fast Bit Error Detection on a LIN Bus

If fast bit error detection is enabled (SCICR3[FBR] = 1) the eSCI compares the transmitted and received data streams when the transmitter is active (not idle). If a mismatch between the transmitted data and the received data is detected, the following actions are taken:

- The transmission is aborted and the byte in the transmit buffer is discarded.
- The ESCISR1[TDRE, TC] flags are set.
- The ESCISR2[BERR] flag is set.

To adjust to varying bus loads, the sample point at which the incoming bit is compared to the one which was transmitted can be selected via the ESCICR4[BESM13] bit. If set, the comparison is performed at RT

clock 13, otherwise it is performed at RT clock 9 (see Figure 21-25, and Section 21.6.13, “Data Sampling”).

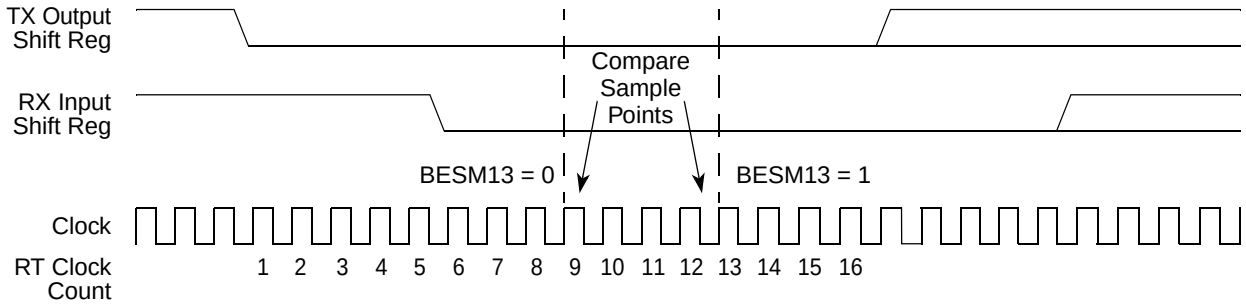


Figure 21-25. Fast Bit Error Detection Timing Diagram

21.6.10 Receiver

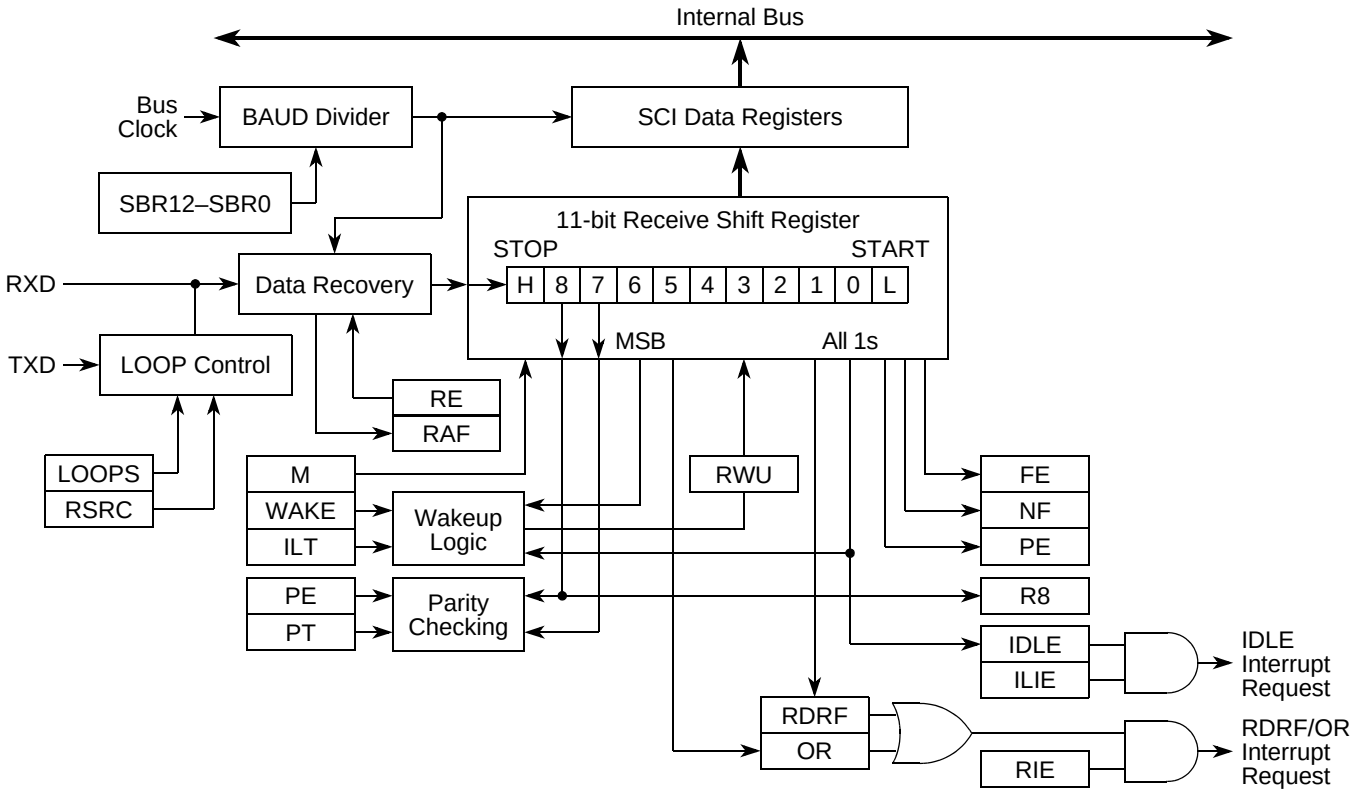


Figure 21-26. eSCI Receiver Block Diagram

21.6.11 Receiver Character Length

The eSCI receiver can accommodate either 8-bit or 9-bit data characters. The state of the M bit in eSCI control register 1 (ESCICR1) determines the length of data characters. When receiving 9-bit data, bit R8 in SCI data register high (ESCIDRH) is the ninth bit (bit 8).

21.6.12 Character Reception

During an eSCI reception, the receive shift register shifts a frame in from the Rx input signal. The SCI data register is the read-only buffer between the internal data bus and the receive shift register.

After a complete frame shifts into the receive shift register the data portion is transferred to the data register. The receive data register full flag, ESCISR1[RDRF], sets to indicate that the received byte can be read. If the receive interrupt enable, ESCICR2[RIE], is set the RDRF flag generates an RDRF interrupt request.

21.6.13 Data Sampling

The receiver samples the Rx input signal at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock (see Figure 21-27) is re-synchronized:

- After every start bit
- After the receiver detects a data bit change from logic 1 to logic 0 (after two or three of the data bit samples at RT8/RT9/RT10 return a valid logic 1 and two or three of the next RT8/RT9/RT10 samples return a valid logic 0)

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

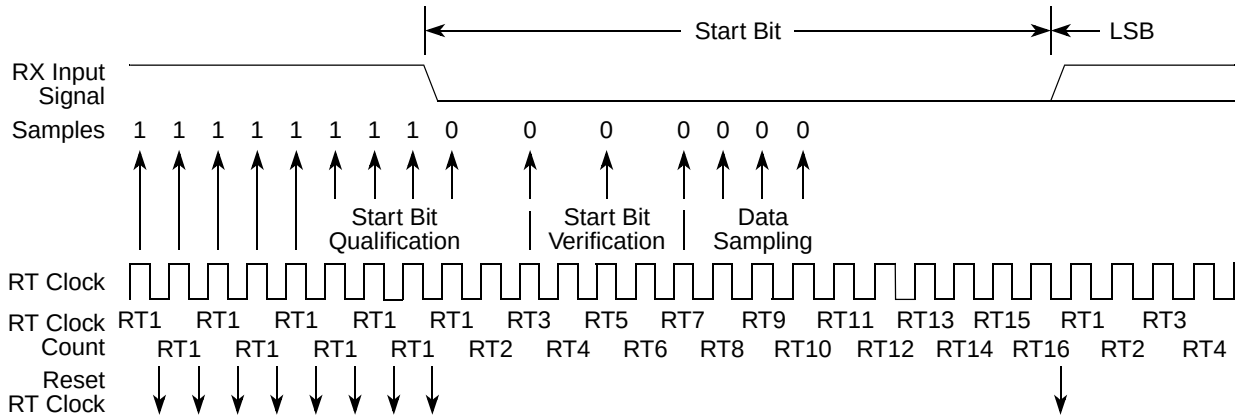


Figure 21-27. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Table 21-27 summarizes the results of the start bit verification samples.

Table 21-27. Start Bit Verification

RT3, RT5, and RT7 Samples	Start Bit Verification	Noise Flag
000	Yes	0
001	Yes	1
010	Yes	1
011	No	0
100	Yes	1
101	No	0

Table 21-27. Start Bit Verification (continued)

RT3, RT5, and RT7 Samples	Start Bit Verification	Noise Flag
110	No	0
111	No	0

If start bit verification is not successful, the RT clock is reset and a new start bit search begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 21-28 summarizes the results of the data bit samples.

Table 21-28. Data Bit Recovery

RT8, RT9, and RT10 Samples	Data Bit Determination	Noise Flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

NOTE

The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (NF) is set, however.

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 21-29 summarizes the results of the stop bit samples.

Table 21-29. Stop Bit Recovery

RT8, RT9, and RT10 Samples	Framing Error Flag	Noise Flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

In Figure 21-28, the verification samples RT3 and RT5 determine that the first low detected was noise and not the beginning of a start bit. The RT clock is reset and a new start bit search begins. The noise flag is not set because the noise occurred before a start bit was found.

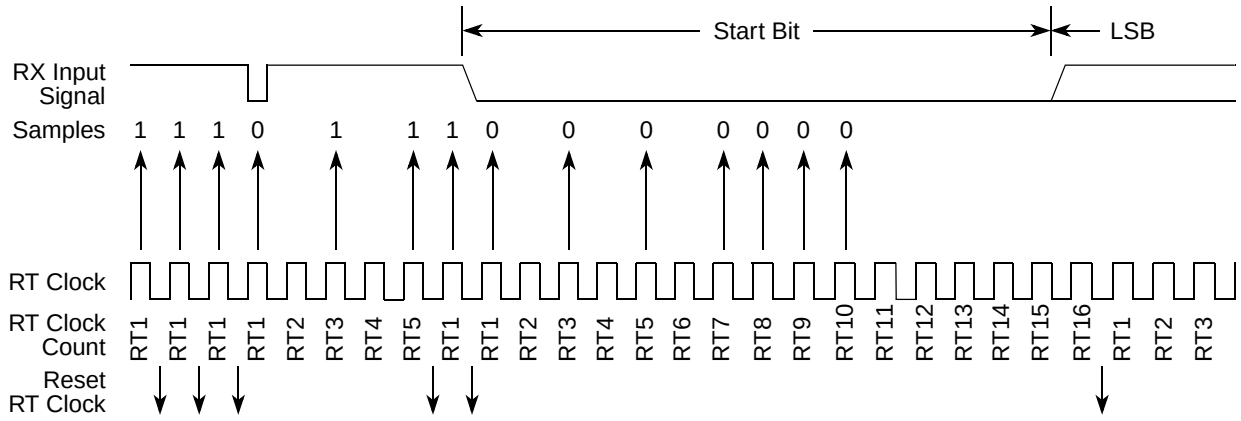


Figure 21-28. Start Bit Search Example 1

In Figure 21-29, verification sample at RT3 is high. The RT3 sample sets the noise flag. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

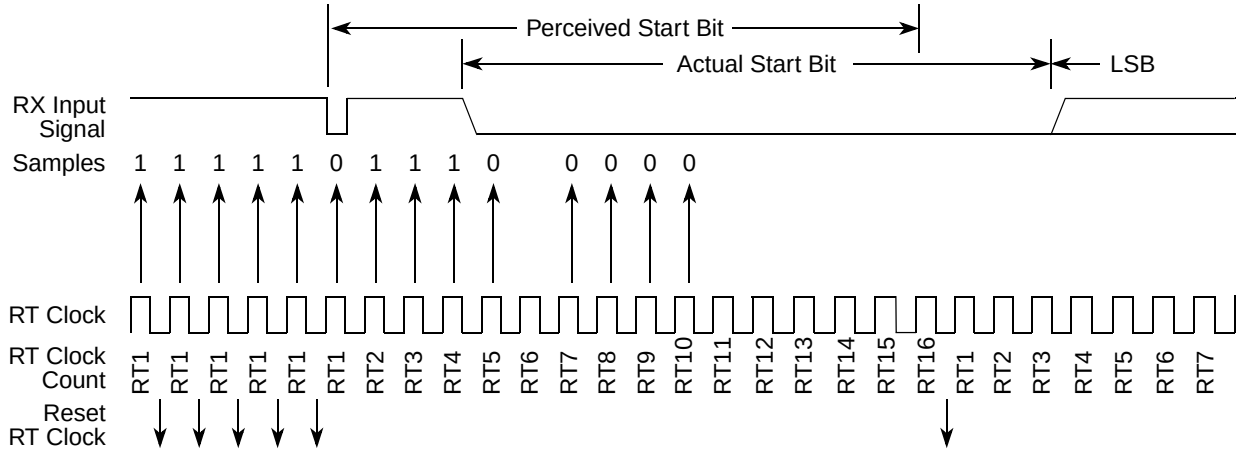


Figure 21-29. Start Bit Search Example 2

In Figure 21-30, a large burst of noise is perceived as the beginning of a start bit, although the test sample at RT5 is high. The RT5 sample sets the noise flag. Although this is a worst-case misalignment of perceived bit time, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

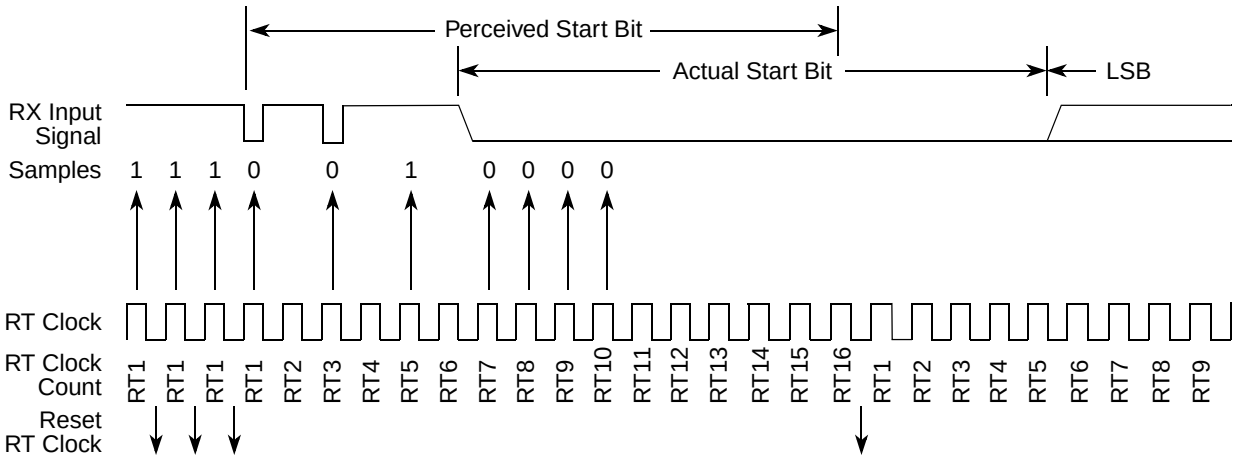


Figure 21-30. Start Bit Search Example 3

Figure 21-31 shows the effect of noise early in the start bit time. Although this noise does not affect proper synchronization with the start bit time, it does set the noise flag.

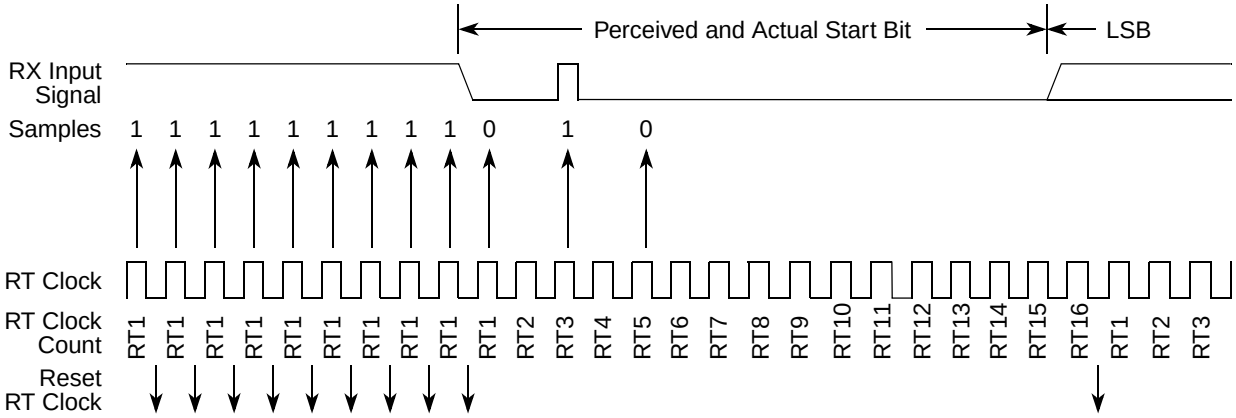


Figure 21-31. Start Bit Search Example 4

Figure 21-32 shows a burst of noise near the beginning of the start bit that resets the RT clock. The sample after the reset is low but is not preceded by three high samples that would qualify as a falling edge. Depending on the timing of the start bit search and on the data, the frame may be missed entirely or it may set the framing error flag.

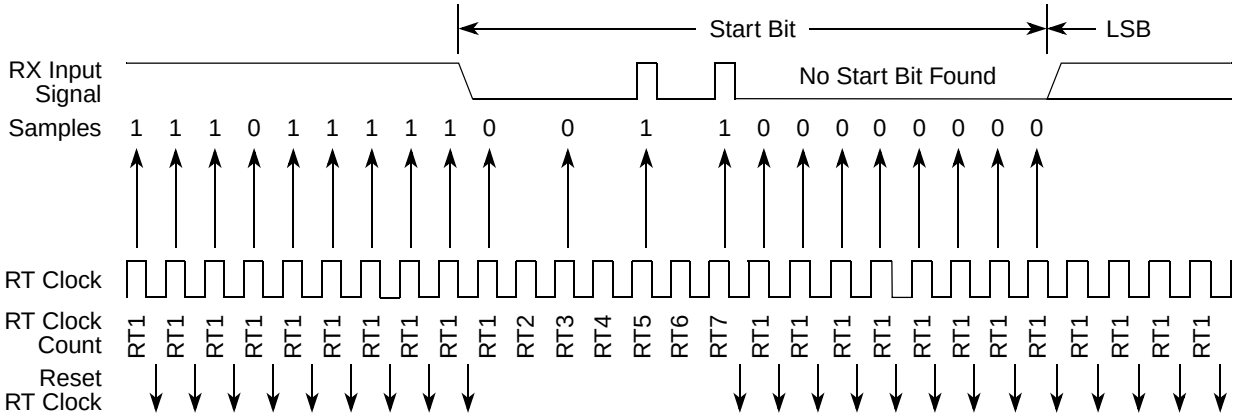


Figure 21-32. Start Bit Search Example 5

In Figure 21-33, a noise burst makes the majority of data samples RT8, RT9, and RT10 high. This sets the noise flag but does not reset the RT clock. In start bits only, the RT8, RT9, and RT10 data samples are ignored.

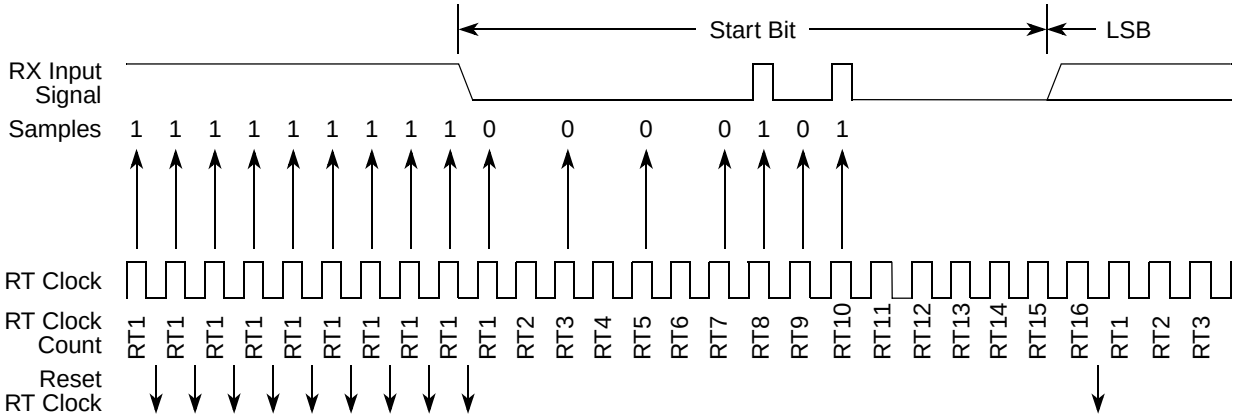


Figure 21-33. Start Bit Search Example 6

21.6.14 Framing Errors

If the data recovery logic does not detect a logic 1 where the stop bit should be in an incoming frame, it sets the framing error flag, ESCISR1[FE]. A break character also sets the FE flag because a break character has no stop bit.

21.6.15 Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples (RT8, RT9, and RT10) to fall outside the actual stop bit. A noise error will occur if the RT8, RT9, and RT10 samples are not all the same logical values. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the RT8, RT9, and RT10 stopbit samples are a logic zero.

As the receiver samples an incoming frame, it re-synchronizes the RT clock on any valid falling edge within the frame. Re-synchronization within frames will correct a misalignment between transmitter bit

times and receiver bit times. In the discussions below, RT_R and RT_T represent the receiver and transmitter RT clocks, respectively.

21.6.15.1 Slow Data Tolerance

Figure 21-34 shows how much a slow received frame can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

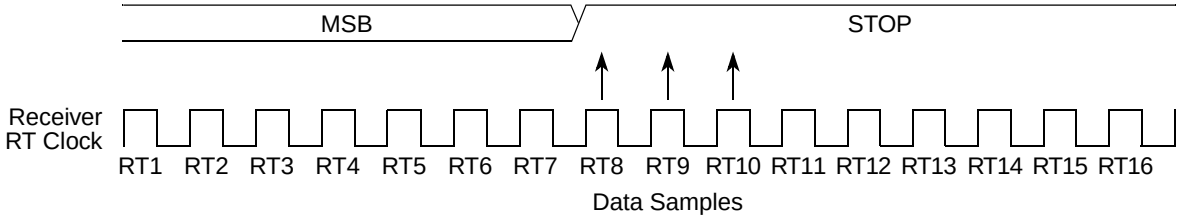


Figure 21-34. Slow Data Tolerance Example

For an 8-bit data character, the number of RT_R cycles required by the receiver to start data sampling of the stop bit is:

$$9 \text{ bit times} \times 16\ RT_R \text{ cycles} + 7\ RT_R \text{ cycles} = 151\ RT_R \text{ cycles} \qquad \textit{Eqn. 21-1}$$

With the misaligned character shown in Figure 21-34, the receiver counts 151 RT_R cycles at the point when the count of the transmitting device is:

$$9 \text{ bit times} \times 16\ RT_T \text{ cycles} = 144\ RT_T \text{ cycles} \qquad \textit{Eqn. 21-2}$$

The maximum difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

$$\frac{151 - 144}{151} \times 100 = 4.63\% \qquad \textit{Eqn. 21-3}$$

For a 9-bit data character, the number of RT_R cycles required by the receiver to start data sampling of the stop bit is:

$$10 \text{ bit times} \times 16\ RT_R \text{ cycles} + 7\ RT_R \text{ cycles} = 167\ RT_R \text{ cycles} \qquad \textit{Eqn. 21-4}$$

With the misaligned character shown in Figure 21-34, the receiver counts 167 RT_R cycles at the point when the count of the transmitting device is:

$$10 \text{ bit times} \times 16\ RT_T \text{ cycles} = 160\ RT_T \text{ cycles} \qquad \textit{Eqn. 21-5}$$

The maximum difference between the receiver count and the transmitter count of a slow 9-bit data character with no errors is:

$$\frac{167 - 160}{167} \times 100 = 4.19\% \qquad \textit{Eqn. 21-6}$$

21.6.15.2 Fast Data Tolerance

Figure 21-35 shows how much a fast received frame can be misaligned. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

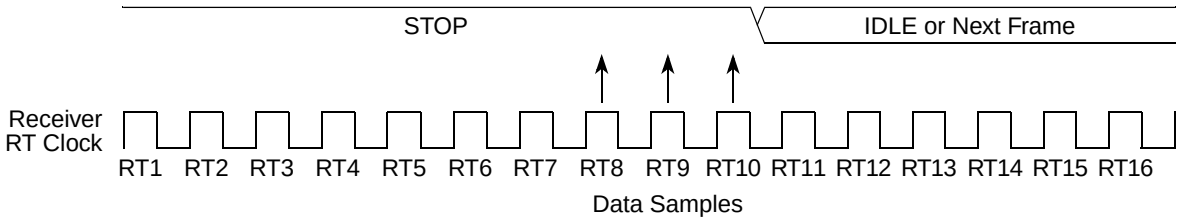


Figure 21-35. Fast Data Tolerance Example

For an 8-bit data character, the number of RT_R cycles required by the receiver to finish data sampling of the stop bit is:

$$9 \text{ bit times} \times 10 \text{ } RT_R \text{ cycles} = 154 \text{ } RT_R \text{ cycles} \qquad \textit{Eqn. 21-7}$$

With the misaligned character shown in Figure 21-35, the receiver counts 154 RT_R cycles at the point when the count of the transmitting device is:

$$10 \text{ bit times} \times 16 \text{ } RT_T \text{ cycles} = 160 \text{ } RT_T \text{ cycles} \qquad \textit{Eqn. 21-8}$$

The maximum difference between the receiver count and the transmitter count of a fast 8-bit data character with no errors is:

$$\frac{160 - 154}{160} \times 100 = 3.75\% \qquad \textit{Eqn. 21-9}$$

For a 9-bit data character, the number of RT_R cycles required by the receiver to finish data sampling of the stop bit is:

$$10 \text{ bit times} \times 16 \text{ } RT_R \text{ cycles} + 10 \text{ } RT_R \text{ cycles} = 170 \text{ } RT_R \text{ cycles} \qquad \textit{Eqn. 21-10}$$

With the misaligned character shown in Figure 21-35, the receiver counts 170 RT_R cycles at the point when the count of the transmitting device is:

$$11 \text{ bit times} \times 16 \text{ } RT_T \text{ cycles} = 176 \text{ } RT_T \text{ cycles} \qquad \textit{Eqn. 21-11}$$

The maximum difference between the receiver count and the transmitter count of a fast 9-bit data character with no errors is:

$$\frac{176 - 170}{176} \times 100 = 3.40\% \qquad \textit{Eqn. 21-12}$$

21.6.16 Receiver Wake-up

To enable the eSCI to ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wake-up bit, ESCICR2[RWU], puts the receiver into standby state during which receiver interrupts are disabled. The eSCI will still load the receive data into the ESCIDRH/ESCIDRL registers, but it will not set the receive data register full (RDRF) flag.

The transmitting device can address messages to selected receivers by including addressing information in the initial frame or frames of each message (refer to Section 21.7.2.3, “Generating a TX frame”).

The WAKE bit in eSCI control register 1 (ESCICR1) determines how the eSCI is brought out of the standby state to process an incoming message. The WAKE bit enables either idle line wake-up or address mark wake-up.

21.6.16.1 Idle Input Line Wake-up (WAKE = 0)

In this wake-up method, an idle condition on the Rx Input signal clears the ESCICR2[RWU] bit and wakes up the eSCI. The initial frame or frames of every message contain addressing information. The CPU can read and evaluate the addressing information and decide whether it should receive the frame. If it decides not to receive, it can set the eSCI’s RWU bit and return the eSCI to the standby state. The RWU bit remains set and the receiver remains on standby until another idle character appears on the Rx Input signal.

Idle line wake-up requires that messages be separated by at least one idle character and that no message contains idle characters.

The idle character that wakes a receiver does not set the receiver idle bit, ESCISR1[IDLE], or the receive data register full flag, RDRF.

The idle line type bit, ESCICR1[ILT], determines whether the receiver begins counting logic 1s as idle character bits after the start bit or after the stop bit.

21.6.16.2 Address Mark Wake-up (WAKE = 1)

In this wake-up method, a logic 1 in the most significant bit (msb) position of a frame clears the RWU bit and wakes up the eSCI. The logic 1 in the msb position marks a frame as an address frame that contains addressing information. All receivers evaluate the addressing information, and the receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its RWU bit and return to the standby state. The RWU bit remains set and the receiver remains on standby until another address frame appears on the Rx Input signal.

The logic 1 msb of an address frame clears the receiver’s RWU bit before the stop bit is received and sets the RDRF flag.

Address mark wake-up allows messages to contain idle characters but requires that the msb be reserved for use in address frames.

NOTE

With the WAKE bit clear, setting the RWU bit after the Rx Input signal has been idle can cause the receiver to wake up immediately.

21.6.17 Single-Wire Operation

Normally, the eSCI uses two pins for transmitting and receiving. In single-wire operation, the RXD pin is disconnected from the eSCI. The eSCI uses the TXD pin for both receiving and transmitting.

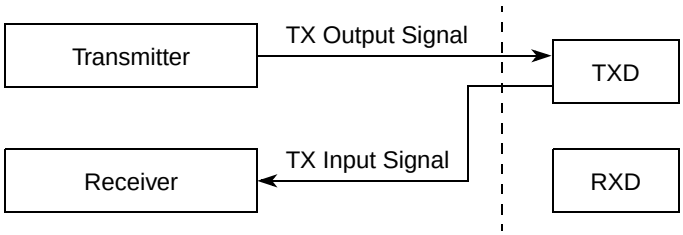


Figure 21-36. Single-Wire Operation (LOOPS = 1, RSRC = 1)

Enable single-wire operation by setting the LOOPS bit and the receiver source bit, RSRC, in eSCI control register 1 (ESCICR1). Setting the LOOPS bit disables the path from the Rx Input signal to the receiver. Setting the RSRC bit connects the receiver input to the output of the TXD pin driver. Both the transmitter and receiver must be enabled (TE = 1 and RE = 1).The TXDIR bit (in ESCICR3) determines whether the TXD pin is going to be used as an input (TXDIR = 0) or an output (TXDIR = 1) in this mode of operation.

21.6.18 Loop Operation

In loop operation the transmitter output goes to the receiver input. The Rx Input signal is disconnected from the eSCI.

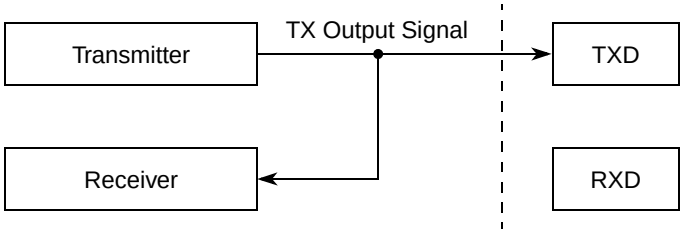


Figure 21-37. Loop Operation (LOOPS = 1, RSRC = 0)

Enable loop operation by setting the LOOPS bit and clearing the RSRC bit in eSCI control register 1 (ESCICR1). Setting the LOOPS bit disables the path from the Rx Input signal to the receiver. Clearing the RSRC bit connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (TE = 1 and RE = 1).

21.6.19 eSCI Operating Mode Details

21.6.19.1 Run Mode

Normal mode of operation.

21.6.19.2 Doze Mode

eSCI operation in doze mode depends on the state of the ESCICR1[SCISDOZ] bit:

- If SCISDOZ is clear, the eSCI operates normally when the system is in doze mode.
- If SCISDOZ is set and the system is in doze mode, the eSCI module enters a power-conservation state as soon as the current operation is completed. In normal eSCI mode, this means the eSCI will

enter doze mode, as soon as the current byte is transmitted or received, respectively. In LIN mode, it will enter doze mode as soon as the current RX or TX frame has completed.

If the eSCI is not disabled in doze mode, an eSCI interrupt request can be used to bring the system out of doze mode.

21.6.19.3 Module Disable

The module disable bit (MDIS) in the eSCI control register 3 can be used to turn off the eSCI. This will prevent the eSCI core being clocked, and thus save power. By default the eSCI is disabled, so the first step for using the eSCI is to enable it by setting the MDIS bit to 0.

21.6.19.4 Stop Mode

The eSCI is inactive during stop mode for reduced power consumption. In order to avoid corrupting data as much as possible, it will prevent the system from entering stop mode before the current operation is completed.

In SCI mode the eSCI will keep the system from entering stop mode until the current byte has been received or transmitted. Thus, no data will be corrupted before the eSCI is ready to shut down. However, since a connected device might continue sending data while the system is stopped, the first byte in the eSCI after waking up from stop mode could still be invalid.

In LIN mode the eSCI will wait at least until the current byte has been received or transmitted. If the LIN FSM (finite state machine) has some more bytes to receive or transmit which do not require processor access (CRC and checksum bytes, last transmit byte of a frame) the eSCI will delay stop mode until these operations are complete, too.

If a LIN frame was aborted, the eDMA controller will be out of sync, and the channel needs to be restarted after leaving stop mode.

21.6.20 Interrupt Operation

This section summarizes how the eSCI generates interrupt requests and how the MCU should acknowledge that request. The interrupt vector offset and interrupt number are shown in Table 6-2 on page 6-85. The eSCI only has a single interrupt signal and all the following interrupts, when generated, are ORed together and issued through that signal.

21.6.20.1 Interrupt Flags and Masks

Table 21-30 lists the sources that can generate an eSCI interrupt, the status flag that indicates the event occurred, and the corresponding enable bit for interrupt. Refer to the listed detail pages for a full description of how each flag bit is set, cleared and masked.

Table 21-30. eSCI Interrupt Sources Summary

Interrupt Source	Interrupt Description	Status Register	Flag Bit	Detail Page	Control Register	Enable Bit	Detail Page
Receiver	Parity error has occurred.	ESCISR1	PF	21-416	ESCICR4	PFIE	21-413
Receiver	Frame error has occurred.	ESCISR1	FE	21-416	ESCICR4	FEIE	21-413
Receiver	Noise has been detected.	ESCISR1	NF	21-416	ESCICR4	NFIE	21-413
Receiver	Overrun condition has occurred.	ESCISR1	OR	21-416	ESCICR4	ORIE	21-413
Receiver	Receiver input has become idle.	ESCISR1	IDLE	21-416	ESCICR2	ILIE	21-412
Receiver	Received data is available in the SCI data register.	ESCISR1	RDRF	21-416	ESCICR2	RIE	21-412
Transmitter	Transmit is complete.	ESCISR1	TC	21-416	ESCICR2	TCIE	21-412
Transmitter	Byte was transferred from ESCIDRH/L to the transmit shift register.	ESCISR1	TDRE	21-416	ESCICR2	TIE	21-412
LIN	Bit error has been detected (only valid in LIN mode).	ESCISR2	BERR	21-416	ESCICR3	IEBERR	21-412
LIN	LIN frame has completed.	LINSTAT1	FRC	21-417	LINCTRL2	FCIE	21-420
LIN	Checksum error has been detected.	LINSTAT1	CKERR	21-417	LINCTRL2	CKIE	21-420
LIN	CRC error has been detected.	LINSTAT1	CERR	21-417	LINCTRL2	CIE	21-420
LIN	Physical bus error has been detected.	LINSTAT1	PBERR	21-417	LINCTRL2	PBIE	21-420
LIN	Response of the slave was too slow (slave timeout).	LINSTAT1	STO	21-417	LINCTRL2	STIE	21-420
LIN	Wake-up character received from a LIN frame.	LINSTAT1	LWAKE	21-417	LINCTRL2	WUIE	21-420
LIN	LIN hardware can accept a control or data byte.	LINSTAT1	TXRDY	21-417	LINCTRL2	TXIE	21-420
LIN	LIN hardware has received a data byte.	LINSTAT1	RXRDY	21-417	LINCTRL2	RXIE	21-420
LIN	The LINRX register has overflowed.	LINSTAT2	OVFL	21-418	LINCTRL3	OFIE	21-420

21.7 Initialization / Application Information

21.7.1 Using the eSCI in 9-bit Data Mode

It may be desirable to set the 9th data bit for each write and to retrieve the 9th data bit for each read. This can be accomplished efficiently by defining the data structure for the eSCI data registers (see Section 21.5.1.3, “eSCI Data Registers (ESCIDRH, ESCIDRL)”) appropriately.

Normally the header files for the eSCI are written in this manner:

```
.
.
.
volatile reg8 ESCIDRH_A;     /* eSCI_A_REGISTER_MAP_OFFSET + 0x06 */
volatile reg8 ESCIDRL_A;     /* eSCI_A_REGISTER_MAP_OFFSET + 0x07 */
.
.
.
```

This structure requires two writes or two reads to access the transmit or receive data:

```
.
.
.
SCI_A->ESCIDRH_A = tx_dat8 << 7;      /* assumes 9th bit is in txdat8[0] */
SCI_A->ESCIDRL_A = tx_dat7_0;
rx_dat8_0 = SCI_A->ESCIDRH_A << 1 | SCI_A->ESCIDRL_A;
.
.
.
```

If the 9th bit is unused, or only written and read occasionally, such a declaration with two 8-bit registers is preferable. However if the 9th bit needs to be written and read on every access to the eSCI data register, a single 16-bit register would be more efficient. Because the register pair is aligned on a 16-bit boundary in the memory map, the header files can be written in this manner:

```
.
.
.
volatile reg16 ESCIDR_A;      /* eSCI_A_REGISTER_MAP_OFFSET + 0x06 */
.
.
.
```

This allows a single write or read to access all 9 bits:

```
.
.
.
SCI_A->ESCIDR_A = tx_dat8_0;
rx_dat8_0 = SCI_A->ESCIDR_A;
.
.
.
```

Note that the 9th data bit is at ESCIDRH[6] for writes and ESCIDRH[7] for reads. Thus, bits 8 and 7:0 of a character cannot be concatenated in the tx_dat8_0 variable before a write, nor will they be concatenated in rx_data8_0 after a read.

If the eDMA is used to service the eSCI, configuring the appropriate eDMA TCD*n* with SADDR = ESCIDRH address and SSIZE = 0b001 will transfer 9-bit frames (refer to Section 12.3.1.16, “Transfer Control Descriptors (TCDn),” on page 12-156). The 9th bit alignment must be considered for the data buffer structure used for a eDMA transfer.

21.7.2 Using the LIN hardware

The eSCI provides special support for the Local Interconnect Network (LIN) protocol. It can be used to automate most tasks of a LIN master. In conjunction with the DMA request interface it is possible to transmit entire frames (or sequences of frames) and receive data from LIN slaves without any CPU intervention. There is no special support for LIN slave mode, if required it should be implemented in software.

A LIN frame consists of a break character (10 or 13 bits, configurably), a sync field, an ID field, n data fields (n could be 0) and a checksum field. The data and checksum bytes are either provided by the LIN

master (TX frame) or by the LIN slave (RX frame). The header fields will always be generated by the LIN master.

Figure 21-38. Typical LIN Frame

The LIN hardware is highly configurable, and allows frames to be generated for LIN slaves from all revisions of the LIN standard. The settings need to be adjusted for the capabilities of the slave device.

In order to activate the LIN hardware, the LIN mode bit in the LINCTRL register needs to be set, other settings like double stop flags after bit errors and automatic parity bit generation, are also available.

The eSCI settings need to be made according to the LIN specification: it needs to be configured for 2-wire operation (2 wires connected to the LIN transceiver) with 8 data bytes and no parity. Normally a 13-bit break is used, but it can also be configured for 10-bit breaks as required by the application.

21.7.2.1 LIN Setup

Since the eSCI is for general purpose, some of the settings are not applicable for LIN operation. The following setup applies for all LIN applications, regardless of the slave type:

a) The module must be enabled by clearing ESCICR3[MDIS]
b) The transmitter and receiver must be enabled (ESCICR2[TE, RE] = 1)
c) The data format must be 8 bits (ESCICR1[M] = 0)
d) Parity must be disabled (ESCICR1[PE] = 0)
e) ESCICR2[TIE, TCIE, RIE] interrupt enable bits should be clear, as the LIN interrupts are used instead
f) The eSCI is placed into LIN mode (LINCTRL1[LIN] = 1)
g) The LIN standard requires that the break character always be 13 bits (ESCICR3[BRK13] = 1)
h) LINCTRL1[LDBG] = 0 and ESCICR3[BSTP] = 1 in order to prevent the LIN FSM from negating the DMA request on bit errors
i) Enable pull-down devices on the RXD_*x* pin (PIM pins PG0, PG2, PG12 or PG14; refer to Section 18.5.1.1, “PIM Port x Pin Configuration Registers (CONFIGn_x),” on page 18-286)
j) Enable appropriate error indicators: ESCICR3[IEBERR], ESCICR4[NFIE, FEIE], LINCTRL2[STIE, PBIE, CIE, CKIE], and LINCTRL3[OFIE]
k) Initially a wake-up character may need to be transmitted on the LIN bus, so that the LIN slaves activate

Other settings such as baud rate, DMA interface, etc., will depend on the LIN slaves to which the eSCI is connected and on the desired operation of the eSCI.

21.7.2.2 Features of the LIN Hardware

The LIN hardware has several features to support different revisions of the LIN slaves. In the TX register it can be configured whether header bits should be included in the checksum (on a frame by frame basis,

to support LIN slaves with different LIN revisions). The LIN control register allows software to determine whether the parity bits in the ID field should be calculated automatically and whether double stop flags should be inserted after a bit error. The BRK13 bit in the eSCI control register 3 decides whether to generate 10 or 13 bit break characters.

The application software can decide to turn off the checksum generation/verification on a per frame basis and handle that function on its own. Also it can decide to let the LIN hardware append two CRC Bytes (Figure 21-39). These are not part of the LIN standard, but could be part of the application layer i.e. they would be treated as data bytes by the LIN protocol. This can be useful when very long frames are transmitted. By default the CRC polynomial used is the same polynomial as for the CAN protocol.

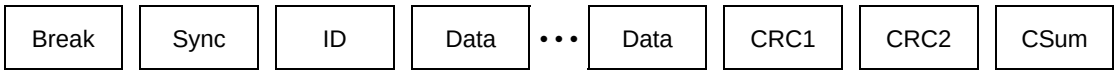


Figure 21-39. LIN Frame With CRC Bytes

It is possible to force a resync of the LIN FSM (finite state machine), with the LRES bit in the LIN control register, however normally the LIN hardware will automatically abort a frame after detecting a bit error.

21.7.2.3 Generating a TX frame

The following procedure illustrates how to generate a basic TX frame.

The frame is controlled via the TX register (LINTX). Initially the application software will need to check the TXRDY bit (either using an interrupt, the TX DMA interface, or by polling the LIN status register). If the bit is set to 1 the register is writable. Before each write the bit needs to be checked (automatic in DMA mode). The first byte written to the TX register has to contain the LIN ID field, then the length of the frame is specified (0 to 255 Bytes), and a control byte (frame direction, checksum/crc settings) are written. (The timeout bits are skipped for TX frames, since they only refer to LIN slaves). After this frame- data is known the LIN hardware will start to generate the LIN frame.

First it will transmit a break field, then the sync field and the ID field. Afterwards the TX register will accept data bytes, and the LIN hardware will transmit these bytes as soon as they are available and can be sent out. After the last step it will automatically append the checksum field.

It is possible to setup an eDMA channel to handle all the tasks required to send a TX frame (see Figure 21-40). For this, the TX DMA channel has to be activated by setting the TXDMA bit. The control information for the LIN frame (ID, message length, TX/RX type, timeout etc.) and the data bytes can be stored at an appropriate memory location. The eDMA controller will then be set up to transfer this block of memory to a location (the TX register). After transmission is complete either the eDMA controller or the LIN hardware can generate an interrupt to the CPU. While the entire communication, bit error and physical bus error checking, check sum and CRC generation (checking on the Rx side) is handled by the eDMA controller and the eSCI.

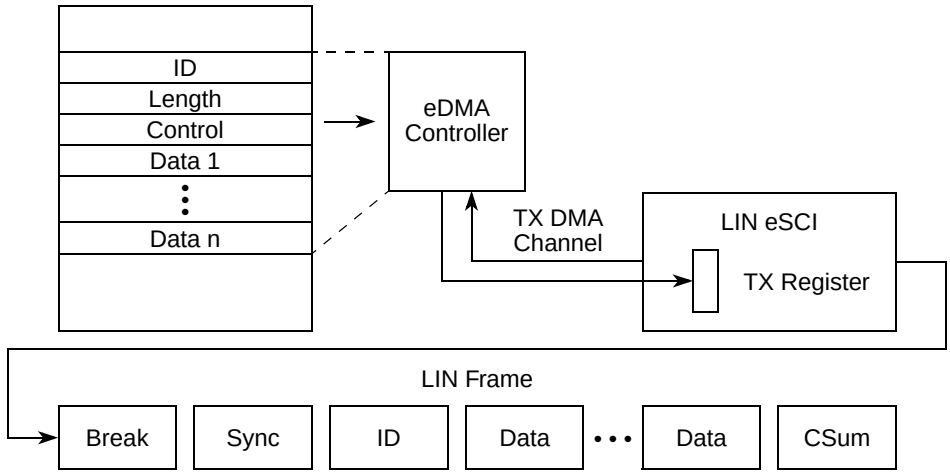


Figure 21-40. DMA Transfer of a TX Frame

21.7.2.4 Generating an RX frame

For RX frames the header information will still be provided by the LIN master - the data, CRC and checksum bytes (as enabled) will be provided by the LIN slave. The LIN master will verify CRC and checksum bytes transmitted by the slave.

For an RX frame the control information needs to be written to the TX register in the same manner as for the TX frames. Additionally the timeout bits need to be written to define the time to complete the entire frame. Afterwards the RXRDY bit needs to be checked (either with an interrupt, RX DMA interface, or by polling) to detect the incoming data bytes. The checksum byte will normally not appear in the RX register, instead the LIN hardware will verify the checksum and raise an interrupt if it is not correct.

Two eDMA channels can be used when executing an RX frame - one to transfer the header/control information from a memory location to the TX register, and one to transfer the incoming data bytes from the RX register to a table in memory (see Figure 21-41). After the last byte has been stored, the eDMA controller can indicate completion to the CPU.

It is also possible to setup a whole sequence of RX and TX frames, and generate a single event at the end of that sequence.

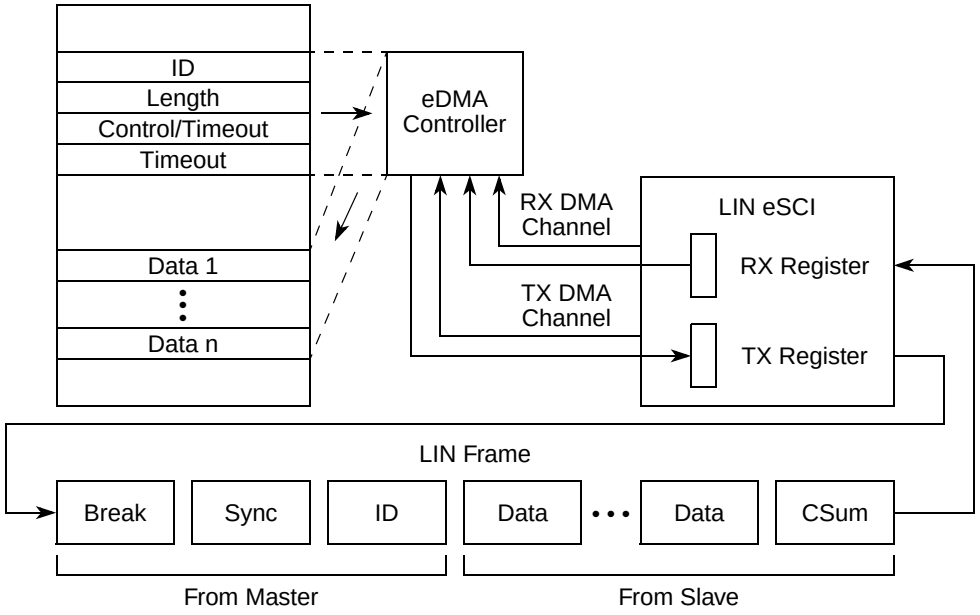


Figure 21-41. DMA Transfer of an RX Frame

21.7.2.5 LIN Error Handling

The LIN hardware can detect several error conditions of the LIN protocol. It will receive every byte that was transmitted, and compare it with the intended values. If there is a mismatch, a bit error will be raised, and the LIN FSM will return to its start state.

For a RX frame the LIN hardware can detect a slave timeout error - the exact value can be set via the timeout bits in the TX register. If the frame is not complete within the number of clock cycles specified there, the LIN FSM will return to its start state, and the STO (slave time-out) interrupt will be raised.

The LIN protocol supports a sleep mode - after 25000 bus cycles of inactivity the bus is assumed to be in sleep mode. Normally entering sleep mode can be avoided, if the LIN master is regularly creating some bus activity. Otherwise the timeout state needs to be detected by the application software - e.g. by setting a timer.

Both LIN masters and LIN slaves can cause the bus to exit sleep mode by sending a break signal. The LIN hardware will generate such a break when the WU bit in the LIN control register is written. After transmitting this break the LIN hardware will not send out data (i.e. not raise the TXRDY flag) before the Wake-up Delimiter period has expired. This period can be selected by setting the WUD bits in the LIN control register.

Break signals sent by a LIN slave are detected by the LIN hardware, and indicated by setting the WAKE flag in the LIN status register.

A physical bus error (LIN bus is permanently stuck at a fixed value) will set several error flags. If the input is permanently low, the eSCI will set the framing error flag (FE) in the eSCI status register. If the RX input remains stuck at a fixed value for 15 cycles after a transmission has started, the LIN hardware will set the PBERR flag in the LIN status register. In addition a bit error may be generated.

21.7.2.6 LIN Wake-up

The LIN hardware automatically detects LIN 1.x wakeup characters and can generate them with the WU bit in LINCTRL1. For LIN 2.0 wakeup characters the requirements are more flexible. Instead of a 0x80 character at the currently selected baudrate, the spec requires a low pulse of 250 ms to 5 ms. For this the baud rate needs to be set to 32 kBaud or lower.

In order to generate a valid wakeup character according to LIN 2.0, the eSCI first needs to be programmed to a baud rate lower than 32 kBaud, then WU can be set. Should the application require a higher baud rate, then this rate can be set once the wakeup character has been transmitted.

For wakeup detection the length of the wakeup pulse depends on the LIN slave node. A wakeup which does not conform to LIN 1.3 will show up as a frame error, FE. Provided that the low pulse is longer than 8 bit-times (this can be controlled by determining the baud rate), the LWAKE flag will be also be set. The application then needs to wait for 10 ms and clear the FE and LWAKE bits.

21.7.2.7 System Wake-up on LIN Bus Activity

It may be desirable to generate a wake-up interrupt to the system when a LIN wakeup character is received. This can be implemented by switching the RXD receive pin into general purpose input mode and setting up the GPI interrupt appropriately so that it can create wake-up interrupts (refer to Section 18.6.3, “General Purpose Input Mode,” on page 18-298), before entering STOP mode. The length of the wakeup pulse needs to conform to the requirements specified for the GPIO interrupt. LIN 2.0 compliant wakeups should fulfill these requirements.

Alternatively, an additional pin can be dedicated just for the wake-up interrupt. This pin would be configured for GPI mode, and externally connected to the RXD receive pin.

Chapter 22
Deserial Serial Peripheral Interface Module (DSPI)

22.1 Overview

MAC7100 Family devices implement up to two Deserial Serial Peripheral Interfaces (DSPI_A and DSPI_B). Fewer DSPI modules and different chip select counts are implemented on family devices. Refer to Table 1-1 on page 1-3 for a general description of each device. Figure 22-1 shows a block diagram of the Deserial Serial Peripheral Interface (DSPI) module.

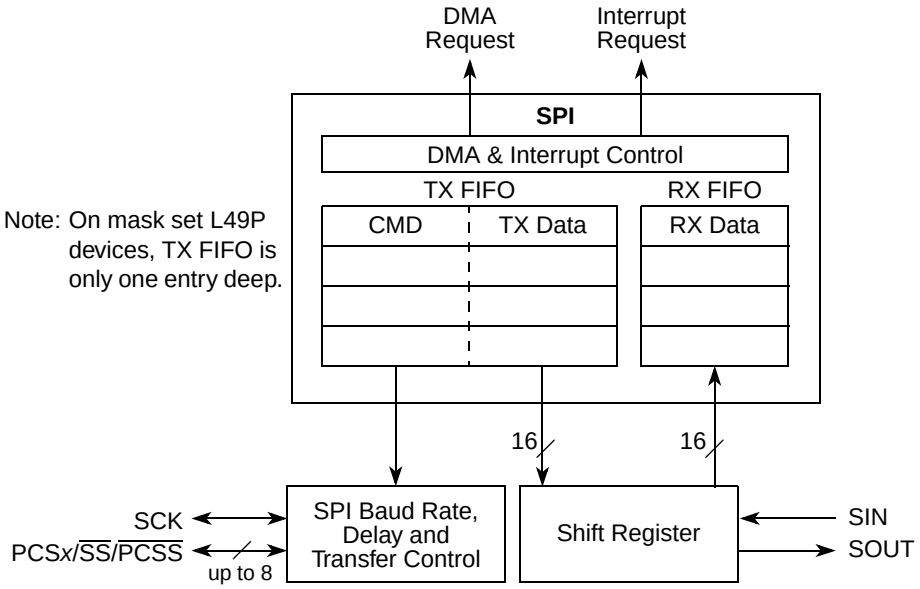


Figure 22-1. DSPI Block Diagram

The MAC7100 family provides the DSPI with a range of chip select lines depending on the device. Each DSPI supports up to eight peripheral chip select lines, offering selection of up to 8 external devices without external hardware, expandable to 256 with an external demultiplexer. One of the chip selects (PCS5_*x* / $\overline{\text{PCSS}}$_*x*) can be used to provide a chip select strobe in order to eliminate decoding glitches generated when the chip selects change. This allows glitch free selection of up to 128 external devices.

The DSPI is implemented with separate transmit and receive FIFOs with a depth of four entries. [1] These FIFOs can be accessed either by the CPU or by the eDMA to enable queuing operations to be performed. Each DSPI has a separate DMA request channel for the transmit and the receive sides of the module.

For queued operations the SPI queues reside in system RAM which is external to the DSPI. Data transfers between the queues and the DSPI FIFOs are accomplished through the use of the eDMA controller or through host software. Figure 22-2 shows a DSPI with external queues in system RAM.

Each of the DSPI modules can be independently disabled by writing to the MDIS bit in the module's MCR. Upon disabling the module, the clock is turned off, although most of the module registers remain available

1. On mask set L49P devices, the TX FIFO is only one entry deep.

to be accessed by the core across the peripheral bus. The MDIS bit is intended to be used when the module is not required in the application.

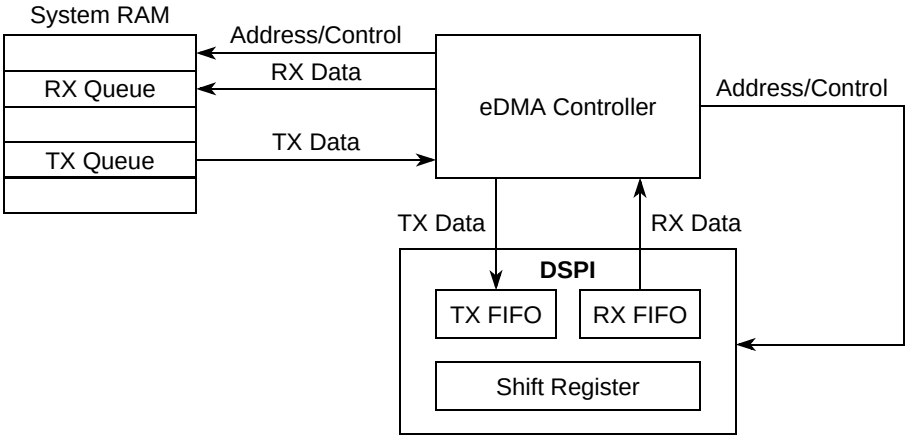


Figure 22-2. DSPI with Queues and DMA

22.2 Features

The DSPI supports these SPI features:

- Full-duplex, three-wire synchronous transfers
- Master and slave mode
- Buffered transmit operation using the 1- to 4-entry TX FIFO [1]
- Buffered receive operation using the 1- to 4-entry RX FIFO
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues
- Visibility into TX and RX FIFOs for ease of debugging
- Programmable transfer attributes on a per-frame basis:
  - Up to six transfer attribute registers available
  - Serial clock with programmable polarity and phase
  - Various programmable delays
  - Programmable serial frame size of 4 to 16 bits, expandable with software control
  - Continuously held chip select capability
- Up to 8 peripheral chip selects (PCS*x*), expandable to 258 with external demultiplexer
- Deglitching support for up to 128 peripheral chip selects with external demultiplexer
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
- 6 Interrupt conditions (all share one interrupt vector):
  - End of queue reached (EOQF)
  - TX FIFO is not full (TFFF)
  - Transfer of current frame complete (TCF)

1. On mask set L49P devices, the TX FIFO is only one entry deep.

  - Attempt to transmit with an empty Transmit FIFO (TFUF)
  - RX FIFO is not empty (RFDF)
  - Frame received while Receive FIFO is full (RFOF)
- Modified SPI transfer formats for communication with slower peripheral devices
- Power-saving architectural features
  - Support for stop and doze modes

22.3 Modes of Operation

The DSPI has five distinct modes:

- Master Mode
- Slave Mode
- Module Disable Mode
- External Stop Mode
- Debug Mode

Master, slave, and module disable modes are module-specific mode while external stop and debug modes are device-specific modes.

The module-specific modes are entered by host software writing to a register. The device-specific modes are controlled by signals external to the DSPI. The device-specific modes are modes that the entire device enters, in parallel to the DSPI being in one of its module-specific modes. See Section 22.6.1, “DSPI Operating Mode Details,” for more details.

22.4 Signal Description

Table 22-1 lists the signals that connect off chip. Note that the Port Integration Module (PIM) must be configured to enable the peripheral function of the appropriate pins (refer to Section 18.6.2, “Peripheral Mode,” on page 18-296) prior to configuring a DSPI channel.

Table 22-1. DSPI Signal Properties

Name	I/O Type	Function	
		Master Mode	Slave Mode
PCS0 / $\overline{SS}$	Output / Input	Peripheral Chip Select 0	Slave Select
PCS[1:4, 6:7]	Output	Peripheral Chip Selects 1–4, 6–7	Unused
PCS5 / $\overline{PCSS}$	Output	Peripheral Chip Select 5 / Peripheral Chip Select Strobe	Unused
SIN	Input	Serial Data In	Serial Data In
SOUT	Output	Serial Data Out	Serial Data Out
SCK	Output / Input	Serial Clock (output)	Serial Clock (input)

22.4.1 PCS0_*x* / $\overline{\text{SS}}$_*x* — Peripheral Chip Select or Slave Select

In master mode, the PCS0 signal is a peripheral chip select output that selects which slave device the current transmission is intended for.

In slave mode, the $\overline{\text{SS}}$ signal is a slave select input signal that allows a SPI master to select the DSPI as the target for transmission.

22.4.2 PCS[1:4, 6:7]_*x* — Peripheral Chip Selects 1–4, 6–7

PCS[1:4, 6:7] are peripheral chip select output signals in master mode. In slave mode these signals are not used. Note that PCS[3, 4, 6, 7] are available only on MAC7136 devices.

22.4.3 PCS5_*x* / $\overline{\text{PCSS}}$_*x* — Peripheral Chip Select 5 or Chip Select Strobe

PCS5 is a peripheral chip select output signal. When the DSPI is in master mode and PCSSE bit in the DSPI*x*_MCR is negated, this signal is used to select which slave device the current transfer is intended for.

$\overline{\text{PCSS}}$ provides a strobe signal that can be used with an external demultiplexer for deglitching of the PCS signals. When the DSPI is in master mode and the PCSSE bit in the DSPI*x*_MCR is set, the $\overline{\text{PCSS}}$ provides the appropriate timing for the decoding of the PCS[0:4, 6:7] signals which prevents glitches from occurring on the demultiplexer output.

This signal is not used in slave mode.

22.4.4 SIN_*x* — Serial Input

SIN is a serial data input signal.

22.4.5 SOUT_*x* — Serial Output

SOUT is a serial data output signal.

