

# 56F8300

Peripheral User Manual

*Also Supports 56F8100 Device Family*

**56F8300**  
**16-bit Hybrid Controllers**

MC56F8300UM  
Rev. 10  
10/2007

[freescale.com](http://freescale.com)



---

**This manual is one of a set of three documents. For complete product information, it is necessary to have all three documents. They are: DSP56800E Reference Manual, MC56F8300 Peripheral User Manual, and Device Technical Data Sheet.**

Order this document by **MC56F8300UM/D - Rev. 10**

**Revision History:**

See each chapter for changes

# TABLE OF CONTENTS

## Preface

Audience . . . . .	xxi
Manual Organization . . . . .	xxi
Suggested Reading . . . . .	xxiii
Manual Conventions . . . . .	xxiv

## Chapter 1 Overview

1.1	Introduction to the 56800E Core. . . . .	1-3
1.1.1	56800E Core Enhancements . . . . .	1-3
1.1.2	56800E Core. . . . .	1-4
1.1.3	System Bus Controller . . . . .	1-9
1.1.4	Operation Method . . . . .	1-10
1.2	Introduction to 56F8300/56F8100 Devices. . . . .	1-10
1.2.1	Applications. . . . .	1-10
1.3	Features . . . . .	1-11
1.3.1	System Architecture and Peripheral Interface. . . . .	1-11
1.3.2	IPBus Bridge (IPBB) . . . . .	1-12
1.3.3	Peripheral Interrupts/Interrupt Controller Module . . . . .	1-14
1.3.4	Peripheral Features. . . . .	1-15
1.4	Energy Information. . . . .	1-21

## Chapter 2 Analog-to-Digital Converter (ADC)

2.1	Introduction. . . . .	2-3
2.2	Features . . . . .	2-3
2.3	Block Diagram . . . . .	2-4
2.4	Functional Description . . . . .	2-5
2.5	Input MUX Function . . . . .	2-7
2.6	ADC Sample Conversion Operation Modes. . . . .	2-8
2.6.1	Normal Operating Mode . . . . .	2-9
2.7	ADC Data Processing. . . . .	2-11
2.8	Sequential Vs. Simultaneous Sampling . . . . .	2-12
2.9	Scan Sequencing . . . . .	2-12
2.9.1	Low Power Operating Mode . . . . .	2-13
2.9.2	ADC STOP Operating Mode. . . . .	2-16

2.10	Calibration	2-16
2.10.1	Calibration Overview	2-16
2.10.2	Calibration Correction Factors	2-18
2.10.3	Calibration Procedure	2-19
2.11	Pin Descriptions	2-24
2.11.1	AN0-AN7 — Analog Input Pins	2-24
2.11.2	Voltage Reference Pins—( $V_{REFH}$ , $V_{REFP}$ , $V_{REFMID}$ , $V_{REFN}$ , and $V_{REFLO}$ )	2-25
2.12	Register Definitions	2-27
2.12.1	ADC Control Register 1 (ADCTL1)	2-29
2.12.2	ADC Control Register 2 (ADCTL2)	2-33
2.12.3	ADC Zero Crossing Control Register (ADZCC)	2-34
2.12.4	ADC Channel List Registers (ADLST1 and ADLST2)	2-35
2.12.5	ADC Sample Disable Register (ADSDIS)	2-36
2.12.6	ADC Status Register (ADSTAT)	2-37
2.12.7	ADC Limit Status Register (ADLSTAT)	2-39
2.12.8	ADC Zero Crossing Status Register (ADZCSTAT)	2-39
2.12.9	ADC Result Registers (ADRSLT0–7)	2-40
2.12.10	ADC Low and High Limit Registers (ADHLMT0-7) and (ADLLMT0-7)	2-41
2.12.11	ADC Offset Registers (ADOFS0–7)	2-42
2.12.12	ADC Power Control Register (ADCPOWER)	2-43
2.12.13	ADC Calibration Register (ADC_CAL)	2-46
2.13	Clocks	2-47
2.13.1	Clock Operation Description	2-47
2.14	Interrupts	2-48