22.4.6 SCK_*x* — Serial Clock

SCK is a serial communication clock signal. In master mode, the DSPI generates the SCK. In slave mode, SCK is an input from an external bus master.

22.5 Memory Map / Register Definition

Table 22-2 shows the DSPI memory map. The Offset listed for each register is the address offset. The total address for each register is the sum of the base address for the DSPI module and the address offset for each register.

Table 22-2. DSPI Memory Map

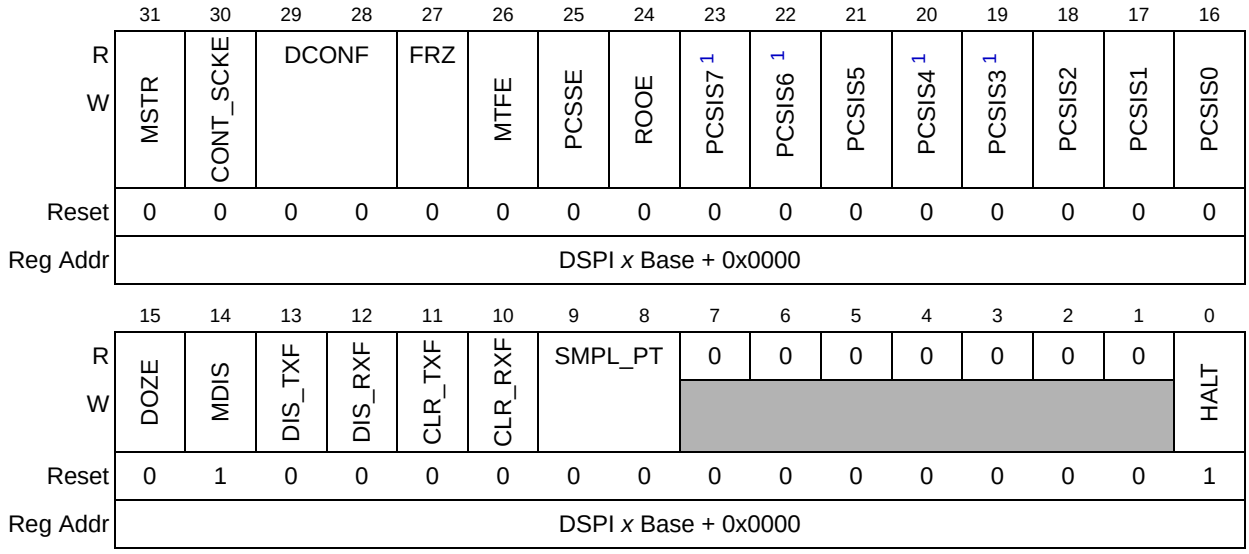
DSPI *x* Offset	Register Description
0x0000	DSPI Module Configuration Register (DSPI*x*_MCR)
0x0004	Reserved
0x0008	DSPI Transfer Count Register (DSPI*x*_TCR)
0x000C	DSPI Clock and Transfer Attributes Register 0 (DSPI*x*_CTAR0)
0x0010	DSPI Clock and Transfer Attributes Register 1 (DSPI*x*_CTAR1)
0x0014	DSPI Clock and Transfer Attributes Register 2 (DSPI*x*_CTAR2) [1]
0x0018	DSPI Clock and Transfer Attributes Register 3 (DSPI*x*_CTAR3) [1]
0x001C	DSPI Clock and Transfer Attributes Register 4 (DSPI*x*_CTAR4) [1]
0x0020	DSPI Clock and Transfer Attributes Register 5 (DSPI*x*_CTAR5) [1]
0x0024–0x0028	Reserved
0x002C	DSPI Status Register (DSPI*x*_SR)
0x0030	DSPI DMA/Interrupt Request Select and Enable Register (DSPI*x*_RSER)
0x0034	DSPI Push TX FIFO Register (DSPI*x*_PUSHR)
0x0038	DSPI Pop RX FIFO Register (DSPI*x*_POPR)
0x003C	DSPI Transmit FIFO Register 0 (DSPI*x*_TXFR0)
0x0040	DSPI Transmit FIFO Register 1 (DSPI*x*_TXFR1) [1]
0x0044	DSPI Transmit FIFO Register 2 (DSPI*x*_TXFR2) [1]
0x0048	DSPI Transmit FIFO Register 3 (DSPI*x*_TXFR3) [1]
0x004C–0x0078	Reserved
0x007C	DSPI Receive FIFO Register 0 (DSPI*x*_RXFR0)
0x0080	DSPI Receive FIFO Register 1 (DSPI*x*_RXFR1)
0x0084	DSPI Receive FIFO Register 2 (DSPI*x*_RXFR2)
0x0088	DSPI Receive FIFO Register 3 (DSPI*x*_RXFR3)
0x008C–0x00B8	Reserved

[1] This register is not present on mask set L49P devices, and the offset must be treated as reserved.

22.5.1 Register Descriptions

22.5.1.1 DSPI Module Configuration Register (DSPI*x*_MCR)

The DSPI*x*_MCR contains bits which configure various attributes associated with DSPI operation. The HALT and MDIS bits can be changed at any time but will only take effect on the next frame boundary. Only the HALT and MDIS bits in the DSPI*x*_MCR may be changed while the DSPI is in the Running state.



1 PCSIS[7:6, 4:3] are available on MAC7136 only. On all other devices, bits 19, 20, 22 and 23 are reserved.

Figure 22-3. DSPI Module Configuration Register (DSPI*x*_MCR)

Table 22-3. DSPI*x*_MCR Field Descriptions

Bits	Name	Description
31	MSTR	Master/slave mode select. The MSTR bit configures the DSPI for either master mode or slave mode. 0 DSPI is in slave mode 1 DSPI is in master mode
30	CONT_SCKE	Continuous SCK enable. The CONT_SCKE bit enables the serial communication clock (SCK) to run continuously. See Section 22.6.6, “Continuous Serial Communications Clock,” for details. 0 Continuous SCK disabled 1 Continuous SCK enabled
29–28	DCONF[1:0]	DSPI configuration. The DCS field selects between the three different configurations of the DSPI. The table below lists the DCONF values for the various configurations. **DCONF** / **Configuration** 00 / SPI 01 / Reserved 10 / Reserved 11 / Reserved
27	FRZ	Freeze. The FRZ bit enables the DSPI transfers to be stopped on the next frame boundary when the device enters debug mode. 0 Do not halt serial transfers 1 Halt serial transfers
26	MTFE	Modified timing format enable. The MTFE bit enables a modified transfer format to be used. See Section 22.6.5.4, “Modified SPI Transfer Format (MTFE = 1, CPHA = 1),” for more information. 0 Modified SPI transfer format disabled 1 Modified SPI transfer format enabled

Table 22-3. DSPI*x*_MCR Field Descriptions (continued)

Bits	Name	Description
25	PCSSE	Peripheral chip select strobe enable. The PCSSE bit enables the PCS5/$\overline{\text{PCSS}}$ to operate as a PCS Strobe output signal. See Section 22.6.4.5, “Peripheral Chip Select Strobe Enable (PCSS),” for more information. 0 PCS5/$\overline{\text{PCSS}}$ is used as the Peripheral Chip Select 5 signal 1 PCS5/$\overline{\text{PCSS}}$ is used as an active-low PCS Strobe signal
24	ROOE	Receive FIFO overflow overwrite enable. The ROOE bit enables an RX FIFO overflow condition to either ignore the incoming serial data or to overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer that generated the overflow is either ignored or shifted into the shift register. If the ROOE bit is set, the incoming data is shifted into the shift register. If the ROOE bit is cleared, the incoming data is ignored. See Section 22.6.7.6, “Receive FIFO Overflow Interrupt Request,” for more information. 0 Incoming data is ignored 1 Incoming data is shifted into the shift register
23–16 [1]	PCSIS*n*	Peripheral chip select 0 – 7 inactive state. The PCSIS*n* bit determines the inactive state of the PCS*n* signal. 0 The inactive state of PCS*n* is low 1 The inactive state of PCS*n* is high
15	DOZE	Doze enable. The DOZE bit provides support for externally controlled doze mode power-saving mechanism. See Section 22.6.1, “DSPI Operating Mode Details,” and Section 7.3, “Power Consumption Considerations,” for details.
14	MDIS	Module disable. The MDIS bit stops the clock to the non-memory mapped logic in the DSPI effectively putting the DSPI in a software controlled power-saving state. See Section 22.6.1, “DSPI Operating Mode Details,” and Section 7.3, “Power Consumption Considerations,” for more information. 0 Enable DSPI clocks. 1 Disable DSPI clocks.
13	DIS_TXF	Disable transmit FIFO. The DIS_TXF bit provides a mechanism to disable the TX FIFO. When the TX FIFO is disabled, the transmit part of the DSPI operates as a simplified double-buffered SPI. See Section 22.6.3.3, “FIFO Disable Operation,” for details. 0 TX FIFO is enabled 1 TX FIFO is disabled
12	DIS_RXF	Disable receive FIFO. The DIS_RXF bit provides a mechanism to disable the RX FIFO. When the RX FIFO is disabled, the receive part of the DSPI operates as a simplified double-buffered SPI. See Section 22.6.3.3, “FIFO Disable Operation,” for details. 0 RX FIFO is enabled 1 RX FIFO is disabled
11	CLR_TXF	Clear TX FIFO. CLR_TXF is used to flush the TX FIFO. Writing a ‘1’ to CLR_TXF clears the TX FIFO Counter. The CLR_TXF bit is always read as zero. 0 Do not clear the TX FIFO Counter 1 Clear the TX FIFO Counter
10	CLR_RXF	Clear RX FIFO. CLR_RXF is used to flush the RX FIFO. Writing a ‘1’ to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. 0 Do not clear the RX FIFO Counter 1 Clear the RX FIFO Counter

Table 22-3. DSPI*x*_MCR Field Descriptions (continued)

Bits	Name	Description
9–8	SMPL_PT[1:0]	Sample Point. SMPL_PT allows the host software to select when the DSPI Master samples SIN in modified transfer format. Figure 22-18 shows where the master can sample the SIN pin. The table below lists the various delayed sample points. SMPL_PT: Number of peripheral bus clock cycles between odd-numbered edge of SCK and sampling of SIN. 00: 0 01: 1 10: 2 11: Reserved
7–1	—	Reserved.
0	HALT	Halt. HALT provides a mechanism for software to start and stop DSPI transfers. See Section 22.6.2, “Start and Stop of DSPI Transfers,” for details on the operation of this bit. 0 Start transfers 1 Stop transfers

1 PCSIS[7:6, 4:3] are available only on MAC7136 devices. On all other devices, bits 19, 20, 22 and 23 are reserved.

22.5.1.2 DSPI Transfer Count Register (DSPI*x*_TCR)

The DSPI*x*_TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management.

NOTE

The user must not write to the DSPI*x*_TCR while the DSPI is in the Running state.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SPI_TCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	DSPI *x* Base + 0x0008															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	DSPI *x* Base + 0x0008															

Figure 22-4. DSPI Transfer Count Register (DSPI*x*_TCR)

Table 22-4. DSPI*x*_TCR Field Descriptions

Bits	Name	Description
31–16	SPI_TCNT [15:0]	SPI Transfer Counter. SPI_TCNT is used to keep track of the number of SPI transfers made. The SPI_TCNT field counts the number of SPI transfers the DSPI makes. The SPI_TCNT field is incremented every time the last bit of a SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the SPI command being executed. The Transfer Counter 'wraps around' i.e. incrementing the counter past 65535 resets the counter to zero.
15–0	—	Reserved.

22.5.1.3 DSPI Clock and Transfer Attributes Registers (DSPI*x*_CTAR*n*)

The DSPI*x*_CTAR*n* registers are used to define different transfer attributes. The user must not write to the DSPI*x*_CTAR*n* registers while the DSPI is in the running state.

In master mode, the DSPI*x*_CTAR*n* registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. When the DSPI is configured as a SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the DSPI*x*_CTARs is used.

In slave mode, a subset of the bitfields in the DSPI*x*_CTAR0 register is used to set the slave transfer attributes. See the individual bit descriptions for details on which bits are used in slave modes.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DBR[1]	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	DSPI *x* Base + 0x00C + (4 × *n*)															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CSSCK				ASC				DT				BR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	DSPI *x* Base + 0x00C + (4 × *n*)															

1 Reserved on mask sets L49P and L47W devices.

Figure 22-5. DSPI Clock and Transfer Attributes Registers (DSPI*x*_CTAR*n*)

Table 22-5. DSPI*x*_CTAR*n* Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>31</td><td>DBR [1]</td><td>Double baud rate. DBR doubles the effective baud rate of SCK. Only used in Master Mode. Halves the Baud Rate division ratio for faster frequencies and odd division ratios for SCK. When set, the duty cycle of SCK depends on the values of the PBR and CPHA fields as shown in the table below. Refer to Section 22.6.4, “DSPI Baud Rate and Clock Delay Generation.” for more details. If the overall baud rate is divide by two or three of the peripheral bus clock, then the DSPIx_MCR[CONT_SCKE, MTFE] bits should be clear.
1 The baud rate is doubled, duty cycle is set by PBR
0 The baud rate is computed normally with a 50/50 duty cycle
<table>
<tr><th>DBR</th><th>CPHA</th><th>PBR</th><th>SCK Duty Cycle</th></tr>
<tr><td>0</td><td>x</td><td>xx</td><td>50 / 50</td></tr>
<tr><td>1</td><td>0</td><td>00</td><td>50 / 50</td></tr>
<tr><td>1</td><td>0</td><td>01</td><td>33 / 66</td></tr>
<tr><td>1</td><td>0</td><td>10</td><td>40 / 60</td></tr>
<tr><td>1</td><td>0</td><td>11</td><td>43 / 57</td></tr>
<tr><td>1</td><td>1</td><td>00</td><td>50 / 50</td></tr>
<tr><td>1</td><td>1</td><td>01</td><td>66 / 33</td></tr>
<tr><td>1</td><td>1</td><td>10</td><td>60 / 40</td></tr>
<tr><td>1</td><td>1</td><td>11</td><td>57 / 43</td></tr>
</table>
</td></tr>
<tr><td>30–27</td><td>FMSZ[3:0]</td><td>Frame size. The FMSZ field selects the number of bits transferred per frame. The FMSZ field is used in master mode and slave mode. The table below lists the frame sizes.
<table>
<tr><th>FMSZ</th><th>Framesize</th></tr>
<tr><td>0000</td><td>Reserved</td></tr>
<tr><td>0001</td><td>Reserved</td></tr>
<tr><td>0010</td><td>Reserved</td></tr>
<tr><td>0011</td><td>4</td></tr>
<tr><td>0100</td><td>5</td></tr>
<tr><td>0101</td><td>6</td></tr>
<tr><td>0110</td><td>7</td></tr>
<tr><td>0111</td><td>8</td></tr>
</table>
<table>
<tr><th>FMSZ</th><th>Framesize</th></tr>
<tr><td>1000</td><td>9</td></tr>
<tr><td>1001</td><td>10</td></tr>
<tr><td>1010</td><td>11</td></tr>
<tr><td>1011</td><td>12</td></tr>
<tr><td>1100</td><td>13</td></tr>
<tr><td>1101</td><td>14</td></tr>
<tr><td>1110</td><td>15</td></tr>
<tr><td>1111</td><td>16</td></tr>
</table>
</td></tr>
<tr><td>26</td><td>CPOL</td><td>Clock polarity. The CPOL bit selects the inactive state of the serial communications clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities.
For more information on continuous selection format, refer to Section 22.6.5.5, “Continuous Selection Format.”
1 The inactive state value of SCK is high
0 The inactive state value of SCK is low
Note: When the continuous selection format is selected (CONT = 1), switching between clock polarities without stopping the DSPI can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge.</td></tr>
<tr><td>25</td><td>CPHA</td><td>Clock phase. The CPHA bit selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings.
1 Data is changed on the leading edge of SCK and captured on the following edge
0 Data is captured on the leading edge of SCK and changed on the following edge</td></tr>
</table>

Table 22-5. DSPIx_CTAR*n* Field Descriptions *(continued)*

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>24</td><td>LSBFE</td><td>LSB first enable. The LSBFE bit selects if the LSB or MSB of the frame is transferred first. This bit is only used in master mode.
1 Data is transferred LSB first
0 Data is transferred MSB first</td></tr>
<tr><td>23–22</td><td>PCSSCK[1:0]</td><td>PCS to SCK delay prescaler. The PCSSCK field selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. This field is only used in master mode. The table below lists the prescaler values. The description for bit field CSSCK in Table 22-5 details how to compute the PCS to SCK delay. Also see Section 22.6.4.2, “PCS to SCK Delay (tCSC).”
<table>
<tr><th>PCSSCK</th><th>PCS to SCK Delay Prescaler Value</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
<tr><td>21–20</td><td>PASC[1:0]</td><td>After SCK delay prescaler. The PASC field selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. This field is only used in master mode. The table below lists the prescaler values. The description for bit field ASC in Table 22-5 details how to compute the After SCK delay. Also see Section 22.6.4.3, “After SCK Delay (tASC).”
<table>
<tr><th>PASC</th><th>After SCK Delay Prescaler Value</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
<tr><td>19–18</td><td>PDT[1:0]</td><td>Delay after transfer prescaler. The PDT field selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. The table below lists the prescaler values. The description for bit field DT in Table 22-5 details how to compute the delay after transfer. Also see Section 22.6.4.4, “Delay after Transfer (tDT).”
<table>
<tr><th>PDT</th><th>Delay after Transfer Prescaler Value</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
</table>

Table 22-5. DSPI*x*_CTAR*n* Field Descriptions *(continued)*

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>17–16</td><td>PBR[1:0]</td><td>Baud rate prescaler. The PBR field selects the prescaler value for the baud rate. This field is only used in master mode. The baud rate is the frequency of the serial communications clock (SCK). The peripheral bus clock is divided by this prescaler value before the baud rate selection takes place. The baud rate prescaler values are listed in the table below. The description for the BR bit field in Table 22-5 details how to compute the baud rate. Also see Section 22.6.4, “DSPI Baud Rate and Clock Delay Generation.”
<table>
<tr><th>PBR</th><th>Baud Rate Prescaler Value</th></tr>
<tr><td>00</td><td>2</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
<tr><td>15–12</td><td>CSSCK[3:0]</td><td>PCS to SCK delay scaler. The CSSCK field selects the scaler value for the PCS to SCK delay. This field is only used in master mode. The PCS to SCK delay is the delay between the assertion of PCS and the first edge of the SCK. The table below lists the scaler values. See Section 22.6.4.2, “PCS to SCK Delay (tCSC),” for more details.
<table>
<tr><th>CSSCK</th><th>PCS to SCK Delay Scaler Value</th><th>CSSCK</th><th>PCS to SCK Delay Scaler Value</th></tr>
<tr><td>0000</td><td>2</td><td>1000</td><td>512</td></tr>
<tr><td>0001</td><td>4</td><td>1001</td><td>1024</td></tr>
<tr><td>0010</td><td>8</td><td>1010</td><td>2048</td></tr>
<tr><td>0011</td><td>16</td><td>1011</td><td>4096</td></tr>
<tr><td>0100</td><td>32</td><td>1100</td><td>8192</td></tr>
<tr><td>0101</td><td>64</td><td>1101</td><td>16384</td></tr>
<tr><td>0110</td><td>128</td><td>1110</td><td>32768</td></tr>
<tr><td>0111</td><td>256</td><td>1111</td><td>65536</td></tr>
</table>
The PCS to SCK delay is a multiple of the peripheral bus clock period and it is computed according to the following equation:
$$t_{CSC} = \frac{1}{f_{IPS}} \times PCSSCK \times CSSCK$$ ***Eqn. 22-1***</td></tr>
</table>

Table 22-5. DSPIx_CTAR*n* Field Descriptions *(continued)*

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>11–8</td><td>ASC[3:0]</td><td>After SCK delay scaler. The ASC field selects the scaler value for the After SCK delay. This field is only used in master mode. The After SCK delay is the delay between the last edge of SCK and the negation of PCS. The table below lists the scaler values. See Section 22.6.4.3, “After SCK Delay (tASC),” for more details.
<table>
<tr><th>ASC</th><th>After SCK Delay Scaler Value</th><th>ASC</th><th>After SCK Delay Scaler Value</th></tr>
<tr><td>0000</td><td>2</td><td>1000</td><td>512</td></tr>
<tr><td>0001</td><td>4</td><td>1001</td><td>1024</td></tr>
<tr><td>0010</td><td>8</td><td>1010</td><td>2048</td></tr>
<tr><td>0011</td><td>16</td><td>1011</td><td>4096</td></tr>
<tr><td>0100</td><td>32</td><td>1100</td><td>8192</td></tr>
<tr><td>0101</td><td>64</td><td>1101</td><td>16384</td></tr>
<tr><td>0110</td><td>128</td><td>1110</td><td>32768</td></tr>
<tr><td>0111</td><td>256</td><td>1111</td><td>65536</td></tr>
</table>
The After SCK delay is a multiple of the peripheral bus clock period, and it is computed according to the following equation:
$$t_{ASC} = \frac{1}{f_{IPS}} \times PASC \times ASC$$ ***Eqn. 22-2***</td></tr>
<tr><td>7–4</td><td>DT[3:0]</td><td>Delay after transfer scaler. The DT field selects the delay after transfer scaler. This field is only used in master mode. The delay after transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The table below lists the scaler values. See Section 22.6.4.4, “Delay after Transfer (tDT).”
<table>
<tr><th>DT</th><th>Delay after Transfer Scaler Value</th><th>DT</th><th>Delay after Transfer Scaler Value</th></tr>
<tr><td>0000</td><td>2</td><td>1000</td><td>512</td></tr>
<tr><td>0001</td><td>4</td><td>1001</td><td>1024</td></tr>
<tr><td>0010</td><td>8</td><td>1010</td><td>2048</td></tr>
<tr><td>0011</td><td>16</td><td>1011</td><td>4096</td></tr>
<tr><td>0100</td><td>32</td><td>1100</td><td>8192</td></tr>
<tr><td>0101</td><td>64</td><td>1101</td><td>16384</td></tr>
<tr><td>0110</td><td>128</td><td>1110</td><td>32768</td></tr>
<tr><td>0111</td><td>256</td><td>1111</td><td>65536</td></tr>
</table>
The delay after transfer is a multiple of the peripheral bus clock period and it is computed according to the following equation:
$$t_{DT} = \frac{1}{f_{IPS}} \times PDT \times DT$$ ***Eqn. 22-3***</td></tr>
</table>

Table 22-5. DSPI*x*_CTAR*n* Field Descriptions *(continued)*

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>3–0</td><td>BR[3:0]</td><td>Baud rate scaler. The BR field selects the scaler value for the baud rate. This field is only used in master mode. The pre-scaled peripheral bus clock is divided by the baud rate scaler to generate the frequency of the SCK. The table below lists the baud rate scaler values. See Section 22.6.4.1, “Baud Rate Generator,” for more details.
<table>
<tr><th>BR</th><th>Baud Rate Scaler Value</th><th>BR</th><th>Baud Rate Scaler Value</th></tr>
<tr><td>0000</td><td>2</td><td>1000</td><td>256</td></tr>
<tr><td>0001</td><td>4</td><td>1001</td><td>512</td></tr>
<tr><td>0010</td><td>6</td><td>1010</td><td>1024</td></tr>
<tr><td>0011</td><td>8</td><td>1011</td><td>2048</td></tr>
<tr><td>0100</td><td>16</td><td>1100</td><td>4096</td></tr>
<tr><td>0101</td><td>32</td><td>1101</td><td>8192</td></tr>
<tr><td>0110</td><td>64</td><td>1110</td><td>16384</td></tr>
<tr><td>0111</td><td>128</td><td>1111</td><td>32768</td></tr>
</table>
The baud rate is computed according to the following equation: [1]
$$\text{SCK baud rate} = \frac{f_{IPS}}{PBR} \times \frac{1 + DBR}{BR} \qquad \textit{Eqn. 22-4}$$</td></tr>
</table>

[1] Mask sets L49P and L47W devices do not implement the double-baud-rate feature; bit 31 is reserved and should be cleared, and DBR = 0 is used for baud rate equations.

22.5.1.4 DSPI Status Register (DSPI*x*_SR)

The DSPI*x*_SR contains status and flag bits. The bits reflect the status of the DSPI and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the DSPI*x*_SR by writing a ‘1’ to that bit. Writing a ‘0’ to a flag bit has no effect. This register is not be writable in module disable or stop mode due to the use of power saving mechanisms (see Section 7.3, “Power Consumption Considerations”).

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TCF	TXRXS	0	EOQF	TFUF	0	TFFF	0	0	0	0	0	RFOF	0	RFDF	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	DSPI *x* Base + 0x002C															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Addr	DSPI *x* Base + 0x002C															

Figure 22-6. DSPI Status Register (DSPI*x*_SR)

Table 22-6. DSPI*x*_SR Field Descriptions

Bits	Name	Description
31	TCF	Transfer complete flag. The TCF bit indicates that all bits in a frame have been shifted out. The TCF bit is set at the end of the frame transfer. The TCF bit remains set until cleared by software. 1 Transfer complete 0 Transfer not complete
30	TXRXS	TX & RX status. The TXRXS bit reflects the status of the DSPI. See Section 22.6.2, "Start and Stop of DSPI Transfers," for information on what causes this bit to be negated or asserted. 1 TX and RX operations are enabled (DSPI is in RUNNING state) 0 TX and RX operations are disabled (DSPI is in STOPPED state)
29	—	Reserved.
28	EOQF	End of queue flag. The EOQF bit indicates that the transmission in progress is the last entry in a queue. The EOQF bit is set when TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. It remains set until cleared by software. When the EOQF bit is set, the TXRXS bit is automatically cleared. End of queue is only detected in master mode. 1 EOQ bit is set in the executing SPI command 0 EOQ bit is not set in the executing SPI command
27	TFUF	Transmit FIFO underflow flag. The TFUF bit indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for DSPI modules operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI operating in SPI slave mode is empty and a transfer is initiated by an external SPI master. The TFUF bit remains set until cleared by software. 1 TX FIFO underflow has occurred 0 TX FIFO underflow has not occurred
26	—	Reserved.
25	TFFF	Transmit FIFO fill flag. The TFFF bit provides a method for the DSPI to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by host software or an acknowledgement from the eDMA controller when the TX FIFO is full. 1 TX FIFO is not full 0 TX FIFO is full
24–20	—	Reserved.
19	RFOF	Receive FIFO overflow flag. The RFOF bit indicates that an overflow condition in the RX FIFO has occurred. The bit is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until cleared by software. 1 RX FIFO overflow has occurred 0 RX FIFO overflow has not occurred
18	—	Reserved.
17	RFDF	Receive FIFO drain flag. The RFDF bit provides a method for the DSPI to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by host software or an acknowledgement from the eDMA controller when the RX FIFO is empty. 1 RX FIFO is not empty 0 RX FIFO is empty
16	—	Reserved.

Table 22-6. DSPI*x*_SR Field Descriptions (continued)

Bits	Name	Description
15–12	TXCTR [3:0]	TX FIFO counter. The TXCTR field indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the DSPI _PUSHR is written. The TXCTR is decremented every time a SPI command is executed and the SPI data is transferred to the shift register.
11–8	TXNXTPTR [3:0]	Transmit next pointer. The TXNXTPTR field indicates which TX FIFO Entry will be transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register. See Section 22.6.3.4, “Transmit First In First Out (TX FIFO) Buffering Mechanism,” for more details.
7–4	RXCTR [3:0]	RX FIFO counter. The RXCTR field indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the DSPI _POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
3–0	POPNXTPT [3:0]	Pop next pointer. The POPNXTPTR field contains a pointer to the RX FIFO entry that will be returned when the DSPI*x*_POPR is read. The POPNXTPTR is updated when the DSPI*x*_POPR is read. See Section 22.6.3.5, “Receive First In First Out (RX FIFO) Buffering Mechanism,” for more details.

22.5.1.5 DSPI DMA/Interrupt Request Select / Enable Register (DSPI*x*_RSER)

The DSPI*x*_RSER serves two purposes. First, it enables flag bits in the DSPI*x*_SR to generate DMA requests or interrupt requests. The DSPI*x*_RSER also selects the type of request to be generated. See the individual bit descriptions for information on the types of requests the bits support.

NOTE

The user must not write to the DSPIx_RSER while the DSPI is in the Running state.

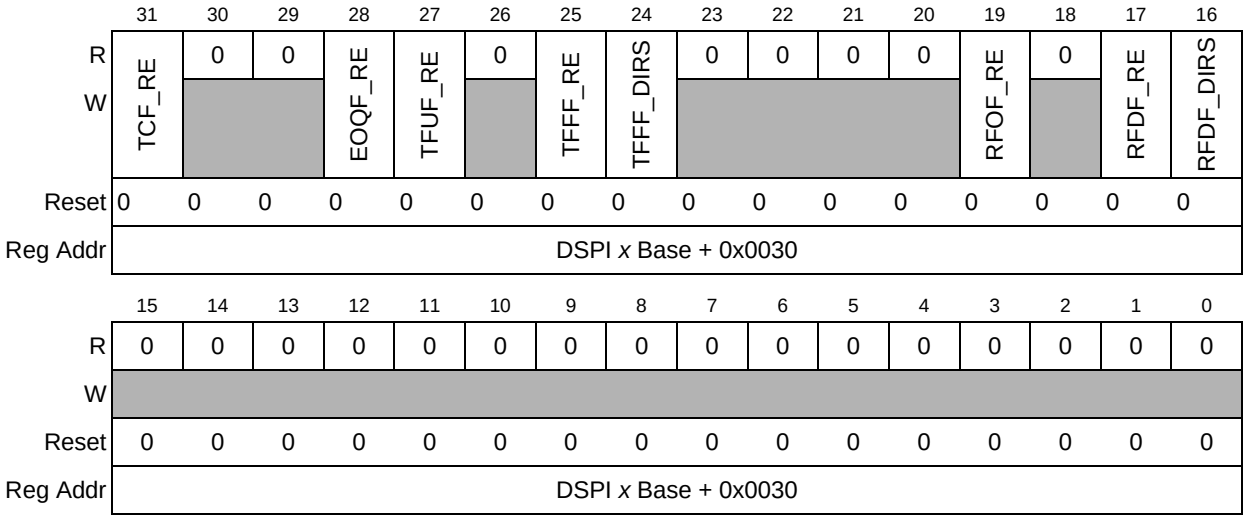


Figure 22-7. DSPI DMA/Interrupt Request Select and Enable Register (DSPI*x*_RSER)

Table 22-7. DSPI*x*_RSER Field Descriptions

Bits	Name	Description
31	TCF_RE	Transmission complete request enable. The TCF_RE bit enables TCF flag in the DSPI*x*_SR to generate an interrupt request. 1 TCF interrupt requests are enabled 0 TCF interrupt requests are disabled
30–29	—	Reserved.
28	EOQF_RE	DSPI finished request enable. The EOQF_RE bit enables the EOQF flag in the DSPI*x*_SR to generate an interrupt request. 1 EOQF interrupt requests are enabled 0 EOQF interrupt requests are disabled
27	TFUF_RE	Transmit FIFO underflow request enable.The TFUF_RE bit enables the TFUF flag in the DSPI*x*_SR to generate an interrupt request. 1 TFUF interrupt requests are enabled 0 TFUF interrupt requests are disabled
26	—	Reserved.
25	TFFF_RE	Transmit FIFO fill request enable. The TFFF_RE bit enables the TFFF flag in the DSPI*x*_SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA requests. 1 TFFF interrupt requests or DMA requests are enabled 0 TFFF interrupt requests or DMA requests are disabled
24	TFFF_DIRS	Transmit FIFO fill DMA or interrupt request select. The TFFF_DIRS bit selects between generating a DMA request or an interrupt request. When the TFFF flag bit in the DSPI*x*_SR is set, and the TFFF_RE bit in the DSPI*x*_RSER is set, this bit selects between generating an interrupt request or a DMA request. 1 DMA request will be generated 0 Interrupt request will be generated
23–20	—	Reserved.
19	RFOF_RE	Receive FIFO overflow request enable. The RFOF_RE bit enables the RFOF flag in the DSPI*x*_SR to generate an interrupt request. 1 RFOF interrupt requests are enabled 0 RFOF interrupt requests are disabled
18	—	Reserved.
17	RFDF_RE	Receive FIFO drain request enable. The RFDF_RE bit enables the RFDF flag in the DSPI*x*_SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 1 RFDF interrupt requests or DMA requests are enabled 0 RFDF interrupt requests or DMA requests are disabled
16	RFDF_DIRS	Receive FIFO drain DMA or interrupt request select. The RFDF_DIRS bit selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the DSPI*x*_SR is set, and the RFDF_RE bit in the DSPI*x*_RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 1 DMA request will be generated 0 Interrupt request will be generated
15–0	—	Reserved.

22.5.1.6 DSPI PUSH TX FIFO Register (DSPI*x*_PUSHR)

The DSPI*x*_PUSHR provides a means to write to the TX FIFO. Data written to this register is transferred to the TX FIFO. See Section 22.6.3.4, “Transmit First In First Out (TX FIFO) Buffering Mechanism,” for more information. Eight or sixteen bit write accesses to the DSPI*x*_PUSHR will transfer 32 bits to the TX FIFO.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CONT	CTAS			EOQ	CTCNT	0	0	PCS7 [1]	PCS6 [1]	PCS5	PCS4 [1]	PCS3 [1]	PCS2	PCS1	PCS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	DSPI *x* Base + 0x0034															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	DSPI *x* Base + 0x0034															

[1] PCS[7:6, 4:3] are available on MAC7136 only. On all other devices, bits 19, 20, 22 and 23 are reserved.

Figure 22-8. DSPI PUSH TX FIFO Register (DSPI*x*_PUSHR)

Table 22-8. DSPI*x*_PUSHR Field Descriptions

Bits	Name	Description
31	CONT	Continuous peripheral chip select enable.The CONT bit selects a continuous selection format. The bit is used in SPI master mode. The bit enables the selected PCS signals to remain asserted between transfers. See Section 22.6.5.5, “Continuous Selection Format,” for more information. 1 Keep Peripheral Chip Select signals asserted between transfers 0 Return Peripheral Chip Select signals to their inactive state between transfers
30–28	CTAS[2:0]	Clock and transfer attributes select. The CTAS field selects which of the DSPI*x*_CTARs is used to set the transfer attributes for the associated SPI frame. The field is only used in SPI master mode. In SPI slave mode DSPI*x*_CTAR0 is used. The table below shows how the CTAS values map to the DSPI*x*_CTARs.

CTAS	Use Clock and Transfer Attributes from
000	DSPI*x*_CTAR0
001	DSPI*x*_CTAR1
010	DSPI*x*_CTAR2 [1]
011	DSPI*x*_CTAR3 [1]
100	DSPI*x*_CTAR4 [1]
101	DSPI*x*_CTAR5 [1]
110	Reserved
111	Reserved

[1] Reserved on mask set L49P devices.

Table 22-8. DSPI*x*_PUSHR Field Descriptions (continued)

Bits	Name	Description
27	EOQ	End Of queue. The EOQ bit provides a means for host software to signal to the DSPI that the current SPI transfer is the last in a queue. At the end of the transfer the EOQF bit in the DSPI*x*_SR is set. 1 The SPI data is the last data to transfer 0 The SPI data is not the last data to transfer
26	CTCNT	Clear SPI_TCNT. The CTCNT provides a means for host software to clear the SPI transfer counter. The CTCNT bit clears the SPI_TCNT field in the DSPI*x*_TCR. The SPI_TCNT field is cleared before transmission of the current SPI frame begins. 1 Clear SPI_TCNT field in the DSPI*x*_TCR 0 Do not clear SPI_TCNT field in the DSPI*x*_TCR
23–16 [1]	PCS*n*	Peripheral chip select 0–7. The PCS bits select which PCS*n* signals will be asserted for the transfer. 1 Assert the PCS*n* signal 0 Negate the PCS*n* signal
15–0	TXDATA[15:0]	Transmit data. The TXDATA field holds SPI data to be transferred according to the associated SPI command.

1 PCS[7:6, 4:3] are available only on MAC7136 devices. On all other devices, bits 19, 20, 22 and 23 are reserved.

22.5.1.7 DSPI POP RX FIFO Register (DSPI*x*_POPR)

The DSPI*x*_POPR provides a means to read the RX FIFO. See Section 22.6.3.5, “Receive First In First Out (RX FIFO) Buffering Mechanism,” for a description of the RX FIFO operations. Eight or sixteen bit read accesses to the DSPI*x*_POPR will read from the RX FIFO and update the counter and pointer.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	DSPI *x* Base + 0x0038															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	DSPI *x* Base + 0x0038															

Figure 22-9. DSPI POP RX FIFO Register (DSPI*x*_POPR)

Table 22-9. DSPI*x*_POPR Field Descriptions

Bits	Name	Description
31–16	—	Reserved. Read-only.
15–0	RXDATA[15:0]	Received data. The RXDATA field contains the SPI data from the RX FIFO entry pointed to by the pop next data pointer.

22.5.1.8 DSPI Transmit FIFO Registers (DSPI*x*_TXFR*n*)

The DSPI*x*_TXFR*n* through DSPIx_TXFR3 registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the DSPI*x*_TXFR*n* registers does not alter the state of the TX FIFO.

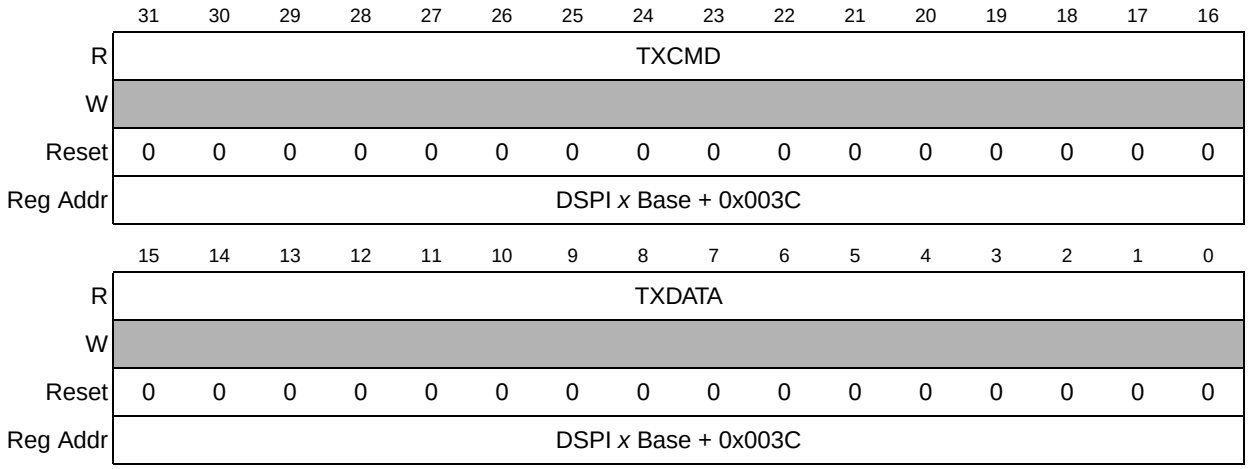


Figure 22-10. DSPI Transmit FIFO Register (DSPI*x*_TXFR0)

Table 22-10. DSPI*x*_TXFR0 Field Descriptions

Bits	Name	Description
31–16	TXCMD[15:0]	Transmit command. The TXCMD field contains the command that sets the transfer attributes for the SPI data. See Section 22.5.1.6, “DSPI PUSH TX FIFO Register (DSPIx_PUSHR),” for details on the command field.
15–0	TXDATA[15:0]	Transmit data. The TXDATA field contains the SPI data to be shifted out.

22.5.1.9 DSPI Receive FIFO Registers (DSPI*x*_RXFR*n*)

The DSPI*x*_RXFR0 through DSPI*x*_RXFR3 registers provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The DSPI*x*_RXFRs are read-only. Reading the DSPI*x*_RXFR*n* registers does not alter the state of the RX FIFO.

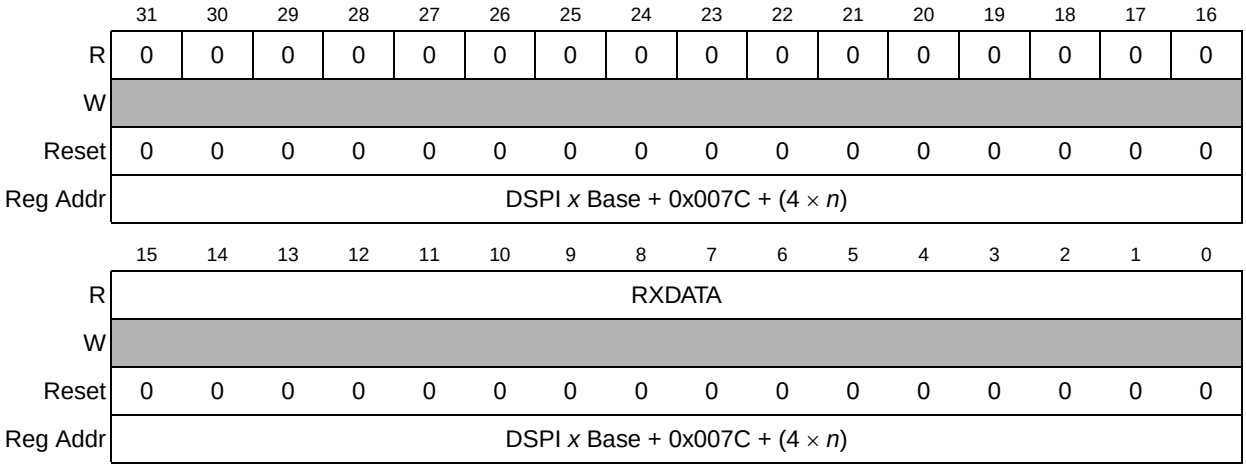


Figure 22-11. DSPI Receive FIFO Registers (DSPI*x*_RXFR*n*)

Table 22-11. DSPI*x*_RXFR*n* Field Description

Bits	Name	Description
31–16	—	Reserved.
15–0	RXDATA[15:0]	Received Data. The RXDATA field contains the received SPI data.

22.6 Functional Description

The Deserial Serial Peripheral Interface (DSPI) module supports full-duplex, synchronous serial communications between devices and peripheral devices.

The DCONF field in the Section 22.5.1.1, “DSPI Module Configuration Register (DSPIx_MCR),” determines the DSPI Configuration. See Table 22-3 for the DSPI configuration values.

The DSPI*x*_CTAR0 and DSPI*x*_CTAR1 registers hold clock and transfer attributes. The SPI configuration can select which CTAR*n* to use on a frame by frame basis by setting a field in the SPI command. See Section 22.5.1.3, “DSPI Clock and Transfer Attributes Registers (DSPIx_CTARn),” for information on the fields of the DSPI*x*_CTARs.

The 16-bit shift register in the master and the 16-bit shift register in the slave are linked by the SOUT and SIN signals to form a distributed 32-bit register. When a data transfer operation is performed, data is serially shifted a pre-determined number of bit positions. Because the registers are linked, data is exchanged between the master and the slave; the data that was in the master’s shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the transfer complete flag (DSPI*x*_SR[TCF]) is set to indicate a completed transfer. Figure 22-12 illustrates how master and slave data is exchanged.

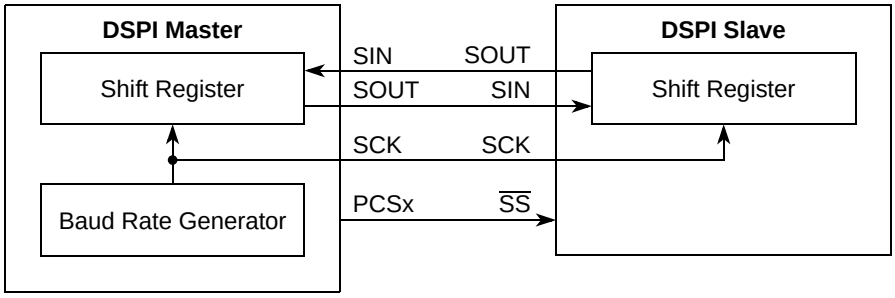


Figure 22-12. SPI Serial Protocol Overview

The DSPI has four peripheral chip select (PCS) signals that are used to select which of the slaves to communicate with. The DSPI transfer protocol and timing properties are described in Section 22.6.5, "Transfer Formats." The transfer rate and delay settings are described in Section 22.6.4, "DSPI Baud Rate and Clock Delay Generation."

22.6.1 DSPI Operating Mode Details

The DSPI has five distinct modes:

- Master Mode
- Slave Mode
- Disabled Mode
- Stop Mode
- Debug Mode

Master, slave, and module disable modes are module-specific mode while external stop and debug modes are device-specific modes.

The module-specific modes are determined by bits in the DSPI*x*_MCR set or cleared by host software. External stop mode and debug mode are modes that are controlled by signals external to the DSPI. These device-specific modes are modes that the entire device can enter, in parallel with the DSPI being configured for one of its module-specific modes.

22.6.1.1 DSPI Master Mode

Master mode allows the DSPI to initiate and control serial communication with peripheral devices. In this mode, the serial communication clock (SCK) signal and the peripheral chip select (PCS*x*) signals are controlled by the DSPI and configured as outputs. It operates as bus master when the DSPI*x*_MCR[MSTR] bit is set. The serial communications clock (SCK) is controlled by the master DSPI.

Master mode transfer attributes are controlled by the SPI command in the current TX FIFO entry. The CTAS field in the SPI command selects which DSPI*x*_CTAR*n* will be used to set the transfer attributes. Transfer attribute control is on a frame by frame basis. See Section 22.6.3, "Serial Peripheral Interface (SPI) Configuration," for more details.

22.6.1.2 DSPI Slave Mode

In slave mode the DSPI responds to transfers initiated by a SPI master. In this mode the DSPI cannot control serial transfers, but rather responds to externally controlled serial transfers. The DSPI operates as bus slave when the DSPI*x*_MCR[MSTR] bit is cleared. In this mode, the SCK signal and the PCS0/$\overline{SS}$ signal are configured as inputs and provided by a bus master. All transfer attributes are controlled by the bus master but clock polarity, clock phase, and numbers of bits to transfer must still be configured in the DSPI slave for proper communications.

In SPI slave mode the slave transfer attributes are set in the DSPI*x*_CTAR0. The DSPI in slave mode transfers data MSB first. The LSBFE field of the associated CTAR is ignored.

22.6.1.3 DSPI Disabled Mode

DSPI disabled mode is a module-specific mode that the DSPI can enter to save power. The DSPI enters the disabled mode when the DSPI*x*_MCR[MDIS] bit is set or when a request for the DSPI to enter doze mode is asserted by the CRG while the DSPI*x*_MCR[DOZE] bit is set.

The clock to the non-memory mapped logic in the DSPI is stopped while in disabled mode. Certain read or write operations have different effects when the DSPI is in disabled mode. Reading the RX FIFO Pop Register will not change the state of the RX FIFO. Likewise, writing to the TX FIFO Push Register will not change the state of the TX FIFO. Clearing either of the FIFOs will not have any affect. Changes to the DIS_TXF and DIS_RXF fields of the DSPI*x*_MCR will not have any affect. All status bits and register flags in the DSPI will return the correct values when read, but writing to them will have no affect. Writing to the DSPI*x*_TCR during disabled mode will not have any affect. Interrupt and DMA request signals cannot be cleared.

22.6.1.4 DSPI Stop Mode

The stop mode is used for device power management. When a request is made to enter stop mode, the DSPI acknowledges the request. When the DSPI is ready to have clocks stopped, an acknowledge signal is asserted. If a serial transfer is in progress, the DSPI waits until it reaches the frame boundary before it asserts an acknowledge. While the clocks are stopped, the DSPI memory-mapped logic is not accessible. The states of the interrupt and DMA request signals cannot be changed while in external stop mode.

The DSPI will not acknowledge the request to enter external stop mode until it has reached a frame boundary. When the DSPI has reached a frame boundary it halts all operations and indicates that it is ready to have clocks stopped. The DSPI exits external stop mode and resumes normal operation once the clocks are turned on. Serial communications or register accesses made while in external stop mode are ignored even if the clocks have not been stopped yet.

22.6.1.5 DSPI Debug Mode

The debug mode is used for system development and debugging. If the device enters debug mode while the DSPI*x*_MCR[FRZ] bit is set, the DSPI stops all serial transfers and enters a stopped state. If the device enters debug mode while the FRZ bit is cleared, the DSPI behavior is unaffected and remains dictated by the module-specific mode and configuration of the DSPI. The DSPI enters debug mode when a debug

request is asserted by Nexus or EICE (see Appendix A, “Debug Interface”). See Figure 22-13 for a state diagram.

22.6.2 Start and Stop of DSPI Transfers

The DSPI has two operating states: STOPPED and RUNNING. The default state of the DSPI is STOPPED. In the STOPPED state, no serial transfers are initiated in master mode and no transfers are responded to in slave mode. The STOPPED state is also a safe state for writing the various configuration registers of the DSPI without causing undetermined results. The TXRXS bit in the DSPI*x*_SR is cleared in this state. In the RUNNING state, serial transfers take place. The TXRXS bit in the DSPI*x*_SR is set in the RUNNING state. Figure 22-13 shows a state diagram of the start and stop mechanism. The transitions are described in Table 22-12.

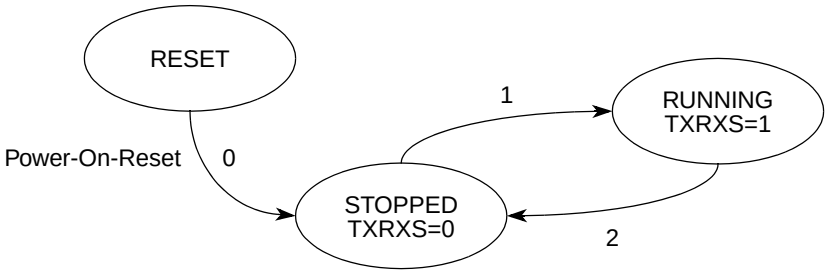


Figure 22-13. DSPI Start and Stop State Diagram

Table 22-12. State Transitions for Start and Stop of DSPI Transfers

Transition #	Current State	Next State	Description
0	RESET	STOPPED	Generic power-on-reset transition
1	STOPPED	RUNNING	The DSPI is started (DSPI transitions to RUNNING) when all of the following conditions are true: • DSPI*x*_SR[EOQF] bit is clear • When device is not in debug mode or the DSPI*x*_MCR[FRZ] bit is clear • DSPI*x*_MCR[HALT] bit is clear
2	RUNNING	STOPPED	The DSPI stops (transitions from RUNNING to STOPPED) after the current frame for any one of the following conditions: • DSPI*x*_SR[EOQF] bit is set • When device is in debug mode and the DSPI*x*_MCR[FRZ] bit is set • DSPI*x*_MCR[HALT] bit is set

State transitions from RUNNING to STOPPED occur on the next frame boundary if a transfer is in progress, or on the next peripheral bus clock cycle if no transfers are in progress.

22.6.3 Serial Peripheral Interface (SPI) Configuration

The SPI Configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The DSPI is in SPI Configuration when the DCONF field in the DSPI*x*_MCR is 0b00. The SPI frames can be from four to sixteen bits long. The data to be transmitted can come from queues stored in RAM external to the DSPI. Host software or the eDMA controller can transfer the SPI data from

the queues to a First-In First-Out (FIFO) buffer. The received data is stored in entries in the Receive FIFO (RX FIFO) buffer. Host software or the eDMA controller transfer the received data from the RX FIFO to memory external to the DSPI. The FIFO buffer operations are described in Section 22.6.3.4, "Transmit First In First Out (TX FIFO) Buffering Mechanism," and Section 22.6.3.5, "Receive First In First Out (RX FIFO) Buffering Mechanism." The interrupt and DMA request conditions are described in Section 22.6.7, "Interrupts/DMA Requests."

22.6.3.1 Master Mode

In SPI master mode the DSPI initiates the serial transfers by controlling the serial communications clock (SCK) and the peripheral chip select (PCS) signals. The SPI command field in the executing TX FIFO entry determines which CTAR*n* will be used to set the transfer attributes and which PCS signal to assert. The command field also contains various bits that help with queue management and transfer protocol. See Section 22.5.1.6, "DSPI PUSH TX FIFO Register (DSPIx_PUSHR)," for details on the SPI command fields. The data field in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin. In SPI master mode, each SPI frame to be transmitted has a command associated with it allowing for transfer attribute control on a frame by frame basis.

22.6.3.2 Slave Mode

In SPI slave mode the DSPI responds to transfers initiated by a SPI bus master. The DSPI does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase and frame size must be set for successful communication with a SPI master. The SPI slave mode transfer attributes are set in the DSPI clock and transfer attributes register 0 (DSPI*x*_CTAR0).

22.6.3.3 FIFO Disable Operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO or RX FIFO. The DSPI operates as a double-buffered simplified SPI when the FIFOs are disabled. The TX and RX FIFOs are disabled separately. The TX FIFO is disabled by setting the DIS_TXF bit in the DSPI*x*_MCR. The RX FIFO is disabled by setting the DIS_RXF bit in the DSPI*x*_MCR.

The FIFO Disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the DSPI*x*_PUSHR and received data is read from the DSPI*x*_POPR. When the TX FIFO is disabled the TFFF, TFUF and TXCTR fields in DSPI*x*_SR behave as if there is a one-entry FIFO but the contents of the DSPI*x*_TXFRs and TXNXTPTR are undefined. When the RX FIFO is disabled the RFDF, RFOF and RXCTR fields in the DSPI*x*_SR behave as if there is a one-entry FIFO but the contents of the DSPI*x*_RXFRs and POPNXTPTR are undefined.

22.6.3.4 Transmit First In First Out (TX FIFO) Buffering Mechanism

The TX FIFO functions as a buffer of SPI data and SPI commands for transmission. The TX FIFO holds up to four entries consisting of a command field and a data field. SPI commands and data are added to the TX FIFO by writing to DSPI*x*_PUSHR (see Section 22.5.1.6, "DSPI PUSH TX FIFO Register (DSPIx_PUSHR)." TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the DSPI status register (DSPI*x*_SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time the DSPI _PUSHR is written or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates which TX FIFO entry will be transmitted during the next transfer. The TXNXTPTR contains the positive offset from DSPI*x*_TXFR0 in number of 32-bit registers. For example, TXNXTPTR = 0b0010 means that the DSPI*x*_TXFR2 contains the SPI data and command for the next transfer. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register.

22.6.3.4.1 Filling the TX FIFO

Host software or other intelligent modules can add (push) entries to the TX FIFO by writing to the DSPI*x*_PUSHR. When the TX FIFO is empty, the TX FIFO Fill Flag (TFFF) in the DSPI*x*_SR is set. The TFFF bit is cleared when TX FIFO is full and the eDMA controller indicates that a write to DSPI*x*_PUSHR is complete or by host software setting the TFFF in the DSPI*x*_SR. The TFFF can generate a DMA request or an interrupt request. See Section 22.6.7.2, “Transmit FIFO Fill Interrupt or DMA Request,” for details.

The DSPI ignores attempts to push data to a full TX FIFO, i.e. the state of the TX FIFO is unchanged. No error condition is indicated.

22.6.3.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter is decremented by one. At the end of a transfer, the TCF bit in the DSPI*x*_SR is set to indicate the completion of a transfer. The TX FIFO is flushed by setting the CLR_TXF bit in DSPI*x*_MCR.

If an external bus master initiates a transfer with a DSPI slave while the slave’s DSPI TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave’s DSPI*x*_SR is set. See Section 22.6.7.4, “Transmit FIFO Underflow Interrupt Request,” for details.

22.6.3.5 Receive First In First Out (RX FIFO) Buffering Mechanism

The RX FIFO functions as a buffer for data received on the SIN signal. The RX FIFO is up to four entries deep. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data is removed (popped) from the RX FIFO by reading the Section 22.5.1.7, “DSPI POP RX FIFO Register (DSPIx_POPR).” RX FIFO entries can only be removed from the RX FIFO by reading the DSPI*x*_POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the Section 22.5.1.4, “DSPI Status Register (DSPIx_SR),” indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the DSPI _POPR is read or SPI data is copied from the shift register to the RX FIFO.

The pop next pointer (POPNXTPTR) field in the DSPI*x*_SR points to the RX FIFO entry that is returned when the DSPI*x*_POPR is read. The POPNXTPTR contains the positive offset from DSPI*x*_RXFR0 in number of 32-bit registers. For example, POPNXTPTR = 0b0010 means that the DSPI*x*_RXFR2 contains

the received SPI data that will be returned when DSPI*x*_POPR is read. The POPNXTPTR field is incremented every time the DSPI*x*_POPR is read.

22.6.3.5.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time a SPI frame is transferred to the RX FIFO the RX FIFO counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the receive FIFO overflow flag (RFOF) bit in the DSPI*x*_SR is set indicating an overflow condition. Depending on the state of the receive FIFO overflow overwrite enable (ROOE) bit in the DSPI*x*_MCR, the data from the transfer that generated the overflow is either ignored or shifted into the shift register. If the ROOE bit is set, the incoming data is shifted into the shift register. If the ROOE bit is cleared, the incoming data is ignored.

22.6.3.5.2 Draining the RX FIFO

Host software or other intelligent modules can remove (pop) entries from the RX FIFO by reading DSPIx_POPR; refer to Section 22.5.1.7, “DSPI POP RX FIFO Register (DSPIx_POPR).” A read of the DSPI*x*_POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored, the RX FIFO counter remains unchanged. The data returned from reading an empty RX FIFO is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the DSPI*x*_SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the eDMA controller indicates that a read from DSPI*x*_POPR is complete or by host software setting the RFDF bit.

22.6.4 DSPI Baud Rate and Clock Delay Generation

The SCK frequency and the delay values for serial transfer are generated by dividing the peripheral bus clock frequency (f_{IPS}) by a prescaler and a scaler with the option of doubling the baud rate. Figure 22-14 shows conceptually how the SCK signal is generated.

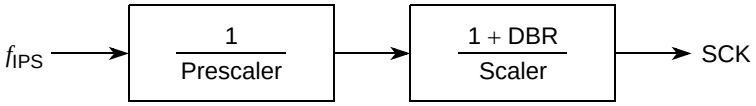


Figure 22-14. Serial Communications Clock Prescalers and Scalers

22.6.4.1 Baud Rate Generator

The baud rate is the frequency of the Serial Communication Clock (SCK). The peripheral bus clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK, with an option to double the baud rate (not available on mask set L49P and L47W devices). The PBR and BR fields in the DSPI*x*_CTARs select the frequency of SCK by the formula in Table 22-5. Table 22-13 shows example computations of the baud rate. Refer to Section 22.7.2, “Baud Rate Settings,” for additional application information.

Table 22-13. Baud Rate Computation Examples

f_{IPS}	PBR	Prescaler	BR	Scaler	DBR	Baud Rate
25.0 MHz	0b10	5	0b0000	2	0	2.5 Mb/s
20.0 MHz	0b00	2	0b0001	4	0	2.5 Mb/s
12.5 MHz	0b10	5	0b0000	2	1	2.5 Mb/s
10.0 MHz	0b00	2	0b0001	4	1	2.5 Mb/s

22.6.4.2 PCS to SCK Delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See Figure 22-16 for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the DSPI*x*_CTAR*n* registers select the PCS to SCK delay by the formula in Section Table 22-5., "DSPIx_CTARn Field Descriptions." Table 22-14 shows example computations of the PCS to SCK delay.

Table 22-14. PCS to SCK Delay Computation Examples

f_{IPS}	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK Delay
25.0 MHz	0b00	1	0b0101	64	2.56 us
20.0 MHz	0b01	3	0b0011	16	2.40 us
12.5 MHz	0b00	1	0b0100	32	2.56 us
10.0 MHz	0b01	3	0b0010	8	2.40 us

22.6.4.3 After SCK Delay (t_{ASC})

The After SCK delay is the length of time between the last edge of SCK and the negation of PCS. See Figure 22-16 and Figure 22-17 for illustrations of the After SCK delay. The PASC and ASC fields in the DSPI*x*_CTAR*n* registers select the After SCK delay by the formula in Section Table 22-5., "DSPIx_CTARn Field Descriptions." Table 22-15 shows example computations of the After SCK delay.