## Chapter 3 Computer Operating Properly (COP)

3.1	Introduction	3-3
3.2	Features	3-3
3.3	Block Diagram	3-3
3.4	Functional Description	3-3
3.5	Register Definitions	3-4
3.5.1	Control Register (COPCTL)	3-5
3.5.2	Timeout Register (COPTO)	3-6
3.5.3	Count Register (COPCTR)	3-7
3.6	Timeout Specifications	3-7
3.7	COP After Reset	3-8
3.8	Clocks	3-8
3.9	Interrupts	3-8

3.10	Resets	3-8
3.11	Wait Mode Operation	3-8
3.12	Stop Mode Operation	3-8
3.13	Debug Mode Operation	3-8

## Chapter 4 External Memory Interface (EMI)

4.1	Introduction	4-3
4.2	Features	4-3
4.3	Functional Description	4-3
4.3.1	Core Interface Detail	4-4
4.4	Block Diagram	4-4
4.5	Register Definitions	4-5
4.6	Register Descriptions	4-7
4.6.1	Chip Select Base Address Registers 0–3 ( $\overline{\text{CSBAR0}}$ – $\overline{\text{CSBAR3}}$ )	4-7
4.6.2	Chip Select Option Registers 0–3 (CSOR0–CSOR3)	4-8
4.6.3	Chip Select Timing Control Registers 0–3 (CSTC0–CSTC3)	4-10
4.6.4	Bus Control Register (BCR)	4-12
4.7	Timing Specifications	4-14
4.7.1	Read Timing	4-14
4.7.2	Write Timing	4-18
4.8	Clocks	4-24
4.9	Interrupts	4-24
4.10	Resets	4-24

## Chapter 5 On-Chip Clock Synthesis (OCCS)

5.1	Introduction	5-4
5.2	Features	5-4
5.3	Block Diagram	5-4
5.4	Functional Description	5-6
5.5	Crystal Oscillator	5-10
5.6	Relaxation Oscillator	5-10
5.6.1	Trimming Frequency on the Internal Relaxation Oscillator	5-10
5.6.2	Switching Clock Sources	5-11
5.7	Phase Locked Loop	5-12
5.7.1	PLL Recommended Range of Operation	5-12
5.7.2	PLL Lock Time Specification	5-17
5.8	PLL Frequency Lock Detector Block	5-18
5.9	Loss of Reference Clock Detector	5-19

5.10	Operating Modes	5-19
5.10.1	Crystal Oscillator	5-20
5.10.2	Ceramic Resonator	5-20
5.10.3	External Clock Source	5-21
5.10.4	Internal Clock Source	5-21
5.11	Register Definitions	5-22
5.11.1	PLL Control Register (PLLCR)	5-23
5.11.2	PLL Divide-By Register (PLLDB)	5-26
5.11.3	PLL Status Register (PLLSR)	5-27
5.11.4	Shutdown Register (SHUTDOWN)	5-29
5.11.5	Oscillator Control Register w/o Relaxation Oscillator (OSCTL)	5-30
5.11.6	Oscillator Control Register w/ Relaxation Oscillator (OSCTL)	5-31
5.12	Interrupts	5-32

## Chapter 6 Flash Memory (FM)

6.1	Introduction	6-3
6.2	Features	6-3
6.3	Block Diagram	6-4
6.4	Memory Map	6-6
6.5	Functional Description	6-9
6.5.1	Read Operation	6-9
6.5.2	Write Operation	6-9
6.5.3	Program and Erase Operation	6-9
6.5.4	Flash Security Operation	6-15
6.5.5	Flash Protection Operation	6-17
6.6	Pin Definitions	6-17
6.7	Unbanked Register Definition	6-17
6.7.1	Clock Divider Register (FMCLKD)	6-18
6.7.2	Configuration Register (FMCR)	6-19
6.7.3	Security Registers (FMSECH and FMSECL)	6-21
6.7.4	Flash Optional Data Registers (FMOPTn)	6-23
6.8	Banked Register Definition	6-24
6.8.1	Protection Register (FMPROT)	6-25
6.8.2	Protection Boot Register (FMPROTB)	6-27
6.8.3	User Status Register (FMUSTAT)	6-28
6.8.4	Command Buffer Register (FMCMD)	6-30
6.9	Interrupts	6-31
6.9.1	Interrupt Operation Description	6-31
6.10	Resets	6-32