Table 22-15. After SCK Delay Computation Examples

f_{IPS}	PASC	Prescaler	ASC	Scaler	After SCK Delay
25.0 MHz	0b00	1	0b0101	64	2.56 us
20.0 MHz	0b01	3	0b0011	16	2.40 us
12.5 MHz	0b00	1	0b0100	32	2.56 us
10.0 MHz	0b01	3	0b0010	8	2.40 us

22.6.4.4 Delay after Transfer (t_{DT})

The delay after transfer is the length of time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See Figure 22-16 for an illustration of the delay after transfer. The PDT and DT fields in the DSPI*x*_CTAR*n* registers select the delay after transfer by the

formula in Table 22-5. Table 22-16 shows example computations of the delay after transfer. Refer to Section 22.7.3, “Delay Settings,” for more information.

Table 22-16. Delay After Transfer Computation Examples

f_{IPS}	PDT	Prescaler	DT	Scaler	Delay after Transfer
25.0 MHz	0b11	7	0b1100	8192	2.29 ms
20.0 MHz	0b01	3	0b1101	16384	2.46 ms
12.5 MHz	0b11	7	0b1011	4096	2.29 ms
10.0 MHz	0b01	3	0b1100	8192	2.46 ms

22.6.4.5 Peripheral Chip Select Strobe Enable ($\overline{\text{PCSS}}$)

The $\overline{\text{PCSS}}$ signal provides a delay to allow the PCS signals to settle after transitioning, thereby avoiding glitches. When the DSPI is in master mode and PCSSE bit is set in the DSPI*x*_MCR, $\overline{\text{PCSS}}$ provides a signal for an external demultiplexer to decode the PCS[0:2] signals into as many as 32 glitch-free PCS signals. Figure 22-15 shows the timing of the $\overline{\text{PCSS}}$ signal relative to PCS signals.

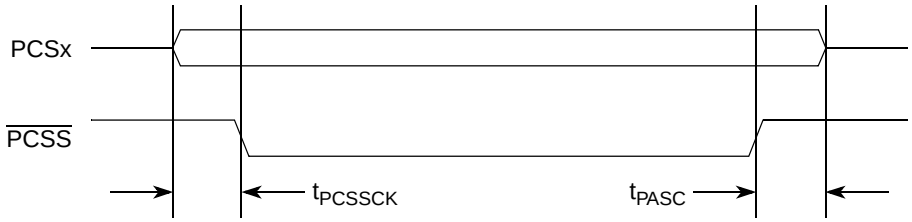


Figure 22-15. Peripheral Chip Select Strobe Timing

The delay between the assertion of the PCS signals and the assertion of $\overline{\text{PCSS}}$ is selected by the PCSSCK field in the DSPI*x*_CTAR*n* based on the following formula:

$$t_{PCSSCK} = \frac{1}{f_{IPS}} \times PCSSCK \qquad \textit{Eqn. 22-5}$$

Table 22-17 shows example computations of the t_{PCSSCK} delay.

Table 22-17. Peripheral Chip Select Strobe Assert Computation Examples

f_{IPS}	PCSSCK	Prescaler	Delay before Assert
25.0 MHz	0b10	5	200 ns
20.0 MHz	0b01	3	150 ns
12.5 MHz	0b00	1	80 ns
10.0 MHz	0b00	1	100 ns

At the end of the transfer the delay between $\overline{\text{PCSS}}$ negation and PCS negation is selected by the PASC field in the DSPI*x*_CTAR*n* based on the following formula:

$$t_{PASC} = \frac{1}{f_{IPS}} \times PASC \qquad \textit{Eqn. 22-6}$$

Table 22-18 shows example computations of the t_{PASC} delay.

Table 22-18. Peripheral Chip Select Strobe Negate Computation Examples

f_{IPS}	PASC	Prescaler	Delay after Negate
25.0 MHz	0b10	5	200 ns
20.0 MHz	0b01	3	150 ns
12.5 MHz	0b00	1	80 ns
10.0 MHz	0b00	1	100 ns

The $\overline{\text{PCSS}}$ signal is not supported when Continuous SCK is enabled (CONT_SCKE=1).

22.6.5 Transfer Formats

The SPI serial communication is controlled by the serial communications clock (SCK) signal and the peripheral chip select (PCS) signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

When the DSPI is the bus master, the CPOL and CPHA bits in the DSPI Clock and Transfer Attributes Registers (DSPI*x*_CTAR*n*) select the polarity and phase of the serial clock, SCK. The polarity bit selects the idle state of the SCK. The clock phase bit selects if the data on SOUT is valid before or on the first SCK edge.

When the DSPI is the bus slave, CPOL and CPHA bits in the DSPI*x*_CTAR0 (SPI) select the polarity and phase of the serial clock. For SPI slaves the DSPI*x*_CTAR0 is used. Even though the bus slave does not control the SCK signal, the clock polarity, clock phase, and number of bits to transfer must be identical for the master device and the slave device to ensure proper transmission.

The DSPI supports four different transfer formats:

- Classic SPI with CPHA=0
- Classic SPI with CPHA=1
- Modified Transfer format with CPHA = 0
- Modified Transfer format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The DSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the DSPI*x*_MCR selects between classic SPI format and modified transfer format. The classic SPI formats are described in Section 22.6.5.1, “Classic SPI Transfer Format (CPHA = 0),” and Section 22.6.5.2, “Classic SPI Transfer Format (CPHA = 1).” The modified transfer formats are described in Section 22.6.5.3, “Modified SPI Transfer Format (MTFE = 1, CPHA = 0),” and Section 22.6.5.4, “Modified SPI Transfer Format (MTFE = 1, CPHA = 1).”

In the SPI Configuration, the DSPI provides the option of keeping the PCS signals asserted between frames. See Section 22.6.5.5, “Continuous Selection Format,” for details.

22.6.5.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in Figure 22-16 is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

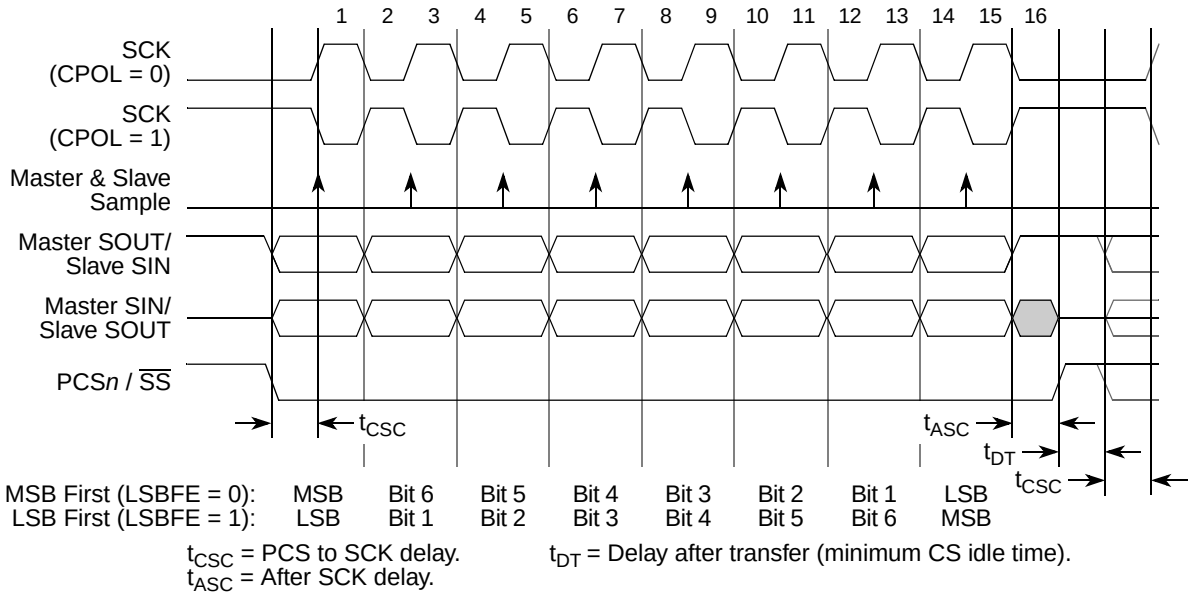


Figure 22-16. DSPI Transfer Timing Diagram (MTFE=0, CPHA=0, FMSZ=8)

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay has elapsed, the master outputs the first edge of SCK. This is the edge used by the master and slave devices to sample the first input data bit on their serial data input signals. At the second edge of the SCK the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

22.6.5.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in Figure 22-17 is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges

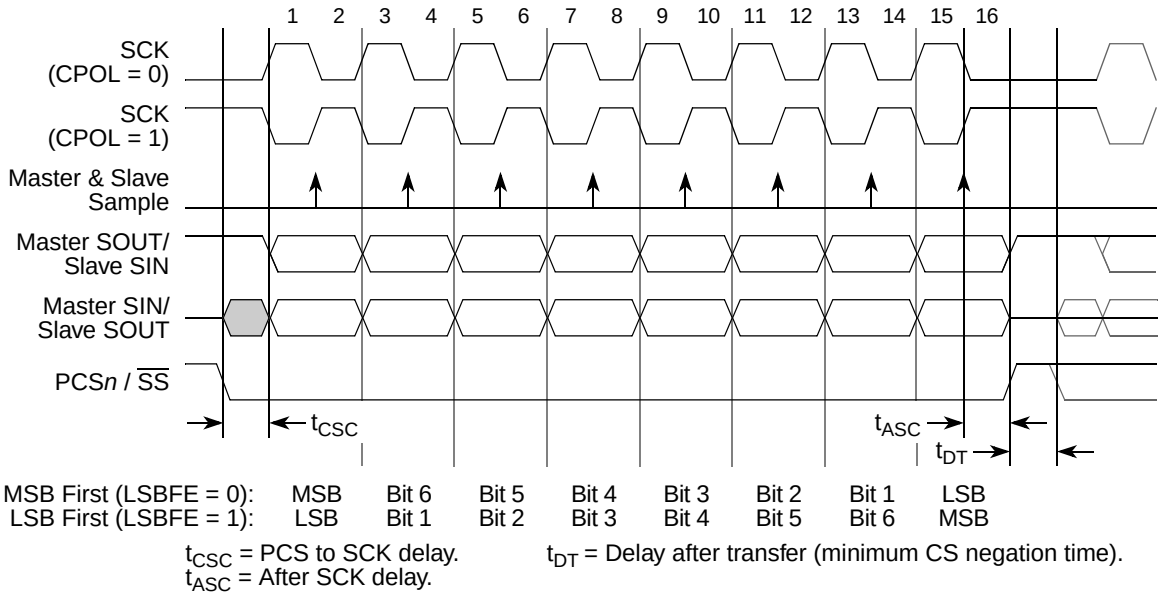


Figure 22-17. DSPI Transfer Timing Diagram (MTFE=0, CPHA=1, FMSZ=8)

The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

22.6.5.3 Modified SPI Transfer Format (MTFE = 1, CPHA = 0)

In the modified transfer format, both the master and the slave sample later in the SCK period than in classic SPI mode to allow for delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed, the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The slave also places new data on the slave SOUT on every odd numbered clock edge.

The master places its second data bit on the SOUT line one peripheral bus clock after an odd numbered SCK edge. The point where the master samples the slave SOUT is selected by writing to the SMPL_PT field in the DSPI*x*_MCR. Table 22-3 lists the number of peripheral bus clock cycles between the active edge of SCK and the master Sample point. The master sample point can be delayed by one or two peripheral bus clock cycles.

Figure 22-18 shows the modified transfer format for CPHA = 0. Only the condition where CPOL = 0 is illustrated. The delayed master sample points are indicated with a lighter shaded arrow.

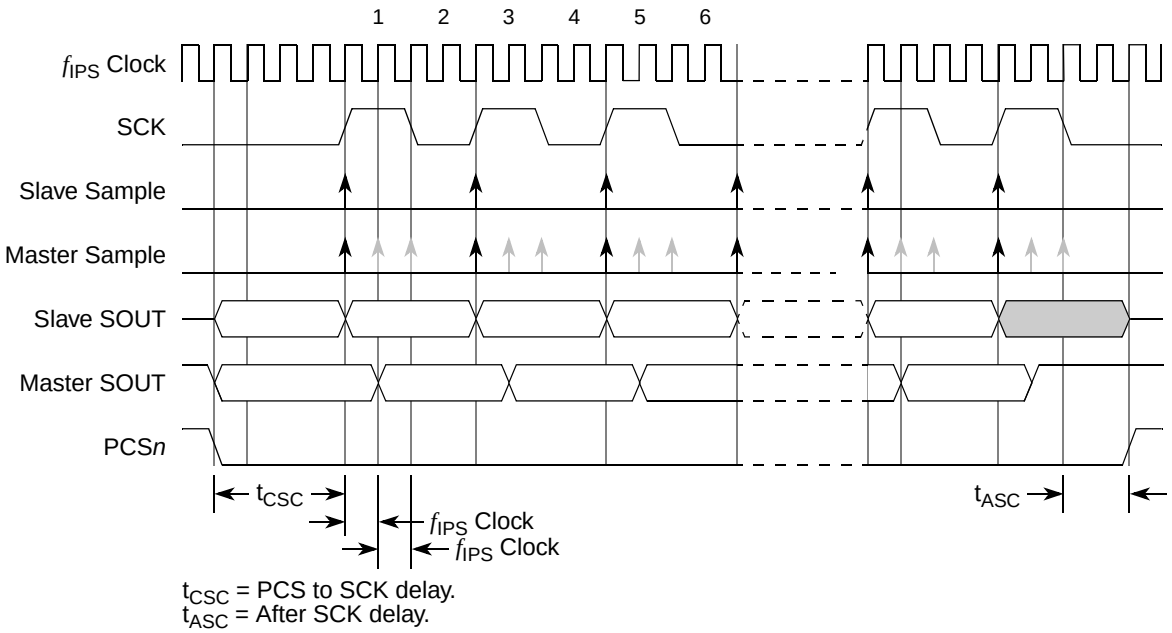


Figure 22-18. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{SCK} = f_{IPS} \div 4$)

22.6.5.4 Modified SPI Transfer Format (MTFE = 1, CPHA = 1)

Figure 22-19 shows the modified transfer format for CPHA = 1. Only the condition where CPOL = 0 is described. At the start of a transfer, the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed, the master and the slave put data on their SOUT pins at the first edge of SCK. The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the 3rd SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. The SCK to PCS delay must be greater or equal to half of the SCK period.

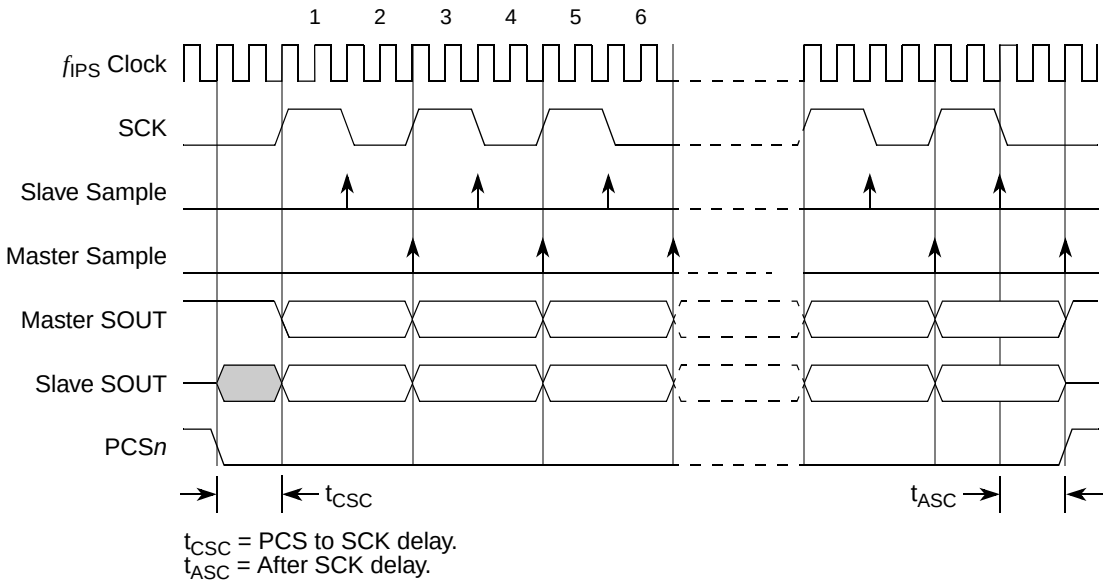


Figure 22-19. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{SCK} = f_{IPS} \div 4$)

22.6.5.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The continuous selection format provides the flexibility to handle both cases. The continuous selection format is enabled for the SPI Configuration by setting the CONT bit in the SPI command.

When the CONT bit = 0, the DSPI drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSIS field in the DSPI*x*_MCR. Figure 22-20 shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 0.

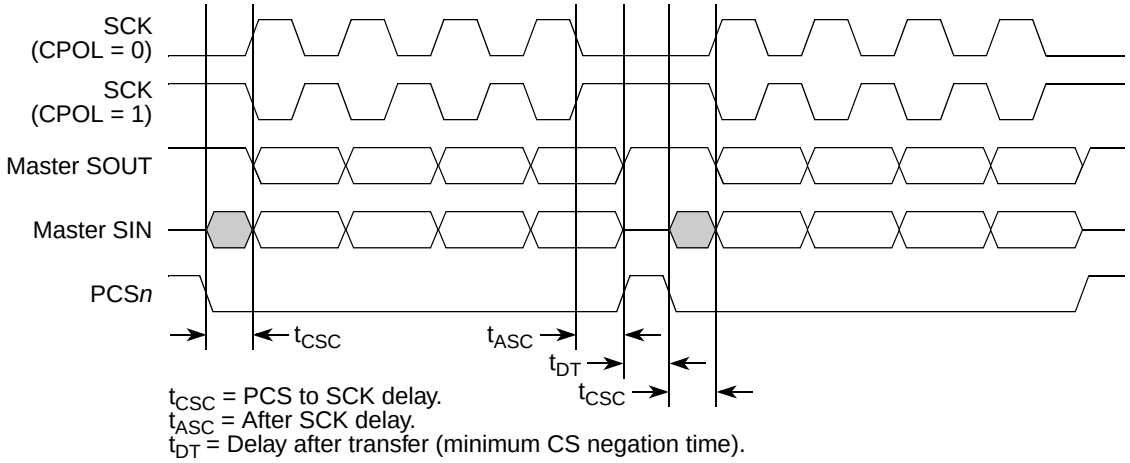


Figure 22-20. DSPI Example of Non-Continuous Format (CPHA=1, CONT=0)

When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The delay between transfers (t_{DT}) is not inserted between the transfers. Figure 22-21 shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

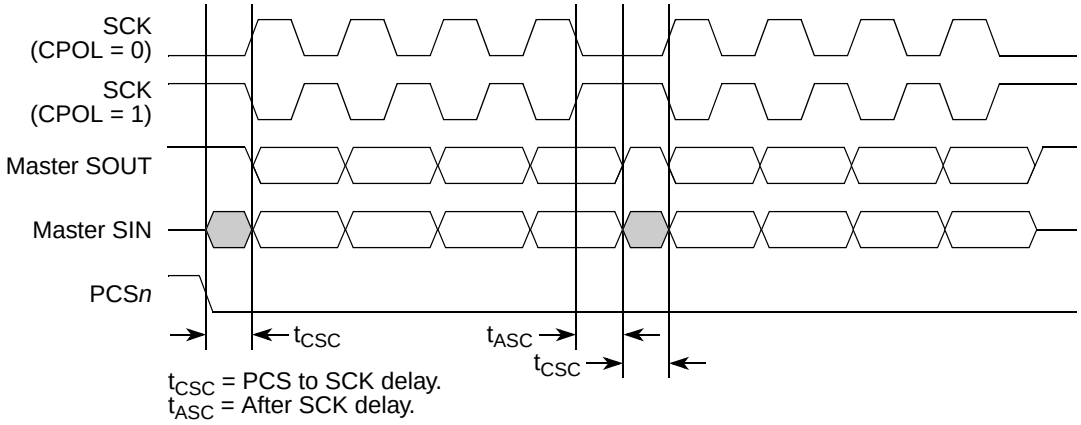


Figure 22-21. DSPI Example of Continuous Transfer (CPHA=1, CONT=1)

Changing CTARs or which PCS signals are asserted between frames while using continuous transfer mode can cause errors in the transfer. The PCS signal should be negated before the CTAR is switched or the PCS definition is changed.

22.6.6 Continuous Serial Communications Clock

The DSPI provides the option of generating a continuous SCK signal for slave peripherals that require a continuous clock. Continuous SCK is enabled by setting the CONT_SCKE bit in the DSPI*x*_MCR.

Continuous SCK is only supported for CPHA=1. Setting CPHA=0 will be ignored if the CONT_SCKE bit is set. Continuous SCK is supported for modified transfer format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- Upon enabling the continuous SCK, CTAR0 will be used. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame shall be used.
- The selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

NOTE

It is recommended that the baud rate is the same for all transfers made while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the DSPI is put into the external stop mode or module disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the After SCK delay. The delay after transfer is fixed at one SCK cycle. Figure 22-22 shows the timing diagram for continuous SCK format with continuous selection disabled.

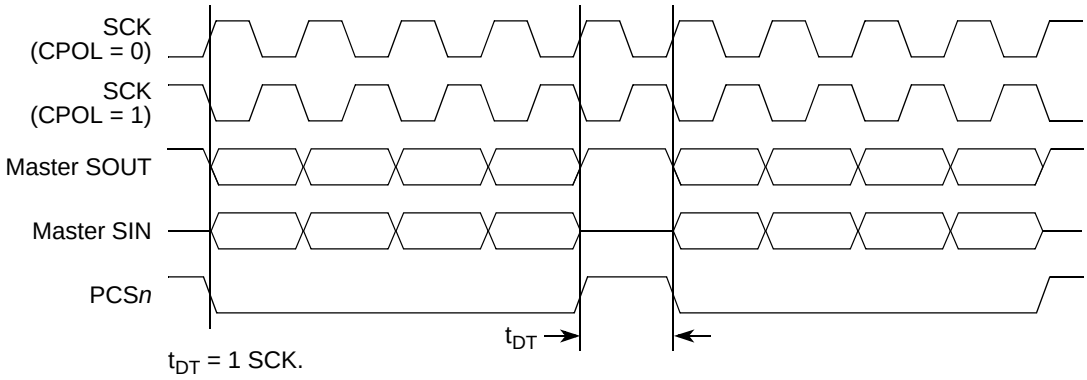


Figure 22-22. DSPI Continuous SCK Timing Diagram (CONT=0)

If the CONT bit in the TX FIFO entry is set or the DCONT in the DSPI_DSICR is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT (SOUT pulled high). This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the transmit FIFO.
- Continuous SCK with CONT bit set and entering STOPPED state (see Section 22.6.2, “Start and Stop of DSPI Transfers”).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

Figure 22-23 shows a timing diagram for Continuous SCK format with Continuous Selection enabled.

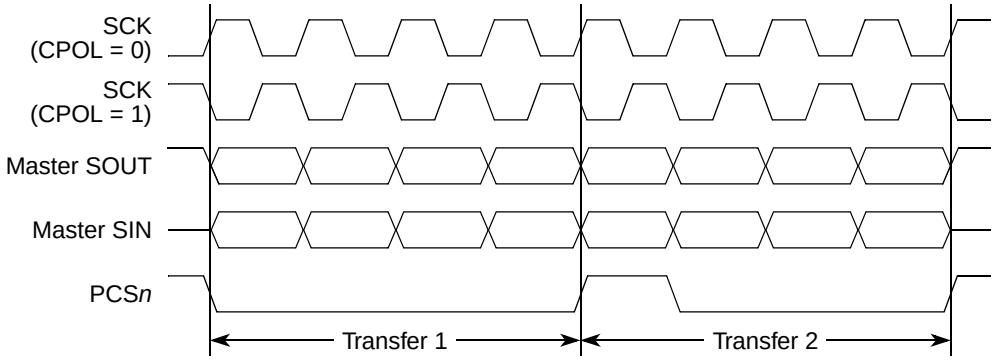


Figure 22-23. DSPI Continuous SCK Timing Diagram (CONT=1)

22.6.7 Interrupts/DMA Requests

The DSPI has four conditions that can only generate interrupt requests and two conditions that can generate interrupt or DMA requests. Table 22-19 lists the six conditions.

Table 22-19. DSPI Interrupt and DMA Request Conditions

Condition	Flag	Interrupt	DMA
End of Queue (EOQ)	EOQF	X	
TX FIFO Fill	TFFF	X	X
Transfer Complete	TCF	X	

Table 22-19. DSPI Interrupt and DMA Request Conditions

Condition	Flag	Interrupt	DMA
TX FIFO Underflow	TFUF	X	
RX FIFO Drain	RFDF	X	X
RX FIFO Overflow	RFOF	X	

Each condition has a flag bit in the Section 22.5.1.4, "DSPI Status Register (DSPIx_SR)," and a Request Enable bit in the Section 22.5.1.5, "DSPI DMA/Interrupt Request Select / Enable Register (DSPIx_RSER)." The TX FIFO Fill Flag (TFFF) and RX FIFO Drain Flag (RFDF) generate interrupt requests or DMA requests depending on the TFFF_DIRS and RFDF_DIRS bits in the DSPI*x*_RSER.

22.6.7.1 End of Queue Interrupt Request

The End of Queue Request indicates that the end of a transmit queue is reached. The End of Queue Request is generated when the EOQ bit in the executing SPI command is asserted and the EOQF_RE bit in the DSPI*x*_RSER is asserted.

22.6.7.2 Transmit FIFO Fill Interrupt or DMA Request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the DSPI*x*_RSER is asserted. The TFFF_DIRS bit in the DSPI*x*_RSER selects whether a DMA request or an interrupt request is generated.

22.6.7.3 Transfer Complete Interrupt Request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the DSPI*x*_RSER.

22.6.7.4 Transmit FIFO Underflow Interrupt Request

The Transmit FIFO Underflow Request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for DSPI modules operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI operating in slave mode and SPI configuration is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the DSPI*x*_RSER is asserted, an interrupt request is generated.

22.6.7.5 Receive FIFO Drain Interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the DSPI*x*_RSER is asserted. The RFDF_DIRS bit in the DSPI*x*_RSER selects whether a DMA request or an interrupt request is generated.

22.6.7.6 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow Request is generated when RX FIFO and the shift register are full and a transfer is initiated. The RFOF_RE bit in the DSPI*x*_RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the DSPI*x*_MCR, the data from the transfer that generated the overflow is either ignored or shifted into the shift register. If the ROOE bit is set, the incoming data is shifted into the shift register. If the ROOE bit is negated, the incoming data is ignored.

22.7 Initialization / Application Information

22.7.1 Changing Queues

This section presents an example of how to change queues for the DSPI. The queues are not part of the DSPI, but the DSPI includes features in support of queue management. Queues are primarily supported in SPI Configuration.

1. The last command word from a queue is executed. The EOQ bit in the command word is set to indicate to the DSPI that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the DSPIx_SR is set.
3. The setting of the EOQF flag will disable both serial transmission, and serial reception of data, putting the DSPI in the STOPPED state. The TXRXS bit is negated to indicate the STOPPED state.
4. The eDMA will continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DSPI eDMA transfers by disabling the DMA request for the eDMA channel assigned to TX FIFO and RX FIFO. This is done by disabling the appropriate channel in the eDMA module or the DMA request channel in the DMAMux module.
6. Ensure all received data in RX FIFO has been transferred to the memory receive queue by reading the RXCNT in DSPI*x*_SR or by checking RFDF in the DSPI*x*_SR after each read operation of the DSPI*x*_POPR.
7. Modify eDMA descriptor of TX and RX channels for “new” queues
8. Flush TX FIFO by setting the CLR_TXF bit in the DSPI*x*_MCR, Flush RX FIFO by setting the CLR_RXF bit in the DSPI*x*_MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the DSPI*x*_TCR.
10. Enable eDMA transfers by enabling the DMA request for the eDMA channel assigned to the DSPI TX FIFO, and RX FIFO by enabling the corresponding eDMA channel and the appropriate DMA request channel in the DMAMux module.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

22.7.2 Baud Rate Settings

Table 22-20 lists the baud rates based on various baud rate prescaler and scaler values (DSPI*x*_CTAR*n*[PBR, BR]). The example baud rates shown assume a 20 MHz peripheral bus frequency, f_{IPS}, (f_{SYS} = 40 MHz) with DSPI_CTAR*n*[DBR] set.

Table 22-20. Example DSPI Baud Rate Values (Hz), f_{IPS} = 20 MHz

DBR = 1		Baud Rate Prescaler Values (DSPI*x*_CTAR*n*[PBR])			
		2	3	5	7
Baud Rate Scaler Values (DSPI*x*_CTAR*n*[BR])	2	10,000,000	6,666,667	4,000,000	2,857,143
	4	5,000,000	3,333,333	2,000,000	1,428,571
	6	3,333,333	2,222,222	1,333,333	952,381
	8	2,500,000	1,666,667	1,000,000	714,286
	16	1,250,000	833,333	500,000	357,143
	32	625,000	416,667	250,000	178,571
	64	312,500	208,333	125,000	89,286
	128	156,250	104,167	62,500	44,643
	256	78,125	52,083	31,250	22,321
	512	39,063	26,042	15,625	11,161
	1024	19,531	13,021	7,813	5,580
	2048	9,766	6,510	3,906	2,790
	4096	4,883	3,255	1,953	1,395
	8192	2,441	1,628	977	698
	16384	1,221	814	488	349
	32768	610	407	244	174

22.7.3 Delay Settings

Table 22-21 shows values for the delay after transfer (t_{DT}) and CS to SCK delay (t_{CSC}) based on the delay prescaler and scaler values (DSPI*x*_CTAR*n*[PDT, DT]). The example delay values shown assume a 20 MHz f_{IPS}.

Table 22-21. Example DSPI Delay Values (ns), f_{IPS} = 20 MHz

Delay Scaler Values (DSPIx_CTARn[DT])	Delay Prescaler Values (DSPIx_CTARn[PDT])			
	1	3	5	7
2	200	300	500	700
4	400	600	1,000	1,400
8	800	1,200	2,000	2,800
16	1,600	2,400	4,000	5,600
32	3,200	4,800	8,000	11,200
64	6,400	9,600	16,000	22,400
128	12,800	19,200	32,000	44,800
256	25,600	38,400	64,000	89,600
512	51,200	76,800	128,000	179,200
1024	102,400	153,600	256,000	358,400
2048	204,800	307,200	512,000	716,800
4096	409,600	614,400	1,024,000	1,433,600
8192	819,200	1,228,800	2,048,000	2,867,200
16384	1,638,400	2,457,600	4,096,000	5,734,400
32768	3,276,800	4,915,200	8,192,000	11,468,800
65536	6,553,600	9,830,400	16,384,000	22,937,600

22.7.4 Calculation of FIFO Pointer Addresses

The user has complete visibility of the TX and RX FIFO contents through the FIFO registers, and valid entries can be identified through a memory mapped pointer and a memory mapped counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). Figure 22-24 illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over to the RX FIFO. See Section 22.6.3.4, “Transmit First In First Out (TX FIFO) Buffering Mechanism,” and Section 22.6.3.5, “Receive First In First Out (RX FIFO) Buffering Mechanism,” for details on the FIFO operation.

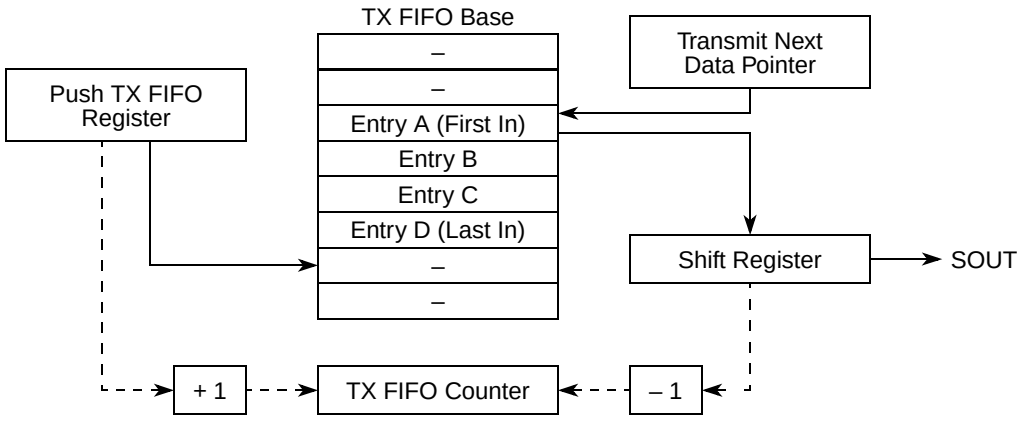


Figure 22-24. DSPI TX FIFO Pointers and Counter

22.7.4.1 Address Calculation for First-in Entry / Last-in Entry in RX FIFO

The address of the first-in entry in the RX FIFO is computed by the following equation:

$$\text{First-in Entry Address} = \text{RX FIFO Base} + 4 \times \text{POPNXTPTR} \qquad \textit{Eqn. 22-7}$$

The address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-in Entry Address} = \text{RX FIFO Base} + 4 \times [\text{Modulo RX FIFO depth}(\text{RXCTR} + \text{POPNXTPTR} - 1)] \qquad \textit{Eqn. 22-8}$$

RX FIFO Base = Base address of RX FIFO

RXCTR = RX FIFO counter

POPNXTPTR = Pop Next Pointer

RX FIFO Depth = Receive FIFO depth (4 for MAC7100 Family devices)

Chapter 23
Controller Area Network Module (FlexCAN)

23.1 Overview

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this application: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth. The FlexCAN module is a full implementation of the CAN protocol specification, Version 2.0 B [Ref. 1], which supports both standard and extended message frames. 32 message buffers (MBs) are supported, which are stored in 544 bytes of embedded RAM dedicated to the FlexCAN module. Refer to Section 23.5.1, "Message Buffer Structure," for the actual number of message buffers available in the MCU. A general block diagram is shown in Table 23-1.

The CAN Protocol Interface (CPI) manages the serial communication on the CAN bus, requesting RAM access for receiving and transmitting message frames, validating received messages, and performing error handling. The Message Buffer Management (MBM) handles message buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms. The Bus Interface Unit (BIU) controls the access to and from the internal interface bus, in order to establish connection to the CPU and to any other modules. Clocks, address and data buses, interrupt outputs, and test signals are accessed through the Bus Interface Unit.

There are up to four FlexCAN modules on the MAC7100 family of devices. Fewer CAN modules are implemented on the subset devices. Refer to the Table 1-1 on page 1-3 for a general description of each device.

On the MAC7100 family of devices, the clock source for the FlexCAN can be selected to be either derived from the PLL clock or the oscillator clock. The FlexCAN also implements a low pass filter which can be used to wake the device up when activity is detected on the bus while also enabling rejection of bus noise to help eliminate instances of false wake. MAC7100 Family devices are implemented with 32 mailboxes on each of their CAN modules.

Each of the FlexCAN modules can be independently disabled by writing to the MDIS bit in the module's configuration register, MCR. Disabling the module will turn off the clock to the module's protocol engine and message buffers, although most of the module's registers remain available to be accessed by the core across the peripheral bus. The MDIS bit is intended to be used when the module is not required in the application.

The detailed use of the CAN protocol is not described in this guide because the protocol is described in the CAN Standards referenced below.

23.1.1 References

1. Controller Area Network – *CAN Specification Version 2.0 Part A, Part B*, Robert Bosch GmbH, 1991.
2. ISO International Standard – *ISO 11898 First Edition 1993 Road Vehicles – Interchange of Digital Information – Controller Area Network (CAN) for high-speed Communication*.

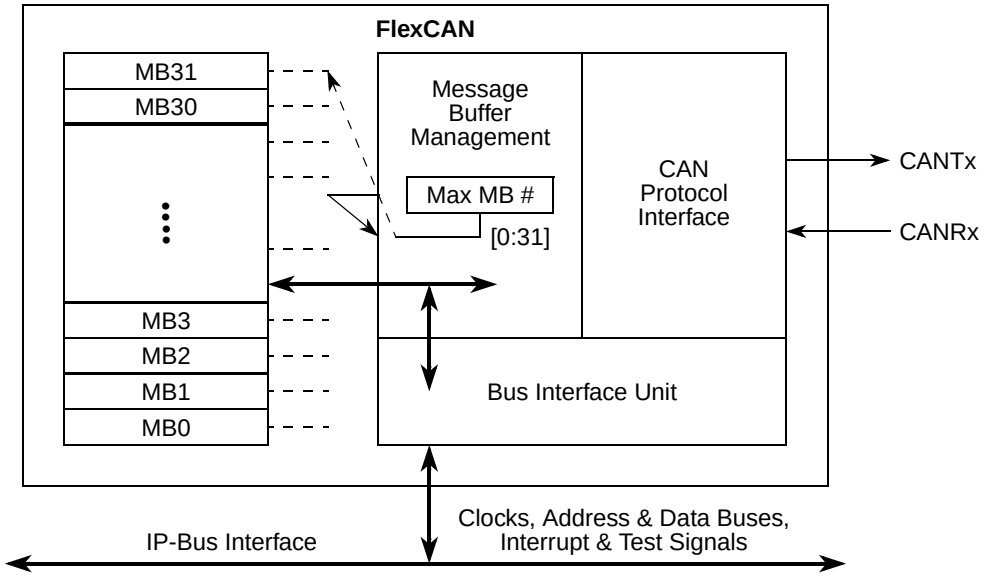


Figure 23-1. FlexCAN Block Diagram

23.2 Features

The FlexCAN module includes these distinctive features:

- Full Implementation of the CAN protocol specification, Version 2.0B
  - — Standard data and remote frames
  - — Extended data and remote frames
  - — Zero to eight bytes data length
  - — Programmable bit rate up to 1 Mb/sec
  - — Content-related addressing
- Flexible message buffers (up to 32) of zero to eight bytes data length
- Each message buffer (MB) configurable as Rx or Tx, all supporting standard and extended messages
- Includes 544 bytes of RAM used for the storage of up to 32 message buffers (MBs)
- Programmable clock source to the CAN Protocol Interface, either bus clock or crystal oscillator
- Unused MB space can be used as general purpose RAM space
- Listen-only mode capability
- Programmable loop-back mode supporting self-test operation
- Three programmable mask registers
- Programmable transmission priority scheme: lowest ID or lowest buffer number
- Time Stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independent of the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes, with programmable wake-up on bus activity

23.3 Modes of Operation

The FlexCAN module has four functional modes: normal mode (user and supervisor), freeze mode, listen-only mode, and loop-back mode. There are also three low power modes: module disabled, doze mode, and stop mode.

- Normal Mode (User or Supervisor) — In normal mode, the module operates receiving and/or transmitting message frames, errors are handled normally, and all the CAN Protocol functions are enabled. User and supervisor modes differ in the access to some restricted control registers.
- Freeze Mode — Freeze mode is enabled when the FRZ bit in the MCR is set. If enabled, freeze mode is entered when the HALT bit in MCR is set or when debug mode is requested at the MCU level. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Section 23.6.8.1, “Freeze Mode,” for more information.
- Listen-Only Mode — The module enters this mode when the LOM bit in the control register is set. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN error passive mode [Ref. 1]. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.
- Loop-Back Mode — The module enters this mode when the LPB bit in the control register is set. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic ‘1’). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.
- Module Disabled Mode — This low power mode is entered when the MDIS bit in the MCR is set. When disabled, the module shuts down the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules. To exit from this, negate the MDIS bit in the MCR. See Section 23.6.8.2, “Module Disabled Mode,” for more information.
- Doze Mode — This low power mode is entered when the DOZE bit in MCR is set and doze mode is requested at MCU level. When in doze mode, the module shuts down the clocks to the CAN Protocol Interface and the Message Buffer Management sub-modules. This mode is exited when the DOZE bit in MCR is negated, when the MCU is removed from doze mode, or when activity is detected on the CAN bus and the self wake-up mechanism is enabled. See Section 23.6.8.3, “Doze Mode,” for more information.
- Stop Mode — This low power mode is entered when stop mode is requested at MCU level. When in stop mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. This mode is exited when the stop mode request is removed or when activity is detected on the CAN bus and the self wake-up mechanism is enabled. See Section 23.6.8.4, “Stop Mode,” for more information.

23.4 Signal Description

The FlexCAN module has two I/O signals connected to the external MCU pins. These signals are summarized in Table 23-1 and described in more detail in the next sub-sections. Note that the Port Integration Module (PIM) must be configured to enable the peripheral function of the appropriate pins (refer to Section 18.6.2, “Peripheral Mode,” on page 18-296) prior to configuring a FlexCAN channel.

Table 23-1. FlexCAN Signal Properties

Signal Name	Direction	Description
CNRX_*x*	Input	CAN Receive Pin
CNTX_*x*	Output	CAN Transmit Pin

23.4.1 CNRX_*x*

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level ‘0’. Recessive state is represented by logic level ‘1’.

23.4.2 CNTX_*x*

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level ‘0’. Recessive state is represented by logic level ‘1’.

23.5 Memory Map / Register Definition

This section describes the registers and data structures in the FlexCAN module. Refer to Chapter 8, “Device Memory Map,” for the base address of the module. Addresses in this section are relative to the base address.

The address space occupied by FlexCAN has 96 bytes for registers starting at the module base address followed by message buffer (MB) storage space in embedded RAM. The complete memory map for a FlexCAN module with 32 MBs capability is shown in Table 23-2. Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be supervisor (S) or unrestricted (U). Most of the registers can be configured to have either supervisor or unrestricted access by programming the SUPV bit in the MCR. These registers are identified as S/U in the Access column of Table 23-2.

Table 23-2. FlexCAN Memory Map

FlexCAN *x* Offset	Register Description	Access Type	Affected by Hard Reset	Affected by Soft Reset
0x0000	FlexCAN Module Configuration Register (MCR)	S	Yes	Yes
0x0004	FlexCAN Control Register (CTRL)	S/U	Yes	No
0x0008	FlexCAN Timer (TIMER)	S/U	Yes	Yes
0x000C	Reserved			
0x0010	FlexCAN Rx Global Mask Register (RXGMASK)	S/U	Yes	No
0x0014	Rx Buffer 14 Mask Register (RX14MASK)	S/U	Yes	No
0x0018	Rx Buffer 15 Mask Register (RX15MASK)	S/U	Yes	No
0x001C	FlexCAN Error Counter Register (ECR)	S/U	Yes	Yes

Table 23-2. FlexCAN Memory Map (continued)

FlexCAN *x* Offset	Register Description	Access Type	Affected by Hard Reset	Affected by Soft Reset
0x0020	FlexCAN Error and Status Register (ESR)	S/U	Yes	Yes
0x0024	Reserved			
0x0028	FlexCAN Interrupt Mask Register (IMASK)	S/U	Yes	Yes
0x002C	Reserved			
0x0030	FlexCAN Interrupt Flags Register (IFLAG)	S/U	Yes	Yes
0x0034–0x005F	Reserved			
0x0060–0x007F	Reserved			
0x0080–0x027F	Message Buffers (MB0–MB31)	S/U	No	No

The FlexCAN module stores CAN messages for transmission and reception using a message buffer structure. Each individual MB is formed by 16 bytes mapped on memory as described in Table 23-3. The FlexCAN module can manage up to 32 message buffers. Table 23-3 shows a Standard/Extended message buffer (MB0) memory map, using 16 bytes total (0x80–0x8F space).

Table 23-3. FlexCAN Message Buffer MB0 Memory Mapping

MB*n* Offset	MB Field Description
0x80	Control and Status (C/S)
0x84	Identifier Field
0x88–0x8F	Data Field 0 – Data Field 7 (1 byte each)

23.5.1 Message Buffer Structure

The message buffer structure used by the FlexCAN module is represented in Figure 23-2. Both extended and standard frames (29-bit identifier and 11-bit identifier, respectively) used in the CAN specification (Version 2.0 Part B) are represented.

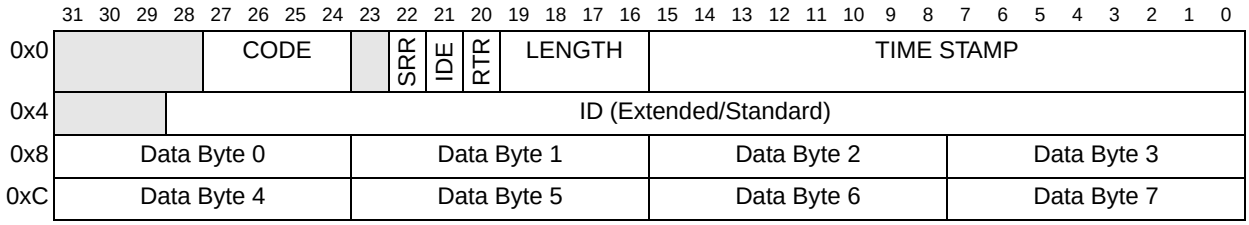


Figure 23-2. FlexCAN Message Buffer Structure

Table 23-4. FlexCAN MB Field Descriptions

Bits	Name	Description
31–28	—	Reserved.
27–24	CODE[3:0]	Message buffer code. This 4-bit field can be accessed (read or write) by the CPU and by the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 23-5 and Table 23-6. See Section 23.6, “Functional Description,” for additional information.

Table 23-4. FlexCAN MB Field Descriptions (continued)

Bits	Name	Description
23	—	Reserved.
22	SRR	Substitute remote request. Fixed recessive bit, used only in extended format. It must be set to '1' by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as arbitration loss. 0 Dominant is not a valid value for transmission in Extended Format frames 1 Recessive value is compulsory for transmission in Extended Format frames
21	IDE	ID extended bit. This bit identifies whether the frame format is standard or extended. 0 Frame format is standard 1 Frame format is extended
20	RTR	Remote transmission request. This bit is used for requesting transmissions of a data frame. If FlexCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN module treats it as bit error. If the value received matches the value transmitted, it is considered as a successful bit transmission. 0 Indicates the current MB has a data frame to be transmitted 1 Indicates the current MB has a remote frame to be transmitted
19–16	LENGTH[3:0]	Length of data in bytes. This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space (see Figure 23-2). In reception, this field is written by the FlexCAN module, copied from the DLC (data length code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR=1, the frame to be transmitted is a remote frame and does not include the DATA field, regardless of the LENGTH field.
15–0	TIME STAMP [15:0]	Free-running counter time stamp. This 16-bit field is a copy of the free-running timer, captured for TX and RX frames at the time when the beginning of the ID field appears on the CAN bus.
28–0	ID[28:0]	Frame identifier. In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases.
31–24, 23–16, 15–8, 7–0	DATA[7:0]	Data field. Up to eight bytes can be used for a data frame. For Rx frames, the data is stored as it is received from the CAN bus. For Tx frames, the CPU prepares the data field to be transmitted within the frame.

Table 23-5. FlexCAN Message Buffer Code for Rx buffers

Rx code BEFORE Rx new frame	Description	Rx code AFTER Rx new frame	Comment
0000	INACTIVE: MB is not active.	—	MB does not participate in the matching process.
0100	EMPTY: MB is active and empty.	0010	MB participates in the matching process. When a frame is received successfully, the code is automatically updated to FULL.

Table 23-5. FlexCAN Message Buffer Code for Rx buffers (continued)

Rx code BEFORE Rx new frame	Description	Rx code AFTER Rx new frame	Comment
0010	FULL: MB is full.	0010	The act of reading the C/S word followed by unlocking the MB does not make the code return to EMPTY. It remains FULL. If a new frame is written to the MB after the C/S word was read and the MB was unlocked, the code still remains FULL.
		0110	If the MB is FULL and a new frame should have been written to this MB before the CPU had time to read it, the MB is overwritten, and the code is automatically updated to OVERRUN. Refer to Section 23.6.5, “Matching Process,” for details about overrun behavior.
0110	OVERRUN: a frame was overwritten into a full buffer.	0010	If the code indicates OVERRUN but the CPU reads the C/S word and then unlocks the MB, when a new frame is written to the MB, the code returns to FULL.
		0110	If the code already indicates OVERRUN, and yet another new frame must be written, the MB will be overwritten again, and the code will remain OVERRUN. Refer to Section 23.6.5, “Matching Process,” for details about overrun behavior.
0XY1 [1]	BUSY: Flexcan is updating the contents of the MB. The CPU should not try to access the MB.	0010	An EMPTY buffer was written with a new frame (XY was 01).
		0110	A FULL/OVERRUN buffer was overwritten (XY was 11).

[1] Note that for Tx MBs (see Table 23-6), the BUSY bit should be ignored upon read.

Table 23-6. FlexCAN Message Buffer Code for Tx buffers

RTR	Initial Tx code	Code after successful transmission	Description
X	1000	–	INACTIVE: MB does not participate in the arbitration process.
0	1100	1000	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
1	1100	0100	Transmit remote frame unconditionally once. After transmission, the MB automatically becomes an Rx MB with the same ID.
0	1010	1010	Transmit a data frame whenever a remote request frame with the same ID is received. This MB participates simultaneously in both the matching and arbitration processes. The matching process compares the ID of the incoming remote request frame with the ID of the MB. If a match occurs this MB is allowed to participate in the current arbitration process and the CODE field is automatically updated to ‘1110’ to allow the MB to participate in future arbitration runs. When the frame is eventually transmitted successfully, the code automatically returns to ‘1010’ to restart the process again.

Table 23-6. FlexCAN Message Buffer Code for Tx buffers (continued)

RTR	Initial Tx code	Code after successful transmission	Description
0	1110	1010	This is an intermediate code that is automatically written to the MB by the MBM as a result of match to a remote request frame. The data frame will be transmitted unconditionally once and then the code will automatically return to '1010'. The CPU can also write this code with the same effect.

23.5.2 Register Descriptions

The FlexCAN registers are described in this section in ascending address order.

23.5.2.1 FlexCAN Module Configuration Register (MCR)

This register defines global system configurations, such as the module operation mode (low power, for example) and maximum message buffer configuration. Most of the fields in this register can be accessed at any time, except the MAXMB field, which should only be changed while the module is in freeze mode.

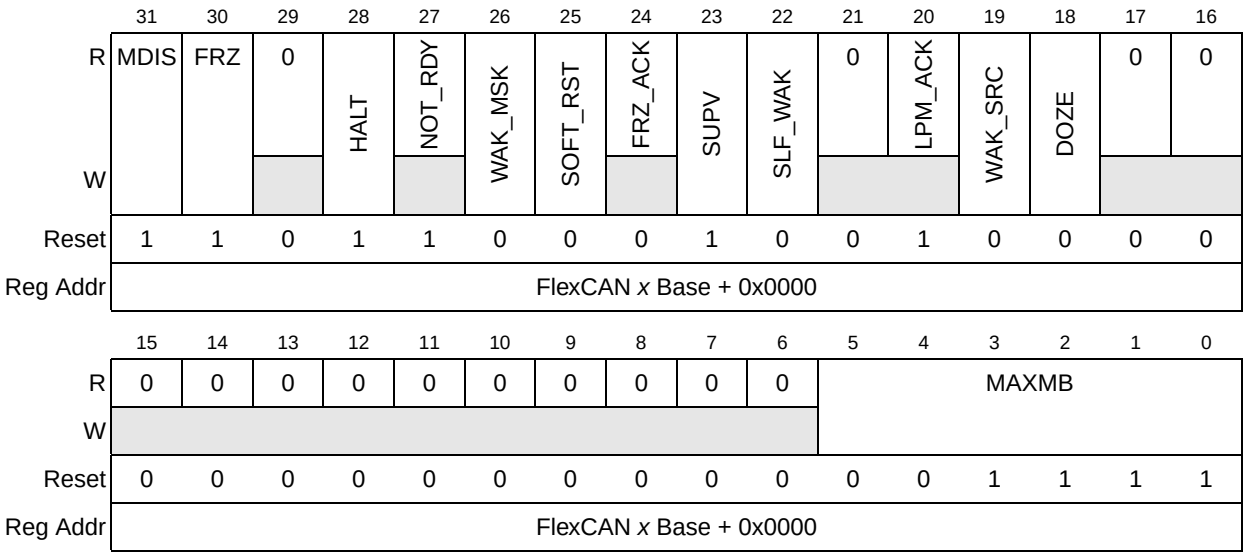


Figure 23-3. FlexCAN Module Configuration Register (MCR)

Table 23-7. MCR Field Descriptions

Bits	Name	Description
31	MDIS	Module disable. This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN shuts down the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules. This is the only bit in MCR not affected by soft reset. See Section 23.6.8.2, "Module Disabled Mode," for more information. 0 Enable the FlexCAN module 1 Disable the FlexCAN module

Table 23-7. MCR Field Descriptions (continued)

Bits	Name	Description
30	FRZ	Freeze enable. The FRZ bit specifies the FlexCAN behavior when the HALT bit in the MCR is set or when debug mode is requested at MCU level. When FRZ is set, FlexCAN is enabled to enter freeze mode. Clearing this bit field causes FlexCAN to exit freeze mode. 0 Not enabled to enter freeze mode 1 Enabled to enter freeze mode
29	—	Reserved.
28	HALT	Halt FlexCAN. Setting this bit puts the FlexCAN module into freeze mode. The CPU should clear it after initializing the message buffers and control register. No reception or transmission is performed by FlexCAN before this bit is cleared. While in freeze mode, the CPU has write access to the error counter register, that is otherwise read-only. freeze mode can not be entered while FlexCAN is in any of the low power modes. See Section 23.6.8.1, “Freeze Mode,” for more information. 0 No freeze mode request. 1 Enters freeze mode if the FRZ bit is asserted.
27	NOT_RDY	FlexCAN not ready. This read-only bit indicates that FlexCAN is either disabled, in doze mode, in stop mode, or in freeze mode. It is cleared once FlexCAN has exited these modes. 0 FlexCAN module is either in normal mode, listen-only mode, or loop-back mode 1 FlexCAN module is either disabled, in doze mode, stop mode, or freeze mode
26	WAK_MSK	Wake-up interrupt mask. This bit enables the wake-up interrupt generation [1]. 0 Wake-up interrupt is disabled 1 Wake-up interrupt is enabled
25	SOFT_RST	Soft reset. When this bit is set, FlexCAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER, ECR, ESR, IMASK, IFLAG. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: CTRL, RXGMASK, RX14MASK, RX15MASK, and all message buffers. The SOFT_RST bit can be set directly by the CPU when it writes to the MCR, but it is also set when global soft reset is requested at MCU level. Since soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFT_RST bit remains set while reset is pending, and is automatically cleared when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in any of the low power modes. The module should be first removed from low power mode, and then soft reset can be applied. 0 No reset request 1 Resets the registers marked as “affected by soft reset” in Table 23-2

Table 23-7. MCR Field Descriptions (continued)

Bits	Name	Description
24	FRZ_ACK	Freeze mode acknowledge. This read-only bit indicates that FlexCAN is in freeze mode and its prescaler is stopped. The freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZ_ACK bit to know when FlexCAN has actually entered freeze mode. If freeze mode request is cleared, then this bit is cleared once the FlexCAN prescaler is running again. If freeze mode is requested while FlexCAN is in any of the low power modes, then the FRZ_ACK bit will only be set when the low power mode is exited. See Section 23.6.8.1, “Freeze Mode,” for more information. 0 FlexCAN not in freeze mode, prescaler running 1 FlexCAN in freeze mode, prescaler stopped
23	SUPV	Supervisor mode. This bit configures some of the FlexCAN registers to be either in supervisor or unrestricted memory space. The registers affected by this bit are marked as S/U in the Access Type column of Table 23-2. Reset value of this bit is ‘1’, so the affected registers start with supervisor access restrictions. 0 Affected registers are in unrestricted memory space 1 Affected registers are in supervisor memory space. Any access without supervisor permission behaves as though the access was done to an unimplemented register location
22	SLF_WAK	Self wake-up. This bit enables the self wake-up feature when FlexCAN is in doze mode or stop mode. If this bit had been asserted by the time FlexCAN entered doze mode or stop mode, then FlexCAN will look for a recessive to dominant transition on the bus during these modes. If a transition from recessive to dominant is detected during doze mode, FlexCAN resumes its clocks and, if enabled to do so, generates a wake-up interrupt to the CPU. If a transition from recessive to dominant is detected during stop mode, then FlexCAN generates, if enabled to do so, a wake-up interrupt to the CPU so that it can resume the clocks globally. This bit cannot be written while the module is in doze mode or stop mode. 0 FlexCAN self wake-up feature is disabled 1 FlexCAN self wake-up feature is enabled
21	—	
20	LPM_ACK	Low power mode acknowledge. This read-only bit indicates that FlexCAN is either disabled, in doze mode, or in stop mode. Either of these low power modes cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPM_ACK bit to know when FlexCAN has actually entered low power mode. See Section 23.6.8.2, “Module Disabled Mode,” Section 23.6.8.3, “Doze Mode,” and Section 23.6.8.4, “Stop Mode,” for more information. 0 FlexCAN not in any of the low power modes 1 FlexCAN is either disabled, in doze mode or in stop mode
19	WAK_SRC	Wake-up source. This bit defines whether the integrated low-pass filter is applied to protect the Rx CAN input from spurious wake-up. See Section 23.6.8.3, “Doze Mode,” and Section 23.6.8.4, “Stop Mode,” for more information. 0 FlexCAN uses the unfiltered Rx input to detect recessive to dominant edges on the CAN bus. 1 FlexCAN uses the filtered Rx input to detect recessive to dominant edges on the CAN bus.

Table 23-7. MCR Field Descriptions (continued)

Bits	Name	Description
18	DOZE	Doze mode enable. This bit defines whether FlexCAN is allowed to enter low power mode when doze mode is requested at MCU level. This bit is automatically reset when FlexCAN wakes up from doze mode upon detecting activity on the CAN bus (self wake-up enabled). 0 FlexCAN is not enabled to enter low power mode when doze mode is requested 1 FlexCAN is enabled to enter low power mode when doze mode is requested
17–6	—	Reserved.
5–0	MAXMB[5:0]	Maximum number of message buffers. This 6-bit field defines the maximum number of message buffers that will take part in the matching and arbitration process. The reset value (0x0F) is equivalent to 16 message buffer (MB) configuration. This field should be changed only while the module is in freeze mode. Maximum MBs in Use = MAXMB + 1 ***Eqn. 23-1*** **Note:** MAXMB must be programmed with a value ≤ 32, otherwise FlexCAN will not transmit or receive frames.

1 On L49P mask set devices, the FlexCAN wake-up request is not handled via the INTC module, therefore no interrupt exception takes place (execution continues inline where the MCU entered the selected low-power mode in response to the wake-up). On later mask sets, the wake-up interrupt shares the vector assigned to the channel error interrupt (see Table 6-2 on page 6-85).

23.5.2.2 FlexCAN Control Register (CTRL)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, loop back mode, listen-only mode, bus-off recovery behavior, and interrupt enabling (e.g., bus-off, error). It also determines the Division Factor for the clock prescaler. Most of the fields in this register should only be changed while the module is disabled or in freeze mode. Exceptions are the BOFF_MSK, ERR_MSK and BOFF_REC bits, that can be accessed at any time.

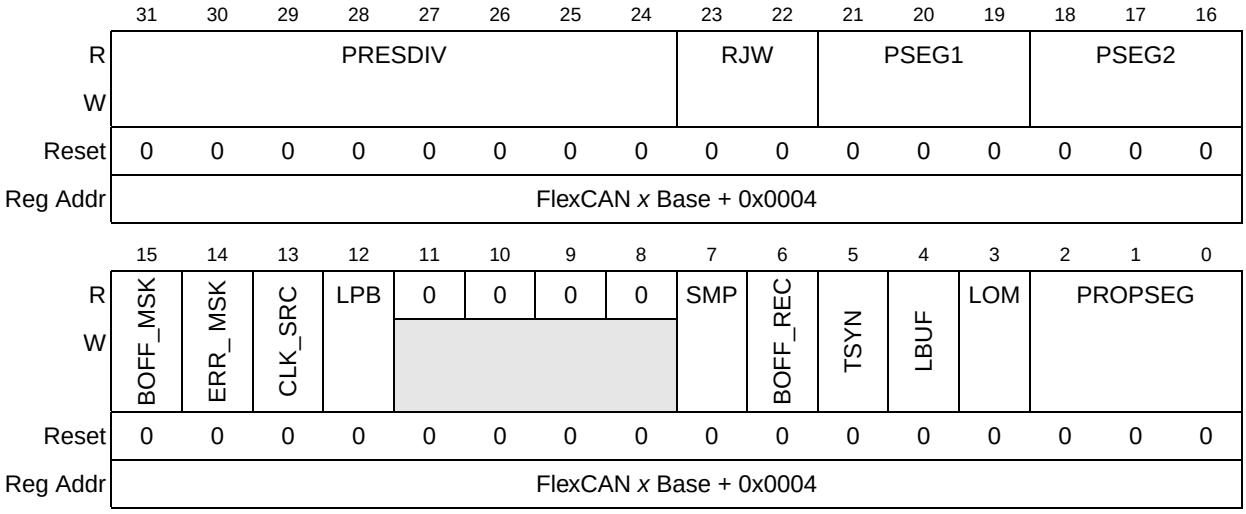


Figure 23-4. FlexCAN Control Register (CTRL)

Table 23-8. CTRL Field Descriptions

Bits	Name	Description
31–24	PRESDIV [7:0]	Prescaler division factor. This 8-bit field defines the ratio between the Can Protocol Interface (CPI) clock frequency and the Serial Clock (SCK) frequency. The S clock period defines the time quantum of the CAN protocol. For the reset value, the S clock frequency is equal to the CPI clock frequency. The Maximum value of this register is 0xFF, that gives a minimum S clock frequency equal to the CPI clock frequency divided by 256. For more information refer to Section 23.6.7.4, “Protocol Timing.” $SCK = \frac{\text{CPI clock frequency}}{\text{PRESDIV} + 1}$ ***Eqn. 23-2***
23–22	RJW[1:0]	Resync jump width. This 2-bit field defines the maximum number of time quanta (one time quantum is equal to the S clock period) that a bit time can be changed by one re-synchronization. The valid programmable values are 0–3. Resync jump width = RJW + 1 ***Eqn. 23-3***
21–19	PSEG1[2:0]	Phase segment 1. This 3-bit field defines the length of Phase Buffer Segment 1 in the bit time. The valid programmable values are 0–7. Phase buffer segment 1 = (PSEG1 + 1) × time-quanta ***Eqn. 23-4***
18–16	PSEG2[2:0]	Phase segment 2. This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmable values are 1–7. Phase buffer segment 2 = (PSEG2 + 1) × time-quanta ***Eqn. 23-5***
15	BOFF_MSK	Bus off mask. This bit provides a mask for the Bus Off Interrupt. 0 Bus Off interrupt disabled 1 Bus Off interrupt enabled
14	ERR_MSK	Error mask. This bit provides a mask for the error Interrupt. 0 Error interrupt disabled 1 Error interrupt enabled
13	CLK_SRC	CAN engine clock source. This bit selects the clock source to the CAN Protocol Interface (CPI) to be either the peripheral clock (driven by the PLL) or the crystal oscillator clock. The selected clock is the one fed to the prescaler to generate the Serial Clock (S clock). In order to guarantee reliable operation, this bit should only be changed while the module is disabled. See Section 23.6.7.4, “Protocol Timing,” for more information. 0 The CAN engine clock source is the oscillator clock 1 The CAN engine clock source is the bus clock
12	LPB	Loop back. This bit configures FlexCAN to operate in loop-back mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic ‘1’). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. 0 Loop back disabled 1 Loop back enabled

Table 23-8. CTRL Field Descriptions (continued)

Bits	Name	Description
11–8	—	Reserved.
7	SMP	Sampling mode. This bit defines the sampling mode of CAN bits at the Rx Input. 0 Just one sample is used to determine the Rx bit value 1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples, a majority rule is used
6	BOFF_REC	Bus off recovery mode. This bit defines how FlexCAN recovers from bus-off state. If this bit is cleared, automatic recovering from bus-off state occurs according to the CAN Specification 2.0B. If the bit is set, automatic recovering from Bus Off is disabled and the module remains in bus-off state until the bit is cleared by the user. If the bit is cleared before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFF_REC bit had never been set. If the bit is cleared after 128 sequences of 11 recessive bits occurred, then FlexCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFF_REC bit can be set again during Bus Off, but it will only be effective the next time the module enters Bus Off. If BOFF_REC was cleared when the module entered Bus Off, setting it during Bus Off will not be effective for the current Bus Off recovery. 0 Automatic recovering from bus off-state enabled, according to CAN Spec 2.0 part B 1 Automatic recovering from bus-off state disabled
5	TSYN	Timer sync mode. This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special “SYNC” message (i.e., global network time). 0 Timer Sync feature disabled 1 Timer Sync feature enabled
4	LBUF	Lowest buffer transmitted first. This bit defines the ordering mechanism for message buffer transmission. 0 Buffer with lowest ID is transmitted first 1 Lowest number buffer is transmitted first
3	LOM	Listen-only mode. This bit configures FlexCAN to operate in listen-only mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN error passive mode [Ref. 1]. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message. 0 FlexCAN module is in normal active operation, listen-only mode is deactivated 1 FlexCAN module is in listen-only mode operation
2–0	PROPSEG [2:0]	Propagation segment. This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7. Propagation segment time = (PROPSEG + 1) × time-quanta ***Eqn. 23-6*** **Note:** A time-quantum = one S clock period.

23.5.2.3 FlexCAN Timer Register (TIMER)

This register represents a 16-bit free running counter that can be read and written by the CPU. [1] The timer starts from 0x0000 after Reset, counts linearly to 0xFFFF, and wraps around.

The timer is clocked by the FlexCAN bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. During freeze mode, the timer is not incremented.

The timer value is captured at the beginning of the identifier (ID) field of any frame on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

Writing to the timer is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

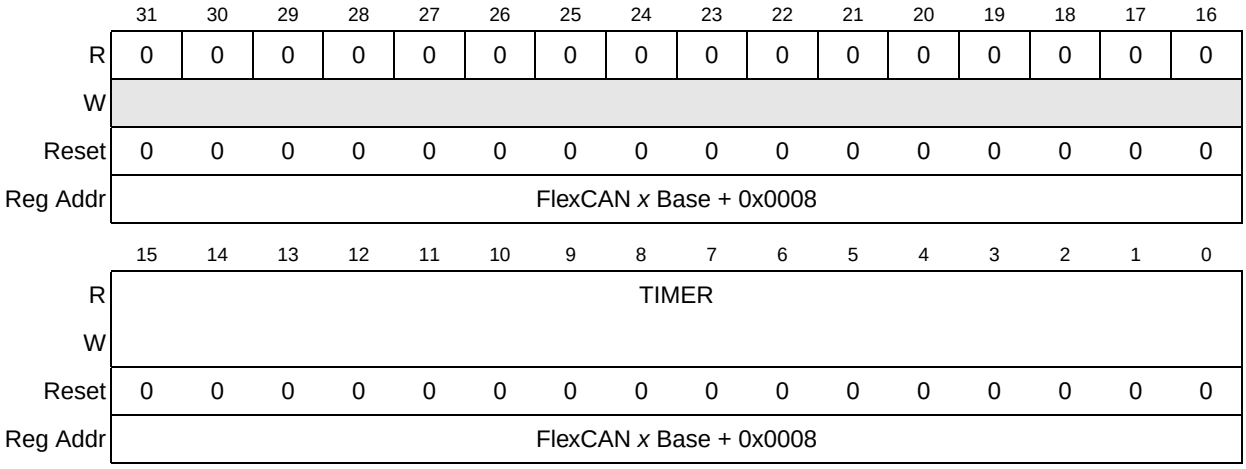


Figure 23-5. FlexCAN Timer (TIMER)

23.5.2.4 Rx Mask Registers

These registers are used as acceptance masks for a received frame ID. Three masks are defined: a global mask, used for Rx buffers 0–13 and 16–63, and two extra masks dedicated for buffers 14 and 15. The meaning of each mask bit is the following:

- Mask bit = 0: the corresponding incoming ID bit is “don’t care”
- Mask bit = 1: the corresponding ID bit is checked against the incoming ID bit, to see if a match exists

Note that these masks are used both for Standard and Extended ID formats. The value of mask registers should not be changed while in normal operation, as locked frames that matched a message buffer (MB) through a mask may be transferred into the MB (upon release) but may no longer match. Table 23-9 shows some examples of ID masking for Standard and Extended message buffers.

1. On mask set L49P devices, it is not possible to read the timer reliably when not in Freeze Mode.

Table 23-9. Mask examples for Standard/Extended Message Buffers

	Base ID ID28.....ID18	IDE	Extended ID ID17.....ID0	Match
MB2 ID	1 1 1 1 1 1 1 1 0 0 0	0		
MB3 ID	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	
MB4 ID	0 0 0 0 0 0 0 1 1 1 1	0		
MB5 ID	0 0 0 0 0 0 0 1 1 1 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	
MB14 ID	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	
Rx Global Mask	1 1 1 1 1 1 1 1 1 1 0		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1	
Rx Msg in [1]	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	3
Rx Msg in [2]	1 1 1 1 1 1 1 1 0 0 1	0		2
Rx Msg in [3]	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0	
Rx Msg in [4]	0 1 1 1 1 1 1 1 0 0 0	0		
Rx Msg in [5]	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	14
Rx 14 Mask	0 1 1 1 1 1 1 1 1 1 1		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0	
Rx Msg in [6]	1 0 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	
Rx Msg in [7]	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	14

1 Match for Extended Format (MB3).

2 Match for Standard Format. (MB2).

3 Mismatch for MB3 because of ID0.

4 Mismatch for MB2 because of ID28.

5 Mismatch for MB3 because of ID28, Match for MB14 (uses Rx_14_Mask).

6 Mismatch for MB14 because of ID27 (uses Rx_14_Mask).

7 Match for MB14 (uses Rx_14_Mask).

23.5.2.4.1 FlexCAN Rx Global Mask Register (RXGMASK)

The Rx Global Mask bits are applied to all Rx Identifiers excluding Rx buffers 14–15, that have their specific Rx mask registers. Access to this register is unrestricted.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	MI[28:16]											MI[17:16]	
W																
Reset	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
Reg Addr	FlexCAN x Base + 0x0010															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MI[15:0]															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reg Addr	FlexCAN x Base + 0x0010															

Figure 23-6. FlexCAN Rx Global Mask Register (RXGMASK)

Table 23-10. RXGMASK Field Descriptions

Bits	Name	Description
31–29	—	Reserved.
28–18	MI[28:18]	Standard ID mask bits. The same mask bits for the Standard and Extended Formats.
17–0	MI[17:0]	Extended ID mask bits. Used to mask comparison only in Extended Format.

23.5.2.4.2 FlexCAN Rx 14 Mask Register (RX14MASK)

The RX14MASK register has the same structure as the Rx global mask register and is used to mask message buffer 14. Access to this register is unrestricted.

- Address Offset: 0x14
- Reset Value: 0x1FFF_FFFF

23.5.2.4.3 FlexCAN Rx 15 Mask Register (RX15MASK)

The RX15MASK register has the same structure as the Rx global mask register and is used to mask message buffer 15. Access to this register is unrestricted.

- Address Offset: 0x18
- Reset Value: 0x1FFF_FFFF

23.5.2.5 FlexCAN Error Counter Register (ECR)

This register has 2 8-bit fields reflecting the value of two FlexCAN error counters: transmit error counter (TX_ERR_COUNTER) and receive error counter (RX_ERR_COUNTER). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module. Both counters are read-only except in freeze mode, [1] where they can be written by the CPU.

Writing to the error counter register while in freeze mode is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

FlexCAN responds to any bus state as described in the protocol, e.g. transmit error-active or error-passive flag, delay its transmission start time (error-passive), and avoid any influence on the bus when in bus-off state. The following are the basic rules for FlexCAN bus state transitions.

- If the value of TX_ERR_COUNTER or RX_ERR_COUNTER increases to be greater than or equal to 128, the FLT_CONF field in the error and status register is updated to reflect error-passive state.

1. On mask set L49P devices, it is not possible to read the counter reliably when not in Freeze Mode.

- If the FlexCAN state is error-passive, and either TX_ERR_COUNTER or RX_ERR_COUNTER decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLT_CONF field in the error and status register is updated to reflect error-active state.
- If the value of TX_ERR_COUNTER increases to be greater than 255, the FLT_CONF field in the error and status register is updated to reflect bus-off state, and an interrupt may be issued. The value of TX_ERR_COUNTER is then reset to zero.
- If FlexCAN is in bus-off state, then TX_ERR_COUNTER is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, TX_ERR_COUNTER is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the TX_ERR_COUNTER. When TX_ERR_COUNTER reaches the value of 128, the FLT_CONF field in the error and status register is updated to be error-active and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TX_ERR_COUNTER value.
- If during system start-up, only one node is operating, then its TX_ERR_COUNTER increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACK_ERR bit in the error and status register). After the transition to error-passive state, the TX_ERR_COUNTER does not increment anymore by acknowledge errors. Therefore the device never goes to the bus-off state.
- If the RX_ERR_COUNTER increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to error-active state.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	FlexCAN x Base + 0x001C															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RX_ERR_COUNTER								TX_ERR_COUNTER							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	FlexCAN x Base + 0x001C															

Figure 23-7. FlexCAN Error Counter Register (ECR)

23.5.2.6 FlexCAN Error and Status Register (ESR)

This register reflects various error conditions, some general status of the device, and it is the source of three interrupts to the CPU. The reported error conditions (bits 15–10) are those that have occurred since the last time the CPU read this register. The CPU read action clears bits 15–10. Bits 9–3 are status bits.

Most bits in this register are read-only, except BOFF_INT, WAK_INT and ERR_INT, that are interrupt flags that can be cleared by writing ‘1’ to them (writing ‘0’ has no effect). See Section 23.6.9, “Interrupts,” for more details.

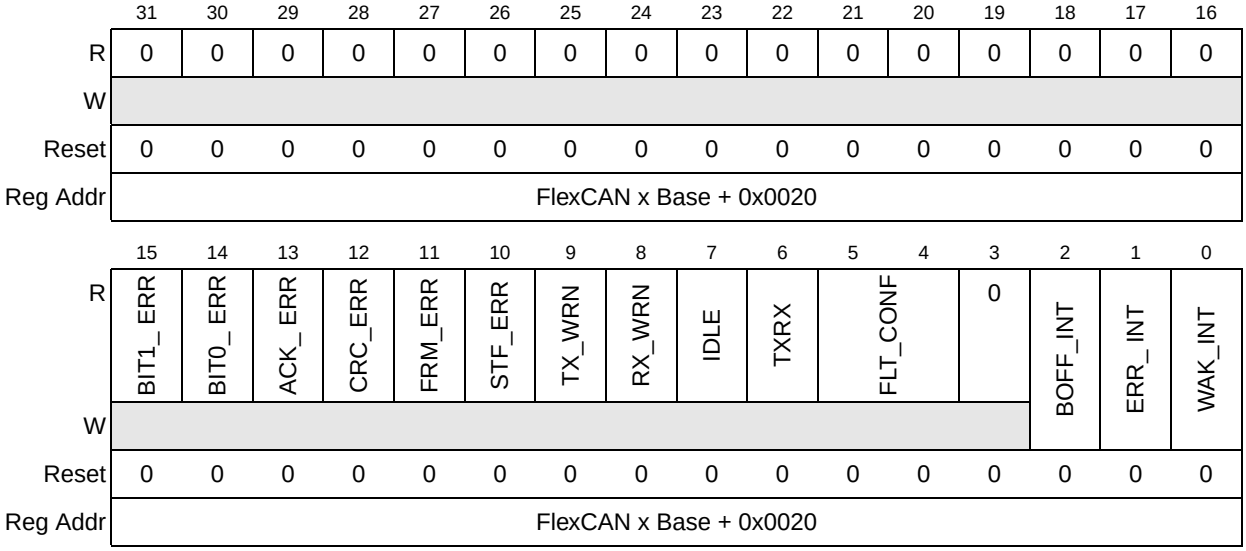


Figure 23-8. FlexCAN Error and Status Register (ESR)

Table 23-11. ESR Field Descriptions

Bits	Name	Description
31–16	—	Reserved.
15	BIT1_ERR	Bit 1 error. This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 0 No such occurrence 1 At least one bit sent as recessive is received as dominant **Note:** This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits.
14	BIT0_ERR	Bit 0 error. This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 0 No such occurrence 1 At least one bit sent as dominant is received as recessive
13	ACK_ERR	Acknowledge error. This bit indicates that an acknowledge error has been detected by the transmitter node, i.e., a dominant bit has not been detected during the ACK SLOT. 0 No such occurrence 1 An ACK error occurred since last read of this register
12	CRC_ERR	Cyclic redundancy code error. This bit indicates that a CRC error has been detected by the receiver node, i.e., the calculated CRC is different from the received. 0 No such occurrence 1 A CRC error occurred since last read of this register.

Table 23-11. ESR Field Descriptions (continued)

Bits	Name	Description
11	FRM_ERR	Form error. This bit indicates that a form error has been detected by the receiver node, i.e., a fixed-form bit field contains at least one illegal bit. 0 No such occurrence 1 A form error occurred since last read of this register
10	STF_ERR	Stuffing error. This bit indicates that a stuffing error has been detected. 0 No such occurrence. 1 A stuffing error occurred since last read of this register.
9	TX_WRN	TX error counter. This bit indicates that repetitive errors are occurring during message transmission. 0 No such occurrence 1 TX_ERR_COUNTER ≥ 96
8	RX_WRN	Rx error counter. This bit indicates when repetitive errors are occurring during message reception. 0 No such occurrence 1 RX_ERR_COUNTER ≥ 96
7	IDLE	CAN bus IDLE state. This bit indicates when CAN bus is in IDLE state. 0 No such occurrence 1 CAN bus is now IDLE
6	TXRX	Current FlexCAN status (transmitting/receiving). This bit indicates if FlexCAN is transmitting or receiving a message when the CAN bus is not in IDLE state. This bit has no meaning when IDLE is asserted. 0 FlexCAN is receiving a message (IDLE=0) 1 FlexCAN is transmitting a message (IDLE=0)
5–4	FLT_CONF [1:0]	Fault confinement state. This 2-bit field indicates the confinement state of the FlexCAN module, as shown below. If the LOM bit in the control register is asserted, the FLT_CONF field will indicate error-passive. Since the control register is not affected by soft reset, the FLT_CONF field will not be affected by soft reset if the LOM bit is asserted. 00 Error active 01 Error passive 1x Bus off
3	—	Reserved.
2	BOFF_INT	Bus Off Interrupt. This bit is set when FlexCAN enters bus-off state. If the corresponding mask bit in the control register (BOFF_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing a ‘1’ to it. Writing ‘0’ has no effect. 0 No such occurrence 1 FlexCAN module entered bus-off state

Table 23-11. ESR Field Descriptions (continued)

Bits	Name	Description
1	ERR_INT	Error interrupt. This bit indicates that at least one of the error bits (bits 15-10) is set. If the corresponding mask bit in the control register (ERR_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing a ’1’ to it. Writing ’0’ has no effect. 0 No such occurrence 1 Indicates setting of any error bit in the ESR
0	WAK_INT	Wake-up interrupt. When FlexCAN is in doze mode or stop mode and a recessive to dominant transition is detected on the CAN bus and if the WAK_MSK bit in the MCR is set, an interrupt is generated to the CPU. This bit is cleared by writing a ’1’ to it. Writing ’0’ has no effect. 0 No such occurrence 1 Indicates a recessive to dominant transition received on the CAN bus when the FlexCAN module is in doze mode or stop mode

23.5.2.7 FlexCAN Interrupt Mask Register (IMASK)

This register allows to enable or disable any number of a range of 32 message buffer Interrupts. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (i.e., when the corresponding IFLAG bit is set).

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BUF31M	BUF30M	BUF29M	BUF28M	BUF27M	BUF26M	BUF25M	BUF24M	BUF23M	BUF22M	BUF21M	BUF20M	BUF19M	BUF18M	BUF17M	BUF16M
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	FlexCAN x Base + 0x0028															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF15M	BUF14M	BUF13M	BUF12M	BUF11M	BUF10M	BUF9M	BUF8M	BUF7M	BUF6M	BUF5M	BUF4M	BUF3M	BUF2M	BUF1M	BUF0M
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	FlexCAN x Base + 0x0028															

Figure 23-9. FlexCAN Interrupt Mask Register (IMASK)

Table 23-12. IMASK Field Descriptions

Bits	Name	Description
31–0	BUF*n*M	Buffer MB Mask. Each bit enables or disables the respective FlexCAN message buffer (MB0 to MB31) Interrupt. 0 The corresponding buffer Interrupt is disabled 1 The corresponding buffer Interrupt is enabled **Note:** Setting or clearing a bit in IMASK can assert or negate an interrupt request, if the corresponding IFLAG bit is set.

23.5.2.8 FlexCAN Interrupt Flags Register (IFLAG)

This register defines the flags for 32 message buffer interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG bit. If the corresponding IMASK bit is set, an interrupt will be generated. The Interrupt flag must be cleared by writing a ’1’ to it. Writing ‘0’ has no effect.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BUF31I	BUF30I	BUF29I	BUF28I	BUF27I	BUF26I	BUF25I	BUF24I	BUF23I	BUF22I	BUF21I	BUF20I	BUF19I	BUF18I	BUF17I	BUF16I
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	FlexCAN *x* Base + **0x0030**															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF15I	BUF14I	BUF13I	BUF12I	BUF11I	BUF10I	BUF9I	BUF8I	BUF7I	BUF6I	BUF5I	BUF4I	BUF3I	BUF2I	BUF1I	BUF0I
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	FlexCAN x Base + 0x0030															

Figure 23-10. FlexCAN Interrupt Flags Register (IFLAG)

Table 23-13. IFLAG Field Descriptions

Bits	Name	Description
31–0	BUF*n*I	Buffer MB interrupt. Each bit flags the respective FlexCAN message buffer (MB0 to MB31) interrupt. 0 No such occurrence 1 The corresponding buffer has successfully completed transmission or reception

23.6 Functional Description

23.6.1 Overview

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed of a set of up to 32 message buffers (MB) that store configuration and control data, time stamp, message ID and data (see Section 23.5.1, “Message Buffer Structure”). Any MB can work as a transmission or reception buffer. An arbitration algorithm decides the prioritization of MBs to be transmitted based on either the message ID or the MB ordering. A matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. Data coherency mechanisms are implemented to guarantee data integrity during MB manipulation by the CPU.

Before proceeding with the functional description, an important concept must be explained. A message buffer is said to be “active” at a given time if it can participate in the matching and arbitration algorithms

that are happening at that time. An Rx MB with a 0b0000 code is inactive (refer to Table 23-5). Similarly, a Tx MB with a 0b1000 code is inactive (refer to Table 23-6). A MB not programmed with either 0b0000 or 0b1000 will be temporarily deactivated (will not participate in the current arbitration/matching run) when the CPU writes to the C/S field of that MB (see Section 23.6.6, “Data Coherence”).

23.6.2 Transmit Process

In order to transmit a CAN frame, the CPU should prepare a message buffer for transmission by executing the following procedure:

- Write 0b1000 to the CODE field of the control and status word to keep the MB inactive
- Write the ID word
- Write the data bytes
- Write the LENGTH, control, and CODE fields of the control and status word to activate the MB

The first and last steps are mandatory. The first write to the control and status word is important in case there was pending reception or transmission. The write operation immediately deactivates the MB, removing it from any currently ongoing arbitration or ID matching processes, giving time for the CPU to program the rest of the MB (see Section 23.6.6.1, “Message Buffer Deactivation”). Once the MB is activated in the fourth step, it will participate in the arbitration process and eventually be transmitted according to its priority. At the end of the successful transmission, the value of the free running timer is written into the TIME STAMP field, the CODE field in the control and status word is updated, a status flag is set in the interrupt flag register and an interrupt is generated if allowed by the corresponding interrupt mask register bit. The new CODE field after transmission depends on the code that was used to activate the MB in step four (see Table 23-5 and Table 23-6 in Section 23.5.1, “Message Buffer Structure”).

23.6.3 Arbitration Process

The arbitration process is an algorithm executed by the message buffer management (MBM) that scans the whole MB memory looking for the highest priority message to be transmitted. All MBs programmed as transmit buffers will be scanned to find the lowest ID [1] or the lowest MB number, depending on the LBUF bit on the control register. The arbitration process is triggered in the following events:

- During the CRC field of the CAN frame
- During the error delimiter field of the CAN frame
- During Intermission, if the winner MB defined in a previous arbitration was deactivated, or if there was no MB to transmit, but the CPU wrote to the C/S word of any MB after the previous arbitration finished
- When MBM is in idle or bus-off state and the CPU writes to the C/S word of any MB
- Upon leaving freeze mode

Once the highest priority MB is selected, it is transferred to a temporary storage space called Serial Message Buffer (SMB), which has the same structure as a normal MB but is not user accessible. This operation is called “move-out”. At the first opportunity window on the CAN bus, the message on the SMB

1. Actually, if LBUF is negated, the arbitration considers not only the ID, but also the RTR and IDE bits placed inside the ID at the same positions they are transmitted in the CAN frame.

is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the DLC (data length code) value is bigger.

23.6.4 Receive Process

To be able to receive CAN frames, the CPU must prepare one or more message buffers for reception by executing the following steps:

- Write 0b0000 to the control and status (C/S) word CODE field to keep the MB inactive
- Write the ID word
- Write 0b0100 to the CODE field of the C/S word to activate the MB

The first and last steps are mandatory. The first write to the C/S word is important in case there was a pending reception or transmission. The write operation immediately deactivates the MB, removing it from any currently ongoing arbitration or matching process, giving time for the CPU to program the rest of the MB (see Section 23.6.6.1, “Message Buffer Deactivation”). Once the MB is activated in the third step, it will be able to receive CAN frames that match the programmed ID. At the end of a successful reception, the MB is updated by the MBM as follows:

- The value of the free-running timer is written into the TIME STAMP field
- The received ID, data (8 bytes at most), and LENGTH fields are stored
- The CODE field in the C/S word is updated (see Table 23-5 and Table 23-6 in Section 23.5.1, “Message Buffer Structure”)
- A status flag is set in the interrupt flag register and an interrupt is generated if allowed by the corresponding interrupt mask register bit

Upon receiving the MB interrupt, the CPU should service the received frame using the following procedure:

- Read the control and status word (mandatory – activates an internal lock for this buffer)
- Read the ID field (optional – needed only if a mask was used)
- Read the DATA field
- Read the free running timer (optional – releases the internal lock)

Upon reading the control and status word, if the BUSY bit is set in the CODE field, then the CPU should defer the access to the MB until this bit is negated. Reading the free running timer is not mandatory. If not executed the MB remains locked, unless the CPU reads the C/S word of another MB. Note that only a single MB is locked at a time. The only mandatory CPU read operation is the one on the control and status word to assure data coherency (see Section 23.6.6, “Data Coherence”).

The CPU should synchronize to frame reception by the status flag bit for the specific MB in one of the IFLAG registers and not by the CODE field of that MB. Polling the CODE field does not work because once a frame was received and the CPU services the MB (by reading the C/S word followed by unlocking the MB), the CODE field will not return to EMPTY. It will remain FULL, as explained in Table 23-5. If the CPU tries to work around this behavior by writing to the C/S word to force an EMPTY code after reading the MB, the MB is actually deactivated from any currently ongoing matching process. As a result, a newly received frame matching the ID of that MB may be lost. In summary: never do polling by directly reading the C/S word of the MBs. Instead, read the IFLAG registers.

Note that the received ID field is always stored in the matching MB, thus the contents of the ID field in an MB may change if the match was due to masking. Also note that FlexCAN does receive frames transmitted by itself if there exists an Rx matching MB.

23.6.5 Matching Process

The matching process is an algorithm executed by the MBM that scans the whole MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. Only MBs programmed to receive will participate in the matching process for received frames.

While the ID, DLC and DATA fields are retrieved from the CAN bus, they are stored temporarily in the Serial Message Buffer (SMB). The matching process takes place during the CRC field. If a matching ID is found in one of the MBs, the contents of the SMB will be transferred to the matched MB during the 6th bit of the end-of-frame field of the CAN protocol. This operation is called “move-in”. If any protocol error (CRC, ACK, etc.) is detected, then the move-in operation does not happen.

An MB with a matching ID is said to be “free to receive” a new frame if the following conditions are satisfied:

- The MB is not locked (see Section 23.6.6.2, “Message Buffer Lock Mechanism”)
- The CODE field is either EMPTY or else it is FULL or OVERRUN, but the CPU has already serviced the MB (read the C/S word and then unlocked the MB)

If the matching algorithm finds an MB with a matching ID that is not “free to receive” the new frame, then FlexCAN will overwrite the matching MB (unless it is locked) and set the CODE field to OVERRUN (refer to Table 23-5 and Table 23-6). If the matching MB is locked, then the new message remains in the SMB, waiting for the MB to be unlocked (see Section 23.6.6.2, “Message Buffer Lock Mechanism”).

Matching to a range of IDs is possible by using ID Acceptance Masks. During the matching algorithm, if a mask bit is set, then the corresponding ID bit is compared. If the mask bit is cleared, the corresponding ID bit is “don’t care”. Please refer to Section 23.5.2.4, “Rx Mask Registers,” for more information.

23.6.6 Data Coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in Section 23.6.2, “Transmit Process,” and Section 23.6.4, “Receive Process.” Any form of CPU accessing a MB structure within FlexCAN other than those specified may cause FlexCAN to behave in an unpredictable way.

23.6.6.1 Message Buffer Deactivation

If the CPU wants to change the function of an active MB, the recommended procedure is to first put the module into freeze mode and then change the CODE field of that MB. This is a safe procedure because FlexCAN waits for pending CAN bus and MB moving activities to finish before entering freeze mode. Nevertheless, a mechanism is provided to maintain data coherence when the CPU writes to the C/S word of active MBs out of freeze mode.

Any CPU write access to the C/S word of an MB causes that MB to be excluded from the transmit or receive processes during the current matching or arbitration round. This mechanism is called MB deactivation. It is temporary, affecting only for the current match/arbitration round.

The purpose of deactivation is data coherency. The match/arbitration process scans the MBs to decide which MB to transmit or receive. If the CPU updates the MB in the middle of a match or arbitration process, the data of that MB may no longer be coherent, therefore deactivation of that MB is done.

Even with the coherence mechanism described above, writing to the control and status word of active MBs when not in freeze mode may produce undesirable results. Examples are:

- Matching and arbitration are one-pass processes. If MBs are deactivated after they are scanned, no re-evaluation is done to determine a new match/winner. If an Rx MB with a matching ID is deactivated during the matching process after it was scanned, then this MB is marked as invalid to receive the frame, and FlexCAN will keep looking for another matching MB within the ones it has not scanned yet. If it can not find one, then the message will be lost. Suppose, for example, that two MBs have a matching ID to a received frame, and the user deactivated the first matching MB after FlexCAN has scanned the second. The received frame will be lost even if the second matching MB was “free to receive”.
- If a Tx MB containing the lowest ID is deactivated after FlexCAN has scanned it, then FlexCAN will look for another winner within the MBs that it has not scanned yet. Therefore, it may transmit an MB with ID that may not be the lowest at the time because a lower ID might be present in one of the MBs that it had already scanned before the deactivation.
- There is a point in time until which the deactivation of a Tx MB causes it not to be transmitted (end of move-out). After this point, it is transmitted but no interrupt is issued and the CODE field is not updated.

23.6.6.2 Message Buffer Lock Mechanism

Besides MB deactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the control and status word of an “active not empty” Rx MB, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and thus it sets an internal lock flag for that MB. The lock is released when the CPU reads the free-running timer [1] (global unlock operation), or when it reads the control and status word of another MB. The MB locking is done to prevent a new frame to be written into the MB while the CPU is reading it.

Suppose, for example, that FlexCAN has already received and stored a message into one of the MBs. Suppose now that the CPU decides to read that MB at the same time another message with an ID that matches that MB is arriving. When the CPU reads the control and status word, the MB is locked. The new message arrives and the matching algorithm finds out that the matching MB is not “free to receive” because it is locked, so the new message will remain in the SMB waiting for the MB to be unlocked. If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the CODE field of the MB or in the error and status register.

1. On mask set L49P devices, it is not possible to read the timer reliably when not in Freeze Mode.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the CODE field is asserted. If the CPU reads the control and status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

NOTE

If the BUSY bit is asserted or if the MB is empty, then reading the control and status word does not lock the MB.

Deactivation takes precedence over locking. If the CPU deactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the MB.

23.6.7 CAN Protocol Related Features

23.6.7.1 Remote Frames

A remote frame is a special kind of frame. The user can program an MB to be a request remote frame by writing the MB as Transmit with the RTR bit set. After the remote request frame is transmitted successfully, the MB becomes a Receive message buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, its ID is compared to the IDs of the transmit message buffers with the CODE field 0b1010. If there is a matching ID, then this MB frame will be transmitted. Note that if the matching MB has the remote transmission request (RTR) bit set, then FlexCAN will transmit a remote frame as a response.

A received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match.

In the case that a remote request frame was received and matched a MB, this message buffer immediately enters the internal arbitration process, but is considered as a normal Tx MB, with no higher priority. The data length of this frame is independent of the data length code (DLC) field in the remote frame that initiated its transmission.

23.6.7.2 Overload Frames

FlexCAN transmits overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of intermission
- Detection of a dominant bit at the 7th bit (last) of end-of-frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of error frame delimiter or overload frame delimiter

23.6.7.3 Time Stamp

The value of the free-running timer is sampled at the beginning of the identifier (ID) field on the CAN bus, and is stored at the end of “move-in” in the TIME STAMP field, providing network behavior with respect to time.

Note that the free-running timer can be reset upon a specific frame reception, enabling network time synchronization. Refer to TSYN description in Section 23.5.2.2, “FlexCAN Control Register (CTRL).”

23.6.7.4 Protocol Timing

Figure 23-11 shows the structure of the clock generation circuitry that feeds the CAN Protocol Interface (CPI) sub-module. The clock source bit (CLK_SRC) in the CTRL register defines whether the internal clock is connected the oscillator clock (OSCCLK) or the Peripheral Bus Clock. In order to guarantee reliable operation, the clock source should be selected while the module is disabled (MDIS set in the module configuration register).

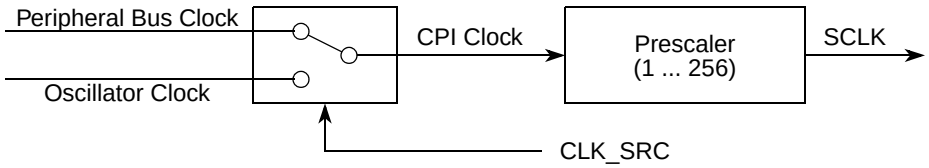


Figure 23-11. CAN Engine Clocking Scheme

The crystal oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The crystal oscillator clock has better jitter performance than PLL generated clocks.

The FlexCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The control register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW. See Section 23.5.2.2, “FlexCAN Control Register (CTRL).”

The PRESDIV field controls a prescaler that generates the Serial Clock (SCLK), whose period defines the ‘time quantum’ used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{(\text{Prescaler Value})} \qquad \textit{Eqn. 23-7}$$

A bit time is subdivided into three segments [1] (reference Figure 23-12 and Table 23-14):

1. SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
2. Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL register so that their sum (plus 2) is in the range of 4 to 16 time quanta
3. Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL register (plus 1) to be 2 to 8 time quanta long

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})} \qquad \textit{Eqn. 23-8}$$

1. For further explanation of the underlying concepts please refer to ISO/DIS 11519–1, Section 10.3. Reference also the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

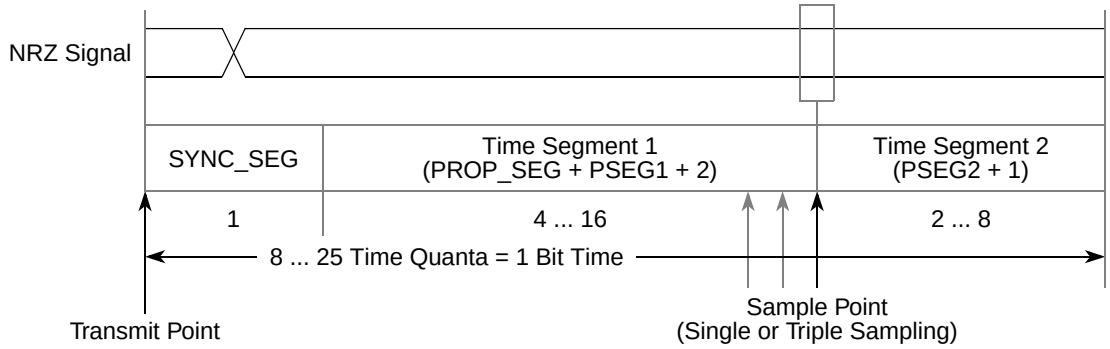


Figure 23-12. Segments Within the Bit Time

Table 23-14. Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Table 23-15 gives an overview of the CAN compliant segment settings and the related parameter values.

NOTE

It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard. For bit time calculations, use an IPT (Information Processing Time) of 2, which is the value implemented in the FlexCAN module.

Table 23-15. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	Time Segment 2	Re-synchronization Jump Width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

23.6.7.5 Arbitration and Matching Timing

During normal transmission or reception of frames, the arbitration, matching, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in Figure 23-13.

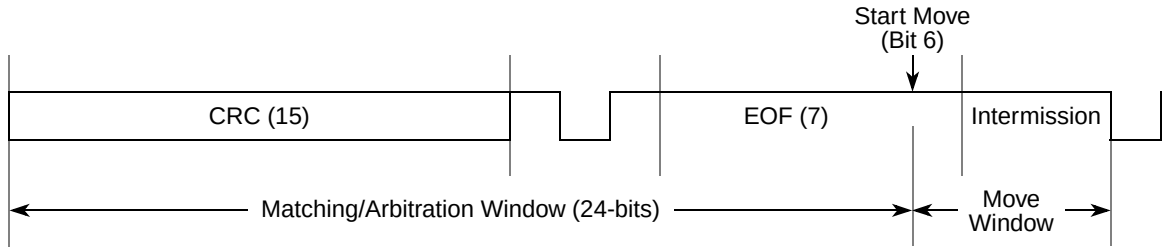


Figure 23-13. Arbitration, Match, and Move Time Windows

When doing matching and arbitration, FlexCAN needs to scan the whole message buffer memory during the available time slot. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in Table 23-15
- The peripheral clock frequency can not be smaller than the oscillator clock frequency, i.e. the PLL can not be programmed to divide down the oscillator clock
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in Table 23-16

Table 23-16. Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate

Number of Message Buffers	Minimum Ratio
16	8
32	8

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, so the oscillator clock frequency should be at least 8 times the CAN bit rate. The minimum frequency ratio specified in Table 23-16 can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRESDIV, PROPSEG, PSEG1, PSEG2). As an example, taking the case of 32 message buffers, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRESDIV + 1) should be at least 2. For the prescaler factor equal to one and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least 2.

23.6.8 FlexCAN Operating Mode Details

23.6.8.1 Freeze Mode

This mode is entered by setting the HALT bit in the MCR or when the MCU is put into debug mode. In both cases it is also necessary that the FRZ bit is asserted in the MCR and the module is not in any of the low power modes (disabled, doze, stop). When freeze mode is requested during transmission or reception, FlexCAN does the following:

1. Waits to be in either intermission, passive error, bus-off or idle state
2. Waits for all internal activities like arbitration, matching, move-in, and move-out to finish

3. Ignores the Rx input pin and drives the Tx pin as recessive
4. Stops the prescaler, thus halting all CAN protocol activities
5. Grants write access to the error counters register, which is read-only in other modes
6. Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting freeze mode, the user must wait for the FRZ_ACK bit to be asserted in the MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In freeze mode, all memory mapped registers are accessible.

Exiting freeze mode is done in one of the following ways:

- CPU negates the FRZ bit in the MCR
- The MCU is removed from debug mode and/or the HALT bit is cleared

Once out of freeze mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

23.6.8.2 Module Disabled Mode

This low power mode is entered when the MDIS bit in the MCR is set. If the module is disabled during freeze mode, it shuts down the clocks to the CPI and MBM sub-modules, sets the LPM_ACK bit and clears the FRZ_ACK bit. If the module is disabled during transmission or reception, FlexCAN does the following:

1. Waits to be in either idle or bus-off state, or else waits for the third bit of Intermission and then checks it to be recessive
2. Waits for all internal activities like arbitration, matching, move-in, and move-out to finish
3. Ignores its Rx input pin and drives its Tx pin as recessive
4. Shuts down the clocks to the CPI and MBM sub-modules
5. Sets the NOT_RDY and LPM_ACK bits in the MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the free-running timer, the error counter register and the message buffers, which cannot be accessed when the module is disabled. Exiting from this mode is done by negating the MDIS bit, which will resume the clocks and negate the LPM_ACK bit.

23.6.8.3 Doze Mode

This is a system low power mode in which the CPU bus is kept alive and a global doze mode request is sent to all peripherals asking them to enter low power mode. When doze mode is globally requested, the DOZE bit in MCR needs to have been set previously for doze mode to be triggered. If doze mode is triggered during freeze mode, FlexCAN shuts down the clocks to the CPI and MBM sub-modules, sets the LPM_ACK bit and clears the FRZ_ACK bit. If doze mode is triggered during transmission or reception, FlexCAN does the following:

1. Waits to be in either idle or bus-off state, or else waits for the third bit of intermission and checks it to be recessive
2. Waits for all internal activities like arbitration, matching, move-in, and move-out to finish

3. Ignores its Rx input pin and drives its Tx pin as recessive
4. Shuts down the clocks to the CPI and MBM sub-modules
5. Sets the NOT_RDY and LPM_ACK bits in MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the free-running timer, the error counter register, and the message buffers, which cannot be accessed in doze mode.

Exiting doze mode is done in one of the following ways:

- CPU removing the doze mode request
- CPU clearing the DOZE bit of the MCR
- Self wake mechanism

In the self wake mechanism, if the SLF_WAK bit in the MCR was set at the time FlexCAN entered doze mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN negates the DOZE bit and resumes its clocks. It also sets the WAK_INT bit in the ESR and, if enabled by the WAK_MSK bit in the MCR, generates a wake-up interrupt to the CPU. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up. Table 23-17 details the effect of SLF_WAK and WAK_MSK upon wake-up from doze mode.

Table 23-17. FlexCAN Wake-up from Doze Mode

SLF_WAK	WAK_MSK	FlexCAN Clocks Enabled	Wake-up Interrupt Generated
0	0	No	No
0	1	No	No
1	0	Yes	No
1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line while in doze mode. See the WAK_SRC bit in Section 23.5.2.1, “FlexCAN Module Configuration Register (MCR).” This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments.

23.6.8.4 Stop Mode

This is a system low power mode in which all MCU clocks are stopped for maximum power savings. If FlexCAN receives the global stop mode request during freeze mode, it sets the LPM_ACK bit, clears the FRZ_ACK bit and then sends a stop acknowledge signal to the CPU, in order to shut down the clocks globally. If stop mode is requested during transmission or reception, FlexCAN does the following:

1. Waits to be in either idle or bus-off state, or else waits for the third bit of Intermission and checks it to be recessive
2. Waits for all internal activities like arbitration, matching, move-in, and move-out to finish

3. Ignores its Rx input pin and drives its Tx pin as recessive
4. Sets the NOT_RDY and LPM_ACK bits in the MCR
5. Sends a stop acknowledge signal to the CPU, so that it can shut down the clocks globally

Exiting stop mode is done in one of the following ways:

- CPU resuming the clocks and removing the stop mode request
- CPU resuming the clocks and stop mode request as a result of the self wake mechanism

In the self wake mechanism, if the SLF_WAK bit in the MCR was set at the time FlexCAN entered stop mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN sets the WAK_INT bit in the ESR and, if enabled by the WAK_MSK bit in the MCR, generates a wake-up interrupt to the CPU. [1] Upon receiving the interrupt, the CPU should resume the clocks and remove the stop mode request. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up. Table 23-18 details the effect of SLF_WAK and WAK_MSK upon wake-up from stop mode. Note that wake-up from stop mode only works when both bits are set.

Table 23-18. FlexCAN Wake-up from Stop Mode

SLF_WAK	WAK_MSK	MCU Clocks Enabled	Wake-up Interrupt Generated
0	0	No	No
0	1	No	No
1	0	No	No
1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line while in stop mode. See the WAK_SRC bit in Section 23.5.2.1, “FlexCAN Module Configuration Register (MCR).” This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments.

23.6.9 Interrupts

The module can generate interrupts from up to 36 sources (32 interrupts due to message buffers (MBs) and 4 interrupts due to ORed interrupts from MBs, bus-off, error, and wake-up). The number of actual sources depends on the configured number of message buffers.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each buffer is assigned a flag bit in the IFLAG registers.

1. On L49P mask set devices, the FlexCAN wake-up request is not handled via the INTC module, therefore no interrupt exception takes place (execution continues inline where the MCU entered the selected low-power mode in response to the wake-up). On later mask sets, the wake-up interrupt shares the vector assigned to the channel error interrupt (see Table 6-2 on page 6-85).

The bit is set when the corresponding buffer completes a successful transmission/reception and is cleared when the CPU writes a ’1’ to it (unless another interrupt is generated at the same time).

NOTE

It must be guaranteed that the CPU only clears the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

A combined interrupt for all MBs is also generated by an OR of all the interrupt sources from MBs. This interrupt gets generated when any of the MBs generates an interrupt. In this case the CPU must read the IFLAG registers to determine which MB caused the interrupt.

The other 3 interrupt sources (bus off, error, and wake-up [1]) generate interrupts like the MB ones, and can be read from the error and status register. The bus-off and error interrupt mask bits are located in the control register (see Section 23.5.2.2, “FlexCAN Control Register (CTRL)”), and the wake-up interrupt mask bit is located in the configuration register (see Section 23.5.2.1, “FlexCAN Module Configuration Register (MCR)”).

23.6.10 Bus Interface

The CPU access to FlexCAN registers are subject to the following rules:

- Read and write access to supervisor registers in user mode results in access error.
- Read and write access to unimplemented or reserved address space also results in access error. Any access to unimplemented MB space results in access error.
- If MAXMB is programmed with a value smaller than the available number of MBs, then the unused MB memory space can be used as general purpose RAM space. Note that reserved words within RAM cannot be used. As an example, suppose FlexCAN is configured with 32 MBs and MAXMB is programmed with zero. The maximum number of MBs in this case becomes one. The MB memory starts at 0x0060, but the space from 0x0060 to 0x007F is reserved (for SMB usage), and the space from 0x0080 to 0x008F is used by the one MB. This leaves us with the space from 0x0090 to 0x027F available for general purpose use.

NOTE

Unused MB space must not be used as general purpose RAM while the FlexCAN is transmitting and receiving CAN frames.

23.7 Initialization / Application Information

23.7.1 FlexCAN Initialization Sequence

The FlexCAN module may be reset in three ways:

1. On L49P mask set devices, the FlexCAN wake-up request is not handled via the INTC module, therefore no interrupt exception takes place (execution continues inline where the MCU entered the selected low-power mode in response to the wake-up). On later mask sets, the wake-up interrupt shares the vector assigned to the channel error interrupt (see Table 6-2 on page 6-85).

- MCU level hard reset, which resets all memory mapped registers asynchronously
- MCU level soft reset, which resets some of the memory mapped registers synchronously (refer to Table 23-2 to see what registers are affected by soft reset)
- SOFT_RST bit in MCR, which has the same effect as the MCU level soft reset

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFT_RST bit remains set while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in any of the low power modes. The low power mode should be exited and the clocks resumed before applying soft reset.

The clock source (CLK_SRC bit) should be selected while the module is disabled. After the clock source is selected and the module is enabled (MDIS bit cleared), FlexCAN automatically goes to freeze mode. In freeze mode, FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in the MCR are set, the internal state machines are disabled and the FRZ_ACK and NOT_RDY bits in the MCR are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the message buffer contents are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into freeze mode (see Section 23.6.8.1, “Freeze Mode”). The following is a generic initialization sequence applicable to the FlexCAN module:

- Initialize the control register
  - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
  - Determine the bit rate by programming the PRESDIV field
  - Determine the internal arbitration mode (LBUF bit)
- Initialize the message buffers
  - The control and status word of all message buffers must be initialized
  - Other entries in each message buffer should be initialized as required
- Initialize MASK registers for acceptance mask as needed
- Set required interrupt mask bits in the IMASK registers (for all MB interrupts), in CTRL register (for bus off and error interrupts) and in the MCR for wake-up interrupt
- Negate the HALT bit in MCR

Starting with the last event, FlexCAN attempts to synchronize to the CAN bus.

23.7.2 FlexCAN Addressing and RAM Size Configurations

The RAM configuration that is implemented within the FlexCAN module is:

- 544 bytes for maximum of 32 message buffers

In this configuration the user can program the maximum number of MBs that will take part in the match and arbitration processes using the MAXMB field in the MCR. For the 32 MB configuration, MAXMB can be any number between 0–31.

Chapter 24
Inter-Integrated Circuit Bus Module (I^2C)

24.1 Overview

The Inter-Integrated Circuit Bus (I^2C or IIC) is a two wire bidirectional serial bus that provides a simple and efficient method of data exchange between devices. It minimizes the number of external connections between devices and does not require an external address decoder. The block diagram of the I^2C module is shown in Figure 24-1.

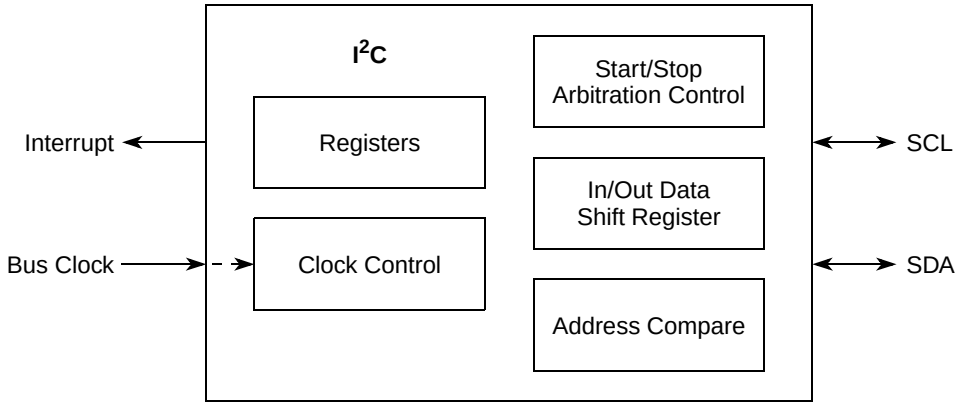


Figure 24-1. I^2C Module Block Diagram

The I^2C bus is suitable for applications requiring occasional communications over a short distance between a number of devices. It also provides flexibility, allowing additional devices to be connected to the bus for further expansion and system development. The interface is designed to operate up to 100kbps with maximum bus loading and timing. MAC7100 Family devices are capable of operating at higher baud rates, up to a maximum of the module clock divided by 20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by the maximum bus capacitance of 400pF.

The I^2C module can be independently disabled by writing to the IBDIS bit in the module control register (IBCR). Disabling the module turns off the clock to the module, although the module registers remain available for accessed by the core across the peripheral bus. The IBDIS bit is intended to be used when the module is not required in the application. Following a $\overline{\text{RESET}}$ operation the IBDIS bit is set, causing the module to be disabled.

24.2 Features

The I^2C module has the following key features:

- Compatible with I^2C Bus standard
- Multi-master operation
- Software programmable for one of 256 serial clock frequencies
- Software selectable acknowledge bit

- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- Basic DMA request interface

24.2.1 DMA Request Interface

A simple DMA request interface is implemented such that the I^2C can request data transfers with minimal support from the CPU. DMA mode is enabled by setting the DMAEN bit in the control register (IBCR). The DMA interface is only available when the I^2C module is configured for master mode. At least 3 bytes of data per frame must be transferred from/to the I^2C when using DMA mode, although in practice it is efficient to use DMA mode only when there is a large number of data bytes to transfer per frame.

Two internal signals, TX request and RX request, are used to generate DMA requests when the I^2C module requires data to be written or read from the data register.

Further details on using the DMA interface can be found in Section 24.7, “Initialization / Application Information.”

24.3 Modes of Operation

The I^2C module operates in one of four modes as determined by the MCU mode, plus one module-specific mode. The module must be in normal mode to execute I^2C bus operations. The disabled, doze and stop modes provide reduced power consumption as needed. Refer to Section 24.6.1, “I2C Operating Mode Details.”

24.4 Signal Description

Each Inter-Integrated Circuit (I^2C) module has 2 external signals. Note that the port integration module (PIM) must be configured to enable the peripheral function of the appropriate pins (refer to Section 18.6.2, “Peripheral Mode,” on page 18-296) prior to configuring an I^2C channel.

24.4.1 SCL

This is the bidirectional serial clock signal, compatible with the I^2C-bus specification.

24.4.2 SDA

This is the bidirectional serial data signal, compatible with the I^2C-bus specification.

24.5 Memory Map / Register Definition

The memory map for the I^{2}C module is given below in Table 24-1. The address listed for each register is the address offset. The actual address for each register is the sum of the base address for the I^{2}C module and the address offset for each register.

Table 24-1. I^{2}C Memory Map

I^{2}C Offset	Register Description	Access
0x0000	I^{2}C Bus Address Register (IBAD)	R/W
0x0001	I^{2}C Bus Frequency Divider Register (IBFD)	R/W
0x0002	I^{2}C Bus Control Register (IBCR)	R/W
0x0003	I^{2}C Bus Status Register (IBSR)	R/W
0x0004	I^{2}C Bus Data I/O Register (IBDR)	R/W

24.5.1 Register Descriptions

24.5.1.1 I^{2}C Bus Address Register (IBAD)

The IBAD register contains the address the I^{2}C module will respond to when addressed as a slave; note that this is not the address sent on the bus during the address transfer.

	7	6	5	4	3	2	1	0
R	ADR							0
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	I^{2}C Base + 0x0000							

Figure 24-2. I^{2}C Bus Address Register (IBAD)

Table 24-2. IBAD Field Descriptions

Bits	Name	Description
7–1	ADR[6:0]	Slave address. The slave address recognized by the I^{2}C Bus module. The default mode of I^{2}C bus is slave mode for an address match on the bus.
0	—	Reserved.

24.5.1.2 I^2C Bus Frequency Divider Register (IBFD)

The I^2C bus clock generator is implemented as a prescale divider that utilizes three input parameters defined by the IBFD register.

	7	6	5	4	3	2	1	0
R W	IBC[7:6]		IBC[5:3]			IBC[2:0]		
Reset	0	0	0	0	0	0	0	0
Reg Addr	I^2C Base + 0x0001							

Figure 24-3. I^2C Bus Frequency Divider Register (IBFD)

Table 24-3. IBFD Field Descriptions

Bits	Name	Description
7–6	IBC[7:6]	Prescaler shift register. 00 → 01 01 → 02 10 → 04 11 → RESERVED
5–3	IBC[5:3]	Prescaler divider.
2–0	IBC[2:0]	Shift register tap point.

	scl2start (clocks)	scl2stop (clocks)	scl2tap (clocks)	tap2tap (clocks)
000	2	7	4	1
001	2	7	4	2
010	2	9	6	4
011	6	9	6	8
100	14	17	14	16
101	30	33	30	32
110	62	65	62	64
111	126	129	126	128

	SCL Tap (clocks)	SDA Tap (clocks)
000	5	1
001	6	1
010	7	2
011	8	2
100	9	3
101	10	3
110	12	4
111	15	4

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of Table 24-3. All subsequent tap points are separated by $2^{IBC[5:3]}$ as shown in the tap2tap column in Table 24-3. The SCL tap is used to generate the SCL period and the SDA tap is used to determine the delay from the falling edge of SCL to the change of state of SDA i.e. the SDA hold time

IBC[7:6] defines the multiplier factor MUL. The values of MUL are shown in Table 24-3.

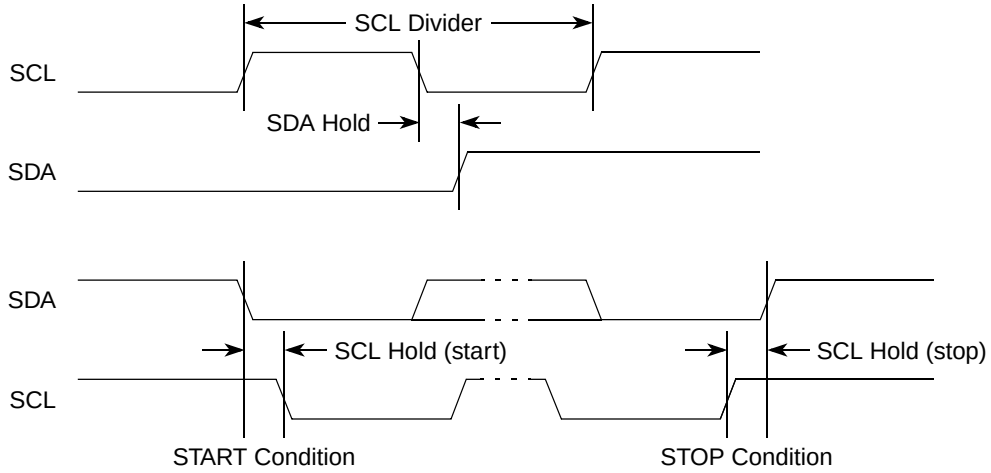


Figure 24-4. SCL Divider and SDA Hold

The equation used to generate the divider values from the IBFD bits is:

$$\text{SCL Divider} = \text{MUL} \times (2 \times (2 + \text{scl2tap} + (\text{tap2tap} \times (\text{SCL\_Tap} - 1)))) \quad \textit{Eqn. 24-1}$$

The SDA hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in Table 24-4. The equation used to generate the SDA Hold value from the IBFD bits is:

$$\text{SDA Hold} = \text{MUL} \times (3 + \text{scl2tap} + (\text{tap2tap} \times (\text{SDA\_Tap} - 1))) \quad \textit{Eqn. 24-2}$$

The equation for SCL Hold values to generate the start and stop conditions from the IBFD bits is:

$$\text{SCL Hold (start)} = \text{MUL} \times (\text{scl2start} + (\text{tap2tap} \times (\text{SCL\_Tap} - 1))) \quad \textit{Eqn. 24-3}$$

$$\text{SCL Hold (stop)} = \text{MUL} \times (\text{scl2stop} + (\text{tap2tap} \times (\text{SCL\_Tap} - 1))) \quad \textit{Eqn. 24-4}$$

Table 24-4. I^2C Divider and Hold Values

	IBC[7:0]	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL = 1	00	20	7	6	11
	01	22	7	7	12
	02	24	8	8	13
	03	26	8	9	14
	04	28	9	10	15
	05	30	9	11	16
	06	34	10	13	18
	07	40	10	16	21
	08	28	7	10	15
	09	32	7	12	17
	0A	36	9	14	19
	0B	40	9	16	21
	0C	44	11	18	23
	0D	48	11	20	25
	0E	56	13	24	29
	0F	68	13	30	35
	10	48	9	18	25
	11	56	9	22	29
	12	64	13	26	33
	13	72	13	30	37
	14	80	17	34	41
	15	88	17	38	45
	16	104	21	46	53
	17	128	21	58	65
	18	80	9	38	41
	19	96	9	46	49
	1A	112	17	54	57
	1B	128	17	62	65
	1C	144	25	70	73
	1D	160	25	78	81
	1E	192	33	94	97
	1F	240	33	118	121
	20	160	17	78	81
	21	192	17	94	97
	22	224	33	110	113
	23	256	33	126	129
	24	288	49	142	145
	25	320	49	158	161
	26	384	65	190	193
	27	480	65	238	241
	28	320	33	158	161
	29	384	33	190	193
	2A	448	65	222	225
	2B	512	65	254	257
	2C	576	97	286	289
	2D	640	97	318	321
	2E	768	129	382	385
	2F	960	129	478	481
	30	640	65	318	321
	31	768	65	382	385
	32	896	129	446	449
	33	1024	129	510	513
	34	1152	193	574	577
	35	1280	193	638	641
	36	1536	257	766	769
	37	1920	257	958	961
	38	1280	129	638	641
	39	1536	129	766	769
	3A	1792	257	894	897
	3B	2048	257	1022	1025
	3C	2304	385	1150	1153
	3D	2560	385	1278	1281
	3E	3072	513	1534	1537
	3F	3840	513	1918	1921

Table 24-4. I²C Divider and Hold Values (continued)

	IBC[7:0]	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL = 2	40	40	14	12	22
	41	44	14	14	24
	42	48	16	16	26
	43	52	16	18	28
	44	56	18	20	30
	45	60	18	22	32
	46	68	20	26	36
	47	80	20	32	42
	48	56	14	20	30
	49	64	14	24	34
	4A	72	18	28	38
	4B	80	18	32	42
	4C	88	22	36	46
	4D	96	22	40	50
	4E	112	26	48	58
	4F	136	26	60	70
	50	96	18	36	50
	51	112	18	44	58
	52	128	26	52	66
	53	144	26	60	74
	54	160	34	68	82
	55	176	34	76	90
	56	208	42	92	106
	57	256	42	116	130
	58	160	18	76	82
	59	192	18	92	98
	5A	224	34	108	114
	5B	256	34	124	130
	5C	288	50	140	146
	5D	320	50	156	162
	5E	384	66	188	194
	5F	480	66	236	242
	60	320	28	156	162
	61	384	28	188	194
	62	448	32	220	226
	63	512	32	252	258
	64	576	36	284	290
	65	640	36	316	322
	66	768	40	380	386
	67	960	40	476	482
	68	640	28	316	322
	69	768	28	380	386
	6A	896	36	444	450
	6B	1024	36	508	514
	6C	1152	44	572	578
	6D	1280	44	636	642
	6E	1536	52	764	770
	6F	1920	52	956	962
	70	1280	36	636	642
	71	1536	36	764	770
	72	1792	52	892	898
	73	2048	52	1020	1026
	74	2304	68	1148	1154
	75	2560	68	1276	1282
	76	3072	84	1532	1538
	77	3840	84	1916	1922
	78	2560	36	1276	1282
	79	3072	36	1532	1538
	7A	3584	68	1788	1794
	7B	4096	68	2044	2050
	7C	4608	100	2300	2306
	7D	5120	100	2556	2562
	7E	6144	132	3068	3074
	7F	7680	132	3836	3842

Table 24-4. I^2C Divider and Hold Values (continued)

	IBC[7:0]	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL = 4	80	80	28	24	44
	81	88	28	28	48
	82	96	32	32	52
	83	104	32	36	56
	84	112	36	40	60
	85	120	36	44	64
	86	136	40	52	72
	87	160	40	64	84
	88	112	28	40	60
	89	128	28	48	68
	8A	144	36	56	76
	8B	160	36	64	84
	8C	176	44	72	92
	8D	192	44	80	100
	8E	224	52	96	116
	8F	272	52	120	140
	90	192	36	72	100
	91	224	36	88	116
	92	256	52	104	132
	93	288	52	120	148
	94	320	68	136	164
	95	352	68	152	180
	96	416	84	184	212
	97	512	84	232	260
	98	320	36	152	164
	99	384	36	184	196
	9A	448	68	216	228
	9B	512	68	248	260
	9C	576	100	280	292
	9D	640	100	312	324
	9E	768	132	376	388
	9F	960	132	472	484
	A0	640	68	312	324
	A1	768	68	376	388
	A2	896	132	440	452
	A3	1024	132	504	516
	A4	1152	196	568	580
	A5	1280	196	632	644
	A6	1536	260	760	772
	A7	1920	260	952	964
	A8	1280	132	632	644
	A9	1536	132	760	772
	AA	1792	260	888	900
	AB	2048	260	1016	1028
	AC	2304	388	1144	1156
	AD	2560	388	1272	1284
	AE	3072	516	1528	1540
	AF	3840	516	1912	1924
	B0	2560	260	1272	1284
	B1	3072	260	1528	1540
	B2	3584	516	1784	1796
	B3	4096	516	2040	2052
	B4	4608	772	2296	2308
	B5	5120	772	2552	2564
	B6	6144	1028	3064	3076
	B7	7680	1028	3832	3844
	B8	5120	516	2552	2564
	B9	6144	516	3064	3076
	BA	7168	1028	3576	3588
	BB	8192	1028	4088	4100
	BC	9216	1540	4600	4612
	BD	10240	1540	5112	5124
	BE	12288	2052	6136	6148
	BF	15360	2052	7672	7684

24.5.1.3 I²C Bus Control Register (IBCR)

	7	6	5	4	3	2	1	0
R	IBDIS	IBIE	MS/$\overline{SL}$	Tx/$\overline{Rx}$	TXAK	0	DMAEN	IBDOZE
W						RSTA		
Reset	1	0	0	0	0	0	0	0
Reg Addr	I^2C Base + 0x0002							

Figure 24-5. I^2C Bus Control Register (IBCR)

Table 24-5. IBCR Field Descriptions

Bits	Name	Description
7	IBDIS	I^2C bus disable. Refer to Section 24.6.1, "I2C Operating Mode Details," for more information. 0 Module enabled. 1 Module disabled and held in reset state.
6	IBIE	I^2C bus interrupt enable. 0 Interrupt request disabled. Note, does not clear IBSR register IBIF bit. 1 Interrupt request enabled. Interrupt requested if IBSR register IBIF bit is set.
5	MS/$\overline{SL}$	Master/slave mode select. 0 Slave mode. When changed from 1 to 0, a STOP signal is generated on the bus and the operation mode changes from master to slave. A STOP signal should be generated only if the IBIF flag is set. MS/$\overline{SL}$ is cleared without generating a STOP signal when the master loses arbitration. 1 Master mode. When changed from 0 to 1, a START signal is generated on the bus and the master mode is selected.
4	Tx/$\overline{Rx}$	Transmit/receive mode select. This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit of the IBSR. In master mode this bit should be set according to the type of transfer required. 0 Receive 1 Transmit
3	TXAK	Transmit acknowledge enable. This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The I^2C module will always acknowledge address matches, provided it is enabled, regardless of the value of TXAK. Note that values written to this bit are only used when the I^2C Bus is a receiver, not a transmitter. 0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte of data 1 No acknowledge signal response is sent (i.e., acknowledge bit = 1)
2	RSTA	Repeat start. 0 Writing a 0 is ignored. Always reads as zero. 1 Writing a 1 to this bit generates a repeated START condition on the bus, provided this module is the current bus master. Attempting a repeated start if the bus is owned by another master will result in loss of arbitration.