## Chapter 7 FlexCAN (FC)

7.1	Introduction	7-3
7.2	Features	7-3
7.3	Block Diagram	7-4
7.4	Typical CAN System Diagram	7-4
7.5	Message Buffers	7-6
7.5.1	Message Buffer Structure	7-6
7.6	Functional Overview	7-9
7.6.1	Transmit Process	7-9
7.6.2	Receive Process	7-10
7.6.3	Message Buffer Handling	7-11
7.6.4	Lock/Release/Busy Mechanism and SMB Usage	7-13
7.6.5	Remote Frames	7-13
7.6.6	Overload Frames	7-14
7.6.7	Time Stamp	7-14
7.6.8	Listen-Only Mode	7-15
7.6.9	Bit Timing	7-15
7.6.10	FlexCAN Initialization/Reset Sequence	7-16
7.7	Special Operating Modes	7-17
7.7.1	Debug Mode	7-17
7.7.2	Stop Mode for Power Saving	7-18
7.7.3	Auto Power Save Mode	7-20
7.8	Register Definitions	7-20
7.8.1	Module Configuration Register (FCMCR)	7-25
7.8.2	Control Register 0 (FCCTL0)	7-28
7.8.3	Control Register 1 (FCCTL1)	7-30
7.8.4	Free-Running Timer (FCTMR)	7-31
7.8.5	Maximum Message Buffer Register (FCMAXMB)	7-32
7.8.6	Receive Mask Registers	7-32
7.8.7	Receive Global Mask (FCRXGMASK_H and FCRXGMASK_L)	7-33
7.8.8	Receive Buffer 14 Mask Registers (FCRX14MASK_H /_L)	7-35
7.8.9	Receive Buffer 15 Mask Registers (FCRX15MASK_H /_L)	7-35
7.8.10	Error and Status Register (FCSTATUS)	7-36
7.8.11	Interrupt Mask Register 1 (FCIMASK1)	7-39
7.8.12	Interrupt Flag Register 1 (FCIFLAG1)	7-39
7.8.13	Error Counters (FC_ERR_CNTRS)	7-40
7.9	Interrupts	7-41
7.10	Resets	7-42
7.11	Interaction of FlexCAN Message Buffers and CodeWarrior Debugger	7-42
7.12	Use of Message Buffers as Data RAM	7-43

## Chapter 8 General Purpose Input/Output (GPIO)

8.1	Introduction	8-3
8.2	Features	8-3
8.3	Block Diagram	8-3
8.4	Operating Modes	8-4
8.5	Configurations	8-5
8.6	Register Definitions	8-6
8.6.1	Pull-Up Enable Register (PUR)	8-8
8.6.2	Data Register (DR)	8-9
8.6.3	Data Direction Register (DDR)	8-9
8.6.4	Peripheral Enable Register (PER)	8-9
8.6.5	Interrupt Assert Register (IAR)	8-10
8.6.6	Interrupt Enable Register (IENR)	8-10
8.6.7	Interrupt Polarity Register (IPOLR)	8-11
8.6.8	Interrupt Pending Register (IPR)	8-11
8.6.9	Interrupt Edge Sensitive Register (IESR)	8-12
8.6.10	Push/Pull Output Mode Control Register (PPMODE)	8-12
8.6.11	Raw Data Register (RAWDATA)	8-13
8.7	Clocks and Resets	8-13
8.8	Interrupts	8-13