Table 24-5. IBCR Field Descriptions (continued)

Bits	Name	Description
1	DMAEN	DMA enable. Determines whether DMA TX and RX requests will be asserted when the I^2C module requires data to be read or written to the data register. No Transfer Done interrupts will be generated when this bit is set, however an interrupt will be generated if the loss of arbitration or addressed as slave conditions occur. The DMA mode is only valid when the I^2C module is configured as a Master, and the DMA transfer still requires CPU intervention at the start and the end of each frame of data. See Section 24.7.2, "DMA Application Information," for more details. 1 Disable DMA TX/RX request signals 0 Enable DMA TX/RX request signals.
0	IBSDOZE	DOZE mode enable. Refer to Section 24.6.1, "I2C Operating Mode Details," for more information. 0 Do not support Doze mode: the I^2C module continues to operate normally when the MCU enters Doze mode. 1 Support Doze mode: the I^2C module completes any bus operations in progress and then operation is frozen.

24.5.1.4 I^2C Bus Status Register (IBSR)

	7	6	5	4	3	2	1	0
R	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
W								
Reset	1	0	0	0	0	0	0	0
Reg Addr	I^2C Base + 0x0003							

Figure 24-6. I^2C Bus Status Register (IBSR)

Table 24-6. IBSR Field Descriptions

Bits	Name	Description
7	TCF	Data transferring. While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. Note that this bit is only valid during or immediately following a transfer to or from the I^2C module. 0 Transfer in progress 1 Transfer complete
6	IAAS	Addressed as a slave. When the I^2C bus address register value is matched with the calling address, this bit is set.An interrupt request is asserted if the IBIE is set. The CPU must check the SRW bit and set the Tx/$\overline{Rx}$ mode accordingly. Writing to the I^2C bus control register clears this bit. 0 Not addressed 1 Addressed as a slave
5	IBB	Bus busy. This bit indicates the status of the bus. When a START signal is detected, the IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state. 0 Bus is Idle 1 Bus is busy

Table 24-6. IBSR Field Descriptions (continued)

Bits	Name	Description
4	IBAL	Arbitration lost. This bit is set by hardware when the arbitration procedure is lost, which occurs in the following circumstances: • SDA is sampled low when the master drives a high during an address or data transmit cycle. • SDA is sampled low when the master drives a high during the acknowledge bit of a data receive cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in slave mode. • A stop condition is detected when the master did not request it. This bit must be cleared by writing a one to it. Writing a zero has no effect.
3	—	Reserved.
2	SRW	Slave read/write. When IAAS is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is only valid when the I^2C bus is in slave mode, a complete address transfer has occurred with an address match, and no other transfers have been initiated. By programming this bit, the CPU can select slave transmit/receive mode according to the command of the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
1	IBIF	I^2C bus interrupt. The IBIF bit is set when one of the following conditions occurs: • Arbitration lost (IBAL bit set) • Byte transfer complete (TCF bit set) • Addressed as slave (IAAS bit set) An interrupt request will be generated if the IBIE bit is set. This bit must be cleared by writing a one to it. A write of zero has no effect.
0	RXAK	Received acknowledge. This is the value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits of data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. 0 Acknowledge received 1 No acknowledge received

24.5.1.5 I^2C Bus Data I/O Register (IBDR)

	7	6	5	4	3	2	1	0
R	D[7:0]							
W								
Reset	0	0	0	0	0	0	0	0
Reg Addr	I^2C Base + 0x0004							

Figure 24-7. I^2C Bus Data I/O Register (IBDR)

In master transmit mode, when data is written to IBDR, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred.Note that the Tx/$\overline{\text{Rx}}$ bit in the IBCR

must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the I^2C is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the last byte received while the I^2C is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the I^2C bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of MS/$\overline{SL}$ is used for the address transfer and should comprise the calling address (in position D[7:1]) concatenated with the required R/W bit (in position D0).

24.6 Functional Description

24.6.1 I^2C Operating Mode Details

24.6.1.1 I^2C Module Normal Mode

In order to perform I^2C bus operations, the module must be operating in normal mode. If the MCU is in run mode, the I^2C module is in normal mode unless specifically disabled as described in Section 24.6.1.3, “I2C Module Disabled Mode.”

24.6.1.2 I^2C Module Debug Mode

If the MCU enters debug mode, the I^2C module continues to operate in normal mode.

24.6.1.3 I^2C Module Disabled Mode

A mode that is independent of the MCU mode is the I^2C disabled mode. At any time, the IBDIS bit in the IBCR register may be set to disable the I^2C module. This mode causes all I^2C clocks to halt, which causes the module to draw minimal power while all other MCU peripheral modules may continue to operate normally. All registers remain available to be accessed by the core via the peripheral bus.

The IBDIS bit is set by a reset operation, and therefore the module must be enabled in order to utilize the I^2C bus. If the module is enabled in the middle of a byte transfer by another master on the I^2C bus, the interface behaves as follows:

- If the module is set to slave mode, the current transfer on the bus is ignored and the module starts operating normally when a subsequent start condition is detected.
- If the module is set to master mode, detecting if the bus is busy is not possible, thus if a start cycle is initiated the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the I^2C Bus module losing arbitration, after which bus operation would return to normal.

The IBDIS bit is intended to be used when the module is not required in the application.

24.6.1.4 I²C Module Doze Mode

If the MCU enters doze mode, the contents of the IBSDOZE bit in the IBCR register determines whether the I²C module continues to operate in normal mode or enters doze mode. If the IBSDOZE bit is cleared, the I²C module will remain in normal mode. If the IBSDOZE bit is set, the module will wait for any I²C bus transaction to complete, and then the module will enter doze mode (a transfer is defined as any active data between valid I²C Start and I²C Stop conditions). I²C doze mode stops the module clocks but leaves the registers accessible, thus offering power savings over operation in normal mode.

When the MCU exits doze mode or the IBSDOZE bit is cleared, the I²C clock is turned on again and normal operation is resumed.

24.6.1.5 I²C Module Stop Mode

If the MCU enters stop mode, all clocks stop and therefore all modules stop. This mode causes all I²C clocks to halt, and thus offers maximum power saving. After exiting stop mode, the clocks are turned on again.

24.6.2 I²C Bus Protocol

The I²C bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. A logical AND function is exercised on both lines with external pull-up resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts: START signal, slave address transmission, data transfer and STOP signal. They are described briefly in the following sections and illustrated in Figure 24-8.

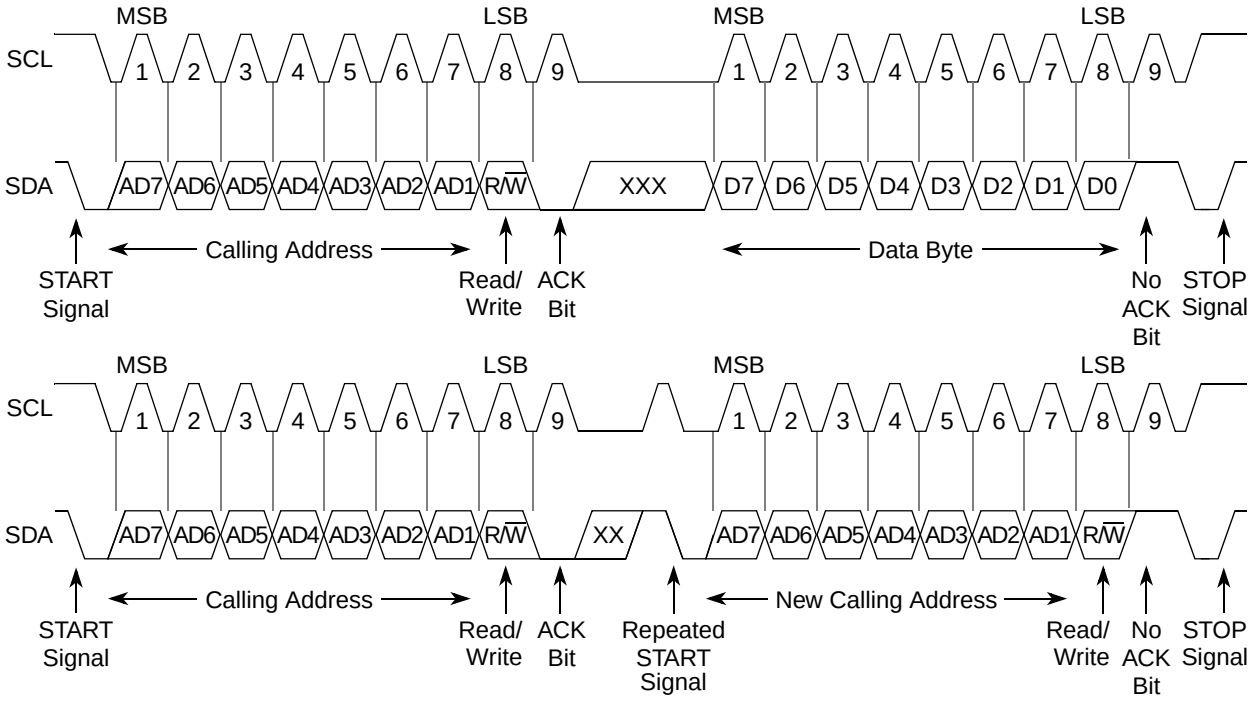


Figure 24-8. I²C Bus Transmission Signals

24.6.2.1 START Signal

When the bus is free, i.e. no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal.As shown in Figure 24-8, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

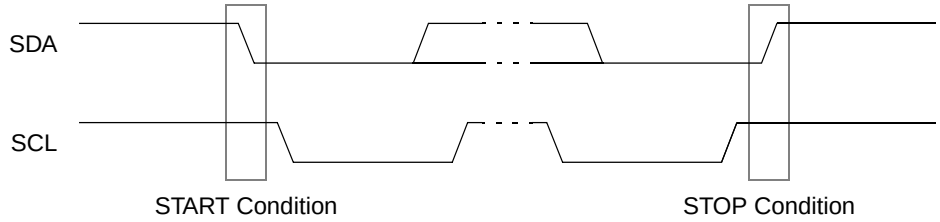


Figure 24-9. I^2C Bus Start and Stop conditions

24.6.2.2 Slave Address Transmission

The first byte of data transfer immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

Read transfer – the slave transmits data to the master.

Write transfer – the master transmits data to the slave.

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 24-8).

No two slaves in the system may have the same address. If the I^2C bus is master, it must not transmit an address that is equal to its own slave address. The I^2C bus cannot be master and slave at the same time.However, if arbitration is lost during an address cycle the I^2C bus will revert to slave mode and operate correctly, even if it is being addressed by another master.

24.6.2.3 Data Transfer

Once successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 24-8. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte must be followed by an acknowledge bit, which is signalled from the receiving device by pulling the SDA low at the ninth clock. Therefore, one complete data byte transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop signal to abort the data transfer or a start signal (repeated start) to commence a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means 'end of data' to the slave, so the slave releases the SDA line for the master to generate a STOP or START signal.

24.6.2.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL is at logical “1” (see Figure 24-8).

The master can generate a STOP even if the slave has generated an acknowledge, at which point the slave must release the bus.

24.6.2.5 Repeated START Signal

As shown in Figure 24-8, a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

24.6.2.6 Arbitration Procedure

The I^2C bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic “1” while another master transmits logic “0”. The losing masters immediately switch over to slave receive mode and stop driving the SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

24.6.2.7 Clock Synchronization

Since wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and once a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the low to high change in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 24-10). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods.The first device to complete its high period pulls the SCL line low again.

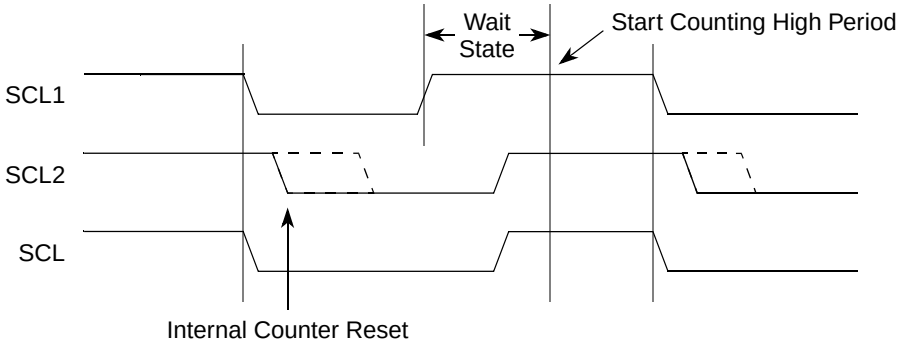


Figure 24-10. I^2C Bus Clock Synchronization

24.6.2.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait state until the slave releases the SCL line.

24.6.2.9 Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it.If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

24.6.3 Interrupts

The I^2C module uses only one interrupt vector as defined in Chapter 6, “Exceptions,” and Chapter 10, “Interrupt Controller Module (INTC).”

There are three types of internal interrupts in the I^2C. The interrupt service routine can determine the interrupt type by reading the IBSR register.

I^2C interrupts can be generated on

- Arbitration lost condition (IBAL bit set)
- Byte transfer condition (TCF bit set)
- Address detect condition (IAAS bit set)

The I^2C interrupt request is enabled by the IBIE bit in the IBCR register. It must be cleared by writing ‘1’ to the IBIF bit of the IBSR register.

24.7 Initialization / Application Information

24.7.1 I²C Programming Examples

24.7.1.1 Initialization Sequence

Reset will put the I²C bus control register to its default state. Before the interface can be used to transfer serial data, an initialization procedure must be carried out, as follows:

1. Update the frequency divider register (IBFD) and select the required division ratio to obtain SCL frequency from system clock.
2. Update the I²C bus address register (IBAD) to define its slave address.
3. Clear the IBDIS bit of the I²C bus control register (IBCR) to enable the I²C interface system.
4. Modify the bits of the I²C bus control register (IBCR) to select master/slave mode, transmit/receive mode and interrupt enable or not.

24.7.1.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the I²C Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB, which is set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I²C is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of the sequence of events which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
while (bit 5, IBSR ==1)    // wait in loop for IBB flag to clear
bit4 and bit 5, IBCR = 1   // set transmit & master mode, i.e. generate start condition
IBDR = calling_address     // send the calling address to the data register
while (bit 5, IBSR ==0)    // wait in loop for IBB flag to be set
```

24.7.1.3 Post-Transfer Software Response

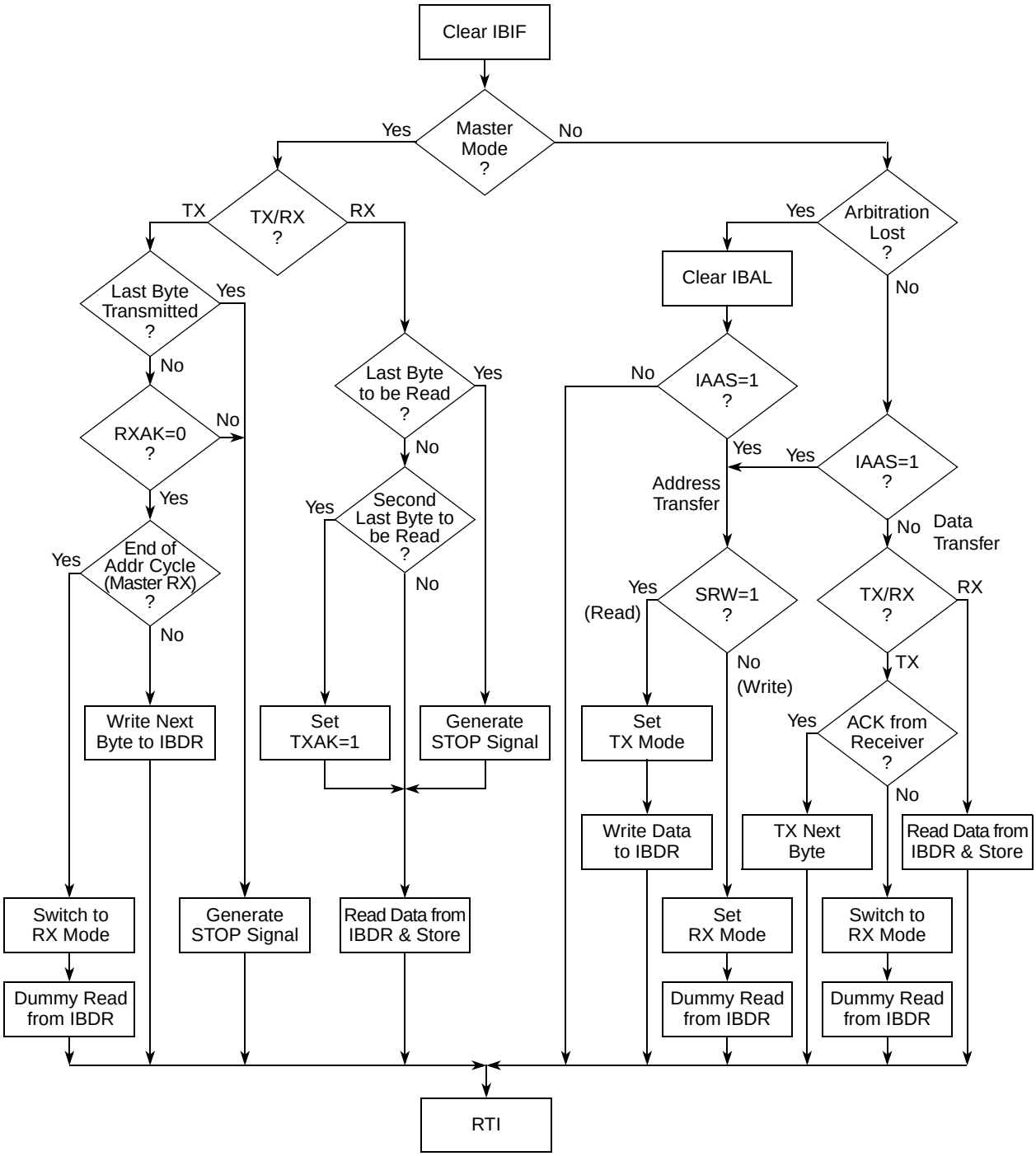


Figure 24-11. I^2C Typical Interrupt Routine Flow Chart

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The I^2C Bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. The IBIF (interrupt flag) can be cleared by writing 1 (in the interrupt service routine, if interrupts are used).

The TCF bit will be cleared to indicate data transfer in progress by reading the IBDR data register in receive mode or writing the IBDR in transmit mode. The TCF bit should not be used as a data transfer complete flag as the flag timing is dependent on a number of factors including the I^2C bus frequency. This bit may not conclusively provide an indication of a transfer complete situation. It is recommended that transfer complete situations are detected using the IBIF flag

Software may service the I^2C I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Note that polling should monitor the IBIF bit rather than the TCF bit since their operation is different when arbitration is lost.

Note that when an interrupt occurs at the end of the address cycle, the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit in IBDR, then the Tx/Rx bit should be toggled at this stage.

During slave mode address cycles (IAAS=1) the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For slave mode data cycles (IAAS=0) the SRW bit is not valid. The Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

The following is an example software sequence for 'master transmitter' in the interrupt routine.

```
clear bit 1, IBSR                                    // Clear the IBIF flag
if (bit 5, IBCR ==0)
      slave_mode()                                 // run slave mode routine
if (bit 4, IBCR ==0))
     receive_mode()                              // run receive_mode routine
if (bit 0, IBSR == 1)                                        // if NO ACK
     end();                                          // end transmission
else
IBDR = data_to_transmit                        // transmit next byte of data
```

24.7.1.4 Generation of STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a stop condition is generated by a master transmitter.

```
if (tx_count == 0) or     // check to see if all data bytes have been transmitted
   (bit 0, IBSR == 1) {                                 // or if no ACK generated
      clear bit 5, IBCR                                // generate stop condition
   }
else {
IBDR = data_to_transmit                  // write byte of data to DATA register
   tx_count --                                                // decrement counter
   }                                                    // return from interrupt
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the transmit acknowledge bit (TXAK) before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must first be generated. The following is an example showing how a STOP signal is generated by a master receiver.

```
rx_count --                                          // decrease the rx counter
if (rx_count ==1)                                 // 2nd last byte to be read ?
   bit 3, IBCR = 1                                              // disable ACK
```

```
if (rx_count == 0)                                  // last byte to be read ?
  bit 1, IBCR = 0                                     // generate stop signal
else
data_received = IBDR                                // read RX data and store
```

24.7.1.5 Generation of Repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
bit 2, IBCR = 1                              // generate another start ( restart)
IBDR == calling_address                           // transmit the calling address
```

24.7.1.6 Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred. Interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR for slave transmits or dummy reading from IBDR in slave receive mode. The slave will drive SCL low in-between byte transfers SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an 'end of data' signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

24.7.1.7 Arbitration Lost

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices that lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission, while the bus is being engaged by another master, the hardware will inhibit the transmission, switch the MS/SL bit from 1 to 0 without generating a STOP condition, generate an interrupt to CPU and set the IBAL to indicate that the attempt to engage the bus is failed. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

24.7.2 DMA Application Information

The DMA interface on the I^2C is not completely autonomous and requires intervention from the CPU to start and to terminate the frame transfer. DMA mode is only valid for Master transmit and Master receive modes. Software must ensure that the DMA enable bit in the control register is not set when the I^2C module is configured in master mode.

The eDMA controller must only transfer one byte of data per Tx/Rx request. This is because there is no FIFO on the I^2C module.

The CPU should also keep the I^2C interrupt enabled during a DMA transfer to detect the arbitration lost condition and take action to recover from this situation. The address match condition will not occur in DMA mode as the I^2C should never be configured for slave operation.

The following sections detail how to set up a DMA transfer and what intervention is required from the CPU. It is assumed that the system eDMA controller is configured to generate an interrupt after a certain number of DMA transfers have taken place.

24.7.2.1 DMA Mode, Master Transmit

The following flow diagram details exactly the operation for using a eDMA controller to transmit “n” data bytes to a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the last data byte) can be transferred by the eDMA controller. The last data byte must be transferred by the CPU.

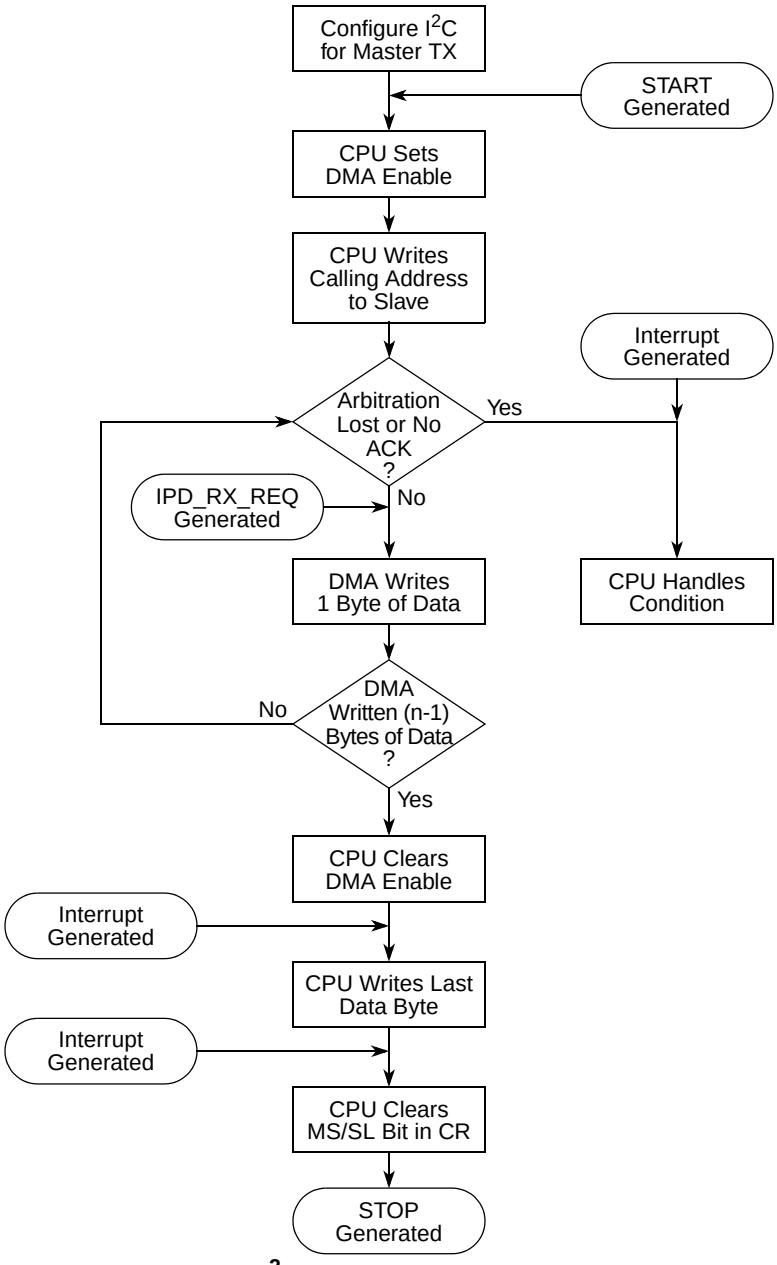


Figure 24-12. I^2C Master Transmit in DMA Mode

24.7.2.2 DMA Mode, Master Receive

The following flow diagram details the exact operation for using the eDMA controller to receive “n” data bytes from a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the two last data bytes) can be read by the eDMA controller. The last two data bytes must be transferred by the CPU.

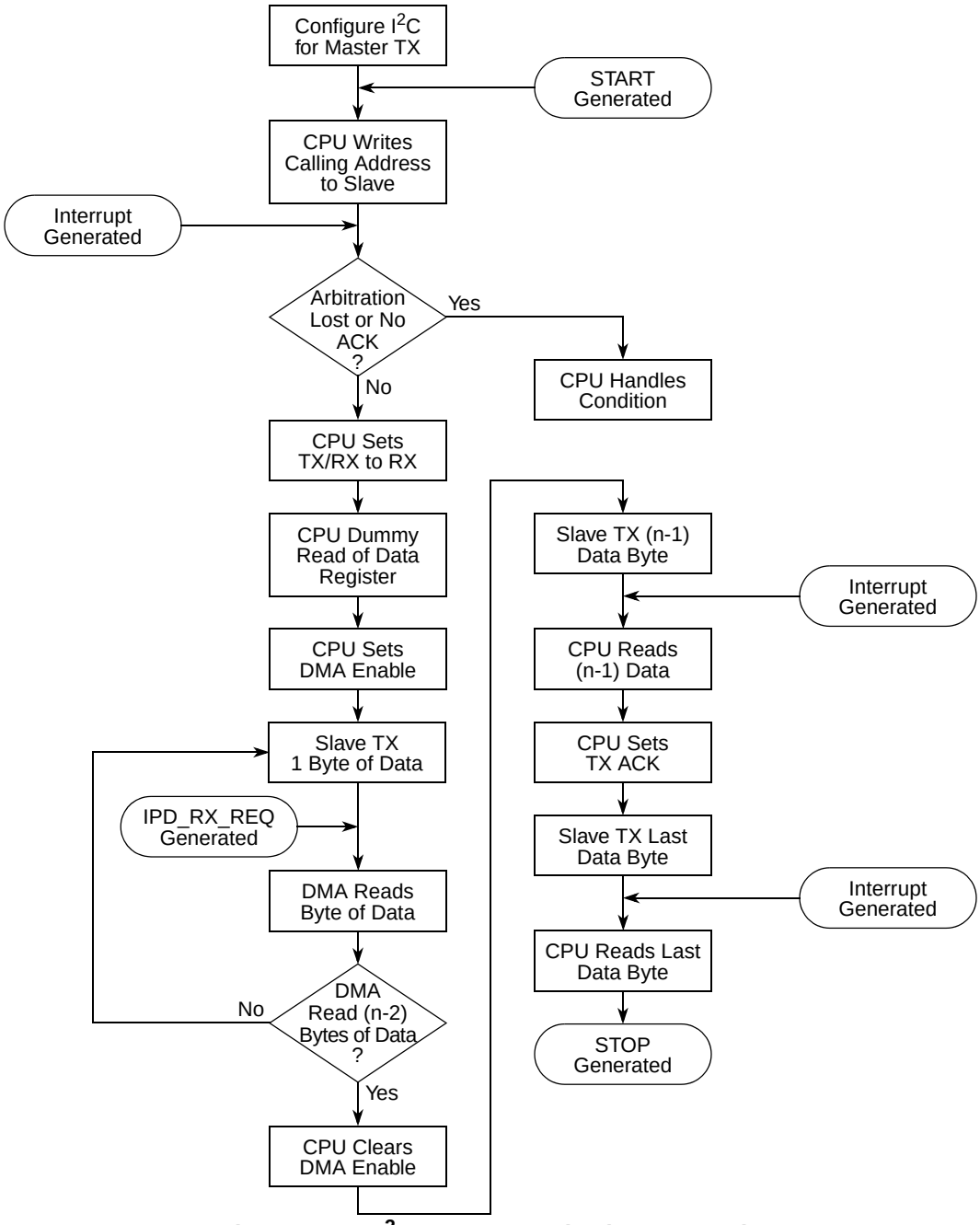


Figure 24-13. I2C Master Receive in DMA Mode

Chapter 25
Periodic Interrupt Timer Module (PIT)

25.1 Overview

As shown in Figure 25-1, the PIT is an array of timers that can be used to raise interrupts and trigger eDMA channels. It also provides a dedicated Real Time Interrupt Timer (RTI), which runs on a separate clock and can be used for system wake-up.

The PIT provides eleven programmable timers offering a range of functions. All of the timers are 24-bit wide down-counters. Once zero is reached, a trigger can be generated, then the counter is reloaded with the start value and continues to be decremented. The value of the decrementer can be read at anytime.

The PIT has one Real Time Interrupt (RTI), four general purpose timers, four dedicated timers used to trigger eDMA transfers and two timers used to trigger ATD conversions.

Timer 0 is used to provide the RTI and is used only to wake the system from low power modes. The four general purpose timers, Timer 1 to Timer 4 can be used to generate interrupts if required, or to trigger eDMA channels 0 to 3. Timer 5 to Timer 8 are used to trigger eDMA channels 4 to 7. Timers 9 and 10 are used for the ATD triggers SYSTRIG1 and SYSTRIG0 respectively. Refer to Table 25-9 for a summary of feature assignments.

All of the timers receive their clock from the peripheral bus clock f_{IPS}, with the exception of the RTI which uses the oscillator or PLL clock as its source. This allows the RTI to operate while in pseudo stop mode.

The PIT module can be independently disabled by writing to the MDIS bit in the PITCTRL register when the module is not required in the application. Disabling the module turns off all clocks to the module, with the exception of the RTI clock. Setting the MDIS bit disables Timer 1 – Timer 10 while allowing the RTI to continue operating; and allowing most module registers to remain accessible by the core across the peripheral bus.

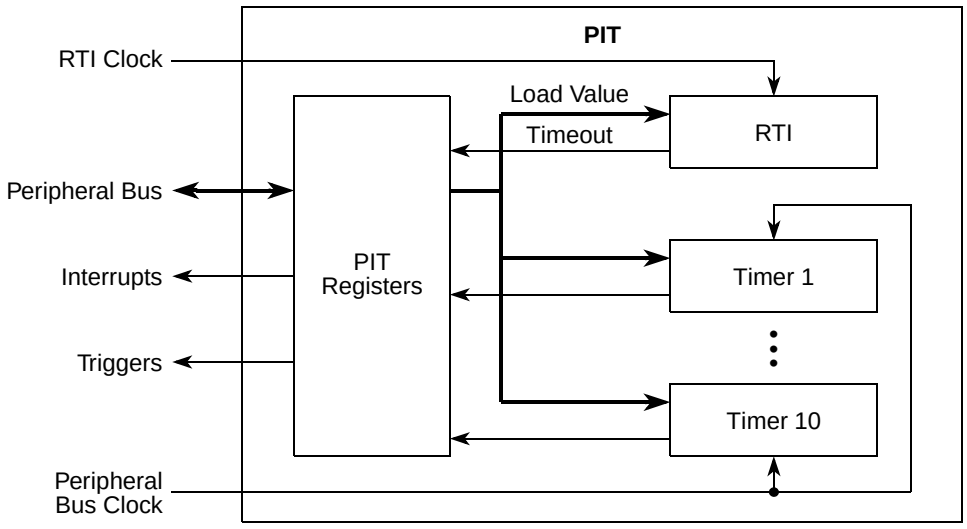


Figure 25-1. PIT Block Diagram

25.2 Features

The main features of this module are:

- Independent timeout periods for each of the ten 24-bit timers
- One real-time interrupt (RTI) timer to wake-up the CPU in wait or pseudo-stop mode
- Eight timers that can be configured to generate DMA trigger pulses
- Four timers that can be configured to generate interrupts instead of DMA triggers
- Two timers that can be configured to generate ATD trigger pulses
- All interrupts are maskable
- Interrupt requests can be asserted even when the bus clock is switched off
- Power savings with a separate clock for the RTI (all other timers share one common core clock)

25.3 Modes of Operation

This subsection describes briefly all operating modes supported by the PIT.

- Run Mode. All functional parts of the PIT are running during normal Run Mode.
- Debug Mode. All timers are frozen when the device enters debug mode.
- Doze Mode. The RTI timer can run in Doze mode depending on the settings in the CRG (Clock and Reset Generator Module).
- Stop Mode. Depending on the setting of the CRG PSTP bit, Stop Mode can be differentiated between Full Stop Mode and Pseudo-Stop Mode:
  - Full Stop Mode (PSTP=0). The PIT module is frozen.
  - Pseudo-Stop Mode (PSTP=1). RTI continues to run if enabled, other timers are frozen.

25.4 Memory Map / Register Definition

Table 25-1 gives an overview of all PIT registers. The first timer (timer 0) is the RTI timer. Register Address = Base Address + PIT Offset (refer to Table 8-10 on page 8-99 for the PIT base address).

Table 25-1. PIT Memory Map

PIT Offset	Register Description	Access
0x0000	PIT RTI Load Value Register	R/W
0x0004	PIT Timer Load Value Register 1	R/W
0x0008	PIT Timer Load Value Register 2	R/W
0x000C	PIT Timer Load Value Register 3	R/W
0x0010	PIT Timer Load Value Register 4	R/W
0x0014	PIT Timer Load Value Register 5	R/W
0x0018	PIT Timer Load Value Register 6	R/W
0x001C	PIT Timer Load Value Register 7	R/W
0x0020	PIT Timer Load Value Register 8	R/W
0x0024	PIT Timer Load Value Register 9	R/W
0x0028	PIT Timer Load Value Register 10	R/W

Table 25-1. PIT Memory Map (continued)

PIT Offset	Register Description	Access
0x0032 – 0x007F	Reserved	
0x0080	PIT RTI Current Value Register	R
0x0084	PIT Timer Current Value Register 1	R
0x0088	PIT Timer Current Value Register 2	R
0x008C	PIT Timer Current Value Register 3	R
0x0090	PIT Timer Current Value Register 4	R
0x0094	PIT Timer Current Value Register 5	R
0x0098	PIT Timer Current Value Register 6	R
0x009C	PIT Timer Current Value Register 7	R
0x00A0	PIT Timer Current Value Register 8	R
0x00A4	PIT Timer Current Value Register 9	R
0x00A8	PIT Timer Current Value Register 10	R
0x00AC – 0x009F	Reserved	
0x00100	PIT Interrupt Flags Register	R/W
0x00104	PIT Interrupt Enable Register	R/W
0x00108	PIT Interrupt/DMA Select Register	R/W
0x0010C	PIT Timer Enable Register	R/W
0x00110	PIT Control Register	R/W
0x0114 – 0x01FC	Reserved	

25.4.1 Register Descriptions

This section describes, in address order, all the PIT registers and their individual bits.

NOTE

The RTI registers should be set only when the RTI clock is enabled – for example, not if the system is in doze mode and the RTI clock is switched off in doze mode.

25.4.1.1 PIT RTI / Timer Load Value Registers (TLVAL*n*)

These registers select the timeout period for the timer interrupts. In the case of the RTI, it will take several cycles until this value is synchronized into the RTI clock domain. For all other timers the value change is visible immediately.

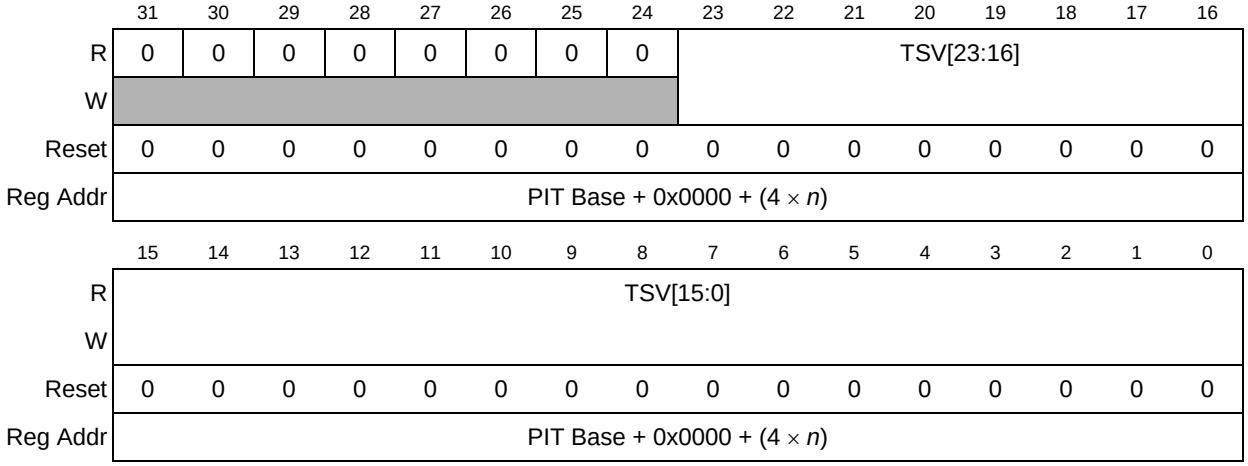


Figure 25-2. PIT RTI/Timer Load Value Registers (TLVAL*n*)

Table 25-2. TLVAL*n* Field Descriptions

Bits	Name	Description
31–24	—	Reserved.
23–0	TSV[23:0]	Timer start value [1]. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again (see Section 25.4.1.6, “PIT Timer Enable Register (PITEN)”).

[1] For the RTI, the timer should not be set to a value lower than 32 cycles, otherwise interrupts may be lost, as it takes several cycles to clear the RTI interrupt. For the other timers, this limit does not apply, however there will be practical limits, since the processor will require several cycles to service an interrupt.

25.4.1.2 PIT RTI / Timer Current Value Registers (TVAL*n*)

These registers indicate the current timer value of each decrementer. In the case of the RTI, this will show a value which is several cycles old, since it originates from a potentially different clock domain.

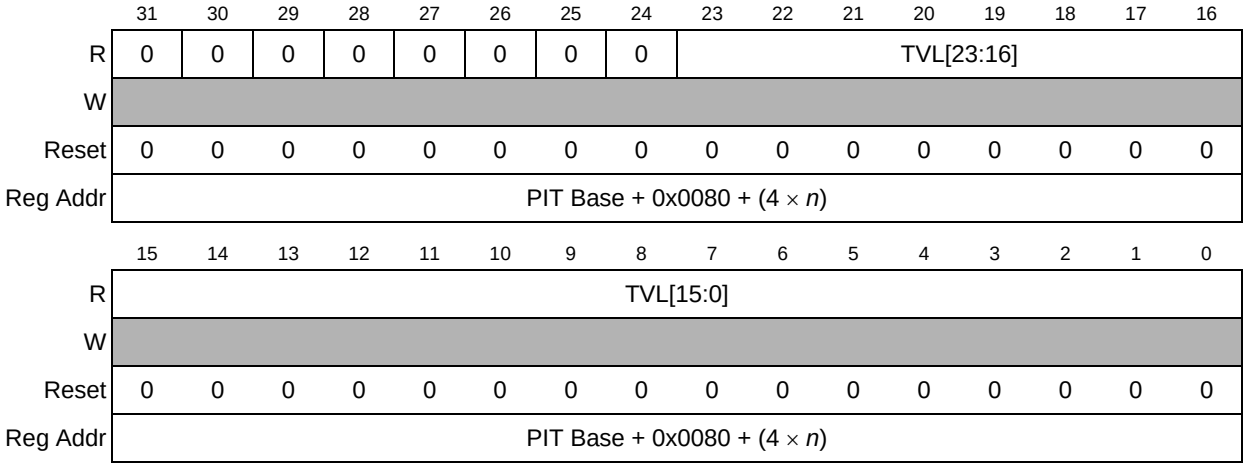


Figure 25-3. PIT Current RTI / Timer Values (TVAL*n*)

Table 25-3. TVAL*n* Field Descriptions

Bits	Name	Description
31–24	—	Reserved.
23–0	TVL[23:0]	Current timer value. These bits represent the current timer value. Note that the timer uses a down counter.

25.4.1.3 PIT Interrupt Flags Register (PITFLG)

This register holds the PIT interrupt flags. Timer 0 is the special RTI timer, which can be used to wake-up the device.

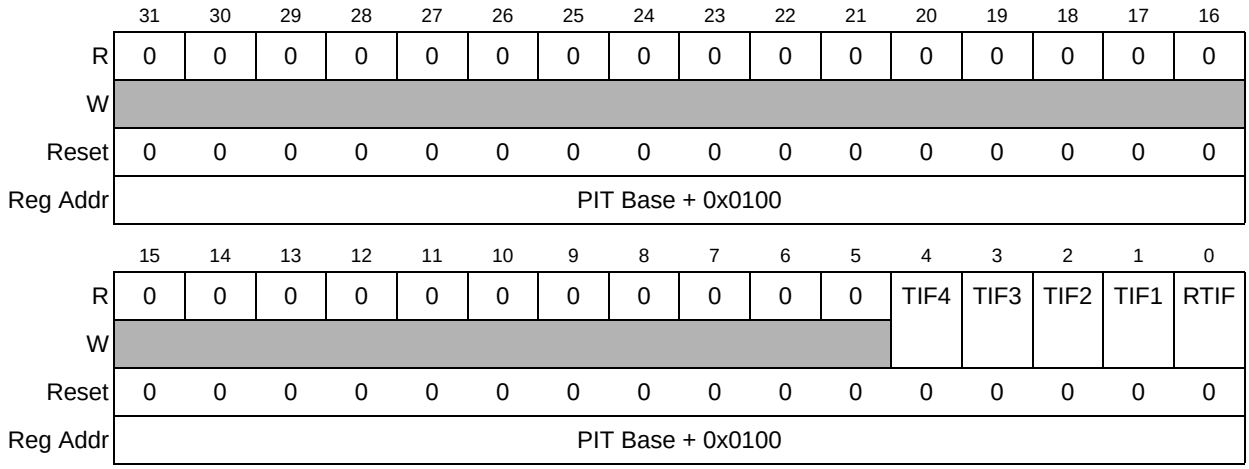


Figure 25-4. PIT Interrupt Flags Register (PITFLG)

Table 25-4. PITFLG Field Descriptions

Bits	Name	Description
31–5	—	Reserved.
4–1	TIF*n*	Real Time Interrupt Flags for Timer 1-4. TIFx is set to 1 at the end of the timer period. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (TIEx=1 and ISELx=1), TIFx causes an interrupt request. 0 Time-out has not yet occurred 1 Time-out has occurred
0	RTIF	Real time interrupt flag. RTIF is set to 1 at the end of the RTI period. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (RTIE=1), RTIF causes an interrupt request. 0 RTI time-out has not yet occurred 1 RTI time-out has occurred

25.4.1.4 PIT Interrupt Enable Register (PITINTEN)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	PIT Base + 0x0104															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	TIE4	TIE3	TIE2	TIE1	RTIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	PIT Base + 0x104															

Figure 25-5. PIT Interrupt Enable Register (PITINTEN)

Table 25-5. PITINTEN Field Descriptions

Bits	Name	Description
31–5	—	Reserved.
4–1	TIE*n*	Timer interrupt enable. 0 Interrupt requests from Timer *n* are disabled 1 Interrupt is requested when corresponding TIF*n* is set
0	RTIE	Real time interrupt enable. 0 Interrupt requests from RTI are disabled 1 Interrupt is requested when RTIF is set

When an interrupt is pending (TIF/RTIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF/RTIF flag must be cleared first.

25.4.1.5 PIT Interrupt/DMA Select Register (PITINTSEL)

This register determines whether a channel generates an interrupt or is used for DMA triggering.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	PIT Base + 0x0108															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	ISEL4	ISEL3	ISEL2	ISEL1	1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Reg Addr	PIT Base + 0x0108															

Figure 25-6. PIT Interrupt/DMA Select Register (PITINTSEL)

Table 25-6. PITINTSEL Field Description

Bits	Name	Description
31–5	—	Reserved.
4–1	ISEL*n*	Interrupt selector. 0 Timer *n* generates a DMA trigger [1] 1 Timer *n* generates an interrupt if corresponding TIE*n* if bit is set
0	—	Reserved.

[1] The corresponding DMA channel number is *n*-1. See Figure 17-3 on page 17-263.

25.4.1.6 PIT Timer Enable Register (PITEN)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	PIT Base + 0x010C															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	PEN10	PEN9	PEN8	PEN7	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	RTIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	PIT Base + 0x010C															

Figure 25-7. PIT Timer Enable Register (PITEN)

Table 25-7. PITEN Field Descriptions

Bits	Name	Description
31–11	—	Reserved.
10–1	PEN*n*	PIT timer enable 0 Timer *n* is disabled 1 Timer *n* is active
0	RTIEN	RTI timer enable 0 Timer *n* is disabled 1 Timer *n* is active

25.4.1.7 PIT Control Register (PITCTRL)

This register controls whether the clock for the timers 1-10 is enabled. The RTI timer (timer 0) runs on a separate clock which is controlled by the CRG.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	DOZE	MDIS	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Reg Addr	PIT Base + 0x0110															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	PIT Base + 0x0110															

Figure 25-8. PIT Control Register (PITCTRL)

Table 25-8. PITCTRL Field Descriptions

Bits	Name	Description
31–26	—	Reserved.
25	DOZE	Disable module in doze mode. This is used to disable timers 1–10 in Doze Mode. The RTI (timer 0) is not affected by this bit. 0 Clock for Timers 1–10 stays active in Doze Mode (default) 1 Clock for Timers 1–10 is turned off in Doze Mode
24	MDIS	Module disable. This is used to disable timers 1–10. The RTI (timer 0) is not affected by this bit. The module should be enabled before any setup is done. 0 Clock for Timers 1–10 is enabled 1 Clock for Timers 1–10 is disabled (default)
23–0	—	Reserved.

25.5 Functional Description

The PIT module has 11 timers: one RTI timer dedicated to wake-up the processor from low power modes and 10 timers for general-purpose use (e.g., interrupt generation, DMA triggering or ATD triggering).

25.5.1 General

As shown in Table 25-9, Timer 0 provides the RTI, primarily used to wake the system from low power modes. The four general purpose timers, Timer 1 to Timer 4 can be used to generate interrupts or trigger eDMA channels 0 to 3. Timer 5 to Timer 8 are used to trigger eDMA channels 4 to 7. Timers 9 and 10 are used for the ATD triggers SYSTRIG1 and SYSTRIG0 respectively.

Table 25-9. PIT Timer Feature Assignments

Timer	Interrupt Generation	eDMA / DMA Mux Trigger	ATD Trigger
0	Vector 0x16 [1]	—	—
1	Vector 0x13	Channel 0	—
2	Vector 0x14	Channel 1	—
3	Vector 0x15	Channel 2	—
4	Vector 0x16	Channel 3	—
5	—	Channel 4	—
6	—	Channel 5	—
7	—	Channel 6	—
8	—	Channel 7	—
9	—	—	SYSTRIG1
10	—	—	SYSTRIG0

[1] On mask set L49P devices, RTI only causes a wake-up and does not generate an interrupt via the INTC.

25.5.2 PIT Operating Mode Details

25.5.2.1 PIT Module Normal Mode

In order to perform timing operations, the module must be operating in normal mode. If the MCU is in run mode, the PIT module is in normal mode unless specifically disabled as described in Section 25.5.2.3, “PIT Module Disabled Mode.”

25.5.2.2 PIT Module Debug Mode

If the MCU enters debug mode, timers 1–10 are frozen. [1] When the MCU exits debug mode the timers resume counting. The RTI counter continues to run in debug mode if the BDMCTL[RSBCK] bit is set, or freezes if the bit is clear.

1. On mask set L49P devices, timers 1–10 continue to run in debug mode (RTI behaves as described).

25.5.2.3 PIT Module Disabled Mode

A mode that is independent of the MCU mode is the PIT disabled mode. At any time, the PITCTRL[MDIS] bit may be set to disable timers 1–10. This mode causes most of the PIT clocks to halt, which causes the module to draw minimal power while all other MCU peripheral modules may continue to operate normally. The RTI clock continues to run. All registers remain available to be accessed by the core via the peripheral bus. The PITCTRL[MDIS] bit is intended to be used when the module is not required in the application, not as a “module freeze” control bit.

25.5.2.4 PIT Module Doze Mode

If the MCU enters doze mode, the PITCTRL[DOZE] bit determines whether the PIT module continues to operate in normal mode or enters doze mode. If PITCTRL[DOZE] is clear, the PIT module will remain in normal mode. If PITCTRL[DOZE] is set, the module clocks (except for the RTI clock) are stopped but registers remain accessible, thus offering power savings over operation in normal mode.

The CLKSEL[RTIDOZE] bit in the CRG module determines whether the RTI stops in doze mode. Refer to Section 4.3.6.10.4, “Doze Mode,” on page 4-71 for details.

When the MCU exits doze mode or the PITCTRL[DOZE] bit is cleared, the PIT clocks are turned on again and normal operation is resumed.

25.5.2.5 PIT Module Stop Mode

If the MCU enters stop mode, most chip clocks stop and therefore most module functions stop. When the MCU enters stop mode, the peripheral bus clocks halt, and thus timers 1–10 clocks are frozen to offer maximum power saving. After exiting stop mode, the clocks are turned on again and the timers resume normal operation.

The RTI clock may continue to run in stop mode to allow the RTI counter to bring the device out of stop mode after the appropriate time period. The behavior of the RTI clock in stop mode is controlled via configuration of the CRG module. If the CRG CLKSEL[PSTP] bit is set the oscillator continues to run (pseudo-stop), and if PLLCTL[PRE] is also set the RTI clock is enabled, thus allowing the RTI counter to continue operation. Refer to Section 4.3.6.10.5, “Stop Mode,” on page 4-75 and Table 4-15 on page 4-81 for details.

25.5.3 Timer / RTI

The timers generate triggers at periodic intervals, when enabled. The load value is taken from the TLVAL*n* registers, then decremented until 0 is reached. This creates a trigger and then they load their respective start value again. Each time a timer reachers 0, it will generate a trigger pulse, and set the interrupt flag.

All interrupts can be enabled or masked by setting the PITINTEN[TIE/RTIE] bits and selecting interrupts in the PITINTSEL register. A new interrupt can be generated only after the previous one is cleared. Because the RTI counter uses either the oscillator or PLL clock while the rest of the PIM module, including the peripheral bus interface, uses the lower frequency peripheral bus clock, best case interrupt service routine latency and clock ratios must be taken into account when determining the RTI period. Any RTI load value less than or equal to 32 will result in an RTI period that is too short for the CPU to service it.

The current counter value of a timer can be read via the TVAL*n* registers. The value of TVAL0 can be delayed considerably, as it is synchronized to the bus clock from the RTI clock.

As shown in Figure 25-9, the counter period can be restarted by disabling and then enabling the timer with the PITEN register.

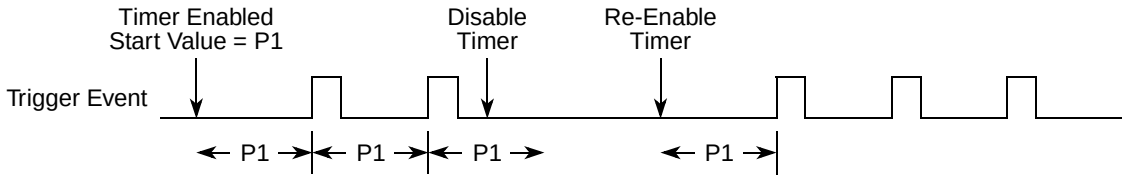


Figure 25-9. Stopping and Starting a Timer

As shown in Figure 25-10, the counter period of a running timer can be modified by first disabling the timer, setting a new load value and then re-enabling the timer.

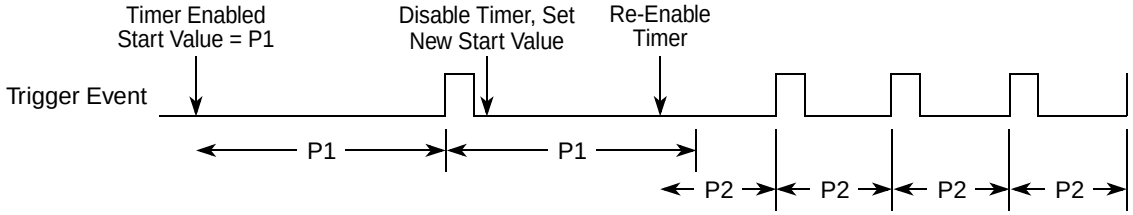


Figure 25-10. Modifying a Running Timer Period

As shown in Figure 25-11, the counter period may be changed without restarting the timer by writing the TLVAL register with a new value. This value is loaded after the next trigger event.

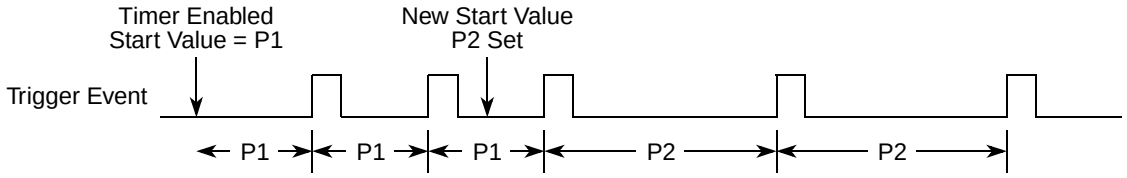


Figure 25-11. Dynamically Setting a New Load Value

25.5.4 Interrupts

25.5.4.1 General

The interrupts generated by the PIT are listed in Table 25-10. Refer to Table 25-9 and Table 6-2 on page 6-85 for related vector numbers, addresses and priorities.

Table 25-10. PIT Interrupt Sources

Interrupt Source	Local Enable
Real Time Interrupt	PITINTEN[RTIE]
Timer Interrupts	PITINTEN[TIE[4:1]]

25.5.4.2 Description of Interrupt Operation

25.5.4.2.1 Real Time Interrupt

The PIT generates a real time interrupt when the programmed RTI period elapses. The RTI interrupt is disabled within the PIT by clearing the PITINTEN[RTIE] register bit. The real time interrupt flag (PITFLG[RTIF]) is set when a timeout occurs, and is cleared by writing a 1 to the RTIF bit. On mask L49P devices, the RTI cannot be used as a general purpose interrupt, as it is only used by the CRG for periodic wake-up from low power modes. On later mask set devices, the RTI generates an interrupt via the INTC (and shares an interrupt vector with Timer 4). RTI wake-up is enabled only if the CRG PLLCTL[PRE] bit is set and the RTI interrupt is enabled. Refer to Section 4.3.6.6, “Real Time Interrupt (RTI),” on page 4-66 for more information.

25.5.4.2.2 Timer Interrupts

The PIT can generate timer interrupts via the first four timer channels when the selected interrupt period elapses. Timer interrupts are disabled within the PIT by clearing the PITINTEN[TIE*n*] register bits. The timer interrupt flags (PITFLG[TIF*n*] register bits) are set when a timeout occurs on the associated timer, and are cleared by writing a 1 to that TIF*n* bit. To activate a timer interrupt it must also be switched from trigger mode into interrupt mode, using the PITINTSEL register.

The timer interrupts are general-purpose in the sense that they are handled via the INTC module with a unique interrupt vector for each timer (note that Timer 4 shares a single interrupt vector with the RTI timer). Refer to Chapter 10, “Interrupt Controller Module (INTC),” and Table 6-2 on page 6-85 for more information.

25.6 Initialization / Application Information

Although the PIT provides significant functionality to the system via real-time interrupts alone, interaction with the ATD module(s) and the eDMA Controller, via the DMAMux, allows for enhanced system configurations in time-sensitive applications. Refer to Figure 25-12 for an illustration of all available PIT interconnections to other peripherals.

25.6.1 Example Configuration

In this example configuration:

- the PIT clock (f_{IPS}) has a frequency of 25 MHz
- the RTI clock (OSCCLK) has a frequency of 10 MHz
- the RTI timer is set up to create a wake-up interrupt every 500 ms
- timer 1 is set up to create an interrupt every 5.12 ms
- timer 8 is set up to create a trigger event every 30 ms.

First, the PIT module must be activated by writing a 0 to the MDIS bit in the PITCTRL register.

The 25 MHz f_{IPS} clock frequency equates to a clock period of 40 ns and the 10 MHz RTI clock frequency equates to a clock period of 100 ns. Therefore the RTI timer period is 500 ms ÷ 100 ns = 5,000,000 cycles,

the timer 1 period is 5.12 ms ÷ 40 ns = 128,000 cycles and the timer 8 period is 30 ms ÷ 40 ns = 750,000 cycles. The value of the appropriate TLVAL*n* register is calculated using the equation:

$$\text{TLVAL}n = \left(\frac{\text{Desired time period}}{\text{Peripheral bus clock period}}\right) - 1 \qquad \textit{Eqn. 25-1}$$

This means that TLVAL0 is written with 0x004C_4B3F, TLVAL1 with 0x0001_F3FF and TLVAL8 with 0x000B_71AF.

To generate the interrupt, the interrupt line must be enabled by writing a 1 to the RTIE bit in the PITINTEN register. There is no need to modify PITINTSEL, as the RTI timer is always used for interrupts and never for trigger events. To start the RTI, PEN0 in the PIT Timer Enable register (PITEN) is set.

The interrupt for Timer 1 is enabled by setting TIE1 in the PITINTEN register and the Interrupt/DMA selector ISEL1 (in PITINTSEL) is set to 1. The timer is started by writing a 1 to bit PEN1 in the PITEN register.

Timer 8 is used only for triggering. Only timers 0–4 have interrupt capability. Therefore Timer 8 is started by writing a 1 to bit PEN8 in the PITEN register.

It is also possible to setup all timers and start them simultaneously by writing to the PITEN register. However, in this case the RTI does not start in synchronization, as it is running on a separate clock.

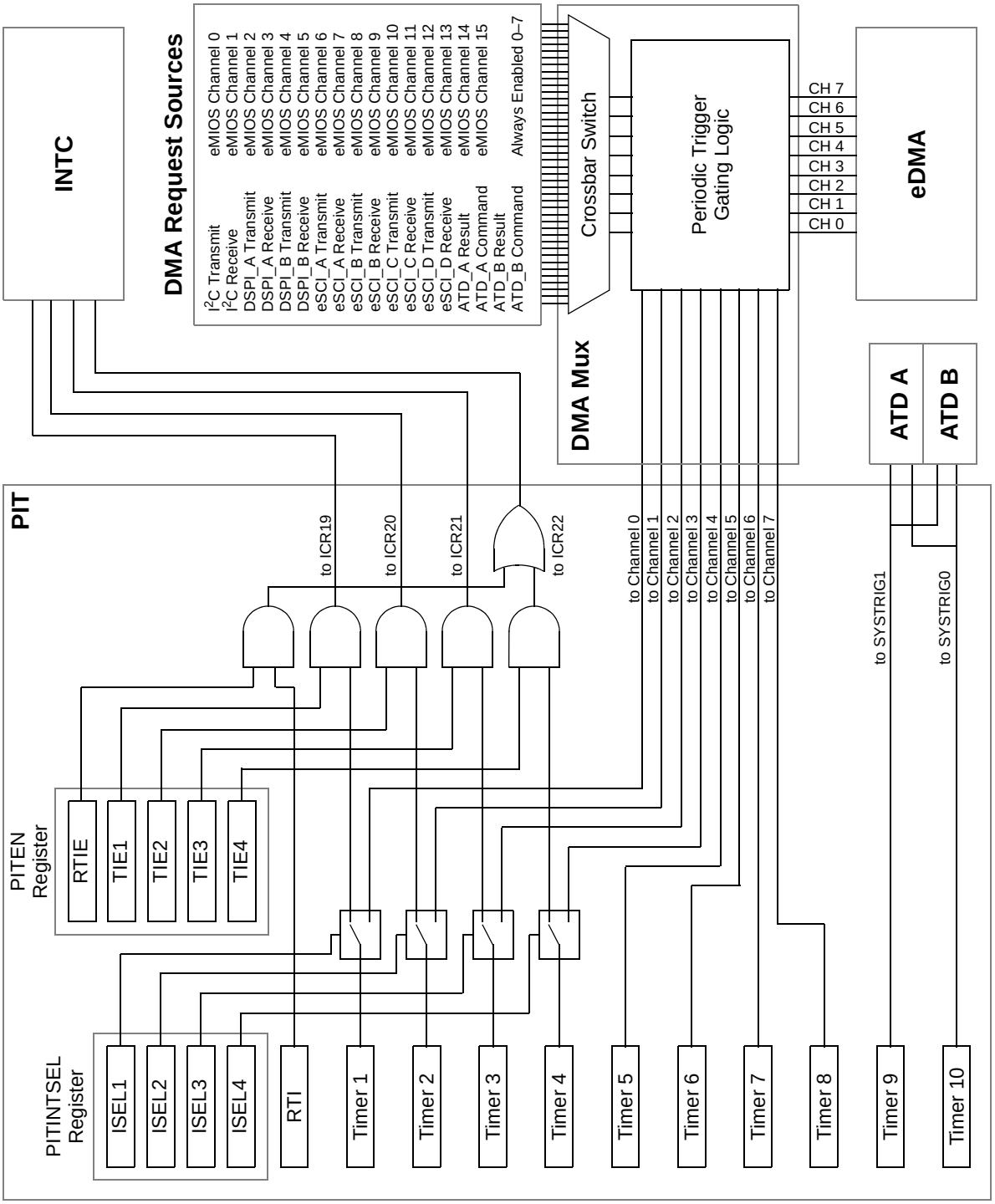


Figure 25-12. PIT Trigger Connections to Other Modules

Chapter 26
System Services Module (SSM)

26.1 Overview

The System Services Module (SSM), shown in Figure 26-1, contains information on device configuration, the status of the eDMA controller, control of how accesses to reserved memory space is handled, and control of how Port F is used for debug purposes. [1]

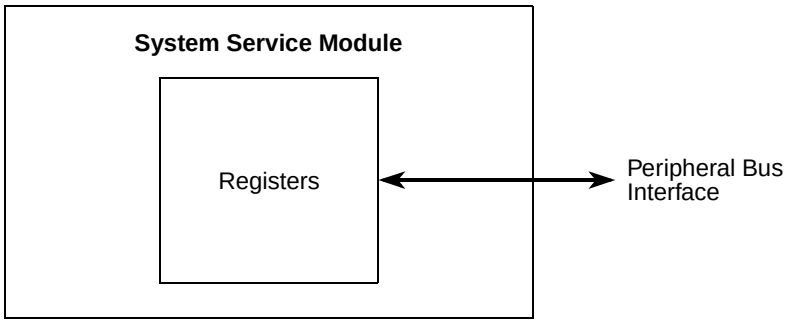


Figure 26-1. SSM Block Diagram

26.2 Features

The SSM includes these distinctive features:

- System Configuration and Status
  - Memory sizes and status
  - Security status
  - Device mode
  - eDMA status
  - Reserved address space protection
- Debug status port configuration

26.3 Modes of Operation

The SSM operates identically in all system modes.

1. On mask set L49P devices, the SSM also includes registers to detect the source of the last wake-up event (see Section 26.4.1.3) and control the swapping of Port C and H registers in (see Section 26.4.1.5). On later mask set devices, the wake-up source register is not needed and the Port C/H swapping is handled via the PIM.

26.4 Memory Map / Register Definition

This section provides a detailed description of all memory-mapped registers in the SSM. 8-, 16- and 32-bit reads and writes are allowed to all registers, provided the read or write does not access any address defined as reserved. Any access to the offset range 0x0020 through 0x3FF will cause a bus abort.

Table 26-1 shows the memory map for the SSM. Note that all addresses are offsets; the absolute address is calculated by adding the module base address specified in Chapter 8, "Device Memory Map," to the base offset of the SSM register.

Table 26-1. SSM Memory Map

SSM Offset	Register Description	Access
0x0000	Reserved	–
0x0002	SSM Current System Status Register (STATUS)	Read
0x0004	SSM System Memory Configuration Register (MEMCONFIG)	Read
0x0006	Reserved	–
0x0008	SSM Wake-up Source Register (WAKEUP) [1] or SSM Error Configuration Register (ERROR) [2]	Read/Write
0x000A	Reserved [2]	–
0x000C	SSM Port Select Register (PORTSEL) [3]	Read/Write
0x000E	SSM Debug Status Port Control Register (DEBUGPORT) [4]	Read/Write
0x0010 — 0x3FFF	Reserved	–

1 L49P mask set devices only (32-bit register).
2 Non-L49P mask set devices only (16-bit register).
3 On non-L49P mask set devices, this register is not implemented and the offset is reserved.
4 On L49P mask set devices, this register is not implemented and the offset is reserved.

26.4.1 Register Descriptions

The following memory-mapped registers are available in the SSM. Those bits shaded for writes are read-only, meaning that writes will have no effect. To ensure compatibility with future implementations, these bits should be ignored when read and written as zero.

26.4.1.1 SSM Current System Status Register (STATUS)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	NEXUS [1]		EIM_ACK [2]	EIM_SIZE [2]		SEC	MODE		DMAIDLE	DMAACTCH			
W																
Reset	0	0	0	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0	0	0	0	0
Reg Addr	SSM Base + 0x0002															

1 Not implemented on mask set L49P devices.
2 Not implemented on mask set L49P or L61W devices.

Figure 26-2. SSM Current System Status Register (STATUS)

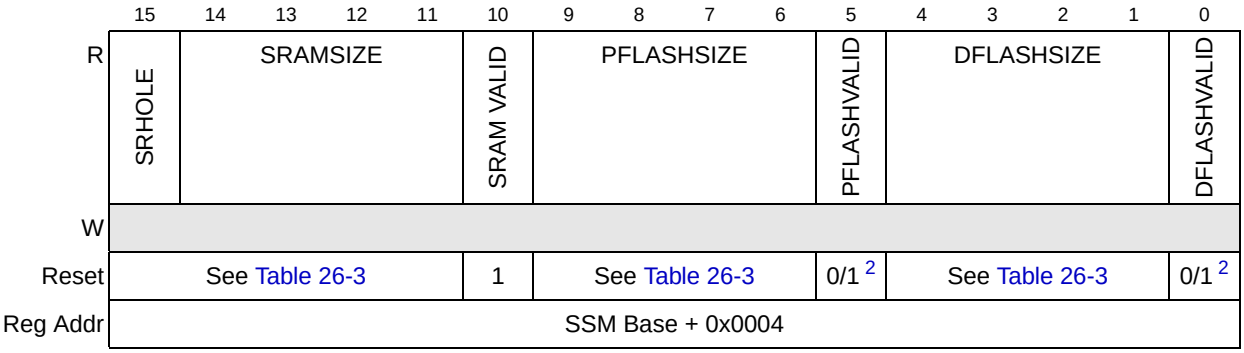
Table 26-2. STATUS Field Descriptions

Bits	Name	Description
15–13	—	Reserved, read as zero, writes are ignored.
12–11	NEXUS [1]	Nexus Status. This field reflects the current status of the Nexus port. 00 No Nexus hardware attached 01 No Nexus hardware attached 10 Nexus hardware attached to Primary port 11 Nexus hardware attached to Secondary port
10	EIM_ACK [2]	EIM Acknowledge. This bit reflects the configuration of the EIM acknowledge hardware configuration. 0 The EIM global chip select uses auto acknowledge 1 The EIM global chip select uses external acknowledge
9–8	EIM_SIZE [2]	EIM Port Size. This field reflects the configuration of the EIM port size latched at $\overline{\text{RESET}}$. 00 32-bit port size 01 8-bit port size 10 16-bit port size 11 16-bit port size
7	SEC	Security Status. This bit reflects the current security state of the program Flash. 1 Program Flash is secured 0 Program Flash is not secured
6–5	MODE[1:0]	Device Mode. This field reflects the current mode of the device. 00 Single-Chip 01 Expanded 10 Data Flash Boot 11 Single-Chip
4	DMAIDLE	eDMA is idle. This bit reflects the current status of the eDMA. It is primarily used for debug purposes. 0 eDMA is performing a bus read or write 1 eDMA is idle
3–0	DMAACTCH [3:0]	Active eDMA channel. This field identifies which eDMA channel, if any, is currently executing bus transactions. Note that this value is only valid if DMAIDLE = 0. These bits are primarily used for debug purposes. *nnnn* Number of active channel

1 Not implemented on mask set L49P devices.

2 Not implemented on mask set L49P or L61W devices.

26.4.1.2 SSM System Memory Configuration Register (MEMCONFIG)



1 Not implemented on mask set L49P and L47W mask set devices.

2 Reset value depends on the Chip Mode and security status of the Program Flash.

Figure 26-3. SSM System Memory Configuration Register (MEMCONFIG)

Table 26-3. MEMCONFIG Field Descriptions

Bits	Name	Description
15	SRHOLE	SRAM Map Hole. This bit indicates that the upper quarter of the memory range indicated by the SRAMSIZE field is not used.
14–11	SRAMSIZE[3:0]	SRAM size. This field identifies the size of the on-chip SRAM memory. SRHOLE / SRAMSIZE / Size 0 / 0000–0011 / No SRAM 0 / 0100 / 4 Kbytes 0 / 0101 / 8 Kbytes 0 / 0110 / 16 Kbytes (MAC71*x*2 devices) 0 / 0111 / 32 Kbytes (MAC71*x*1 devices) 0 / 1000 / 64 Kbytes 1 / 0000–0011 / No SRAM 1 / 0100 / 3 Kbytes 1 / 0101 / 6 Kbytes 1 / 0110 / 12 Kbytes 1 / 0111 / 24 Kbytes 1 / 1000 / 48 Kbytes (MAC71*x*6 devices)
10	SRAMVALID	SRAM valid. This bit identifies whether or not the on-chip SRAM is visible in the system memory map. 0 SRAM is not visible 1 SRAM is visible
9–6	PFLASHSIZE[3:0]	Program Flash size. This field identifies the size of the on-chip program Flash memory. 0000–1000 No program Flash 1001 64 Kbytes 1010 128 Kbytes 1011 256 Kbytes (MAC71*x*2 devices) 1100 512 Kbytes (MAC71*x*1 devices) 1101 1 Mbytes (MAC71*x*6 devices) 1110–1111 No program Flash

Table 26-3. MEMCONFIG Field Descriptions (continued)

Bits	Name	Description
5	PFLASHVALID	Program Flash valid. This bit identifies whether or not the on-chip program Flash is visible in the system memory map. 0 Program Flash is not visible 1 Program Flash is visible
4–1	DFLASHSIZE[3:0]	Data Flash size. This field identifies the size of the on-chip data Flash memory. 0000–0011 No data Flash 0100 4 Kbytes 0101 8 Kbytes 0110 16 Kbytes 0111 32 Kbytes (MAC71*xx* devices) 1001–1111 No data Flash
0	DFLASHVALID	Data Flash valid. This bit identifies whether or not the on-chip data Flash is visible in the system memory map 0 Data Flash is not visible 1 Data Flash is visible

26.4.1.3 SSM Wake-up Source Register (WAKEUP)

This register can be read after exiting a low power mode to determine the source(s) of a system wake-up. Note that multiple sources may be asserted simultaneously. To clear a wake-up source, the interrupt/wake-up source must be cleared in the specific module that generated it.

NOTE

This register is present only on mask set L49P devices. In later devices it is not needed (all wake-up events are handled via the INTC and thus have unique vector numbers) and is not present.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	SSM Base + 0x0008 [1]															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	SOURCE													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	SSM Base + 0x0008 [1]															

[1] L49P mask set devices only. On later devices, this offset is occupied by the ERROR register.

Figure 26-4. SSM Wake-up Source Register (WAKEUP)

Table 26-4. WAKEUP Field Descriptions

Bits	Name	Description
31–14	—	Reserved.
13–0	SOURCE [13:0]	These read-only bits specify which wake-up source(s) are currently asserted. Table 26-5 identifies wake-up source bit assignments.

Table 26-5. WAKEUP SOURCE Field Detail

Bit	Wake-up Source
SOURCE0	Periodic Interrupt Timer Channel 0 (RTI)
SOURCE1	Interrupt (from the INTC)
SOURCE2	FlexCAN A Bus Activity
SOURCE3	FlexCAN B Bus Activity
SOURCE4	FlexCAN C Bus Activity
SOURCE5	FlexCAN D Bus Activity
SOURCE6	External Wake-up on Port A
SOURCE7	External Wake-up on Port B
SOURCE8	External Wake-up on Port C
SOURCE9	External Wake-up on Port D
SOURCE10	External Wake-up on Port E
SOURCE11	External Wake-up on Port F
SOURCE12	External Wake-up on Port G
SOURCE13	External Wake-up on Port H

26.4.1.4 SSM Error Configuration Register (ERROR)

This read/write register controls error handling for accesses to reserved off-platform peripheral addresses. The protected offset range for each module is listed in Table 26-7.

NOTE

This register is not present on mask set L49P devices.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BABORT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	SSM Base + 0x0008 [1]															

1 This offset is occupied by the WAKEUP register on L49P mask set devices.

Figure 26-5. SSM Error Configuration Register (ERROR)

Table 26-6. ERROR Field Descriptions

Bits	Name	Description
15–1	—	Reserved.
0	BABORT	This bit enables bus aborts on illegal accesses to off-platform peripherals. 0 Illegal accesses to off-platform peripherals do not cause an exception. 1 Illegal accesses to off-platform peripherals cause prefetch or data abort exceptions.

Table 26-7. ERROR[BABORT] Protected Address Ranges

Module	Offset
VREG	0x0008 – 0x3FFF
CRG	0x0010 – 0x3FFF
CFM registers	0x0003 – 0x0007 0x000C – 0x000F 0x001C – 0x001F 0x0021 – 0x0023 0x0025 – 0x0043 0x0047 – 0x0049 0x004C – 0x3FFF
DMA Mux	0x0010 – 0x3FFF
PIM	0x0200 – 0x03BF 0x03CF – 0x3FFF
ATD A	0x0004 – 0x000B
ATD B	0x0018 – 0x3FFF

Module	Offset
eMIOS	0x0014 – 0x3FFF
eSCI A	0x0020 – 0x3FFF
eSCI B	0x0020 – 0x3FFF
eSCI C	0x0020 – 0x3FFF
eSCI D	0x0020 – 0x3FFF
DSPI A	0x00D0 – 0x3FFF
DSPI B	0x00D0 – 0x3FFF
FlexCAN A	0x000C – 0x000F
FlexCAN B	0x0034 – 0x007F
FlexCAN C	0x0480 – 0x087F
FlexCAN D	0x0980 – 0x3FFF
I^2C	0x0008 – 0x3FFF
PIT	0x0200 – 0x3FFF
SSM	0x0020 – 0x3FFF

Behavior of on-platform peripherals (eDMA, AIPS, EIM and MCM) are not affected, as they will produce prefetch or data abort exceptions for illegal accesses regardless of the value of the BABORT bit. In addition, some peripherals allow access to certain registers only in supervisor mode. Refer to the individual module chapters for more details.

26.4.1.5 SSM Port Select Register (PORTSEL)

This register allows software to switch Port C and H in the memory map to ease software portability between device variations (in particular, the MAC7101, MAC7106, MAC7111 and MAC7116). Please refer to Chapter 18, “Port Integration Module (PIM),” for detailed information on the memory map of Port C and Port H.

NOTE

This register is implemented only on L49P mask set devices. Refer to Section 18.5.1.7 on page 18-290 for later devices.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PORTHSEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	SSM Base + 0x000C															

Figure 26-6. SSM Port Select Register (PORTSEL)

Table 26-8. PORTSEL Field Descriptions

Bits	Name	Descriptions
15–1	—	Reserved.
0 [1]	PORTHSEL	Port H select. This bit configures the locations of Port C and H in the memory map. 0 Maintain the standard port order in the Port Integration Module memory map 1 Swap Port C and Port H in the Port Integration Module memory map

[1] If Port H is not available on a particular device, the PORTHSEL bit should always be clear.

26.4.1.6 SSM Debug Status Port Control Register (DEBUGPORT)

This read/write register is used to select Port F as an optional debug status port and the debug data set to be transmitted. Refer to Section 26.6.4, “Using the DEBUGPORT Register,” for details on how use of this register the impacts GPIO and peripheral functions.

NOTE

This functionality is not present on mask set L49P devices.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DEBUG_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	SSM Base + 0x000E (this offset is reserved on L49P mask set devices)															

Figure 26-7. SSM Debug Status Port Control Register (DEBUGPORT)

Table 26-9. DEBUGPORT Field Descriptions

Bits	Name	Description
15–3	—	Reserved.
2–0	DEBUG_ MODE[2:0]	This field selects the alternate debug functionality for Port F. Refer to Table 26-10 for descriptions of the debug data provided in each mode. 000 No alternate function 001 Mode 1 selected 010 Mode 2 selected 011 Mode 3 selected 100 Mode 4 selected 101 Mode 5 selected 110 Mode 6 selected 111 Reserved

Table 26-10. Port F Debug Status Mode Signal Assignments

Port F Pin	Function					
	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
PF0	System is entering STOP mode	System is entering STOP mode	STATUS0	MEMCONFIG0	Reserved	Reserved
PF1	System is ready to enter STOP mode	Platform has entered STOP mode	STATUS1	MEMCONFIG1	Reserved	Reserved
PF2	System has entered STOP mode	ATD A has entered STOP mode	STATUS2	MEMCONFIG2	Reserved	Reserved
PF3	System has entered DOZE mode	ATD B has entered STOP mode	STATUS3	MEMCONFIG3	Reserved	Reserved
PF4	System has entered DEBUG mode	eSCI A has entered STOP mode	STATUS4	MEMCONFIG4	Reserved	Reserved
PF5	Core is held (not running)	eSCI B has entered STOP mode	STATUS5	MEMCONFIG5	Reserved	Reserved
PF6	VREG wake-up RC oscillator clock	eSCI C has entered STOP mode	STATUS6	MEMCONFIG6	JTAG lockout recovery started	Reserved
PF7	VREG wake-up RC oscillator clock running	eSCI D has entered STOP mode	STATUS7	MEMCONFIG7	JTAG lockout recovery w/ PTIMER load started	Reserved
PF8	External wake-up RC oscillator clock	FlexCAN A has entered STOP mode	STATUS8	MEMCONFIG8	JTAG lockout recovery running	Reserved
PF9	External wake-up RC oscillator clock running	FlexCAN B has entered STOP mode	STATUS9	MEMCONFIG9	Reserved	Reserved
PF10	Reserved	FlexCAN C has entered STOP mode	STATUS10	MEMCONFIG10	Reserved	Reserved
PF11	Reserved	FlexCAN D has entered STOP mode	STATUS11	MEMCONFIG11	Reserved	Reserved
PF12	Reserved	PIT has entered STOP mode	STATUS12	MEMCONFIG12	Reserved	Reserved
PF13	Reserved	I^2C has entered STOP mode	STATUS13	MEMCONFIG13	Reserved	Reserved
PF14	Reserved	DSPI A has entered STOP mode	STATUS14	MEMCONFIG14	Reserved	Reserved
PF15	Reserved	DSPI B has entered STOP mode	STATUS15	MEMCONFIG15	Reserved	Reserved

26.5 Functional Description

This section provides a complete functional description of the System Services Module. The primary purpose of the SSM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

26.5.1 System Configuration / Status

The SSM provides three or four registers to aid in configuring software and debugging a system:

- STATUS
- MEMCONFIG
- ERROR (mask sets later than L49P only)
- DEBUGPORT (mask sets later than L49P only)
- PORTSEL (L49P mask set only)

The STATUS register is a 16-bit read-only register that reflects the current status of the eDMA, and is primarily used for debug. By reading the STATUS register, it can be determined which eDMA channel, if any, is currently active. This information should not be used as part of the execution of application code, but may be useful for debug purposes. Bits [15:5] of the STATUS register are reserved for future use, and a read from these bits may return any value. Therefore, it is necessary to mask out these bits when reading from this register (refer to Section 26.6.1, “Using the STATUS Register”).

The MEMCONFIG register contains a value which describes the memory configuration of the device. Table 26-11 shows examples of the contents of this register for several devices in the MAC7100 family. For example, the MAC7101 device has the memory configuration of 512 Kbytes program Flash, 32 Kbytes data Flash and 32 Kbytes SRAM. This configuration results in the following MEMCONFIG register value (assuming all memories are visible in the memory map):

- MEMCONFIG[14:11]SRAMSIZE`0b0111` 32 Kbytes
- MEMCONFIG[9:6]PFLASHSIZE`0b1100`512 Kbytes
- MEMCONFIG[4:1]DFLASHSIZE`0b0111` 32 Kbytes
- MEMCONFIG[15:0]`0x3F2F`

Table 26-11. MEMCONFIG Field Details

Device	MEMCONFIG Contents
MAC71*x*1	0b0011_1x11_00x0_111x
MAC71*x*2	0b0011_0x10_11x0_111x
MAC71*x*6	0b0100_0x11_01x0_111x

The ERROR register (mask sets later than L49P only) is 16-bit read/write register that is used to control the handling of illegal accesses to off-platform peripherals. After reset, the BABORT bit is clear, and no protection against illegal accesses is performed. If a read or write cycle is executed to a reserved area within any IPS peripheral memory map, the results will be undefined (read cycles will retrieve undefined data, write cycles may cause unexpected behavior). In order to protect against errant code, initialization

routines can set the BABORT bit, which enables ABORT errors when reserved areas within IPS peripherals are accessed.

The DEBUGPORT register (mask sets later than L49P only) is a 16-bit read/write register that is used to configure the optional debug status port on Port F pins.

The PORTSEL register (L49P mask set only) is a 16-bit read/write register that is used to switch functionality on various special ports in the system. The memory map of Port C and Port H in the Port Integration Module may be swapped to maintain binary compatibility between custom and standard parts by setting or clearing the PORTHSEL bit. Bits [15:1] of the PORTSEL register are reserved for future use, and a read from these bits may return any value. Therefore, it is necessary to mask out these bits when reading from this register (refer to Section 26.6.6, “Using the PORTSEL Register”). For mask sets later than L49P, this function is controlled within the PIM module; refer to Section 18.5.1.7 on page 18-290 for more information.

26.5.2 Wake-up Source Identification

The WAKEUP register (L49P mask set only) is a 16-bit read-only register that is designed to reduce the overall latency when waking up the system from a low power mode. Each of the 14 wake-up sources in the system has a corresponding bit (see Table 26-5) in the WAKEUP register, which can be used to identify the source(s) of a system wake-up with a single bus access. Note that multiple sources may be active at the same time. To clear a wake-up source, you must clear it in the originating module (i.e. Clear the FlexCAN_A wake-up on bus activity flag by writing to the appropriate register(s) in the FlexCAN_A module). This function is not needed on mask sets later than L49P, as all wake-up sources are routed through the INTC and thus have unique vector numbers to identify them.

26.6 Initialization / Application Information

26.6.1 Using the STATUS Register

Bits [15:5] of the STATUS register are reserved for future use; therefore these bits should be masked out after reading this register, as shown in the following code example:

```
In File registers.h:
        #define SSM_BASE_ADDRESS     0xFC008000     /* Example only! */
        /* Following example assumes short is 16-bits */
        volatile unsigned short *STATUS = (volatile unsigned short *)
(SSM_BASE_ADDRESS+0x0002);

In File main.c:
        #include "registers.h"
                 :
                 :
        unsigned char dma_active;
        unsigned char dma_channel;
        /* Do single read of STATUS register to "capture" state */
        dma_channel = (unsigned char) *STATUS & 0x001f;
        dma_active  = dma_channel >> 4;
        dma_channel = dma_channel & 0x0f;
```

Because the eDMA executes independently of the processor core, using the STATUS register in software is discouraged, as the latency for the execution of the above code may be longer than the eDMA is actually active. Use of this register is intended for debug purposes only.

26.6.2 Using the MEMCONFIG Register

Bit 15 of the MEMCONFIG register is reserved for future use; therefore you should mask this bit out when reading this register, as shown in the following code example:

```
In File registers.h:
        #define SSM_BASE_ADDRESS     0xFC008000     /* Example only! */
        /* Following example assumes short is 16-bits */
        volatile unsigned short *MEMCONFIG = (volatile unsigned short *)
(SSM_BASE_ADDRESS+0x0004);

In File main.c:
        #include "registers.h"
                 :
                 :
        unsigned short memconfig;
        unsigned char sram_valid;
        unsigned char sram_size;
        unsigned char pflash_valid;
        unsigned char pflash_size;
        unsigned char dflash_valid;
        unsigned char dflash_size;
        memconfig    = *MEMCONFIG & 0x7fff; /*bit 15 is masked*/
        sram_valid   = (unsigned char) (memconfig & 0400) >> 10;
        pflash_valid = (unsigned char) (memconfig & 0020) >>  5;
        dflash_valid = (unsigned char) (memconfig & 0001) >>  0;
        sram_size    = (unsigned char) (memconfig & 7800) >> 11;
        pflash_size  = (unsigned char) (memconfig & 02C0) >>  6;
        dflash_size  = (unsigned char) (memconfig & 001e) >>  1;
```

26.6.3 Using the ERROR Register

Additional information to be provided later.

26.6.4 Using the DEBUGPORT Register

The debug status function overrides the PIM and eMIOS module configurations for Port F. If DEBUGPORT[DEBUG_MODE] is set to any non-zero value, PF[15:0] are driven with debug status information. When DEBUG_MODE = 0b000 PF[15:0] pin configuration is controlled by the PIM and eMIOS modules.

When DEBUG_MODE is non-zero and the MCU enters a low-power mode, debug status information continues to be driven onto the pins.

Additional information to be provided later.

26.6.5 Using the WAKEUP Register

NOTE

This register is present only on mask set L49P devices. In later devices it is not needed (all wake-up events are handled via the INTC and thus have unique vector numbers) and is not present.

Bits [31:14] of the WAKEUP register are reserved for future use; therefore, you should mask these bits out when reading this register, as shown in the following code example:

```
In File registers.h:
        #define SSM_BASE_ADDRESS     0xFC008000     /* Example only! */
        /* Following example assumes int is 32-bits */
        volatile unsigned int *WAKEUP = (volatile unsigned int *) (SSM_BASE_ADDRESS+0x0008);

In File main.c:
        #include "registers.h"
                :
                :
        unsigned int sources;
        sources = *WAKEUP & 0x0003ffff; /*bits [31:14] are masked*/
```

Alternatively, for better access to individual sources, the following code will also work:

```
In File registers.h:
        #define SSM_BASE_ADDRESS     0xFC008000     /* Example only! */
        /* Following example assumes int is 32-bits */
        typedef union {
                unsigned int regval;
                struct {
                        unsigned int rti    :1;
                        unsigned int can0   :1;
                        unsigned int can1   :1;
                        unsigned int can2   :1;
                        unsigned int can3   :1;
                        unsigned int port_a :1;
                        unsigned int port_b :1;
                        unsigned int port_c :1;
                        unsigned int port_d :1;
                        unsigned int port_e :1;
                        unsigned int port_f :1;
                        unsigned int port_g :1;
                        unsigned int port_h :1;
                        } bitval;
                } WAKEUP;
        volatile WAKEUP *wakeup_reg = (volatile WAKEUP *) (SSM_BASE_ADDRESS+0x0008);

In File main.c:
        #include "registers.h"
                :
                :
        unsigned int rti_wakeup;
        rti_wakeup = wakeup_reg->bitval.rti;
```

26.6.6 Using the PORTSEL Register

NOTE

This register is implemented only on L49P mask set devices. Refer to Section 18.7.6.1 on page 18-315 for later devices.

Bits [15:1] of the PORTSEL register are reserved for future use; therefore you should mask these bits out when reading this register, as shown in the following code example:

```
In File registers.h:
        #define SSM_BASE_ADDRESS     0xFC008000     /* Example only! */
        /* Following example assumes short is 16-bits, int is 32-bits */
        typedef union {
                unsigned short regval;
                struct {
                        unsigned int porth_select :1;
                        } bitval;
                } PORTSEL;
        volatile PORTSEL *portsel_reg = (volatile PORTSEL *) (SSM_BASE_ADDRESS+0x000C);