## Chapter 9 Joint Test Action Group Port (JTAG)

9.1	Introduction	9-3
9.2	Features	9-3
9.3	Block Diagram	9-4
9.4	Functional Description	9-4
9.4.1	Master TAP Instructions	9-4
9.5	TAP Controller	9-7
9.5.1	Operation	9-8
9.6	Memory Map	9-11
9.7	Pin Description	9-11
9.8	JTAG Port Architecture	9-12
9.9	JTAG Boundary Scan Register	9-12
9.10	Clocks	9-12
9.10.1	TCK	9-12
9.11	Interrupts	9-13
9.12	Resets	9-13
9.12.1	TRST Reset	9-13



## Chapter 10 Power Supervisor (PS)

10.1	Introduction	10-3
10.2	Features	10-3
10.3	Block Diagram	10-3
10.4	Functional Description	10-4
10.5	Register Definitions	10-6
10.5.1	Power Supervisor Control Register (LVICTLR)	10-6
10.5.2	Power Supervisor Status Register (LVISR)	10-7
10.6	Suggestions for LVI Interrupt Service Routines	10-8

## Chapter 11 Pulse Width Modulator (PWM)

11.1	Introduction	11-3
11.2	Features	11-3
11.3	Block Diagram	11-3
11.4	Functional Description	11-4
11.4.1	Prescaler	11-4
11.4.2	PWM Generator	11-4
11.4.3	Independent or Complementary Channel Operation	11-8
11.4.4	Deadtime Generators	11-9
11.4.5	Automatic Deadtime Correction	11-16
11.4.6	Asymmetric PWM Output	11-17
11.4.7	Output Polarity	11-18
11.5	Software Output Control	11-20
11.6	PWM Generator Loading	11-22
11.6.1	Load Enable	11-22
11.6.2	Load Frequency	11-22
11.6.3	Reload Flag	11-23
11.6.4	Synchronization Output	11-25
11.6.5	Initialization	11-25
11.7	Fault Protection	11-27
11.7.1	Fault Pin Filter	11-28
11.7.2	Automatic Fault Clearing	11-28
11.7.3	Manual Fault Clearing	11-28
11.8	Operating Modes	11-29
11.9	Pin Descriptions	11-30
11.9.1	PWM0–PWM5 Pins—(PWM0–5)	11-30
11.9.2	FAULT0–FAULT3 Pins—(FAULT0–3)	11-30
11.9.3	IS2 Pins—(IS0–2)	11-30

11.10	Register Definitions	11-31
11.10.1	PWM Control Register (PMCTL)	11-32
11.10.2	PWM Fault Control Register (PMFCTL)	11-36
11.10.3	PWM Fault Status and Acknowledge Register (PMFSA)	11-36
11.10.4	PWM Output Control Register (PMOUT)	11-37
11.10.5	PWM Counter Register (PMCNT)	11-39
11.10.6	PWM Counter Modulo Register (PWMCM)	11-39
11.10.7	PWM Value Registers (PWMVAL0–5)	11-40
11.10.8	PWM Deadtime Register (PMDEADTM)	11-40
11.10.9	PWM Disable Mapping Registers (PMDISMAP1-2)	11-41
11.10.10	PWM Configure Register (PMCFG)	11-42
11.10.11	PWM Channel Control Register (PMCCR)	11-44
11.10.12	PWM Port Register (PMPORT)	11-46
11.10.13	PWM Internal Correction Control Register (PMICCR)	11-46
11.11	Clocks	11-48
11.12	Interrupts	11-48
11.13	Resets	11-48

## Chapter 12

### Quadrature Decoder (DEC)

12.1	Introduction	12-3
12.2	Features	12-3
12.3	Block Diagram	12-4
12.4	Functional Description	12-5
12.4.1	Positive Vs. Negative Direction	12-5
12.4.2	Position Counter	12-5
12.4.3	Position Difference Counter	12-6
12.4.4	Revolution Counter	12-6
12.4.5	Holding and Initializing Registers	12-6
12.4.6	Prescaler for Slow or Fast Speed Measurement	12-7
12.4.7	Pulse Accumulator Functionality	12-7
12.4.8	Glitch Filter	12-7
12.4.9	Edge Detect State Machine	12-7
12.4.10	Watchdog Timer	12-8
12.5	Operating Modes	12-8
12.6	Pin Descriptions	12-8
12.6.1	Phase A Input (PHASEA)	12-9
12.6.2	Phase B Input (PHASEB)	12-9
12.6.3	Index Input (INDEX)	12-9
12.6.4	Home Switch Input (HOME)	12-10
12.7	Register Definitions	12-10