In File main.c:
        #include "registers.h"
                :
                :
        /* Swap port assignments... */
        /* Set only the Port H select bit! */
        portsel_reg->bitval.porth_select = 1;
```

Appendix A
Debug Interface

A.1 Overview

The MAC7100 family of devices offers debugging with the Embedded ICE (E-ICE) and a NEXUS 2 Plus interface of the core. Embedded ICE offers debug features such as setting breakpoints or watchpoints, modifying and reading memory contents. Embedded ICE uses the standard JTAG serial interface and Test Access Port (TAP) and is compatible with existing ARM7 tool chains. The NEXUS interface provides real time program trace capability and also uses the JTAG port, as well as providing an auxiliary port.

The Nexus interface is enabled based on the level latched on the PF1/eMIOS1/NEXPR pin during the assertion of $\overline{\text{RESET}}$ and the $\overline{\text{EVTI}}$ / $\overline{\text{EVTI}}$' signal. If the level latched on NEXPR is high, the Nexus interface can be enabled based on the state of the $\overline{\text{EVTI}}$ / $\overline{\text{EVTI}}$' pin at any time. If the level latched on NEXPR is low, the Nexus interface is not enabled.

NEXPR must be used in conjunction with NEXPS to select the position of the Nexus port. The auxiliary port can be provided in two pin positions depending on the device and package. In the 208 BGA and 144 LQFP packages it is possible for the NEXUS port to be available on either the low byte of Port A or the low byte of Port E. Selection of the port position is via the value of the PF0/eMIOS0/NEXPS pin latched during the assertion of $\overline{\text{RESET}}$. If the level latched on NEXPS is low and Nexus is enabled, the Nexus port will be available in the primary position, PA[6:0]. If the level latched on NEXPS is high and Nexus is enabled, the Nexus port will be available in the alternate position, PE[6:0]. On smaller packages the Nexus port is available only in the alternate position (Port E).

Table 26-12. Nexus Port Configuration Summary

PF1 / NEXPR (during reset)	PF0 / NEXPS (during reset)	$\overline{\text{EVTI}}$ / $\overline{\text{EVTI}}$' (at any time)	Nexus Port Configuration
Low	x	x	Nexus auxiliary port disabled
High	Low	High	Nexus auxiliary port disabled
High	Low	Low	Nexus auxiliary port enabled, primary position (PA[6:0])
High	High	High	Nexus auxiliary port disabled
High	High	Low	Nexus auxiliary port enabled, alternate position (PE[6:0])

To supplement the information provided via the E-ICE and NEXUS ports, an optional debug status port may be enabled on Port F. Enabling the debug port and selecting the data set provided is controlled via the SSM (refer to Section 26.4.1.6, “SSM Debug Status Port Control Register (DEBUGPORT),” on page 26-570)). When the debug status port is enabled, Port F GPIO and eMIOS functions are not available.

NOTE

The debug status port is not available on mask set L49P devices.

A.2 E-ICE Interface

For details on the E-ICE interface please refer to Debugging Your System in the *ARM7TDMI-S Users Manual*, or ARM Application Note 31 "*Using Embedded*."

A.3 NEXUS Interface

For details on the NEXUS interface please refer to Appendix B, "A7S Nexus 2 Module," and the Nexus web site: http://www.nexus5001.org/.

Appendix B
A7S Nexus 2 Module

This appendix defines the auxiliary pin functions, transfer protocols and standard development features of a Class 2 device in compliance with IEEE-ISTO 5001-2003. The development features supported are Program Trace, Watchpoint Messaging, and Ownership Trace. The A7S Nexus 2 module also supports three Class 3/4 features: Read/Write Access (via JTAG), Watchpoint Triggering, and processor overrun control.

B.1 Terminology and Introduction

Table B-1. Terms and Definitions

Term	Description
IEEE-ISTO 5001	Consortium and standard for real-time embedded system design. World wide web documentation at http://www.nexus5001.org.
A7S Nexus 2 Module	The Nexus interface implementation on MAC7100 Family devices.
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard.
JTAG IR and DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The selected development register is then accessed via a JTAG Data Register (DR) scan.
EmbeddedICE	The ARM7 EmbeddedICE debug module. This module integrated with each ARM7 processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Messaging (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements.
Standard	The phrase “according to the standard” is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint which does not cause the processor to halt. Instead a pin is used to signal that the condition occurred. A Watchpoint Message is also generated.

B.1.1 A7S Nexus 2 Overview

The A7S Nexus 2 module provides real-time development capabilities for ARM7 series processors in compliance with the IEEE-ISTO Nexus 5001-2003. This module provides development support capabilities for MCUs without requiring address and data pins for internal visibility.

A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which, for ARM7, is used in conjunction with the JTAG port.

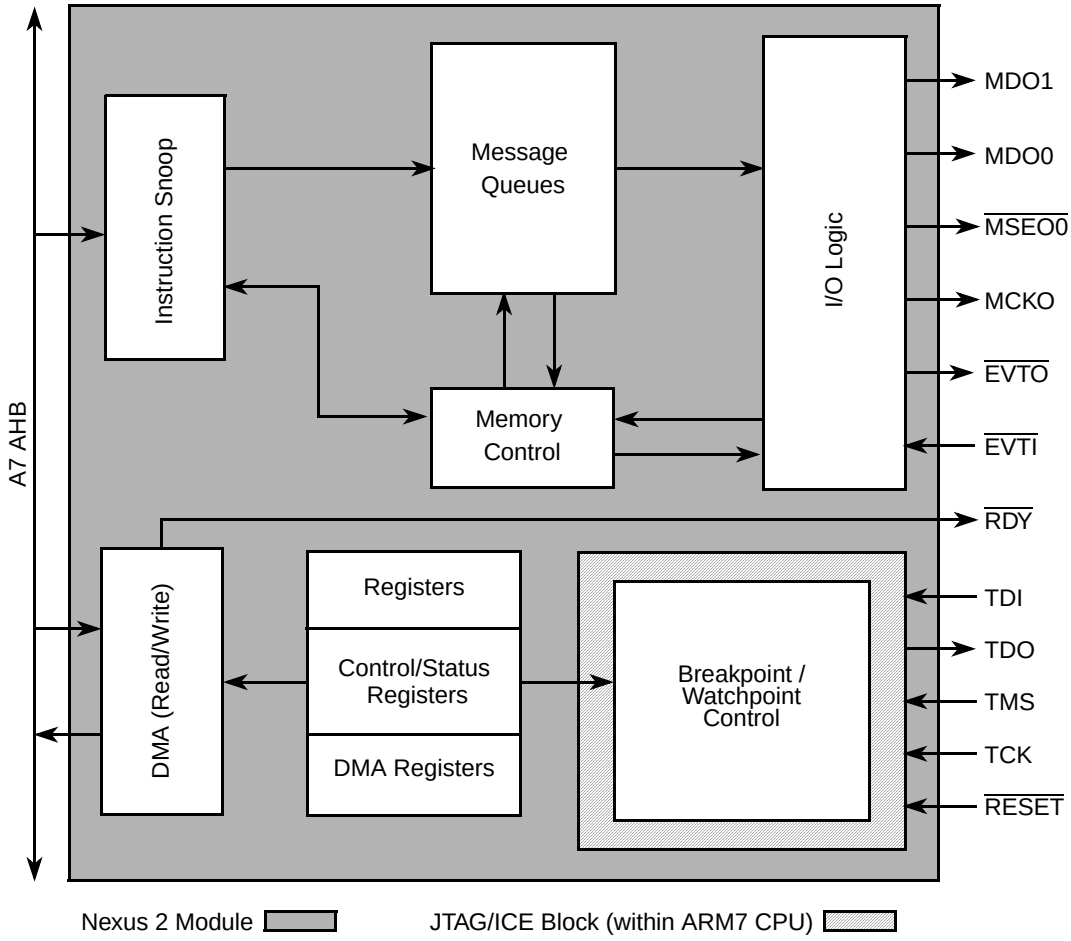


Figure B-1. MAC7100 Family Nexus 2 Functional Block Diagram

B.1.2 Feature List

The A7S Nexus 2 module is compliant with Class 2 of the IEEE-ISTO 5001-2003 standard. The following features are implemented:

1. Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
   - — Program Trace in ARM mode will use Branch/Predicate History Messaging due to the conditional nature of 32-bit ARM instructions
   - — Program Trace in Thumb Mode can be programmed to use the Branch History Messaging method, or use traditional Program Trace - Direct/Indirect Branch Messaging

2. Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
3. Run-time access to the memory map via the JTAG port. This allows for enhanced download/upload capabilities.
4. Watchpoint Messaging via the auxiliary pins
5. Watchpoint Trigger enable of Program Trace Messaging
6. Auxiliary interface for higher data input/output
   - Configurable (min/max) MDO (Message Data Out) pins [1]
   - One or two $\overline{\text{MSEO}}$ (Message Start/End Out) pins [1]
   - One $\overline{\text{RDY}}$ (Read/Write Ready) pin
   - One $\overline{\text{EVTO}}$ (Watchpoint Event) pin
   - One $\overline{\text{EVTI}}$ (Event In) pin
   - One MCKO (Message Clock Out) pin

NOTE

The configuration of the MDO pins is selected during system reset: Full Port Mode (FPM, maximum number of MDO pins) or Reduced Port Mode (RPM, minimum number of MDO pins). The values for maximum and minimum MDO are determined by the specific integration.

The configuration of the MSEO pins is hard-wired to 1 for MAC7100 family devices.

7. Registers for Program Trace, Ownership Trace, Watchpoint Trigger, and Read/Write Access.
8. Programmable processor stall function to mitigate message queue overrun risk.
9. All features controllable and configurable via the JTAG port

B.1.3 Modes of Operation

There are three basic modes of operation for the A7S Nexus 2 block.

- Reset
- Normal
- Disabled

B.1.3.1 Reset

The reset configuration is received via the $\overline{\text{EVTI}}$ pin to enable or disable the A7S Nexus 2 module. $\overline{\text{EVTI}}$ is sampled synchronously at the exit from the JTAG Test-Logic-Reset state. Reset configuration information must be valid on $\overline{\text{EVTI}}$ during the JTAG “Test Logic Reset” state (see Section B.6, “IEEE 1149.1 State Machine and RD/WR Sequences”).

1. MAC7100 family devices implement two MDO signal and one $\overline{\text{MSEO}}$ signal.

The A7S Nexus 2 module will disable (drive inactive) the output pins during the JTAG Test-Logic-Reset state, and when a Power-on-Reset (POR) event occurs.

B.1.3.2 Normal

If $\overline{\text{EVTI}}$ is asserted at the exit from JTAG Test-Logic-Reset, the A7S Nexus 2 module will be enabled. The module will be ready to accept control input via the JTAG pins.

The A7S Nexus 2 module may also be enabled by loading the NEXUS-ACCESS instruction into the JTAG Instruction Register. Once the A7S Nexus 2 module has been enabled, it will remain enabled until entry into the JTAG Test-Logic-Reset state.

B.1.3.3 Disabled

If $\overline{\text{EVTI}}$ is negated at the exit from JTAG Test-Logic-Reset, the A7S Nexus 2 module will be disabled. No trace output will be provided, auxiliary port output pins ($\overline{\text{MDO}}$, $\overline{\text{MSEO}}$, MCKO) will be disabled (driven inactive), and Nexus 2 registers will not be accessible for reads or writes.

NOTE

If there is no debug/development tool connected to the chip, the $\overline{\text{EVTI}}$ pin should be held de-asserted in order to keep the Nexus 2 module disabled.

The A7S Nexus 2 module may be enabled after the exit from JTAG Test-Logic-Reset state by loading the NEXUS-ACCESS instruction into the JTAG Instruction Register.

B.1.4 TCODEs supported

The A7S Nexus 2 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages. The A7S Nexus 2 block supports the public TCODEs seen in Table B-2.

Table B-2. Public TCODEs Supported

Message Name	Packet Size (bits)		Packet Name	Packet Type	Packet Description
	Min.	Max.			
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier (multiple Nexus configuration)
	8	8	STATUS	fixed	Debug Status Register (DS[31:24])
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier (multiple Nexus configuration)
	32	32	PROCESS	fixed	Task/Process ID tag
Program Trace [1]	6	6	TCODE	fixed	TCODE number = 3 [2]
Direct Branch Message [2]	4	4	SRC	fixed	source processor identifier (multiple Nexus configuration)
	1	8	I-CNT	variable	# sequential instructions executed since last taken branch

Table B-2. Public TCODEs Supported (continued)

Message Name	Packet Size (bits)		Packet Name	Packet Type	Packet Description
	Min.	Max.			
Program Trace [1] Indirect Branch Message [2]	6	6	TCODE	fixed	TCODE number = 4 [2]
	4	4	SRC	fixed	source processor identifier (multiple Nexus configuration)
	1	8	I-CNT	variable	# sequential instructions executed since last taken branch
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier (multiple Nexus configuration)
	5	5	ERROR	fixed	error code (Refer to Table B-3)
Program Trace [1] Direct Branch Message with Sync [2]	6	6	TCODE	fixed	TCODE number = 11 [2]
	4	4	SRC	fixed	source processor identifier (multiple Nexus configuration)
	1	8	I-CNT	variable	# sequential instructions executed since last taken branch
	1	32	F-ADDR	variable	full target address (leading zero truncated)
Program Trace [1] Indirect Branch Message with Sync [2]	6	6	TCODE	fixed	TCODE number = 12 [2]
	4	4	SRC	fixed	source processor identifier (multiple Nexus configuration)
	1	8	I-CNT	variable	# sequential instructions executed since last taken branch
	1	32	F-ADDR	variable	full target address (leading zero truncated)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier (multiple Nexus configuration)
	4	4	WPHIT	fixed	# indicating watchpoint source(s)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier (multiple Nexus configuration)
	4	4	RCODE	fixed	resource code (Refer to Table B-4)
	1	32	HIST	variable	branch / predicate instruction history
Program Trace [1] Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 [2]
	4	4	SRC	fixed	source processor identifier (multiple Nexus configuration)
	1	8	I-CNT	variable	# sequential instructions executed since last taken branch
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
	1	32	HIST	variable	branch / predicate instruction history
Program Trace [1] Indirect Branch History Message with Sync	6	6	TCODE	fixed	TCODE number = 29 [2]
	4	4	SRC	fixed	source processor identifier (multiple Nexus configuration)
	1	8	I-CNT	variable	# sequential instructions executed since last taken branch
	1	32	F-ADDR	variable	full target address (leading zero truncated)
	1	32	HIST	variable	branch / predicate instruction history
Program Trace [1] Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier (multiple Nexus configuration)
	4	4	ECODE	fixed	event correlated with program flow (Refer to Table B-5)
	1	8	I-CNT	variable	# sequential instructions executed since last taken branch
	1	32	HIST	variable	branch / predicate instruction history

1 Due to the conditional nature of 32-bit ARM7 instructions, when in ARM7 mode, Program Trace will be implemented using Branch History/Predicate Instruction Messages. When in Thumb mode, the user can select between traditional Program Trace using Direct/Indirect Branch Messages, or Branch History Messages.

2 If the Branch History method is selected in Thumb mode, this TCODE will not be sent.

Table B-3. Error Code Encoding (TCODE = 8)

Error Code	Description
00000	Ownership Trace overrun
00001	Program Trace overrun
00010	Reserved
00011	Read/write access error
00101	Invalid access opcode (Nexus Register unimplemented)
00110	Watchpoint overrun
00111	Program Trace and Ownership Trace overrun
01000	(Program Trace or Ownership Trace) and Watchpoint overrun
01001–10111	Reserved
11000	BTM lost due to collision w/ higher priority message
11001–11111	Reserved

Table B-4. Resource Code Encoding (TCODE = 27)

Resource Code	Description
0000	Reserved for future functionality
0001	Branch / Predicate Instruction History Buffer
0010–1111	Reserved for future functionality

Table B-5. Event Code Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010–1111	Reserved for future functionality

B.2 External Signal Description

The A7S Nexus 2 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

B.2.1 Pins Implemented

The A7S Nexus 2 module implements one $\overline{\text{EVTI}}$ and one or two $\overline{\text{MSEO}}$ pins. [1] It also implements up to sixteen MDO pins, [1] one $\overline{\text{RDY}}$ pin, one $\overline{\text{EVTO}}$ pin, and one clock output pin (MCKO). The output pins are synchronized to the Nexus 2 output clock (MCKO).

1. The MAC7100 family implements one $\overline{\text{MSEO}}$ pint and two MDO pins.

All Nexus 2 input functionality is controlled through the JTAG port in compliance with IEEE 1149.1 (see Table B.3.4 for details). The JTAG pins are incorporated as I/O to the ARM7 processor.

Table B-6. JTAG Pins for A7S Nexus 2

JTAG Pins	Input/ Output	Description of JTAG Pins (included in ARM7 CPU)
TDO	O	The Test Data Output pin is the serial output for test instructions and data. TDO is three-stateable and is actively driven in the “shift-IR” and “shift-DR” controller states. TDO changes on the falling edge of TCK.
TDI	I	The Test Data Input pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
TMS	I	The Test Mode Select input pin is used to sequence the JTAG test controllers’ state machine. TMS is sampled on the rising edge of TCK and has an internal pull-up resistor.
TCK	I	The Test Clock input pin is used to synchronize the test logic, and control register access through the JTAG port.
$\overline{\text{RESET}}$	I	The Test Reset input pin is used to asynchronously initialize the JTAG controller. On MAC7100 devices, this is the same $\overline{\text{RESET}}$ signal used to reset the chip.

Table B-7. A7S Nexus 2 Auxiliary Pins

Auxiliary Pins	Input/ Output	Description of Auxiliary Pins
MCKO	O	Message Clock Out is a Nexus generated output clock to development tools for timing of MDO and $\overline{\text{MSEO}}$ pin functions. MCKO is programmable through the DC Register.
MDO[*n*:0] [1]	O	Message Data Out are output pins used for OTM and BTM. External latching of MDO shall occur on rising edge of the Nexus 2 message clock (MCKO).
$\overline{\text{MSEO}}$[1:0] [1]	O	Message Start/End Out are output pins which indicate when a message on the MDO pins has started, when a variable length packet has ended, and when the message has ended. External latching of $\overline{\text{MSEO}}$ shall occur on rising edge of the Nexus 2 clock (MCKO). One or two pin $\overline{\text{MSEO}}$ functionality is determined at integration time.
$\overline{\text{RDY}}$	O	Ready is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB transfer (Nexus read or write) and is held asserted until the JTAG state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, $\overline{\text{RDY}}$ remains de-asserted.
$\overline{\text{EVTO}}$	O	Event Out is an output which, when asserted, indicates one of two events has occurred based on the EOC bits in the DC Register. $\overline{\text{EVTO}}$ is held asserted for one cycle of MCKO: 1. one of two watchpoints has occurred (DBGRNG[1:0] from ARM7) and EOC = 0b00, or 2. debug mode was entered (DBGACK from ARM7) and EOC = 0b01
$\overline{\text{EVTI}}$	I	Event In is an input which, when asserted, will initiate one of two events based on the EIC bits in the DC Register (if the Nexus module is enabled at reset - see Table B.1.3.3): 1. Program Trace synchronization messages (provided Program Trace is enabled and EIC = 0b00), or 2. Debug request (EDBGRQ) to ARM7 EmbeddedICE module (provided EIC = 0b01 and this feature is implemented).

[1] The MAC7100 family implements one $\overline{\text{MSEO}}$ pint and two MDO pins.

B.2.2 Pin Protocol

The protocol for the ARM7 processor transmitting messages via the auxiliary pins shall be accomplished with the $\overline{\text{MSEO}}$ pin(s) function (outlined in Table B-8).

$\overline{\text{MSEO}}$ is used to signal the end of variable-length packets, and not fixed length packets. $\overline{\text{MSEO}}$ is sampled on the rising edge of the message clock (MCKO).

Table B-8. $\overline{\text{MSEO}}$ Pin(s) Protocol

$\overline{\text{MSEO}}$ Function	Single $\overline{\text{MSEO}}$ data (serial)	Dual $\overline{\text{MSEO}}$ data
Start of message	1–1–0	11–00
End of message	0–1–1–(more 1's)	00 (or 01)–11–(more 11's)
End of variable length packet	0–1–0	00–01
Message transmission	0's	00's
Idle (no message)	1's	11's

Section Figure B-2., "Single Pin MSEO Transfers," illustrates the state diagram for single pin $\overline{\text{MSEO}}$ transfers.

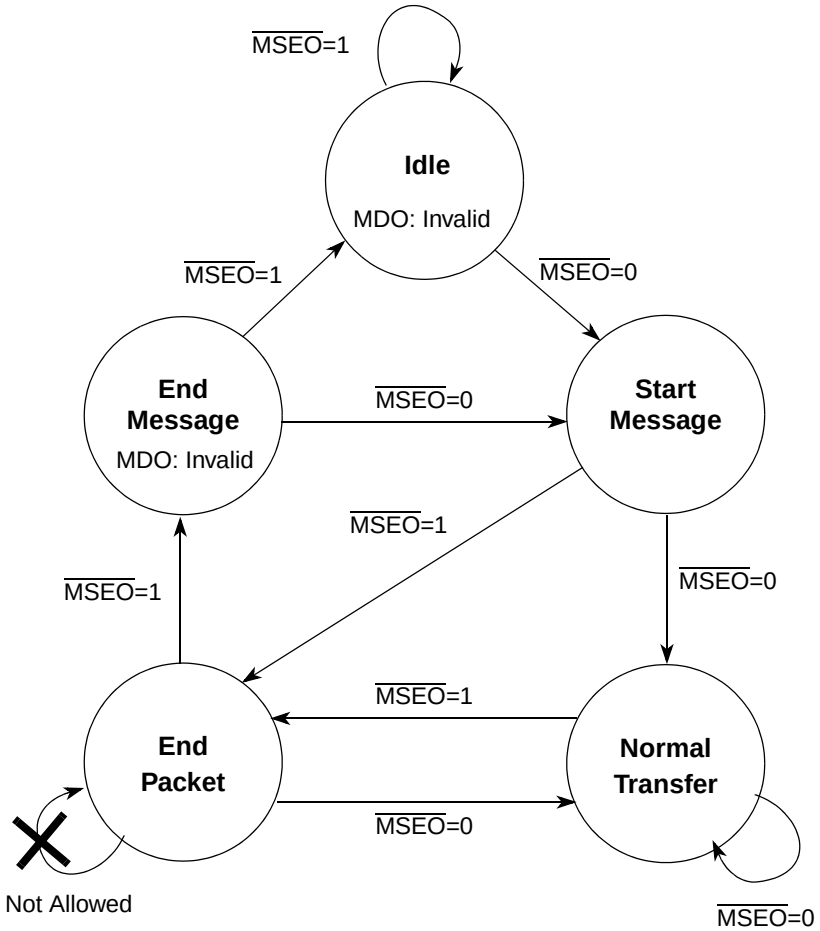


Figure B-2. Single Pin $\overline{\text{MSEO}}$ Transfers

Note that the “End Message” state does not contain valid data on the MDO pins. Also, It is not possible to have two consecutive “End Packet” messages. This implies the minimum packet size for a variable length packet is 2x the number of MDO pins. This ensures that a false end of message state is not entered by emitting two consecutive 1’s on the $\overline{\text{MSEO}}$ pin before the actual end of message.

Section Figure B-3., “Two Pin MSEO Transfers,” illustrates the state diagram for two pin $\overline{\text{MSEO}}$ transfers.

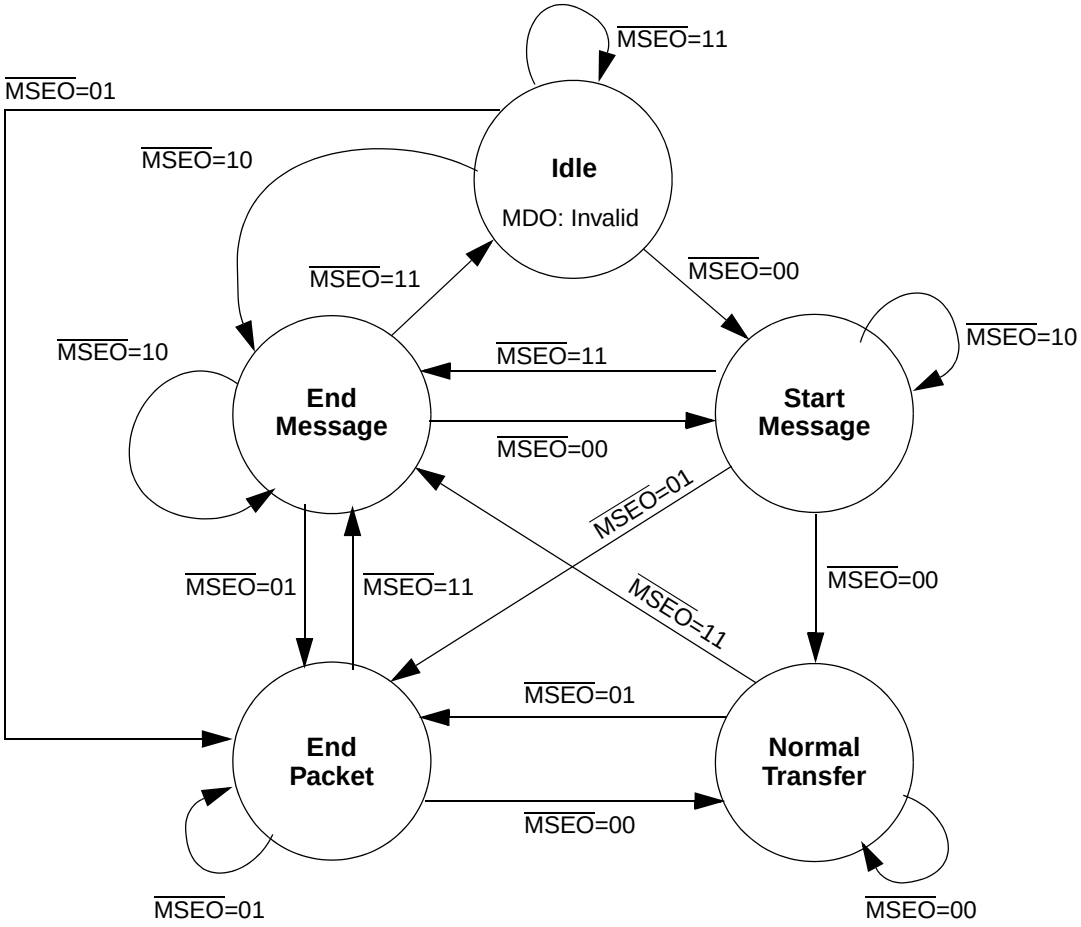


Figure B-3. Two Pin $\overline{\text{MSEO}}$ Transfers

The two-pin $\overline{\text{MSEO}}$ option is more efficient that the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one $\overline{\text{MSEO}}$ pin option. The two-pin option also allows for consecutive “End Packet” states. This can be an advantage when small, variable sized packets are transferred.

NOTE

The “End Message” state may also indicate the end of a variable-length packet as well as the end of the message when using the two-pin option.

B.2.3 Rules for Output Messages

ARM7 based Class 2 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.
- For example, if the MDO port is 2 bits wide, and the unique portion of an indirect branch address is 5 bits, then the remaining 1 bit of MDO must be packed with a 0.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is impossible to know which bits are from the last packet and which bits are from the next packet.)

B.2.4 Examples

The following are examples of Program Trace Messages.

Table B-9 illustrates an example Indirect Branch Message (traditional - Thumb mode) with 2 MDO / 1 $\overline{\text{MSEO}}$ configuration. Table B-10 illustrates the same example with the 8 MDO / 2 $\overline{\text{MSEO}}$ configuration.

Note that T0 and S0 are the least significant bits where:

- T*n* = TCODE number (fixed)
- S*n* = Source processor (fixed)
- I*n* = Number of instructions (variable)
- A*n* = Unique portion of the address (variable)

Note that during clock 13, the MDO pins are ignored in the single $\overline{\text{MSEO}}$ case.

Table B-9. Indirect Branch Message Example (2 MDO / 1 $\overline{\text{MSEO}}$)

Clock	MDO[1:0]		$\overline{\text{MSEO}}$	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	0	Normal Transfer
7	I3	I2	0	Normal Transfer
8	I5	I4	0	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table B-10. Indirect Branch Message Example (8 MDO / 2 $\overline{\text{MSEO}}$)

Clock	MDO[7:0]								$\overline{\text{MSEO}}$[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I5	I4	I3	I2	I1	I0	S3	S2	0	1	End Packet
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table B-11 and Table B-12 illustrate examples of Direct Branch Messages: one with 2 MDO / 1 $\overline{\text{MSEO}}$, and one with 8 MDO / 2 $\overline{\text{MSEO}}$.

Note that T0 and I0 are the least significant bits where:

- T*n* = TCODE number (fixed)
- S*n* = Source processor (fixed)
- I*n* = Number of Instructions (variable)

Table B-11. Direct Branch Message Example (2 MDO / 1 $\overline{\text{MSEO}}$)

Clock	MDO[1:0]		MSEO	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message
8	T1	T0	0	Start Message

Table B-12. Direct Branch Message Example (8 MDO / 2 $\overline{\text{MSEO}}$)

Clock	MDO[7:0]								$\overline{\text{MSEO}}$[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

B.3 A7S Nexus 2 Programmers Model

This section describes the A7S Nexus 2 programmers model. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. See Table B.3.4 for details on Nexus register access.

B.3.1 JTAG ID Register (ID)

This JTAG ID Register (located within the ARM7 CPU) provides key development attributes to the development tool concerning the ARM7 processor and the A7S Nexus 2 block. This register is fixed for each ARM7 embedded system.

This register is accessed through the standard JTAG IR/DR paths.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VERSION				PART NUMBER											
W																
Reset	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PART NUMBER				MANUFACTURER ID											1
W																
Reset	M	M	M	M	0	0	0	0	0	0	0	1	1	1	0	1

Figure B-4. JTAG ID Register (ID)

Table B-13. ID Field Descriptions

Bits	Name	Description
31–28	VERSION[3:0]	Embedded Product Version Number x MAC7100 specific value (TBD)
27–12	PART NUMBER[15:0]	ARM7 Based Part Number xxxx MAC7100 specific value (TBD)
11–1	MANUFACTURER ID[10:0]	Manufacturer ID Number 00E Freescale (previously Motorola)
0	JTAG	Fixed per IEEE 1149.1 (JTAG) 1 Always set

B.3.2 A7S Nexus 2 Memory Map / Register Definition

Table B-14. A7S Nexus 2 Memory Map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address
Client Select Control Register (CSC) [1]	0x1	R	0x02	—
Development Control Register (DC)	0x2	R/W	0x04	0x05
Development Status Register (DS)	0x4	R	0x08	—
User Base Address Register (UBA)	0x6	R/W	0x0C	0x0D
Read / Write Access Control Register (RWCS)	0x7	R/W	0x0E	0x0F
Read/Write Access Address Register (RWA)	0x9	R/W	0x12	0x13
Read/Write Access Data Register (RWD)	0xA	R/W	0x14	0x15
Watchpoint Trigger Register (WT)	0xB	R/W	0x16	0x17
Reserved	0x0D to 0x3F	—	0x1A to 0x7E	0x1B to 7F

[1] The MAC7100 family utilizes a single-client implementation.

B.3.3 A7S Nexus 2 Register Descriptions

B.3.3.1 Client Select Control Register (CSC)

The CSC Register determines which Nexus client is under development.

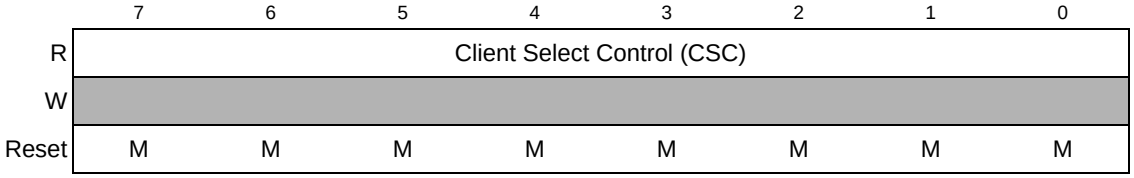


Figure B-5. Client Select Control Register (CSC)

Table B-15. CSC Field Description

Bits	Name	Description
7–0	CSC[7:0]	Client Select Control xx MAC7100 specific value (TBD)

B.3.3.2 Development Control Register (DC)

The Development Control Register is used to control basic development features of A7S Nexus 2.

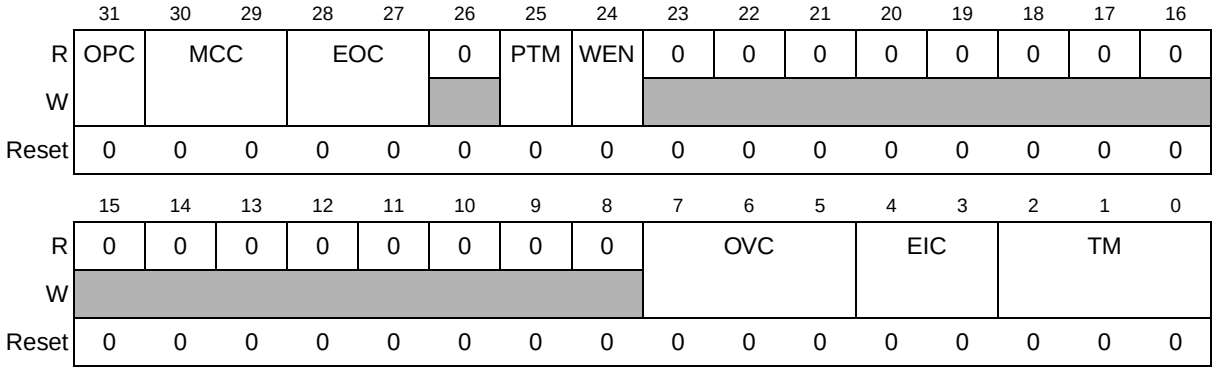


Figure B-6. Development Control Register (DC)

Table B-16. DC Field Description

Bits	Name	Description
31	OPC [1]	Output Port Mode Control 0 Reduced Port Mode configuration (2 MDO pins) 1 Full Port Mode configuration (8 MDO pins)
30–29	MCC[1:0] [1]	MCKO Clock Frequency Control 00 MCKO is 1x processor clock frequency 01 MCKO is 1/2x processor clock frequency 10 MCKO is 1/4x processor clock frequency 11 MCKO is 1/8x processor clock frequency
28–27	EOC[1:0]	EVTO Control 00 EVTO upon occurrence of Watchpoint (DBGRNG[1] or [0]) 01 EVTO upon entry into Debug Mode (DBGACK) 1x Reserved
26	—	Reserved for future use 0 Always reads as 0
25	PTM	Program Trace Method (Thumb mode only) 0 Program Trace in Thumb mode uses Branch History Messages 1 Program Trace in Thumb mode uses traditional Branch Messages
24	WEN	Watchpoint Trace Enable 0 Watchpoint Messaging disabled 1 Watchpoint Messaging enabled
23–8	—	Reserved for future use 0000 Always reads as 0

Table B-16. DC Field Description (continued)

Bits	Name	Description
7–5	OVC[2:0]	Overrun Control 000 Generate overrun messages 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 011 Delay processor for BTM / OTM overruns (FIFOFULL) 1xx Reserved
4–3	EIC[1:0]	EVTI Control 00 EVTI for synchronization (Program Trace) 01 EVTI for Debug request (EDBGRQ) (if implemented) 10 EVTI disabled 11 Reserved
2–0	TM[2:0]	Trace Mode 000 No Trace 1xx Program Trace enabled x1x Reserved xx1 Ownership Trace enabled

[1] The Output Port Mode Control bit (OPC) and MCKO Clock Control bits (MCC) must only be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC also only be modified in one of these two modes.

B.3.3.3 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited or a MAC7100 family defined Low Power Mode is entered, a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DBG	LPS			LPC		0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure B-7. Development Status Register (DS)

Table B-17. DS Field Description

Bits	Name	Description
31	DBG	ARM7 CPU Debug Mode Status 0 CPU not in Debug Mode 1 CPU in Debug Mode (DBGACK asserted)
30–28	LPS[2:0] [1]	ARM7 System Low Power Mode Status 000 Normal (Run) mode xxx System Low Power Status (MCU level)
27–26	LPC[1:0] [2]	ARM7 CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in powered-down state (STANDBYWFI or equivalent) 1x Reserved
25–0	—	Reserved for future use Always reads as 0

1 Functionality is determined at the platform or MCU integration level. Any entry into a system-level low power mode (LPS ≠ 0b000) will trigger a Debug Status Message sending the DS register value to the external tool. These bits are recommended for non-CPU related power-down modes. It is the tool's responsibility to decode the specific type of low power mode based on the encodings for the specific MCU.

2 Must be tied to the logic which indicates that the ARM7 processor is in a powered-down state. The CPU low power state may be independent of the MCU defined low power mode(s) determined by the LPS bits.

B.3.3.4 User Base Address Register (UBA)

For ARM7 processors, Ownership Trace Messaging is implemented using the Nexus defined User Base Address Register. The User Base Address Register defines the memory mapped base address for the Ownership Trace Register (OTR). The operating system writes the ID for the current task/process in the OTR.

The UBA is read and written to by the external development tool.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Ownership Trace Register Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Ownership Trace Register Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure B-8. User Base Address Register (UBA)

NOTE

It is recommended that the UBA only be modified while system reset is asserted or in debug mode. Caution should be taken when modifying the UBA when system reset is negated.

B.3.3.5 Read/Write Access Control / Status Register (RWCS)

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to ARM7 Advanced High-performance Bus (AHB) memory mapped resources when the processor is halted or during runtime. The RWCS Register also provides Read/Write Access Status information per Table B-19.

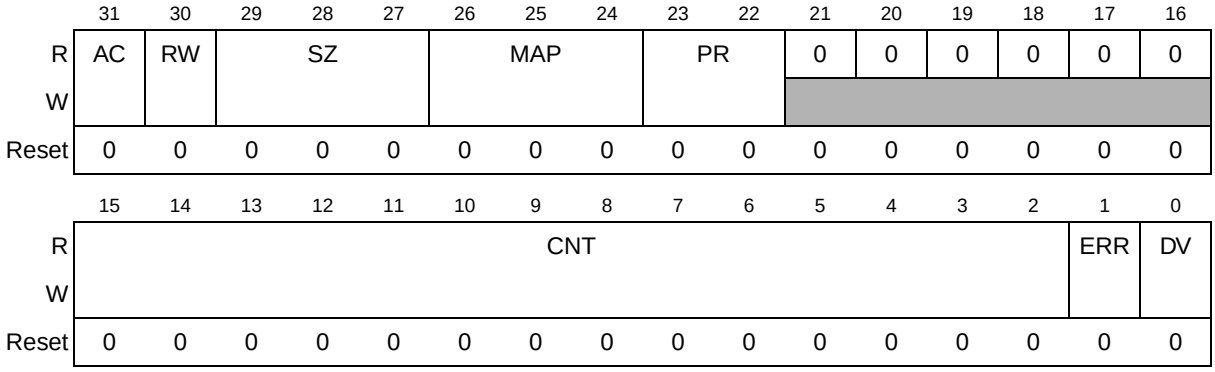


Figure B-9. Read / Write Access Control Register (RWCS)

Table B-18. RWCS Field Description

Bits	Name	Description
31	AC	Access Control 0 End access 1 Start access
30	RW	Read/Write Select 0 Read access 1 Write access
29–27	SZ[2:0]	Word Size 000 8-bit (byte) 001 16-bit (halfword) 010 32-bit (word) 011-111 Reserved (default to word)
26–24	MAP[2:0]	Map Select 000 Primary memory map 001-111 Reserved
23–22	PR[1:0]	Read/Write Access Priority 00 Lowest access priority 01 Reserved (default to lowest priority) 10 Reserved (default to lowest priority) 11 Highest access priority
21–16	—	Reserved for future functionality
15–2	CNT[13:0]	Access Control Count hhhh Number of accesses of word size SZ
1	ERR	Read/Write Access Error (see Table B-19)
0	DV	Read/Write Access Data Valid (see Table B-19)

Table B-19. Read / Write Access Status Bit Encoding

ERR	DV	Read Action	Write Action
0	0	Read Access has not completed	Write Access completed without error
1	0	Read Access error has occurred	Write Access error has occurred
0	1	Read Access completed without error	Write Access has not completed
1	1	Not Allowed	Not allowed

B.3.3.6 Read/Write Access Data Register (RWD)

The Read/Write Access Data Register provides the data to/from AHB memory mapped locations when initiating a read or a write access.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Read/Write Data															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Read/Write Data															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure B-10. Read/Write Access Data Register (RWD)

B.3.3.7 Read/Write Access Address Register (RWA)

The Read/Write Access Address Register provides the AHB memory mapped address to be accessed when initiating a read or a write access.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Read/Write Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Read/Write Address															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure B-11. Read/Write Access Address Register (RWA)

B.3.3.8 Watchpoint Trigger Register (WT)

The Watchpoint Trigger Register allows the two watchpoints defined within the ARM7 EmbeddedICE logic to trigger actions. These watchpoints can control Program Trace enable and disable. The WT bits can be used to produce an address related “window” for triggering Trace Messages.

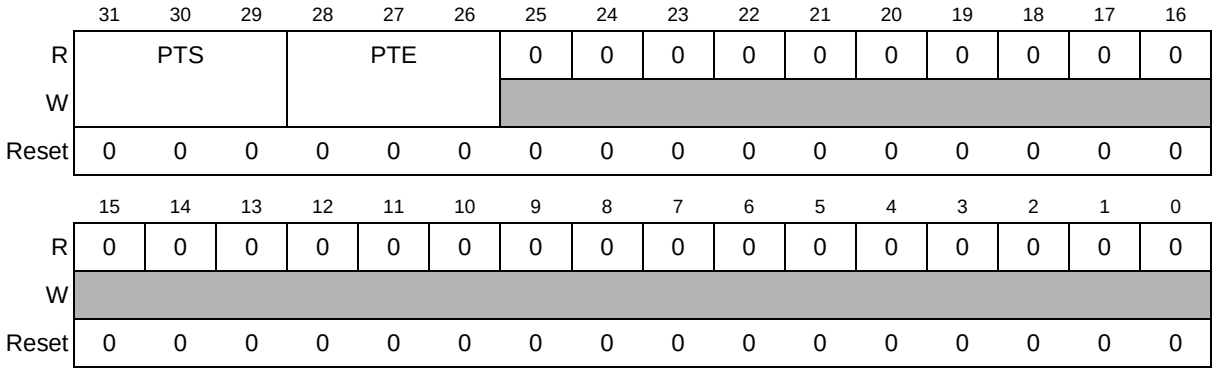


Figure B-12. Watchpoint Trigger Register (WT)

Table B-20. WT Field Description

Bits [1]	Name	Description
31–29	PTS[2:0]	Program Trace Start Control 000 Trigger disabled 001 Use ARM7 Watchpoint #1 (DBGRNG[0]) 010 Use ARM7 Watchpoint #2 (DBGRNG[1]) 011 Use ARM7 Watchpoint #3 (DBGRNG[2]) [2] 100 Use ARM7 Watchpoint #4 (DBGRNG[3]) [2] 101–111 Reserved
28–26	PTE[2:0]	Program Trace End Control 000 Trigger disabled 001 Use ARM7 Watchpoint #1 (DBGRNG[0]) 010 Use ARM7 Watchpoint #2 (DBGRNG[1]) 011 Use ARM7 Watchpoint #3 (DBGRNG[2]) [2] 100 Use ARM7 Watchpoint #4 (DBGRNG[3]) [2] 101–111 Reserved
25–0	—	Reserved for future functionality (read as 0)

1 The WT bits will ONLY control Program Trace if the TM bit within the Development Control Register (DC) have not already been set to enable Program Trace.

2 Mask set L49P implements only two watchpoints, DBGRNG[0] and DBGRNG[1].

NOTE

The WT bits will control Program Trace only if the DC[TM] field has not already been set to enable Program Trace.

B.3.4 Nexus Register Access via JTAG

Access to Nexus register resources is enabled by loading a single instruction (“NEXUS-ACCESS”) into the JTAG Instruction Register (IR). For the A7S Nexus 2 block, the JTAG IR value is programmable at the platform or chip integration level. It can be programmed to any of the non-ARM7 implemented IR values. Table below shows the current ARM7 IR values (as defined by ARM).

Table B-21. ARM7 JTAG Instructions

IR[3:0]	Usable for NEXUS-ACCESS?	JTAG Instruction
0x0	No	EXTEST (ARM720T)
0x1	Yes	Not publicly implemented
0x2	No	SCAN_N (ARM7TDMI-S)
0x3	No	SAMPLE/PRELOAD (ARM720T)
0x4	No	RESTART (ARM7TDMI-S)
0x5	No	CLAMP (ARM720T)
0x6	Yes	Not publicly implemented
0x7	No	HIGHZ (ARM720T)
0x8 [1]	Yes	recommended for NEXUS-ACCESS
0x9	No	CLAMPZ (ARM720T)
0xA	Yes	Not publicly implemented
0xB	Yes	Not publicly implemented
0xC	No	INTEST (ARM7TDMI-S)
0xD	Yes	Not publicly implemented
0xE	No	IDCODE (ARM7TDMI-S)
0xF	No	BYPASS (ARM7TDMI-S)

[1] It is recommended to use IR[3:0] = 0x8 for “NEXUS-ACCESS”

Once the JTAG “NEXUS-ACCESS” instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in Table B-14

Reading/writing of a Nexus register then requires two passes through the Data-Scan (DR) path of the JTAG state machine (see Section B.6, “IEEE 1149.1 State Machine and RD/WR Sequences”).

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (see Table B-14), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:

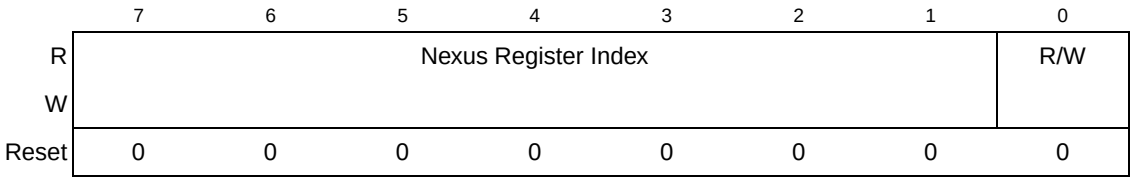


Figure B-13. JTAG DR for Nexus Register Access

Table B-22. JTAG DR Field Values for Nexus Register Access

Bits	Name	Description
7–1	Nexus Register Index[6:0]	Selected from values in Table B-14
0	R/W	Read/Write 0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a) During a read access, data is latched from the selected Nexus register when the JTAG state machine (see Section B.6, “IEEE 1149.1 State Machine and RD/WR Sequences”) passes through the “Capture-DR” state.
   b) During a write access, data is latched into the selected Nexus register when the JTAG state machine (see Section B.6, “IEEE 1149.1 State Machine and RD/WR Sequences”) passes through the “Update-DR” state.

B.3.5 Programming Considerations ($\overline{\text{RESET}}$)

If Nexus 2 register configuration is to occur during system reset (as opposed to debug mode), all Nexus 2 configuration should be completed between the exit from JTAG Test-Logic-Reset state and system reset de-assertion, after the JTAG ID Register has been read by the tool.

B.4 Functional Description

B.4.1 Ownership Trace

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

B.4.1.1 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). The User Base Address Register (UBA), which can be accessed via the JTAG port, contains the address of the Ownership Trace Register (OTR). The OTR is updated by the operating system software to provide task/process ID information.

There are two conditions which will cause an Ownership Trace Message.

1. When new information is updated in the OTR register by the ARM7 processor, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow.
2. When the periodic (255) OTM Message counter expires (after 255 queued messages without an OTM), an OTM will be sent. The data will be sent from the latched OTR data. This allows processors using virtual memory to be regularly updated with the latest process ID.

Ownership trace information is messaged out in the following format:

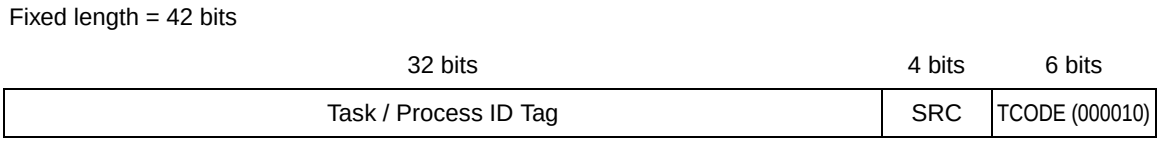


Figure B-14. Ownership Trace Message Format

B.4.1.2 OTM Error Messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only an OTM Message attempts to enter the queue while it is being emptied, the Error Message will incorporate the OTM only error encoding (00000). If both OTM AND either BTM or DTM messages attempt to enter the queue, the Error Message will incorporate the OTM and Program Trace error encoding (00111). If a Watchpoint also attempts to be queued while the FIFO is being emptied, then the Error Message will incorporate error encoding (01000).

NOTE

The OVC bits within the DC Register can be set to delay the CPU by asserting the FIFOFULL signal in order to alleviate (but not eliminate) potential overruns.

Error information is messaged out in the following format (see Table B-3).

Fixed length = 15 bits

5 bits	4 bits	6 bits
ERROR (00000 / 00111 / 01000)	SRC	TCODE (001000)

Figure B-15. Error Message Format

B.4.1.3 OTM Flow

Ownership Trace Messages are generated when the operating system writes to the memory mapped Ownership Trace Register.

The following flow describes the OTM process.

1. For the A7S Nexus 2 module, the OTR register is a memory mapped register, whose address is located in the UBA. The UBA is internal to the Nexus module and can be accessed by the IEEE-ISTO 5001 tool through the JTAG port.
2. Only word writes to the OTR are valid. The data value written into the OTR is latched and formed into the Ownership Trace Message that is queued to be transmitted.
3. OTR reads do not cause Ownership Trace Messages to be transmitted by the A7S Nexus 2 module.
4. If the periodic OTM Message counter expires (after 255 queued messages without an OTM), an OTM is sent using the latched data from the previous OTR.

NOTE

OTM Messages are guaranteed to be transmitted in cases where the OTM collides with another message.

B.4.2 Program Trace

This section details the program trace mechanisms supported by Nexus 2 for the ARM7 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the Class 2 IEEE-ISTO 5001-2003 standard definition.

B.4.2.1 Branch Trace Messaging (BTM)

Traditional Branch Trace Messaging (Thumb mode only) facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception. Direct (or indirect) branches not taken are counted as sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address.

Branch History Messaging (ARM and Thumb modes) facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last predicate instruction, exception, or taken indirect branch, the unique portion of the branch target address or exception vector address, as well as a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one indicates taken, and a value of zero indicates not taken.

B.4.2.1.1 ARM7 Indirect Branch Message Instructions

The table below shows the types of instructions and events which cause Indirect Branch Messages or Branch History Messages to be encoded:

Table B-23. Indirect Branch / Branch History Message Instructions

Source of Indirect Branch Message	Instructions (ARM mode)	Instr. (Thumb mode)
Taken Register / PC Indirect Branch instruction	bx	bx
Sequential instruction w/ PC as destination reg.	any that write to R15 (PC)	any that write to R15 (PC)
Interrupt / exception	swi, undefined instructions	undefined instructions
Return from interrupt / exception	movs, subs, ldm(3)	N/A

NOTE

Instructions with the Program Counter (PC) as the destination register (R15) that are interrupted may or may not cause a BTM to be queued depending on which stage of the pipe the instruction has reached when the interrupt occurs.

B.4.2.1.2 ARM7 Direct Branch Message Instructions

The table below shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message:

Table B-24. Direct Branch Message Instructions

Source of Direct Branch Message	Instructions (ARM mode)	Instr. (Thumb mode)
Taken Direct Branch instruction	b, bl	b(1), b(2), bl

B.4.2.1.3 BTM in ARM mode

Due to the conditional nature of 32-bit ARM instructions, traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch. A value of one indicates the conditional instruction was executed or the direct branch was taken. A value of zero indicates the conditional instruction was not executed or the direct branch was not taken.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

B.4.2.1.4 BTM in Thumb mode

Based on the PTM bit in the DC Register (DC[25]), Program Tracing can utilize either Branch History Messages (DC[25] = 0b0) or traditional Direct/Indirect Branch Messages (DC[25] = 0b1).

Branch History will save bandwidth and keep consistency between methods of Program Trace in ARM and Thumb modes, yet may lose temporal order between branch events and other types of messages. Since direct branches are not messaged, but included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of events that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

B.4.2.2 Branch Trace Message Formats (History and Traditional)

The A7S Nexus 2 block supports three types of traditional BTM Messages - Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages - Indirect Branch History, and Indirect Branch History with Synchronization Messages. Debug Status Messages, Program Correlation Messages and Error Messages are also supported.

B.4.2.2.1 Indirect Branch Messages (History)

Indirect branches include all taken branches whose destination is determined at run time, interrupts and exceptions. If DC[25] is cleared while in Thumb mode, or the ARM7 processor is in full 32-bit ARM mode, indirect branch information is messaged out in the following format

Max length = 82 bits; Min length = 13 bits

1-32 bits	1-32 bits	1-8 bits	4 bits	6 bits
HIST	U-ADDR	I-CNT	SRC	TCODE (011100)

Figure B-16. Indirect Branch Message (History) Format

B.4.2.2.2 Indirect Branch Messages (Traditional)

If DC[25] is set in Thumb mode, indirect branch information is messaged out in the following format:

Max length = 50 bits; Min length = 12 bits

1-32 bits	1-8 bits	4 bits	6 bits
U-ADDR	I-CNT	SRC	TCODE (000100)

Figure B-17. Indirect Branch Message (Traditional) Format

B.4.2.2.3 Direct Branch Messages (Traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. If DC[25] is set while in Thumb mode, direct branch information is messaged out in the following format:

Max length = 18 bits; Min length = 11 bits

1-8 bits	4 bits	6 bits
I-CNT	SRC	TCODE (000100)

Figure B-18. Direct Branch Message Format

NOTE

When DC[25] is cleared in Thumb mode or the ARM7 processor is in full 32-bit ARM mode, Direct Branch Messages will not be transmitted. Instead, each direct branch or predicated instruction will toggle a bit in the history buffer.

B.4.2.2.4 Resource Full Messages

The Resource Full Message is used in conjunction with the Branch History Messages. The Resource Full Message is generated when the internal branch/predicate history buffer is full. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

The current value of the history buffer is transmitted as part of the Resource Full Message. This information can be concatenated by the tool with the branch/predicate history information from

subsequent messages to obtain the complete branch history for a message. The history value is reset by this message, and the I-CNT value is reset as a result of a bit being added to the history buffer

Max length = 46 bits; Min length = 15 bits

1-32 bits	4 bits	4 bits	6 bits
HIST	RCODE (0001)	SRC	TCODE (011011)

Figure B-19. Resource Full Message Format

B.4.2.2.5 Debug Status Messages

Debug Status Messages report low power mode and debug status. Entering/exiting Debug Mode as well as entering a Low Power Mode will trigger a Debug Status Message. Debug status information is sent out in the following format:

Fixed length = 18 bits

8 bits	4 bits	6 bits
DS[31:24]	SRC	TCODE (000000)

Figure B-20. Debug Status Message Format

B.4.2.2.6 Program Correlation Messages

Program Correlation Messages are used to correlate events to the program flow that may not be associated with the instruction stream. In order to maintain accurate instruction tracing information when entering debug mode or a CPU low power mode (where tracing may be disabled), this message is sent upon entry into one of these two modes and includes the instruction count and branch history. Program Correlation is messaged out in the following format

Max length = 54 bits; Min length = 16 bits

1-32 bits	1-8 bits	4 bits	4 bits	6 bits
HIST	I-CNT	ECODE	SRC	TCODE (100001)

Figure B-21. Program Correlation Message Format

B.4.2.2.7 BTM Overflow Error Messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a Program Trace Message attempts to enter the queue while it is being emptied, the Error Message will incorporate the Program Trace only error encoding (00001). If both OTM and Program Trace Messages attempt to enter the queue, the Error Message will incorporate the OTM and Program Trace error encoding (00111). If a Watchpoint also attempts to be queued while the FIFO is being emptied, then the Error Message will incorporate error encoding (01000).

NOTE

The OVC bits within the DC Register can be set to delay the CPU by asserting the FIFOFULL signal in order to alleviate (but not eliminate) potential overruns.

Error information is messaged out in the following format

Fixed length = 15 bits

5 bits	4 bits	6 bits
ERROR (00001 / 00111 / 01000)	SRC	TCODE (001000)

Figure B-22. Error Message Format

B.4.2.2.8 Program Trace Synchronization Messages

A Program Trace Direct/Indirect Branch with Sync. or Indirect Branch History with Sync. Message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table B-25):

- Initial Program Trace Message upon the first direct (traditional only) or indirect branch after exit from system reset or whenever program trace is enabled.
- Upon direct (traditional only) or indirect branch after returning from a Low Power state.
- Upon direct (traditional only) or indirect branch after returning from Debug Mode.
- Upon direct (traditional only) or indirect branch after occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon direct (traditional only) or indirect branch after the periodic program trace counter has expired indicating 255 without-sync Program Trace Messages have occurred since the last with-sync message occurred.
- Upon direct (traditional only) or indirect branch after assertion of the Event In ($\overline{\text{EVTI}}$) pin if the EIC bits within the DC Register have enabled this feature.
- Upon direct (traditional only) or indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred between branches or since the last bit was entered in the history field.
- Upon direct (traditional only) or indirect branch after a BTM Message was lost due to an attempted access to a secure memory location (for chips with security).
- Upon direct (traditional only) or indirect branch after a BTM Message was lost due to a collision entering the FIFO between the BTM Message and any of the following: Error Message, Watchpoint Message, Debug Status Message or Ownership Trace Message.

If the A7S Nexus 2 module is enabled at reset, an $\overline{\text{EVTI}}$ assertion initiates a Program Trace Indirect Branch History with Sync. Message (if Program Trace is enabled) upon the first indirect branch. The message will be a history type message because the ARM7 core will be in full 32-bit ARM mode upon exit from reset. The history field will contain all taken/not taken direct branch and predicated instructions which occur before the first indirect branch.

The formats for Program Trace Direct/Indirect Branch with Sync. Messages and Indirect Branch History with Sync. Messages are as follows

Max length = 82 bits; Min length = 13 bits

1-32 bits	1-32 bits	1-8 bits	4 bits	6 bits
HIST	F-ADDR	I-CNT	SRC	TCODE (011101)

Figure B-23. Indirect Branch History w/ Sync. Message Format

Max length = 50 bits; Min length = 12 bits

1-32 bits	1-8 bits	4 bits	6 bits
F-ADDR	I-CNT	SRC	TCODE (001011 or 001100)

Figure B-24. Direct/Indirect Branch with Sync. Message Format (traditional)

Exception conditions that result in Program Trace Synchronization are summarized in Table B-25.

Table B-25. Program Trace Exception Summary

Exception Condition	Exception Handling
System Reset Negation	Upon entry into JTAG Test-Logic-Reset state, queue pointers, counters, state machines, and registers within the ARM7 Nexus module are reset. Upon the first branch out of system reset (if Program Trace is enabled), the first Program Trace Message is a Direct/Indirect Branch w/ Sync. Message.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next direct/indirect branch will be converted to a Direct/Indirect Branch with Sync. Message.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Direct/Indirect Branch w/ Sync. Message.
Periodic Program Trace Synchronization	A forced synchronization occurs periodically after 255 Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Event In	If the Nexus module is enabled, an $\overline{\text{EVTI}}$ assertion initiates a Direct/Indirect Branch w/ Sync. Message upon the next direct/indirect branch (if Program Trace is enabled and the EIC bits of the DC Register have enabled this feature).
Sequential Instruction Count Overflow	When the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch.
Attempted Access to Secure Memory	For chips which implement security, any attempted branch to secure memory locations will temporarily disable Program Trace and cause the corresponding BTM to be lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will be inaccurate since the re-enable of Program Trace is not necessarily aligned on an instruction boundary.

Table B-25. Program Trace Exception Summary

Exception Condition	Exception Handling
Collision Priority	All Messages have the following priority: Error → WPM → OTM → DS → BTM. A BTM Message which attempts to enter the queue at the same time as an Error Message, Watchpoint Message, Ownership Trace Message or Debug Status Message will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. Instruction counts are not reset when a BTM is lost so subsequent instructions will be added to the preempted message's instruction count until a change of flow or predicated instruction is reached. If a message is generated as a result of the subsequent change of flow, then the instruction count in that message will include the instruction count of the preempted message. Similarly, the history buffer is not reset when a BTM is lost due to collision. In ARM mode, the branch that caused the preempted message will receive a history bit since indirect branches may be conditional in ARM mode.

B.4.2.3 BTM Operation

B.4.2.3.1 Enabling Program Trace

Both types of Branch Trace Messaging can be enabled in one of two ways.

- Setting the TM field of the DC Register to enable Program Trace (DC[2]).
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (ARM7 watchpoints are configured within the CPU).

NOTE

Setting DC[25] will select the traditional Branch Trace Messaging format when in Thumb mode. By default, Branch History format is used (DC[25] = 0b0). Full 32-bit ARM mode always utilizes the Branch History Message format.

B.4.2.3.2 Addressing

The ARM7 architecture supports a processor mode switch into Thumb mode with the least significant bit of the address bus set. The A7S Nexus 2 module ignores this bit and always treats it as if it is zero (all instruction addresses are aligned) for the purpose of program trace messaging.

The relative address feature is compliant with the **IEEE-ISTO 5001-2003** standard recommendations, and is designed to reduce the number of bits transmitted for addresses of Indirect Branch Messages.

The address transmitted is relative to the target address of the instruction which triggered the previous Indirect Branch (or Sync) Message. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant '1' in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address.

Previous Address (A1) = 0x0003FC01, New Address (A2) = 0x0003F365

Message Generation:

```
     A1 = 0000 0000 0000 0011 1111 1100 0000 0001
     A2 = 0000 0000 0000 0011 1111 0011 0110 0101

A1 ⊕ A2 = 0000 0000 0000 0000 0000 1111 0110 0100
                                    └──────────┘
                  Address Message (M1) = 1111 0110 0100
```

Address Re-creation:

```
A1 ⊕ M1 = A2
     A1 = 0000 0000 0000 0011 1111 1100 0000 0001
     M1 = 0000 0000 0000 0000 0000 1111 0110 0100
          ---------------------------------------

     A2 = 0000 0000 0000 0011 1111 0011 0110 0101
```

Figure B-25. Relative Address Generation and Re-creation

B.4.2.3.3 Branch/Predicate Instruction History (HIST)

In full 32-bit ARM mode (and optionally in Thumb mode), BTM messaging will use the Branch History format. The branch history (HIST) packet in these messages provides a history of direct branch execution used for reconstructing the program flow. This packet is implemented as a left-shifting shift register. The register is always pre-loaded with a value of one. This bit acts as a stop bit so that the development tools can determine which bit is the end of the history information. The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one is shifted into the history buffer on a taken direct branch (conditional or unconditional) and on any instruction whose predicate condition resolved as true. A value of zero is shifted into the history buffer on any instruction whose predicate condition executed as false as well as on branches not taken. This will include indirect as well as direct branches not taken.

B.4.2.3.4 Sequential Instruction Count (I-CNT)

The I-CNT packet, is present in all BTM Messages. For traditional Branch Messages (Thumb mode only), I-CNT represents the number of sequential ARM7 instructions, or non-taken branches in between Direct/Indirect Branch Messages.

For Branch History Messages in Thumb mode, I-CNT represents the number of ARM7 instructions executed since the last taken/non-taken direct branch, last taken indirect branch or exception. Not taken indirect branches are considered sequential instructions and cause the instruction count to increment. For Branch History Messages in ARM mode, I-CNT also represents the number of ARM7 instructions executed since the last predicate instruction.

The sequential instruction counter overflows when its value reaches 255. The next BTM Message following an instruction counter overflow will be converted to a synchronization type message.

NOTE

When an undefined instruction causes an exception, the undefined instruction itself will be included in the BTM instruction count.

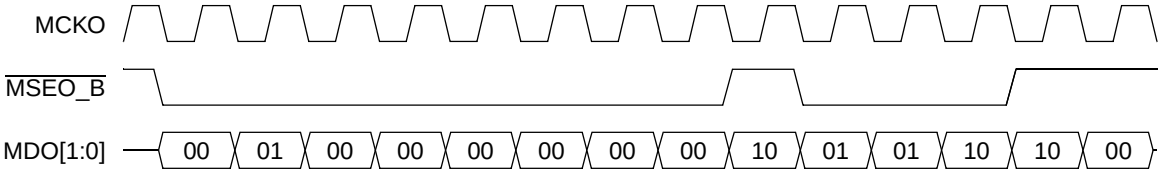
B.4.2.3.5 Program Trace Queueing

A7S Nexus 2 implements a programmable depth queue (32 minimum entry recommended) for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

NOTE

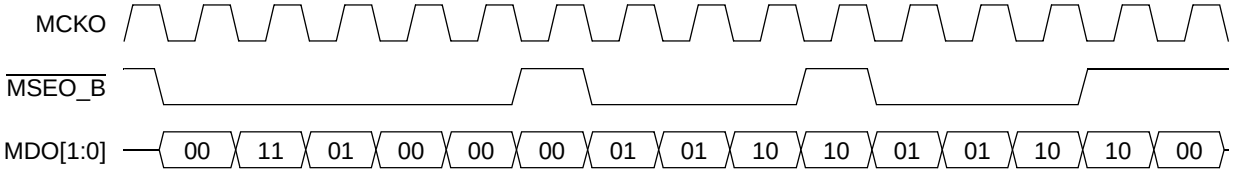
If multiple trace messages need to be queued at the same time, Watchpoint Messages will have the highest priority (WPM → OTM → BTM).

B.4.2.4 Program Trace Timing Diagrams (2 MDO / 1 $\overline{\text{MSEO}}$ configuration)



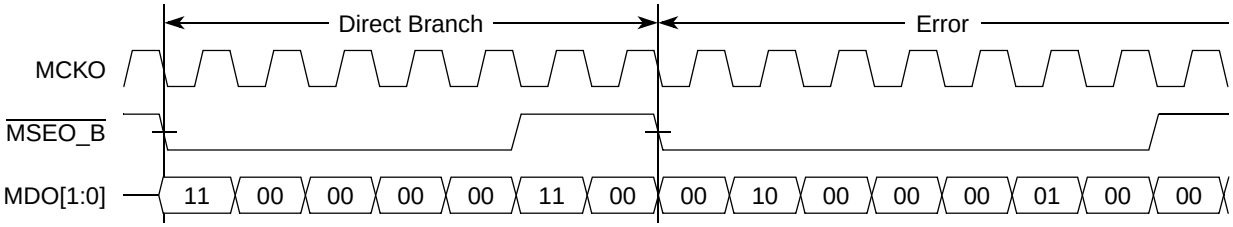
TCODE = 4
Source Processor = 0000
Number of Sequential Instructions = 128
Relative Address = 0xA5

Figure B-26. Program Trace - Indirect Branch Message (Traditional)



TCODE = 28
Source Processor = 0000
Number of Sequential Instructions = 0
Relative Address = 0xA5
Branch History = 0b10100101 (with Stop)

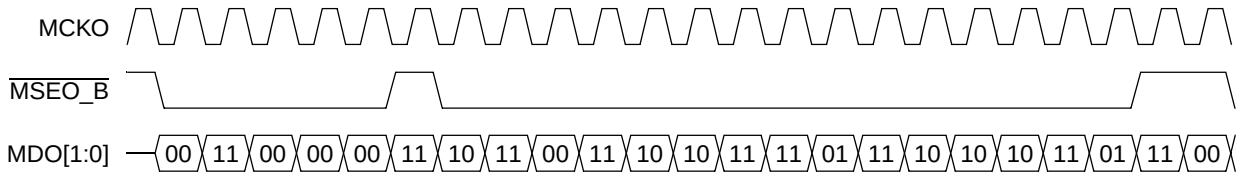
Figure B-27. Program Trace - Indirect Branch Message (History)



DBM:
TCODE = 3
Source Processor = 0000
Number of Sequential Instructions = 3

Error:
TCODE = 8
Source Processor = 0000
Error Code = 1 (Queue Overrun – BTM Only)

Figure B-28. Program Trace - Direct Branch (Traditional) and Error Messages



TCODE = 12
Source Processor = 0000
Number of Sequential Instructions = 3
Full Target Address = 0xDEADFACE

Figure B-29. Program Trace - Indirect Branch with Sync. Message (Traditional)

B.4.3 Watchpoint Support

The A7S Nexus 2 module provides Watchpoint Messaging via the auxiliary pins, as defined by the IEEE-ISTO 5001-2003 standard.

A7S Nexus 2 is not compliant with Class4 breakpoint/watchpoint requirements defined in the standard. The Breakpoint/Watchpoint Control Register is not implemented within Nexus 2.

B.4.3.1 Watchpoint Messaging

Enabling Watchpoint Messaging is done by setting the Watchpoint Enable bit in the DC Register. Watchpoint setting is supported through the ARM7 EmbeddedICE module. Please refer to the debug chapter of the appropriate “ARM7 Technical Reference Manual” for details on Watchpoint initialization.

The Nexus 2 module provides Watchpoint Messaging using the IEEE-ISTO 5001-2003 defined TCODE. The ARM7 EmbeddedICE module is capable of setting up to two address and/or data watchpoints. When either of these watchpoints occur, a watchpoint event signal from the EmbeddedICE module (DBGRNG[1:0]) causes a message to be sent to the queue to be messaged out. This message indicates the watchpoint number.

The occurrence of either watchpoint can be programmed to assert the Event Out ($\overline{\text{EVTO}}$) pin for one period of the output clock (MCKO) (see Table B-7 for additional information on $\overline{\text{EVTO}}$).

Fixed length = 14 bits

4 bits	4 bits	6 bits
Watchpoint Source (00xx)	Src. Proc.	TCODE (001111)

Figure B-30. Watchpoint Message Format

Table B-26. Watchpoint Source Description

Watchpoint Source (4-bits)	Watchpoint Description
00X1	ARM7 Watchpoint #1 (DBGRNG[0])
001X	ARM7 Watchpoint #2 (DBGRNG[1[)

B.4.3.2 Watchpoint Error Message

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a Watchpoint Message attempts to enter the queue while it is being emptied, the Error Message will incorporate the Watchpoint only error encoding (00110). If an OTM and/or Program Trace Message also attempts to enter the queue while it is being emptied, the Error Message will incorporate error encoding (01000).

NOTE

The OVC bits within the DC Register can be set to delay the CPU by asserting the FIFOFULL signal in order to alleviate (but not eliminate) potential overruns.

Error information is messaged out in the following format (see Table B-3):

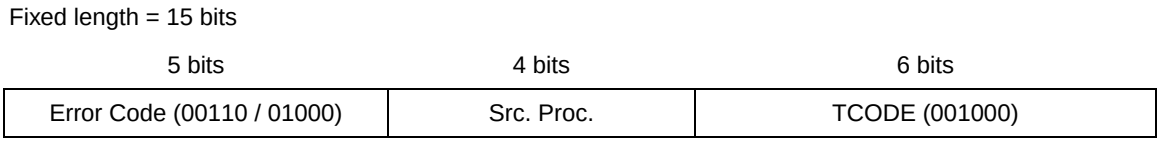


Figure B-31. Error Message Format

B.4.3.3 Watchpoint Timing Diagram (2 MDO / 1 $\overline{\text{MSEO}}$ configuration)

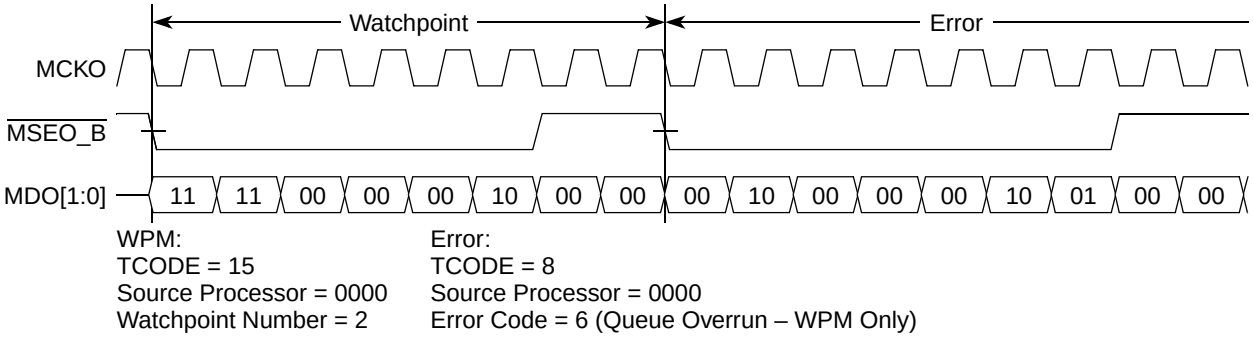


Figure B-32. Watchpoint Message and Watchpoint Error Message

B.4.4 Read/Write Access

The Read/Write access feature allows access to internal memory mapped resources via the JTAG port. The Read/Write mechanism supports single as well as block reads and writes to ARM7 AHB resources.

B.4.4.1 Functional Description

The Nexus 2 module includes the class 3 capability of accessing resources on the ARM7 AHB with multiple configurable priority levels. Internal memory mapped registers and non-cache memory (for processors which implement a cache) can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD).

B.4.4.2 Read/Write Access to Internal Nexus Registers

Access to Nexus register resources is enabled by loading a single instruction ("NEXUS-ACCESS") into the JTAG Instruction Register (IR). For the A7S Nexus 2 block, the JTAG IR value is programmable at the platform or chip integration level. It can be programmed to any of the non-ARM7 implemented IR values (see Table B-21).

Table B-27. JTAG Nexus3 Register Select

JTAG Instruction	JTAG Access Opcode	Read/Write
NEXUS-ACCESS	0xXX (hex) (programmable)	W

Once the JTAG "NEXUS-ACCESS" instruction has been loaded, the JTAG port allows communication with all Nexus registers according to the map in Table B-14.

Reading/writing of a Nexus register then requires two passes through the Data-Scan (DR) path of the JTAG state machine (see Section B.6, "IEEE 1149.1 State Machine and RD/WR Sequences").

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (see Table B-14), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:

(7 bits)	(1 bit)
Nexus Register Index	R/W

RESET value: 0x00

Nexus Register Index	Selected from values in Table B-14
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a) During a read access, data is latched from the selected Nexus register when the JTAG state machine (see Section B.6, "IEEE 1149.1 State Machine and RD/WR Sequences") passes through the "Capture-DR" state.
   b) During a write access, data is latched into the selected Nexus register when the JTAG state machine (see Section B.6, "IEEE 1149.1 State Machine and RD/WR Sequences") passes through the "Update-DR" state.

B.4.4.3 Memory Mapped Register Access via JTAG

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped ARM7 AHB resources can be accessed through Nexus. The following steps are required to access memory mapped resources:

NOTE

Read/Write Access can only access memory mapped resources when system reset is negated.

B.4.4.3.1 Single Write Access

1. Initialize the Read/Write Access Address Register (RWA) through the JTAG access method outlined in Table B.4.4.2 using the Nexus Register Index of 0x9 (see Table B-14). Configure as follows:
   — Write Address → 0xXXXX_XXXX (write address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the JTAG access method outlined in Table B.4.4.2 using the Nexus Register Index of 0x7 (see Table B-14). Configure the bits as follows:
   — Access Control (AC) → 0b1 (to indicate start access)
   — Map Select (MAP) → 0b000 (primary memory map)
   — Access Priority (PR) → 0b00 (lowest priority)
   — Read/Write (RW) → 0b1 (write access)
   — Word Size (SZ) → 0b0xx (32-bit, 16-bit, 8-bit)
   — Access Count (CNT) → 0x0000 or 0x0001(single access)

NOTE

Access Count (CNT) of 0x0000 or 0x0001 will perform a single access.

3. Initialize the Read/Write Access Data Register (RWD) through the JTAG access method outlined in Table B.4.4.2 using the Nexus Register Index of 0xA (see Table B-14). Configure as follows:
   — Write Data → 0xXXXX_XXXX (write data)
4. The Nexus block will then arbitrate for the AHB and transfer the data value from the RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA).
   When the access has completed without error (ERR = 0b0), the Nexus block asserts the $\overline{\text{RDY}}$ pin (see Table B-6 for detail on $\overline{\text{RDY}}$) and clears the DV bit in the RWCS Register. This indicates that the device is ready for the next access.

NOTE

Only the $\overline{\text{RDY}}$ pin as well as the DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

B.4.4.3.2 Block Write Access

1. For a block write access, follow Steps 1, 2, and 3 outlined in Table B.4.4.3.1 to initialize the registers, using a value greater than one (0x0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB and transfer the first data value from the RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA).
   When the transfer has completed without error (ERR = 0b0), the address from the RWA Register is incremented to the next word size (specified in the SZ field) and the number from the CNT field

is decremented. The Nexus block will then assert the $\overline{\text{RDY}}$ pin. This indicates that the device is ready for the next access.

NOTE

The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

3. Repeat Step 3 in Table B.4.4.3.1 until the internal CNT value is zero. When this occurs, the DV bit within the RWCS will be cleared to indicate the end of the block write access.

B.4.4.3.3 Single Read Access

1. Initialize the Read/Write Access Address Register (RWA) through the JTAG access method outlined in Table B.4.4.2 using the Nexus Register Index of 0x9 (see Table B-14). Configure as follows:
   — Read Address → 0xXXXX_XXXX (read address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the JTAG access method outlined in Table B.4.4.2 using the Nexus Register Index of 0x7 (see Table B-14). Configure the bits as follows:
   — Access Control (AC) → 0b1 (to indicate start access)
   — Map Select (MAP) → 0b000 (primary memory map)
   — Access Priority (PR) → 0b00 (lowest priority)
   — Read/Write (RW) → 0b0 (read access)
   — Word Size (SZ) → 0b0xx (32-bit, 16-bit, 8-bit)
   — Access Count (CNT) → 0x0000 or 0x0001(single access)

NOTE

Access Count (CNT) of 0x0000 or 0x0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB and the read data will be transferred from the AHB to the RWD Register.
   When the transfer completed without error (ERR = 0b0), the Nexus block asserts the $\overline{\text{RDY}}$ pin (see Table B-6 for detail on $\overline{\text{RDY}}$) and sets the DV bit in the RWCS Register. This indicates that the device is ready for the next access.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the JTAG access method outlined in Table B.4.4.2 using the Nexus Register Index of 0xA (see Table B-14).

NOTE

Only the $\overline{\text{RDY}}$ pin as well as the DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

B.4.4.3.4 Block Read Access

1. For a block read access, follow Steps 1 and 2 outlined in Table B.4.4.3.3 to initialize the registers, using a value greater than one (0x0001) for the CNT field in the RWCS Register.

2. The Nexus block will then arbitrate for the AHB and the read data will be transferred from the AHB to the RWD Register.

   When the transfer has completed without error (ERR = 0b0), the address from the RWA Register is incremented to the next word size (specified in the SZ field) and the number from the CNT field is decremented. The Nexus block will then assert the $\overline{\text{RDY}}$ pin. This indicates that the device is ready for the next access.

   NOTE

   The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block read access. The original values can be read by the external development tool at any time.

3. The data can then be read from the Read/Write Access Data Register (RWD) through the JTAG access method outlined in Table B.4.4.2 using the Nexus Register Index of 0xA (see Table B-14).
4. Repeat Steps 3 and 4 in Table B.4.4.3.3 until the CNT value is zero. When this occurs, the DV bit within the RWCS is set to indicate the end of the block read access.

B.4.4.4 Error Handling

The A7S Nexus 2 module handles various error conditions as follows:

B.4.4.4.1 AHB Read/Write Error

All address and data errors that occur on read/write accesses to the ARM7 AHB will return a transfer error encoding on the HRESP[1:0] signals. If HRESP[1:0] = 0b01:

1. The access is terminated without re-trying (AC bit is cleared)
2. The ERR bit in the RWCS Register is set
3. The Error Message is sent (TCODE = 8) indicating Read/Write Error

B.4.4.4.2 Access Termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
   a) If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
   b) If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

B.4.4.4.3 Read/Write Access Error Message

The Read/Write Access Error Message is sent out when an AHB access error (read or write) error has occurred.

Error information is messaged out in the following format:

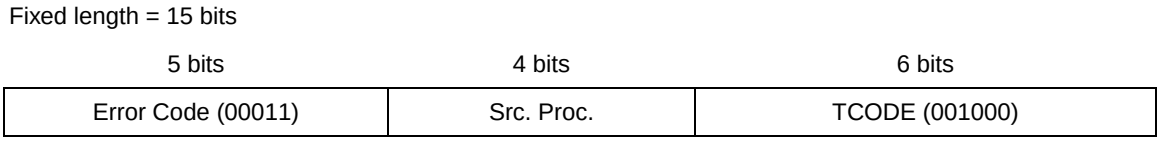


Figure B-33. Error Message Format

B.4.4.5 Timing Diagram

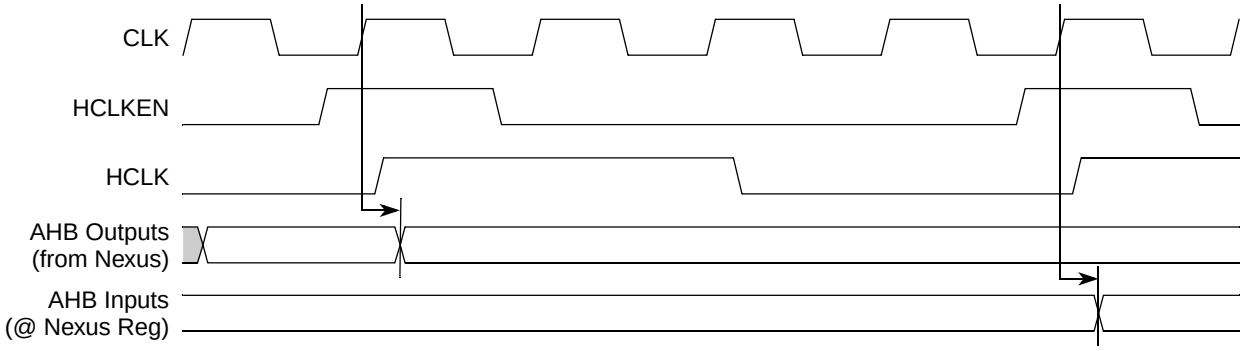


Figure B-34. A7S Nexus 3 DMA clock relationships

The A7S Nexus 2 Read/Write access timing meets the timing requirements for the ARM7 AHB. The Nexus module uses the processor clock gated with an AHB clock enable for all DMA transfers. This clock will correspond to the rising edge of the actual AHB clock. The timing diagram in Figure B-34 above shows the relationship between the processor clock (CLK), the AHB clock (HCLK) and the AHB clock enable (HCLKEN) for DMA writes and reads. Using this clocking method for Nexus read/write access eliminates the need for a separate asynchronous clock input into the A7S Nexus3 module.

B.5 Electrical Characteristics

B.5.1 Maximum Ratings / DC Electrical Specifications

All DC electrical characteristics related to ARM7 and A7S Nexus 2 operation are implementation specific. Please refer to the *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC) for specific device characteristics.

B.5.2 A7S Nexus 2 Auxiliary Pin Timing Specifications

Figure B-35. MCKO Related Timing

Num	MCKO Related Characteristic	Symbol	Min	Max	Unit
	MCKO Frequency of Operation		0	—[1]	MHz
1	MCKO Clock Cycle Time	T_{min}	1/max freq	—[1]	ns
2	MCKO Rise and Fall Times		0	—[1]	ns
3	MCKO Low to MDO Data Valid		0	$0.20 \times T_{min}$	ns
4	MCKO Low to $\overline{\text{EVTO}}$ Valid		0	$0.20 \times T_{min}$	ns
5	$\overline{\text{EVTO}}$ Pulse Width Time		$1 \times T_{min}$	$1 \times T_{min}$	ns

1 The timing specifications of Nexus related clocks (MCKO and TCK) are determined by the MCU and all Nexus related signals are relative to the MCU defined clock specifications in the *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC).

Figure B-36. TCK Related Timing

Num	TCK Related Characteristic	Symbol	Min	Max	Unit
	TCK Frequency of Operation		0	—[1]	MHz
6	TCK Clock Cycle Time	T_{TCK}	1/max freq	—	ns
7	$\overline{\text{EVTI}}$ Pulse Width Time		$4 \times T_{TCK}$	—	ns
8	$\overline{\text{EVTI}}$ to $\overline{\text{TSRT}}$ Negation Setup (@ reset only)		$2 \times T_{TCK}$	—	ns
9	$\overline{\text{EVTI}}$ to $\overline{\text{TSRT}}$ Negation Hold (@ reset only)		$2 \times T_{TCK}$	—	ns

1 The timing specifications of Nexus related clocks (MCKO and TCK) are determined by the MCU and all Nexus related signals are relative to the MCU defined clock specifications in the *MAC7100 Microcontroller Family Hardware Specifications* (MAC7100EC).

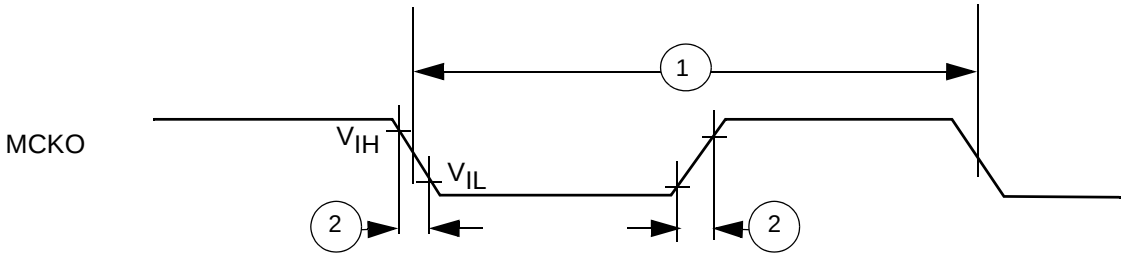


Figure B-37. MCKO Timing

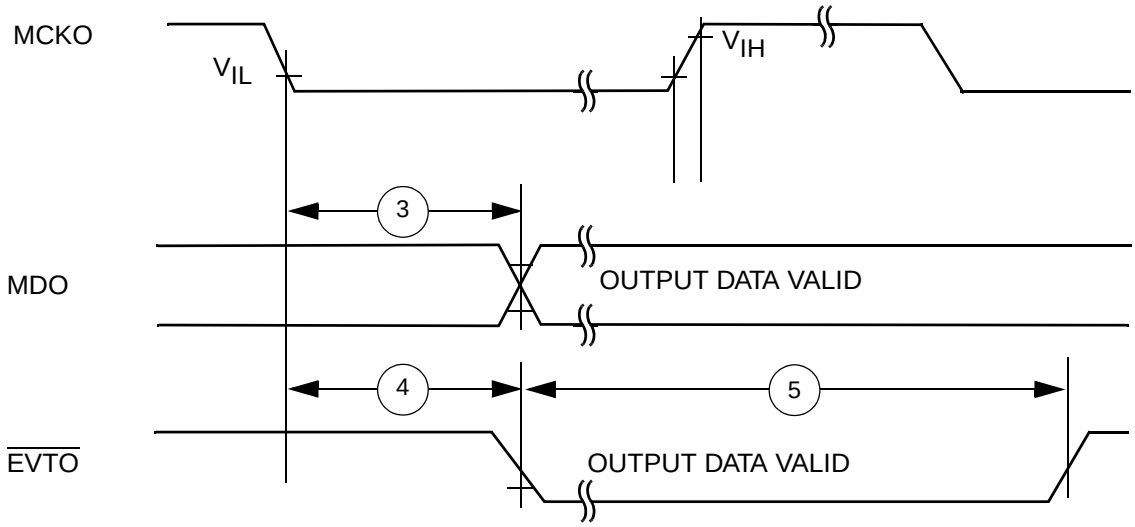


Figure B-38. MDO, $\overline{\text{EVTO}}$ Timing

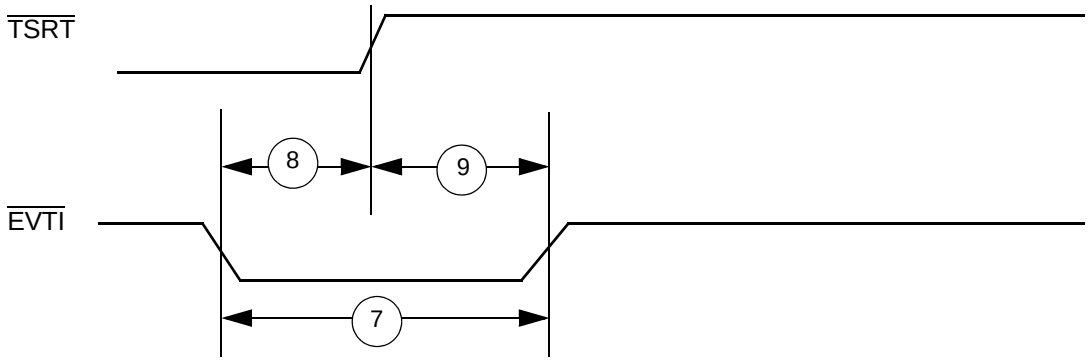


Figure B-39. $\overline{\text{EVTI}}$ Timing

B.6 IEEE 1149.1 State Machine and RD/WR Sequences

B.6.1 JTAG State Machine

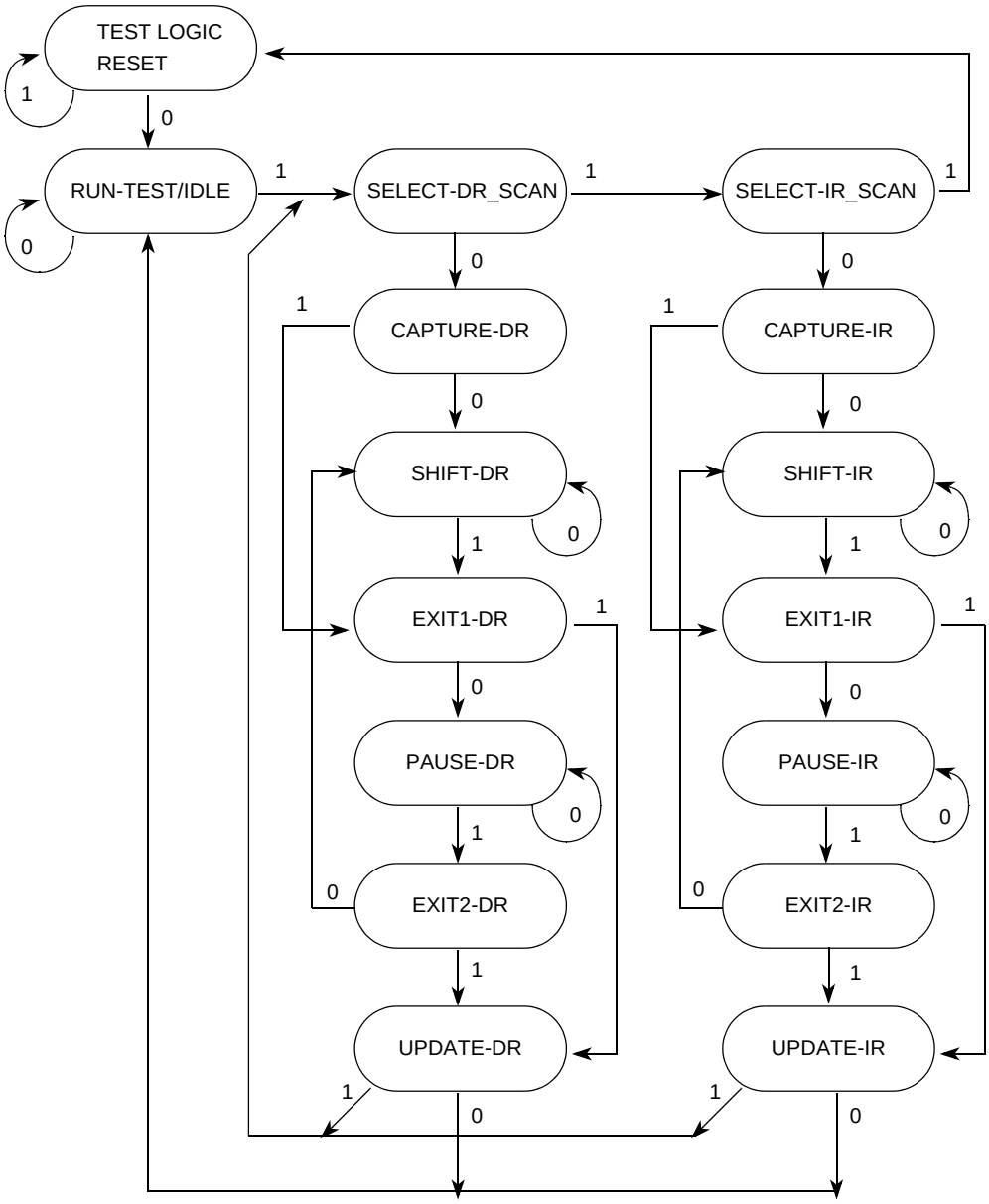


Figure B-40. JTAG State Machine

B.6.2 JTAG Sequence for Accessing Internal Nexus Registers

Table B-28. JTAG Sequence for Accessing Internal Nexus Registers

Step #	TMS Pin	Description
1	1	IDLE → SELECT-DR_SCAN
2	0	SELECT-DR_SCAN → CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR → SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR → EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR → UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR → SELECT-DR_SCAN
8	0	SELECT-DR_SCAN → CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR → SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR → EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR → UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR → RUN-TEST/IDLE (transfer complete - Nexus controller to Reg. Select state)

B.6.3 JTAG Sequence for Read Access of Memory-Mapped Resources

Table B-29. JTAG Sequence for Read Access of Memory-Mapped Resources

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address - data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value - data input on TDI)
5	—	Wait for falling edge of $\overline{\text{RDY}}$ pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #6

B.6.4 JTAG Sequence for Write Access of Memory-Mapped Resources

Table B-30. JTAG Sequence for Write Access of Memory-Mapped Resources

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register
2	37	Write RWCS (initialize write access mode and CNT value - data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address - data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of $\overline{\text{RDY}}$ pin
8	—	If CNT > 0, go back to Step #5

Appendix C
Register Memory Map Quick Reference

C.1 Peripherals Register Memory Map

As described in Chapter 8, “Device Memory Map,” and Chapter 16, “AMBA to IP Bus Bridge Module (AIPS),” each module occupies a 16 KByte space in the device memory map as summarized in Table C-1. Table C-2 through Table C-21 detail each module memory map.

Table C-1. MAC7100 Family Peripheral Memory Map

Address	Module Name	Module Map
0xFC00 0000 – 0xFC00 3FFF	AMBA to IP Bus Bridge (AIPS) Configuration Registers	Table C-2
0xFC00 4000 – 0xFC00 7FFF	Crossbar Bus Switch (XBS) Configuration Registers	Table C-3
0xFC00 8000 – 0xFC00 BFFF	External Interface Module (EIM) Configuration Registers	Table C-4
0xFC04 0000 – 0xFC04 3FFF	Miscellaneous Control Module (MCM)	Table C-5
0xFC04 4000 – 0xFC04 7FFF	Enhanced Direct Memory Access (eDMA) Controller	Table C-6, C-6a
0xFC04 8000 – 0xFC04 BFFF	Interrupt Controller (INTC)	Table C-7
0xFC08 0000 – 0xFC08 3FFF	System Services Module (SSM)	Table C-8
0xFC08 4000 – 0xFC08 7FFF	Direct Memory Access Controller Multiplexer (DMA Mux)	Table C-9
0xFC08 8000 – 0xFC08 BFFF	Clock and Reset Generator (CRG)	Table C-10
0xFC08 C000 – 0xFC08 FFFF	Programmable Interval Timer (PIT)	Table C-11
0xFC09 0000 – 0xFC09 3FFF	Voltage Regulator (VREG)	Table C-12
0xFC09 4000 – 0xFC09 7FFF	CAN controller A (FlexCAN_A)	Table C-13
0xFC09 8000 – 0xFC09 BFFF	CAN controller B (FlexCAN_B)	Table C-13
0xFC09 C000 – 0xFC09 FFFF	CAN controller C (FlexCAN_C) [1]	Table C-13
0xFC0A 0000 – 0xFC0A 3FFF	CAN controller D (FlexCAN_D) [1]	Table C-13
0xFC0A C000 – 0xFC0A FFFF	Inter-IC bus (I^2C)	Table C-14
0xFC0B 4000 – 0xFC0B 7FFF	Serial Peripheral Interface A (DSPI_A)	Table C-15
0xFC0B 8000 – 0xFC0B BFFF	Serial Peripheral Interface B (DSPI_B) [2]	Table C-15
0xFC0C 4000 – 0xFC0C 7FFF	Enhanced Serial Communication Interface A (eSCI_A)	Table C-16
0xFC0C 8000 – 0xFC0C BFFF	Enhanced Serial Communication Interface B (eSCI_B)	Table C-16
0xFC0C C000 – 0xFC0C FFFF	Enhanced Serial Communication Interface C (eSCI_C) [3]	Table C-16
0xFC0D 0000 – 0xFC0D 3FFF	Enhanced Serial Communication Interface D (eSCI_D) [3]	Table C-16
0xFC0D C000 – 0xFC0D FFFF	Enhanced Modular I/O Subsystem (eMIOS)	Table C-17, C-17a
0xFC0E 0000 – 0xFC0E 3FFF	Analog-to-Digital Converter A (ATD_A)	Table C-18
0xFC0E 4000 – 0xFC0E 7FFF	Analog-to-Digital Converter B (ATD_B) [4]	Table C-18
0xFC0E 8000 – 0xFC0E BFFF	Port Integration Module (PIM)	Table C-19, C-20
0xFC0F 0000 – 0xFC0F 3FFF	Common Flash Module (CFM) Registers	Table C-21

[1] For MAC7100 Family devices that do not implement CAN C or D, these memory map areas must be treated as reserved.

[2] For MAC7100 Family devices that do not implement DSPI B, this memory map area must be treated as reserved.

[3] For MAC7100 Family devices that do not implement eSCI C or D, these memory map areas must be treated as reserved.

[4] For MAC7100 Family devices that do not implement ATD B, this memory map area must be treated as reserved.

Table C-2. AIPS Memory Map

Base Address: 0xFC00_0000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	16-255	Master Protection Register A (MPRA)	MPROT0	MPROT1						
0x0004–0x001C	—	Reserved								
0x0020	16-255	Peripheral Access Control Register A (PACRA)	PAC0	PAC1	PAC2					
0x0024	—	Reserved								
0x0028	16-255	Peripheral Access Control Register C (PACRC)	PAC16	PAC17	PAC18					
0x002C	—	Reserved								
0x0040	16-256	Off-Platform Peripheral Access Control Register A (OPACRA)	OPAC0	OPAC1	OPAC2	OPAC3	OPAC4	OPAC5	OPAC6	OPAC7
0x0044	16-256	Off-Platform Peripheral Access Control Register B (OPACRB)	OPAC8			OPAC11		OPAC13	OPAC14	
0x0048	16-256	Off-Platform Peripheral Access Control Register C (OPACRC)		OPAC17	OPAC18	OPAC19	OPAC20			OPAC23
0x004C	16-256	Off-Platform Peripheral Access Control Register D (OPACRD)	OPAC24	OPAC25	OPAC26		OPAC28			
0x0050	16-256	Off-Platform Peripheral Access Control Register E (OPACRE)	OPAC32	OPAC33						

Table C-3. XBS Memory Map

Base Address: 0xFC00_4000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000–0x00FC	—	Reserved								
0x0100	14-205	Priority Register for EIM Slave port	PR_EIM							
0x0104–0x010C	—	Reserved								
0x0110	14-206	Control Register for EIM Slave port	CR_EIM							
0x0114–0x02FC	—	Reserved								
0x0300	14-205	Priority Register for SRAM Slave port	PR_SRAM							
0x0304–0x030C	—	Reserved								
0x0310	14-206	Control Register for SRAM Slave port	CR_SRAM							
0x0314–0x06FC	—	Reserved								
0x0700	14-205	Priority Register for Peripheral Controller Slave port	PR_PC							
0x0704–0x070C	—	Reserved								
0x0710	14-206	Control Register for Peripheral Controller Slave port	CR_PC							

Table C-4. EIM Memory Map

Base Address: 0xFC00_8000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000–0x007C	—	Reserved								
0x0080	13-185	Chip Select Address Register 0	CSAR0							
0x0084	13-186	Chip Select Mask Register 0	CSMR0							
0x0088	13-187	Chip Select Control Register 0					CSCR0			
0x008C	13-185	Chip Select Address Register 1	CSAR1							
0x0090	13-186	Chip Select Mask Register 1	CSMR1							
0x0094	13-187	Chip Select Control Register 1					CSCR1			
0x0098	13-185	Chip Select Address Register 2	CSAR2							
0x009C	13-186	Chip Select Mask Register 2	CSMR2							
0x00A0	13-187	Chip Select Control Register 2					CSCR2			

Table C-5. MCM Memory Map

Base Address: 0xFC04_0000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	11-124 11-125	Processor Core Type Register Device Revision Register	PCT				REV			
0x0004	11-126 11-126	XBS Master Configuration Register XBS Slave Configuration Register	AMC				ASC			
0x0008	11-126	IPS On-Platform Module Configuration Register	IOPMC							
0x000C	11-127	Reset Status Register							MRSR	
0x0010	11-128	Wake-up Control Register							MWCR	
0x0014	11-129	Software Watchdog Timer Control Register					MSWTCR			
0x0018	11-131	SWT Service Register							MSWTSR	
0x001C	11-132	SWT Interrupt Register							MSWTIR	
0x0020	11-132	XBS Address Map Register	AAMR							
0x0024–0x006C	—	Reserved								
0x0070	11-135	Core Fault Address Register	CFADR							
0x0074	11-136 11-137	Core Fault Location Register Core Fault Attributes Register					CFLOC		CFATR	
0x0078	—	Reserved								
0x007C	11-138	Core Fault Data Register	CFDTR							

Table C-6. eDMA Memory Map

Base Address: 0xFC04_4000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	12-144	eDMA Control Register	DMACR							
0x0004	12-145	eDMA Error Status Register	DMAES							
0x0008	—	Reserved								
0x000C	12-147	eDMA Enable Request Register					DMAERQ			
0x0010	—	Reserved								
0x0014	12-148	eDMA Enable Error Interrupt Register					DMAEEI			
0x0018	12-149 12-149 12-150 12-150	eDMA Set Enable Request Register eDMA Clear Enable Request Register eDMA Set Enable Error Interrupt Register eDMA Clear Enable Error Interrupt Request Register	DMASERQ		DMACERQ		DMASEEI		DMACEEI	
0x001C	12-151 12-152 12-152 12-153	eDMA Clear Interrupt Request Register eDMA Clear Error Register eDMA Set START Bit Register eDMA Clear DONE Status Bit Register	DMACINT		DMACERR		DMASSRT		DMACDNE	
0x0020	—	Reserved								
0x0024	12-153	eDMA Interrupt Request Register					DMAINT			
0x0028	—	Reserved								
0x002C	12-154	eDMA Error Register					DMAERR			
0x0030–0x00FC	—	Reserved								
0x0100	12-155	eDMA Channel 0–3 Priority Registers	DCHPRI0		DCHPRI1		DCHPRI2		DCHPRI3	
0x0104	12-155	eDMA Channel 4–7 Priority Registers	DCHPRI4		DCHPRI5		DCHPRI6		DCHPRI7	
0x0108	12-155	eDMA Channel 7–11 Priority Registers	DCHPRI8		DCHPRI9		DCHPRI10		DCHPRI11	
0x010C	12-155	eDMA Channel 12–15 Priority Registers	DCHPRI12		DCHPRI13		DCHPRI14		DCHPRI15	
0x0110	—	Reserved								
0x1000–0x11E0	12-156	Transfer Control Descriptors 0–15	TCD0–15							

Table C-6a. eDMA TCD*n* Memory Map Detail

TCD*n* Offset			
TCD0	0x1000	TCD8	0x1100
TCD1	0x1020	TCD9	0x1120
TCD2	0x1040	TCD10	0x1140
TCD3	0x1060	TCD11	0x1160
TCD4	0x1080	TCD12	0x1180
TCD5	0x10A0	TCD13	0x11A0
TCD6	0x10C0	TCD14	0x11C0
TCD7	0x10E0	TCD15	0x11E0

Word Offset	Page	Description	
TCDn Offset + 0x00	12-157	Source Address	
TCDn Offset + 0x04	12-157	Transfer Attributes	Signed Source Address Offset
TCDn Offset + 0x08	12-158	Minor Byte Transfer Count	
TCDn Offset + 0x0C	12-159	Last Source Address Adjustment	
TCDn Offset + 0x10	12-159	Destination Address	
TCDn Offset + 0x14	12-160	Current Minor Loop Link, Major Loop Count	Signed Destination Address Offset
TCDn Offset + 0x18	12-161	Last Destination Address Adjustment/Scatter Gather Address	
TCDn Offset + 0x1C	12-161	Beginning Minor Loop Link, Major Loop Count	Control and Status

Table C-7. INTC Memory Map

Base Address: 0xFC04_8000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	10-108	INTC Interrupt Pending Register High (Requests 63-32)	IPRH							
0x0004	10-108	INTC Interrupt Pending Register Low (Requests 31-0)	IPRL							
0x0008	10-109	INTC Interrupt Mask Register High (Requests 63-32)	IMRH							
0x000C	10-109	INTC Interrupt Mask Register Low (Requests 31-0)	IMRL							
0x0010	10-110	INTC Interrupt Force Register High (Requests 63-32)	INTFRCH							
0x0014	10-110	INTC Interrupt Force Register Low (Requests 31-0)	INTFRCL							
0x0018	10-111	INTC Module Configuration Register							ICONFIG	
0x001C	10-112 10-112 10-113 10-114	INTC Set Interrupt Mask Register INTC Clear Interrupt Mask Register INTC Current Level Mask Register INTC Saved Level Mask Register	SIMR		CIMR		CLMASK		SLMASK	
0x0020–0x003C	—									
0x0040	10-115	INTC Interrupt Control Register 0–3	ICR0		ICR1		ICR2		ICR3	
0x0044	10-115	INTC Interrupt Control Register 4–7	ICR4		ICR5		ICR6		ICR7	
0x0048	10-115	INTC Interrupt Control Register 8–11	ICR8		ICR9		ICR10		ICR11	
0x004C	10-115	INTC Interrupt Control Register 12–15	ICR12		ICR13		ICR14		ICR15	
0x0050	10-115	INTC Interrupt Control Register 16–19	ICR16		ICR17		ICR18		ICR19	
0x0054	10-115	INTC Interrupt Control Register 20–23	ICR20		ICR21		ICR22		ICR23	
0x0058	10-115	INTC Interrupt Control Register 24–27	ICR24		ICR25		ICR26		ICR27	
0x005C	10-115	INTC Interrupt Control Register 28–31	ICR28		ICR29		ICR30		ICR31	
0x0060	10-115	INTC Interrupt Control Register 32–35	ICR32		ICR33		ICR34		ICR35	
0x0064	10-115	INTC Interrupt Control Register 36–39	ICR36		ICR37		ICR38		ICR39	
0x0068	10-115	INTC Interrupt Control Register 40–44	ICR40		ICR41		ICR42		ICR44	
0x006C	10-115	INTC Interrupt Control Register 45–47	ICR44		ICR45		ICR46		ICR47	
0x0070	10-115	INTC Interrupt Control Register 48–51	ICR48		ICR49		ICR50		ICR51	
0x0074	10-115	INTC Interrupt Control Register 52–55	ICR52		ICR53		ICR54		ICR55	
0x0078	10-115	INTC Interrupt Control Register 56–59	ICR56		ICR57		ICR58		ICR59	
0x007C	10-115	INTC Interrupt Control Register 60–63	ICR60		ICR61		ICR62		ICR63	
0x0080–0x00EA	—	Reserved								
0x00EC	10-115	INTC IRQ Acknowledge Register	IRQIACK							
0x00F0	10-116	INTC FIQ Acknowledge Register	FIQIACK							

Table C-8. SSM Memory Map

Base Address: 0xFC08_0000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	26-564	Current System Status Register					STATUS			
0x0004	26-566	System Memory Configuration Register	MEMCONFIG							
0x0008 [1, 2]	26-567	Wake-up Source Register [1]	WAKEUP [1]							
0x0008 [1, 2]	26-568	Error Configuration Register [2]	ERROR [2]							
0x000C	26-569 26-570	Port Select Register [3] Debug Status Port Control Register [4]	PORTSEL [3]				DEBUGPORT [4]			

1 L49P mask set devices only (32-bit register).

2 Non-L49P mask set devices only (16-bit register).

3 Not implemented on non-L49P mask set devices; offset is reserved.

4 Not implemented on L49P mask set devices; offset is reserved.

Table C-9. DMAMux Memory Map

Base Address: 0xFC08_4000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	17-261	Channel 0–3 Configuration Register	CHCONFIG0		CHCONFIG1		CHCONFIG2		CHCONFIG3	
0x0004	17-261	Channel 4–7 Configuration Register	CHCONFIG4		CHCONFIG5		CHCONFIG6		CHCONFIG7	
0x0008	17-261	Channel 8–11 Configuration Register	CHCONFIG8		CHCONFIG9		CHCONFIG10		CHCONFIG11	
0x000C	17-261	Channel 12–15 Configuration Register	CHCONFIG12		CHCONFIG13		CHCONFIG14		CHCONFIG15	

Table C-10. CRG Memory Map

Base Address: 0xFC08_8000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	4-54 4-54 4-55	CRG Synthesizer Divider Register CRG Reference Divider Register CRG Flags Register	SYNR		REFDV				CRGFLG	
0x0004	4-56 4-57 4-58 4-59	CRG Interrupt Enable Register CRG Clock Select Register CRG PLL Control Register CRG Stop/Doze Control Register	CRGINT		CLKSEL		PLLCTL		SDMCTL	
0x0008	4-60	CRG BDM Control Register	BDMCTL							

Table C-11. PIT Memory Map

Base Address: 0xFC08_C000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	25-552	PIT RTI Load Value Register	TLVAL0							
0x0004	25-552	PIT Timer Load Value Register 1	TLVAL1							
0x0008	25-552	PIT Timer Load Value Register 2	TLVAL2							
0x000C	25-552	PIT Timer Load Value Register 3	TLVAL3							
0x0010	25-552	PIT Timer Load Value Register 4	TLVAL4							
0x0014	25-552	PIT Timer Load Value Register 5	TLVAL5							
0x0018	25-552	PIT Timer Load Value Register 6	TLVAL6							
0x001C	25-552	PIT Timer Load Value Register 7	TLVAL7							
0x0020	25-552	PIT Timer Load Value Register 8	TLVAL8							
0x0024	25-552	PIT Timer Load Value Register 9	TLVAL9							
0x0028	25-552	PIT Timer Load Value Register 10	TLVAL10							
0x002C–0x007C	—	Reserved								
0x0080	25-553	PIT RTI Current Value Register	TVAL0							
0x0084	25-553	PIT Current Timer Value Register 1	TVAL1							
0x0088	25-553	PIT Timer Current Value Register 2	TVAL2							
0x008C	25-553	PIT Timer Current Value Register 3	TVAL3							
0x0090	25-553	PIT Timer Current Value Register 4	TVAL4							
0x0094	25-553	PIT Timer Current Value Register 5	TVAL5							
0x0098	25-553	PIT Timer Current Value Register 6	TVAL6							
0x009C	25-553	PIT Timer Current Value Register 7	TVAL7							
0x00A0	25-553	PIT Timer Current Value Register 8	TVAL8							
0x00A4	25-553	PIT Timer Current Value Register 9	TVAL9							
0x00A8	25-553	PIT Timer Current Value Register 10	TVAL10							
0x00AC–0x009C	—	Reserved								
0x0100	25-553	PIT Interrupt Flags Register	PITFLG							
0x0104	25-554	PIT Interrupt Enable Register	PITINTEN							
0x0108	25-555	PIT Interrupt/DMA Select Register	PITINTSEL							
0x010C	25-555	PIT Timer Enable Register	PITEN							
0x0110	25-556	PIT Control Register	PITCTRL							

Table C-12. VREG Memory Map

Base Address: 0xFC09_0000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	3-33 3-33 3-34 3-35	VREG High Temperature Control Register [1] VREG Control Register [2] VREG Autonomous Periodic Interrupt Control Register [3] VREG Autonomous Periodic Interrupt Trimming Register [3]	VREGHTCL [1]		VREGCTRL [2]		VREGAPICL [3]		VREGAPITR [3]	

1 On mask set L49P devices, this register is not implemented and VREGCTRL is at this offset.

2 On mask set L49P devices, this register is at offset 0x0000.

3 On mask set L49P devices, this register is not implemented and the offset is reserved.

Table C-13. FlexCAN Memory Map

Base Address FlexCAN A: 0xFC09_4000
Base Address FlexCAN B: 0xFC09_8000
Base Address FlexCAN C: 0xFC09_C000
Base Address FlexCAN D: 0xFC0A_0000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	23-498	Module Configuration Register	MCR							
0x0004	23-501	Control Register	CTRL							
0x0008	23-504	Timer Register	TIMER							
0x000C	—	Reserved								
0x0010	23-505	Rx Global Mask Register	RXGMASK							
0x0014	23-506	Rx Buffer 14 Mask Register	RX14MASK							
0x0018	23-506	Rx Buffer 15 Mask Register	RX15MASK							
0x001C	23-506	Error Counter Register	ECR							
0x0020	23-507	Error and Status Register	ESR							
0x0024	—	Reserved								
0x0028	23-510	Interrupt Mask Register	IMASK							
0x002C	—	Reserved								
0x0030	23-511	Interrupt Flag Register	IFLAG							
0x0034–0x007C	—	Reserved								
0x0080–0x027F	23-495	Message Buffers 0–31	MB0–31							

Table C-13a. FlexCAN MB*n* Memory Map Detail

MB*n* Offset																							
MB0	0x0080		MB4	0x00C0		MB8	0x0100		MB12	0x0140		MB16	0x0180		MB20	0x01C0		MB24	0x0200		MB28	0x0240
MB1	0x0090		MB5	0x00D0		MB9	0x0110		MB13	0x0150		MB17	0x0190		MB21	0x01D0		MB25	0x0210		MB29	0x0250
MB2	0x00A0		MB6	0x00E0		MB10	0x0120		MB14	0x0160		MB18	0x01A0		MB22	0x01E0		MB26	0x0220		MB30	0x0260
MB3	0x00B0		MB7	0x00F0		MB11	0x0130		MB15	0x0170		MB19	0x01B0		MB23	0x01F0		MB27	0x0230		MB31	0x0270

Table C-14. I²C Memory Map

Base Address: 0xFC0A_0000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	24-527 24-528 24-533 24-534	I²C Bus Address Register I²C Bus Frequency Divider Register I²C Bus Control Register I²C Bus Status Register	IBAD		IBFD		IBCR		IBSR	
0x0004	24-535	I²C Bus Data I/O Register	IBDR							

Table C-15. DSPI Memory Map

Base Address DSPI A: 0xFC0B_4000
Base Address DSPI B: 0xFC0B_8000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	22-453	DSPI Module Configuration Register	DSPI*x*_MCR							
0x0004	—	Reserved								
0x0008	22-456	DSPI Transfer Count Register	DSPI*x*_TCR							
0x000C	22-457	DSPI Clock and Transfer Attributes Register 0	DSPI*x*_CTAR0							
0x0010	22-457	DSPI Clock and Transfer Attributes Register 1	DSPI*x*_CTAR1							
0x0014	22-457	DSPI Clock and Transfer Attributes Register 2 [1]	DSPI*x*_CTAR2 [1]							
0x0018	22-457	DSPI Clock and Transfer Attributes Register 3 [1]	DSPI*x*_CTAR3 [1]							
0x001C	22-457	DSPI Clock and Transfer Attributes Register 4 [1]	DSPI*x*_CTAR4 [1]							
0x0020	22-457	DSPI Clock and Transfer Attributes Register 5 [1]	DSPI*x*_CTAR5 [1]							
0x0024–0x0028	—	Reserved								
0x002C	22-462	DSPI Status Register	DSPI*x*_SR							
0x0030	22-464	DSPI DMA/Interrupt Request Select and Enable Register	DSPI*x*_RSER							
0x0034	22-466	DSPI Push TX FIFO Register	DSPI*x*_PUSH							
0x0038	22-467	DSPI Pop RX FIFO Register	DSPI*x*_POPR							
0x003C	22-468	DSPI Transmit FIFO Register 0	DSPI*x*_TXFR0							
0x0040	22-468	DSPI Transmit FIFO Register 1 [1]	DSPI*x*_TXFR1 [1]							
0x0044	22-468	DSPI Transmit FIFO Register 2 [1]	DSPI*x*_TXFR2 [1]							
0x0048	22-468	DSPI Transmit FIFO Register 3 [1]	DSPI*x*_TXFR3 [1]							
0x004C–0x0078	—	Reserved								
0x007C	22-468	DSPI Receive FIFO Register 0	DSPI*x*_RXFR0							
0x0080	22-468	DSPI Receive FIFO Register 1	DSPI*x*_RXFR1							
0x0084	22-468	DSPI Receive FIFO Register 2	DSPI*x*_RXFR2							
0x0088	22-468	DSPI Receive FIFO Register 3	DSPI*x*_RXFR3							

[1] This register is not present on mask set L49P devices, and the offset is reserved.

Table C-16. eSCI Memory Map

Base Address eSCI A: 0xFC0C_4000
Base Address eSCI B: 0xFC0C_8000
Base Address eSCI C: 0xFC0C_C000
Base Address eSCI D: 0xFC0D_0000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	21-408 21-409 21-409 21-410	eSCI Baud Rate Register High eSCI Baud Rate Register Low eSCI Control Register 1 eSCI Control Register 2	ESCIBDH		ESCIBDL		ESCICR1		ESCICR2	
0x0004	21-410 21-410 21-414 21-414	eSCI Control Register 3 eSCI Control Register 4 [1] eSCI Data Register High eSCI Data Register Low	ESCICR3		ESCICR4 [1]		ESCIDRH		ESCIDRL	
0x0008	21-415 21-415 21-417 21-417	eSCI Status Register 1 eSCI Status Register 2 LIN Status Register 1 LIN Status Register 2	ESCISR1		ESCISR2		LINSTAT1		LINSTAT2	
0x000C	21-418 21-419 21-419	LIN Control Register 1 LIN Control Register 2 LIN Control Register 3	LINCTRL1		LINCTRL2		LINCTRL3			
0x0010	21-421	LIN TX Register	LINTX							
0x0014	21-423	LIN RX Register	LINRX							
0x0018	21-423 21-423	LIN CRC Polynomial Register 1 LIN CRC Polynomial Register 2	LINCRCP1		LINCRCP2					

[1] This register is not implemented on mask set L49P devices, and the offset is reserved.

Table C-17. eMIOS Memory Map

Base Address: 0xFC0D_C000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	20-354	Module Configuration Register	MCR							
0x0004	20-355	Global Flag Register	GFLAG							
0x0008	20-356	Output Update Disable Register	OUDIS							
0x000C	20-356	Disable Channel Register	UCDIS							
0x0010–0x001F	—	Reserved								
0x0020–0x021F	20-357	Unified Channels 0–15	UC0–15							

Table C-17a. eMIOS UC*n* Memory Map Detail

UC*n* Offset			
UC0	0x0020	UC8	0x0120
UC1	0x0040	UC9	0x0140
UC2	0x0060	UC10	0x0160
UC3	0x0080	UC11	0x0180
UC4	0x00A0	UC12	0x01A0
UC5	0x00C0	UC13	0x01C0
UC6	0x00E0	UC14	0x01E0
UC7	0x0100	UC15	0x0200

Register Offset	Page	Register Description
UC*n* Offset + 0x00	20-357	Channel A Data Register (UCA*n*)
UC*n* Offset + 0x04	20-358	Channel B Data Register (UCB*n*)
UC*n* Offset + 0x08	20-359	UC Counter Register (UCCNT*n*)
UC*n* Offset + 0x0C	20-359	Channel Control Register (UCCR*n*)
UC*n* Offset + 0x10	20-363	UC Status Register (UCSR*n*)
UC*n* Offset + 0x14–0x1F	—	Reserved

Table C-18. ATD Memory Map

Base Address ATD A: 0xFC0E_0000
Base Address ATD B: 0xFC0E_4000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	19-321 19-322 19-323 19-324	ATD Trigger Control Register ATD External Trigger Channel Register ATD Prescaler Register ATD Operating Mode Register	ATDTRIGCTL		ATDETRIGCH		ATDPRE		ATDMODE	
0x0004–0x000B	—	Reserved								
0x000C	19-325 19-326	ATD Interrupt Register ATD Flag Register					ATDINT		ATDFLAG	
0x0010	19-328	ATD Command Word Register	ATDCW							
0x0014	19-330	ATD Result Register	ATDRR							

Table C-19. PIM Memory Map — Port Control Registers

Base Address Port A: 0xFC0E_8000
Base Address Port B: 0xFC0E_8040
Base Address Port C: 0xFC0E_8080
Base Address Port D: 0xFC0E_80C0

Base Address Port E: 0xFC0E_8100
Base Address Port F: 0xFC0E_8140
Base Address Port G: 0xFC0E_8180
Base Address Port H: 0xFC0E_81C0
Base Address Port I: [1] 0xFC0E_8200

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	18-286	Port *x* Pin 0 Configuration Port *x* Pin 1 Configuration	CONFIG0_*x*				CONFIG1_*x*			
0x0004	18-286	Port *x* Pin 2 Configuration [2] Port *x* Pin 3 Configuration	CONFIG2_*x* [2]				CONFIG3_*x*			
0x0008	18-286	Port *x* Pin 4 Configuration Port *x* Pin 5 Configuration	CONFIG4_*x*				CONFIG5_*x*			
0x000C	18-286	Port *x* Pin 6 Configuration Port *x* Pin 7 Configuration	CONFIG6_*x*				CONFIG7_*x*			
0x0010	18-286	Port *x* Pin 8 Configuration Port *x* Pin 9 Configuration	CONFIG8_*x*				CONFIG9_*x*			
0x0014	18-286	Port *x* Pin 10 Configuration Port *x* Pin 11 Configuration	CONFIG10_*x*				CONFIG11_*x*			
0x0018	18-286	Port *x* Pin 12 Configuration Port *x* Pin 13 Configuration	CONFIG12_*x*				CONFIG13_*x*			
0x001C	18-286	Port *x* Pin 14 Configuration Port *x* Pin 15 Configuration	CONFIG14_*x*				CONFIG15_*x*			
0x0020	18-287	Port *x* Interrupt Flag	PORTIFR_*x*							
0x0024	18-288 18-288	Port *x* Data Port *x* Input	PORTDATA_*x*				PORTIR_*x*			
0x0028	18-289	Port *x* Pin 0 Data Port *x* Pin 1 Data Port *x* Pin 2 Data [2] Port *x* Pin 3 Data	PINDATA0_*x*		PINDATA1_*x*		PINDATA2_*x* [2]		PINDATA3_*x*	
0x002C	18-289	Port *x* Pin 4 Data Port *x* Pin 5 Data Port *x* Pin 6 Data Port *x* Pin 7 Data	PINDATA4_*x*		PINDATA5_*x*		PINDATA6_*x*		PINDATA7_*x*	
0x0030	18-289	Port *x* Pin 8 Data Port *x* Pin 9 Data Port *x* Pin 10 Data Port *x* Pin 11 Data	PINDATA8_*x*		PINDATA9_*x*		PINDATA10_*x*		PINDATA11_*x*	
0x0034	18-289	Port *x* Pin 12 Data Port *x* Pin 13 Data Port *x* Pin 14 Data Port *x* Pin 15 Data	PINDATA12_*x*		PINDATA13_*x*		PINDATA14_*x*		PINDATA15_*x*	

1 Port I is available only on mask set L38Y devices; corresponding offsets must be treated as reserved on all other devices.
2 On L49P mask set devices the PD2 function is not available, and this offset is reserved in Port D.

Table C-20. PIM Memory Map — Global Control Registers

Base Address: 0xFC0E_83C0 [1]

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	18-289 18-290	PIM Global Interrupt Status [1] PIM Global Configuration [1]	GLBINT [1]				PIMCONFIG [1]			
0x0004	18-291 18-292	TDI Pin Configuration [1] TDO Pin Configuration [1]	CONFIG_TDI [1]				CONFIG_TDO [1]			
0x0008	18-293 18-294	TMS Pin Configuration [1] TCK Pin Configuration [1]	CONFIG_TMS [1]				CONFIG_TCK [1]			
0x000C	18-294	$\overline{TA}$ / $\overline{AS}$ Pin Configuration [1]	CONFIG_TA [1]							
0x0010–0x001F	—	Reserved								
0x0020	18-295	Port A/B 32-bit Input Register [2]	PORT32IR_AB [2]							
0x0024	18-295	Port C/D 32-bit Input Register [2]	PORT32IR_CD [2]							
0x0028	18-295	Port E/F 32-bit Input Register [2]	PORT32IR_EF [2]							
0x002C	18-295	Port G/H 32-bit Input Register [2]	PORT32IR_GH [2]							
0x0030	18-295	Port B/C 32-bit Input Register [2]	PORT32IR_BC [2]							
0x0034	18-295	Port D/E 32-bit Input Register [2]	PORT32IR_DE [2]							
0x0038	18-295	Port F/G 32-bit Input Register [2]	PORT32IR_FG [2]							
0x003C	18-295	Port H/I 32-bit Input Register [2]	PORT32IR_HI [2]							

1 On mask set L49P devices, these registers are not implemented and the address range is reserved.

2 Mask set L49P, L47W and L61W devices do not implement this register, and the address range is reserved.

Table C-21. CFM Memory Map

Base Address: 0xFC0F_0000

Offset	Page	Register Description	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x0000	15-214 15-216	CFM Module Configuration Register CFM Clock Divider Register	CFMMCR				CFMCLKD			
0x0004	—	Reserved								
0x0008	15-217	CFM Security Register	CFMSEC							
0x000C	—	Reserved								
0x0010	15-218	CFM Program Flash Protection Register	CFMPROT							
0x0014	15-221	CFM Program Flash Supervisor Access Register	CFMSACC							
0x0018	15-222	CFM Program Flash Data Access Register	CFMDACC							
0x001C	—	Reserved								
0x0020	15-223	CFM User Status Register	CFMUSTAT							
0x0024	15-225	CFM Command Register	CFMCMD							
0x0028–0x002C	—	Reserved								
0x0030	15-225	CFM Data Register [1]	CFMDATA [1]							
0x0034–0x0040	—	Reserved								
0x0044	15-220 15-221 15-223	CFM Data Flash Protection Register CFM Data Flash Supervisor Access Register CFM Data Access Register	CFMDFPROT		CFMDFSACC		CFMDFDACC			
0x0048	15-227	CFM Clock Select Register [2]					CFMCLKSEL [2]			

[1] Mask set L49P and L47W devices do not implement this register, and the offset is reserved.

[2] Mask set L49P devices do not implement this register, and the offset is reserved.

Appendix D
Mask Set Differences Summary

D.1 Differences Between L49P, L47W, L61W and L38Y Mask Set Devices

This appendix summarizes all differences between MAC71*x*1, and MAC71*x*2, MAC71*x*6 mask set devices listed in Table D-1. Table D-2 below summarizes the functional differences between mask sets and the impact of those changes on software. Mask sets are listed in the table in chronological order from left to right, and thus the description column of the table refers to functional changes implemented on later mask sets. In order to port code developed for an earlier mask set to a later device, the changes indicated in the impact column must be made. In order to develop new code that is compatible across all mask sets, conditional logic (either run-time, using the MCM registers as described in Section 11.4.1, “Using The PCT And REV Registers,” on page 11-138, or compile-time, using compiler directives) must be provided to properly handle the presence or absence of various functionality.

Table D-1. MAC7100 Family Mask Set to Part Number Correspondence

Mask Set	Part Number(s)
0L49P	Engineering samples: PAC7101, PAC7111, PAC7121, PAC7131, PAC7141
1L49P	Limited production, pre-qualification: PAC7101, PAC7111, PAC7121, PAC7131, PAC7141
0L47W	Fully-qualified, production: PAC7101, PAC7111, PAC7121, PAC7131, PAC7141
1L47W	Fully-qualified, production: MAC7101, MAC7111, MAC7121, MAC7131, MAC7141
0L61W	Engineering samples: PAC7112, PAC7122, PAC7142
0L38Y	Engineering samples: PAC7106, PAC7116, PAC7126, PAC7136

Table D-2. MAC7100 Microcontroller Family Mask Set Differences Summary

Change	Affected Module(s)	Page Num.	Mask Set				Description	Impact
			0L49P 1L49P	0L47W 1L47W	0L91W	0L38Y		
Abort if reserved memory map areas are accessed	All IPS peripherals	—	No	Yes	Yes	Yes	IPS peripherals (VREG, CRG, CFM, DMA Mux, PIM, ATD, eMIOS, eSCI, DSPI, FlexCAN, I^2C, PIT, SSM) generate aborts when unassigned areas of the memory map are accessed. Global enable of aborts is controlled via the SSM (refer to Section 26.4.1.4 on page 26-568).	The change is transparent to existing code, as these areas of memory should not be used and because the abort is not enabled following reset. For enhanced protection against errant code, Initialization routines should be updated to enable the abort, and the abort service routine should be updated to handle the new abort sources.
Data Abort exception for unimplemented peripherals	All IPS peripherals	—	No	No	Yes	Yes	A data abort exception is taken if any access to an unimplemented peripheral is attempted (for example, to eSCI_C on a 2-eSCI device). This includes any 16 Kbyte slot between $FC08 0000 and $FC0F 0000 marked as “Reserved” in Table 8-10 on page 8-99 for a particular device. Refer to Section 26.4.1.4 on page 26-568.	The change is transparent to existing code, as these areas of memory should not be used and because the abort is not enabled following reset. For enhanced protection against errant code, Initialization routines should be updated to enable the abort, and the abort service routine should be updated to handle the new abort sources.
Pull-down added to TEST pin	Device	2-11	No	Yes	Yes	Yes	The TEST pad now has internal pull-down functionality. Refer to Section 2.1.1.9 on page 2-12 for details.	No changes necessary. Circuit boards may be designed to take advantage of this feature.
Internal low power oscillator for fast wake-up added	VREG	3-29	No	Yes	Yes	Yes	The VREG includes an internal 10 kHz oscillator for fast system wake-up. This oscillator is less accurate than the RTI, but requires less power. Refer to Section 3.6.6 on page 3-37 for details.	Minor code modification is required, as the VREG memory map has changed (refer to Table 3-2 on page 3-32). To use the fast wake-up, the VREG must be configured to enable the API oscillator and set the wake-up period (from 0.5 mS to 750 mS).
Internal high temperature sensor added	VREG ATD	3-29 19-317	No	Yes	Yes	Yes	The VREG includes a temperature sensor that can be read via ATD A channel 0. Refer to Section 3.5.1.1 on page 3-33 for details.	VREG configuration code must be changed due to changes in the memory map and register access restrictions (refer to Section 3.5 on page 3-32). In order to utilize the temperature sensor, new VREG and ATD driver code must also be developed; refer to Section 19.7.11 on page 19-348 for details.

Table D-2. MAC7100 Microcontroller Family Mask Set Differences Summary (continued)

Change	Affected Module(s)	Page Num.	Mask Set				Description	Impact
			0L49P 1L49P	0L47W 1L47W	0L61W	0L38Y		
STOP entry status added	CRG	4-45	No	Yes	Yes	Yes	CRGFLG[STPEF] bit added to indicate that STOP mode was entered. STPEF is cleared when the SDMCTL[STOP] bit is written (it may be cleared manually). It is set when the STOP mode is entered, but is not set if a wake-up interrupt occurs before the STOP procedure is completed.	No code modification is required unless it is desired to detect that STOP mode was entered. Refer to Section 4.3.5.3 on page 4-55 for more details.
CRG fast wake-up added	CRG	4-45	No	Yes	Yes	Yes	The CRG can wake-up quickly from stop mode and provide a slow clock to the system. Refer to Section 4.3.5.6 on page 4-58 for the description of the FSTWKP bit.	No code modification is required unless this feature is utilized. To reduce average power consumption, the wake-up service routine may determine if a full wake-up is required or if stop mode can be re-entered prior to restarting the oscillator (refer to Section 4.3.6.10.7 on page 4-79).
New ARM7 core version	ARM7	9-101	No	No	Yes	Yes	The ARM7 TDMI-S core has been upgraded from version rev4p2.04 to version rev4p3.04. Refer to Section 9.1 on page 9-101.	Please refer to the *ARM7TDMI-S Errata List* from ARM for more information.
FlexCAN wake-up handled by INTC	INTC FlexCAN SSM	10-103 23-491 26-563	No	Yes	Yes	Yes	Improved clarity of the FlexCAN wake-up operation and more direct enabling and disabling of the wake-up function via the INTC (versus the L49P method, see Section 26.4.1.3 on page 26-567). For each FlexCAN channel, the wake-up request is combined with the previously implemented error interrupt (refer to Table 6-2, Section 23.5.2.1 on page 23-498 and Section 23.6.8.4 on page 23-521).	INTC configuration must set FlexCAN priority levels high enough to pass through the INTC (refer to Section 10.6.2 on page 10-117). The priority level assigned to the CAN must also be higher than the MCM MWCR[PRILVL] bits (refer to Section 11.3.1.7 on page 11-128) in order for the wake-up to be recognized. The FlexCAN interrupt service routine must be updated to handle the wake-up interrupt source.
RTI wake-up handled by INTC	INTC PIT SSM	10-103 25-549 26-563	No	Yes	Yes	Yes	Improved RTI wake-up operation with direct enabling / disabling of the wake-up function via the INTC (versus L49P method, Section 26.4.1.3 on page 26-567). The RTI wake-up request is combined with the previously implemented PIT Timer 4 interrupt (refer to Table 6-2 on page 6-85and Section 25.5.4.2.1 on page 25-560).	Configuration code for PIT Timer 4 interrupts and/or wake-up may need to be modified to enable or disable the RTI and Timer 4 interrupts, as appropriate. The Timer 4 interrupt service routine must be updated to also handle the RTI.

Table D-2. MAC7100 Microcontroller Family Mask Set Differences Summary (continued)

Change	Affected Module(s)	Page Num.	Mask Set				Description	Impact
			0L49P 1L49P	0L47W 1L47W	0L61W	0L38Y		
SSM WAKEUP register removed	INTC FlexCAN PIT SSM	10-103 23-491 25-549 26-563	No	Yes	Yes	Yes	As support for RTI and FlexCAN wake-ups has been moved to the INTC, the SSM WAKEUP register is no longer required. Refer to Section 26.4.1.3 on page 26-567.	Configuration and interrupt service routine code must be modified, as described for the FlexCAN and RTI wake-up changes elsewhere in this table.
eDMA functionality added	eDMA	12-141	No	Yes	Yes	Yes	The eDMA controller has these new features: • Round-robin arbitration (Section 12.3.1.1) • Channel preemption (Section 12.3.1.15) • Channel linking (Section 12.3.1.16.8) • Scatter-gather (Section 12.3.1.16.8)	Added features provide enhanced functionality, but modifications to existing code are not required unless the new DMA features are to be used.
CLKOUT signal control added	EIM PIM	13-181 18-271	No	Yes	Yes	Yes	Controls for the CLKOUT signal added: • Enable/disable (GPI mode available on PD2) • Drive strength • Pull-up/down • Open drain	Reduces EMI radiation for applications that don't use CLKOUT, and makes an additional general-purpose input available. PIM driver code must to be modified to use this feature (refer to Section 18.7.3 on page 18-313).
EIM clock control added	EIM PIM	13-181 18-271	No	Yes	No [1]	Yes	In modes where the external bus is not available (for example, secured single-chip mode) or when the EIM is manually disabled, the EIM module clock may be gated off to reduce power consumption.	The EIM clock is automatically gated off in all single-chip modes. Power consumption may be reduced in expanded modes when the EIM is not in use by clearing the PIMCONFIG[EIMCLKEN] bit (refer to Section 18.5.1.7 on page 18-290).
Abort if the EIM doesn't match any chip selects	EIM	13-181	No	Yes	No [1]	Yes	If the external bus is enabled and configured to use chip selects, an Abort is generated if a bus cycle address decodes to an external area but no $\overline{CSn}$ matches the address. Most likely occurs only following reset and prior to configuration of $\overline{CS0}$ (refer to Section 13.6.1 on page 13-189).	The change is transparent to existing code, as the chip selects should be used correctly by debugged code. Provides improved protection by preventing errors in chip select configuration. The abort service routine should be updated to handle the new abort source.
Programmable pull-ups/downs in peripheral mode added	EIM PIM ATD eMIOS eSCI DSPI FlexCAN I^2C	13-181 18-271 19-317 20-351 21-405 22-449 23-491 24-525	No	Yes	Yes[1]	Yes	Programmable internal pull-up/down on pins are available when the pin is in peripheral mode (refer to Section 18.7.1 on page 18-303) in addition to GPIO mode, as previously implemented (refer to Section 18.7.2 on page 18-304).	This change is backward compatible with existing code. To use the new functionality, the PIM must be configured appropriately. If an application switches between the GPIO and peripheral functions for a pin, it is important that any pull used in the GPIO mode is also used in the peripheral mode, or the pull should be disabled when switch is performed.

Table D-2. MAC7100 Microcontroller Family Mask Set Differences Summary (continued)

Change	Affected Module(s)	Page Num.	Mask Set				Description	Impact
			0L49P 1L49P	0L47W 1L47W	0L61W	0L38Y		
Multiplexed $\overline{AS}$ signal added	EIM PIM	13-181 18-271	No	Yes	No [1]	Yes	The CONFIG_TA register has been added to the PIM to control multiplexing of the $\overline{TA}$ / $\overline{AS}$ signals on the pin originally used only for $\overline{TA}$ (refer to Section 18.5.1.12 on page 18-294).	This change is backward compatible with existing code, as the pin is configured as the $\overline{TA}$ input by default. Configuration routines must be modified in order to use the $\overline{AS}$ output function.
Single cycle Flash access for low speed operation	CFM	15-209	No	Yes	Yes	Yes	Interleaving of the Flash is disabled when the CPU is running at low speed to allow the core to perform single-cycle instruction fetches. Refer to Section 15.3.1.14 on page 15-227.	This factory programmable option for devices operated at ≤ 25 MHz (estimated) reduces clocks per instruction from an average of 1.2 to exactly 1.
Flash programming integrity check support added	CFM	15-209	No	No	Yes	Yes	Data Signature command and margin-shifting added to provide enhanced ability to verify that the Flash programming is correct. Refer to Section 15.3.1.12 on page 15-225.	This change is backward compatible with existing code. For applications that require more robust Flash content validation, code must be added. Refer to Section 15.5.1 on page 15-248.
PIM global interrupt status register added	PIM SSM	18-271 26-563	No	Yes	Yes	Yes	The PIM GLBLINT register contains eight port interrupt flags in order to determine which ports have pending interrupts with a single bus access (refer to Section 18.5.1.6 on page 18-289).	This change is backward compatible with existing code. PIM interrupt service routine code must be modified to utilize this enhancement (refer to Section 18.6.3.1 on page 18-299).
Port C/H switch moved from the SSM to the PIM	PIM SSM	18-271 26-563	No	Yes	No [2]	Yes	The Port C / H address map switch is moved from the SSM PORTSEL register to the PIM PORTHSEL bit in the PIMCONFIG register (refer to Section 18.5.1.7 on page 18-290).	For those applications using the port C/H switch, configuration code must be modified to use the new PIM register rather than SSM register previously used.
32-bit GPI mode via register concatenation	PIM	18-271	No	No	No	Yes	In order to enhance performance of functions such as data Flash boot mode, where large data sets are loaded via port registers, a 32-bit GPI-only feature is available (refer to Section 18.5.1.13 on page 18-295).	This change is backward-compatible, as no existing registers are modified. Eight new 32-bit read-only registers concatenate adjacent ports (refer to Table 18-3 on page 18-276).
Add Port I	PIM	18-271	No	No	No	Yes	Added Port I with full GPIO capability on all pins (refer to Table 18-2 on page 18-274 and Table 18-4 on 18-285).	No change to existing code required. To use Port I for GPIO, code must be modified to do so, including modification of exception routines to detect external interrupts on Port I.
Port B / DSPI_B chip select bond-out change	PIM DSPI	18-271 22-449	No	Yes	Yes	Yes	PB11 / PCS5_B / PCSS_B is bonded out on pin 57 of MAC7121 devices, in place of PB10 / PCS2_B. Refer to “Device Pin Assignments” in MAC7100EC and Table 2-1 on page 2-13.	Allows glitch-free selection of SPI slaves with an external demux. PIM Port B configuration and data written to DSPIB_PUSHR[PCS*n*] bits must be modified as appropriate.

Table D-2. MAC7100 Microcontroller Family Mask Set Differences Summary (continued)

Change	Affected Module(s)	Page Num.	Mask Set				Description	Impact
			0L49P 1L49P	0L47W 1L47W	0L91W	0L38Y		
eMIOS register A and B operation in PWFM mode changed	eMIOS	20-351	No	Yes	Yes	Yes	Overall prescaler values do not affect the values loaded into UC registers A and B in OPWFM mode. Refer to Section 20.6.7.12 on page 20-380 for a detailed explanation.	For applications using eMIOS channels in OPWFM mode and an overall prescaler value greater than 1, the values loaded into registers A and B must be modified appropriately.
eMIOS double-buffered modes added	eMIOS	20-351	No	No	No	Yes	In order to provide smooth waveform generation while allowing A and B registers to be changed on the fly, double-buffered modes are provided. Refer to Section 20.6.7.15 on page 20-386 through Section 20.6.7.18 on page 20-398 for a detailed descriptions.	Applications using the MC, OPWFM, OPWM and OPWMC modes may be changed to the buffered counterparts in order to simplify real-time driver routines.
eSCI error detection improved	eSCI	21-405	No	No	Yes	Yes	Fast detection of bit errors, on bit boundaries rather than on byte boundaries, added. In case of collisions the eSCI will free up the bus much faster. Refer FBR description in Section 21.5.1.2 on page 21-409.	This change is backward compatible with existing code. eSCI configuration and error routines may be updated to utilize this feature. Since collisions should never happen in normal operation, this will have no impact in a normally functioning system. When errors are detected, eSCI timing will change, which could have a small impact on the timing of the system.
eSCI interrupt handling improved	eSCI	21-405	No	Yes	Yes	Yes	Separate interrupt and mask bits provided for • Overrun (OR), • Noise Flag (NF), • Framing Error (FE) and • Parity Flag (PF). Refer to Section 21.5.1.4 on page 21-415.	No code modification is required to support this change, as the new interrupts are masked by default and the RDRF interrupt can be used to check these flags. Modifying eSCI drivers to utilize the flags will enhance performance, particularly in LIN slave mode.
eSCI flag latching added	eSCI	21-405	No	Yes	Yes	Yes	The SCISR1[TDRE, TC, RDRF, IDLE, OR, NF, FE and PF] bits are latching, indicating that the associated event has occurred at least once since the last time the flag was cleared.	eSCI interrupt service routine code must explicitly clear flags in order to detect new events (refer to Section 21.5.1.4 on page 21-415).
eSCI LIN slave timeout changed	eSCI	21-405	No	Yes	Yes	Yes	The timeout counter has been changed to begin counting at the start of the break-character after the initial LIN header is sent.	No code modifications are required, this is an implementation improvement only (refer to Section 21.7.2 on page 21-442).
Support for double baud rate DSPI operation	DSPI	22-449	No	No	Yes	Yes	DSPI*x*_CTAR*n*[DBR] control bit added to modify the definition of the DSPI*x*_CTAR*n*[PBR, BR] fields, offering doubled SCK baud rates in Master Mode. Refer to Table 22-5 on page 22-458.	This change is backward compatible, as the new divide values are only available if the DRB bit is set. Existing DSPI drivers are not impacted. Refer to Section 22.6.4.1 on page 22-475.

Table D-2. MAC7100 Microcontroller Family Mask Set Differences Summary (continued)

Change	Affected Module(s)	Page Num.	Mask Set				Description	Impact
			0L49P 1L49P	0L47W 1L47W	0L61W	0L38Y		
DSPI Clock and Transfer Attributes Registers added	DSPI	22-449	No	Yes	Yes	Yes	The number of CTARs is increased to 6, providing enhanced control of DSPI Chip Selects. Refer to Table 22-2 on page 22-453.	This change is backward compatible with existing code. Driver code may be changed to use the new CTARs to improve DSPI performance. Refer to CTAS definition in Table 22-8 on page 22-466.
DSPI TX FIFO depth increased	DSPI	22-449	No	Yes	Yes	Yes	FIFO depth on the DSPI TX side increased to match the depth of the RX FIFO. Refer to Table 22-2 on page 22-453 and Section 22.6.3.4 on page 22-473.	This change is backward compatible with existing code. The added TX FIFOs improve performance of high speed DSPI transfers by requiring less frequent servicing by the CPU or eDMA.
Add 4 additional chip selects	DSPI	22-449	No	No	No	Yes	Added 4 chip selects for each DSPI. Refer to Section 22.4.2 on page 22-452, Figure 22-3 on page 22-454 and Figure 22-8 on page 22-466.	No change necessary to existing code. If the user wishes to use the additional chip selects, the code must be modified to do so.
PIT timers freeze in debug mode	PIT	25-549	No	Yes	Yes	Yes	All PIT timers freeze when debug mode is entered and resume counting on debug exit. Refer to Section 25.5.2.2 on page 25-557.	Simplifies debug of code using the PIT.
Hardware status fields added to the SSM STATUS register	SSM	26-563	No	No	Yes[1]	Yes	The SSM STATUS register contains new status information related to hardware configuration (see Section 26.4.1.1 on page 26-564): • Nexus Status (bits 12:11) • EIM Auto Ack (bit 10) • EIM Port Size (bits 9:8)	If existing code reads the STATUS register using byte accesses (STATUS[7:0]) or software masks out bits 15:8, then no changes are necessary. **Note:** 0L61W devices implement only Nexus Status (bits 12:11). Bits 10:8 are not used, and read as 0.
$\overline{TA}$ / $\overline{AS}$ and JTAG pins control added	EIM PIM Debug	13-181 18-271 A-577	No	Yes	Yes[1]	Yes	Additional control for the $\overline{TA}$ / $\overline{AS}$ and JTAG (TDI, TDO, TCLK and TM) pins is provided via the PIM (refer to Section 18.7.4 on page 18-314 and Section 18.7.5 on page 18-314).	No code changes are required unless it is desired to use the new pin functions (programmable drive strength and pull-up/down).

Table D-2. MAC7100 Microcontroller Family Mask Set Differences Summary (continued)

Change	Affected Module(s)	Page Num.	Mask Set				Description	Impact
			0L49P 1L49P	0L47W 1L47W	0M16Y	0L38Y		
Debug information on Port F added	SSM Debug	26-563 A-577	No	Yes	Yes	Yes	The DEBUGPORT register has been added to the SSM to enable delivery of additional debug information on Port F (refer to Section 26.4.1.6 on page 26-570 and Section A.1 on page A-577).	This change is backward compatible with existing code. SSM configuration code must be modified to enable enhanced debug support. If enabled, Port F and/or eMIOS signals are not available.
Nexus watchpoints added	Debug	A-577 B-596	No	Yes	Yes	Yes	Two additional watchpoints allow Nexus to turn on/off tracing at certain points in the code. Refer to Section B.3.3.8 on page B-596.	No end-user code changes are required, although debug tool chain software may need to be updated to utilize the additional watchpoints.

[1] This device does not implement the EIM, so references to EIM features, registers and signals such as $\overline{AS}$ / $\overline{TA}$, should be ignored.

[2] This device does not implement Port H. All references to Port H features, registers and signals should be ignored.

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Technical Information Center
3-20-1, Minami-Azabu, Minato-ku
Tokyo 106-0047, Japan
0120 191014 or +81 3 3440 3569
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 26668334
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

MAC7100RM
Rev. 1.0
10/2004