12.7.1	Decoder Control Register (DECCR) . . . . .	12-12
12.7.2	Filter Interval Register (FIR) . . . . .	12-15
12.7.3	Watchdog Timer Register (WTR) . . . . .	12-16
12.7.4	Position Difference Counter Register (POSD) . . . . .	12-17
12.7.5	Position Difference Counter Hold Register (POSDH) . . . . .	12-17
12.7.6	Revolution Counter Register (REV) . . . . .	12-18
12.7.7	Revolution Hold Register (REXH) . . . . .	12-18
12.7.8	Upper Position Counter Register (UPOS) . . . . .	12-18
12.7.9	Lower Position Counter Register (LPOS) . . . . .	12-18
12.7.10	Upper Position Hold Register (UPOSH) . . . . .	12-19
12.7.11	Lower Position Hold Register (LPOSH) . . . . .	12-19
12.7.12	Upper Initialization Register (UIR) . . . . .	12-19
12.7.13	Lower Initialization Register (LIR) . . . . .	12-20
12.7.14	Input Monitor Register (IMR) . . . . .	12-20
12.8	Interrupts . . . . .	12-21

## Chapter 13 Serial Communications Interface (SCI)

13.1	Introduction . . . . .	13-3
13.2	Features . . . . .	13-3
13.3	Block Diagram . . . . .	13-4
13.4	Functional Description . . . . .	13-4
13.4.1	Data Frame Format . . . . .	13-5
13.4.2	Baud Rate Generation . . . . .	13-6
13.4.3	Transmitter . . . . .	13-7
13.4.4	Receiver . . . . .	13-9
13.5	Special Operating Modes . . . . .	13-16
13.5.1	Single-Wire Operation . . . . .	13-16
13.5.2	Loop Operation . . . . .	13-17
13.5.3	Low-Power Options . . . . .	13-17
13.6	Register Definitions . . . . .	13-18
13.6.1	SCI Baud Rate Register (SCIBR) . . . . .	13-19
13.6.2	SCI Control Register (SCICR) . . . . .	13-20
13.6.3	SCI Status Register (SCISR) . . . . .	13-23
13.6.4	SCI Data Register (SCIDR) . . . . .	13-25
13.7	Clocks . . . . .	13-26
13.8	Resets . . . . .	13-26

13.9	Interrupts	13-26
13.9.1	Transmitter Empty Interrupt	13-26
13.9.2	Transmitter Idle Interrupt	13-26
13.9.3	Receiver Full Interrupt	13-26
13.9.4	Receive Error Interrupt	13-27

## Chapter 14 Serial Peripheral Interface (SPI)

14.1	Introduction	14-3
14.2	Features	14-3
14.3	Block Diagram	14-4
14.4	Operating Modes	14-4
14.4.1	Master Mode	14-5
14.4.2	Slave Mode	14-6
14.4.3	Wired OR Mode	14-6
14.5	Pin Descriptions	14-7
14.5.1	Master In/Slave Out (MISO)	14-7
14.5.2	Master Out/Slave In (MOSI)	14-8
14.5.3	Serial Clock (SCLK)	14-8
14.5.4	Slave Select ( $\overline{SS}$ )	14-8
14.6	Transmission Formats	14-9
14.6.1	Data Transmission Length	14-9
14.6.2	Data Shift Ordering	14-9
14.6.3	Clock Phase and Polarity Controls	14-9
14.6.4	Transmission Format When CPHA = 0	14-10
14.6.5	Transmission Format When CPHA = 1	14-11
14.6.6	Transmission Initiation Latency	14-12
14.7	Transmission Data	14-13
14.8	Error Conditions	14-15
14.8.1	Overflow Error	14-15
14.8.2	Mode Fault Error	14-17
14.9	Register Definitions	14-18
14.9.1	SPI Status and Control Register (SPSCR)	14-19
14.9.2	SPI Data Size and Control Register (SPDSR)	14-23
14.9.3	SPI Data Receive Register (SPDRR)	14-25
14.9.4	SPI Data Transmit Register (SPDTR)	14-25
14.10	Resets	14-25
14.11	Interrupts	14-26

## Chapter 15 Temperature Sensor System (TSENSOR)

15.1	Introduction	15-3
15.2	Features	15-3
15.3	Block Diagram	15-4
15.4	Functional Description	15-4
15.5	Operating Modes	15-5
15.6	Pin Descriptions	15-6
15.7	Register Definition	15-6
15.7.1	Temperature Sensor Control Register (TSENSOR_CTRL)	15-6
15.8	Interrupts	15-7
15.9	Resets	15-7

## Chapter 16 Quad Timer (TMR)

16.1	Introduction	16-3
16.2	Features	16-3
16.3	Block Diagram	16-4
16.4	Functional Description	16-4
16.4.1	Compare Registers Usage	16-5
16.4.2	Compare Preload Registers	16-6
16.4.3	Capture Register Usage	16-7
16.5	Operating Modes	16-7
16.5.1	Stop Mode	16-7
16.5.2	Count Mode	16-8
16.5.3	Edge Count Mode	16-9
16.5.4	Gated Count Mode	16-10
16.5.5	Quadrature Count Mode	16-10
16.5.6	Signed Count Mode	16-11
16.5.7	Triggered Count Mode	16-12
16.5.8	One-Shot Mode	16-13
16.5.9	Cascade Count Mode	16-14
16.5.10	Pulse Output Mode	16-15
16.5.11	Fixed Frequency PWM Mode	16-17
16.5.12	Variable Frequency PWM Mode	16-17
16.6	Register Definitions	16-20
16.6.1	Timer Compare Registers 1 (TMRCMP1)	16-22
16.6.2	Timer Compare Registers 2 (TMRCMP2)	16-23
16.6.3	Timer Capture Registers (TMRCAP)	16-23
16.6.4	Timer Load Registers (TMRLOAD)	16-24

16.6.5	Timer Hold Registers (TMRHOLD) . . . . .	16-24
16.6.6	Timer Counter Registers (TMRCNTR) . . . . .	16-25
16.6.7	Timer Control Registers (TMRCTRL) . . . . .	16-25
16.6.8	Timer Status and Control Registers (TMRSCR) . . . . .	16-29
16.6.9	Timer Comparator Load Registers 1 (TMRCMPLD1) . . . . .	16-31
16.6.10	Timer Comparator Load Registers 2 (TMRCMPLD2) . . . . .	16-31
16.6.11	Timer Comparator Status/Control Registers (TMRCOMSCR) . . . . .	16-32
16.7	Clocks . . . . .	16-33
16.8	Interrupts . . . . .	16-33

## Chapter 17 Voltage Regulator (VREG)

17.1	Introduction . . . . .	17-3
17.2	Features . . . . .	17-3
17.3	Block Diagram . . . . .	17-3
17.4	Functional Description . . . . .	17-4
17.5	Operating Modes . . . . .	17-4
17.6	Memory Map . . . . .	17-4
17.7	Pin Descriptions . . . . .	17-5
17.7.1	Output Voltage ( $V_{out}$ ) . . . . .	17-5
17.7.2	Input Voltage ( $V_{in}$ ) . . . . .	17-5
17.7.3	Capacitor Pins ( $V_{cap1}$ , $V_{cap2}$ , $V_{cap3}$ , and $V_{cap4}$ ) . . . . .	17-5
17.7.4	On-Chip Regulator Disable (OCR_DIS) . . . . .	17-5
17.8	Clocks . . . . .	17-5
17.9	Resets . . . . .	17-6
17.10	Interrupts . . . . .	17-6

## Appendix A Glossary

A.1	Glossary . . . . .	A-3
-----	--------------------	-----

## Appendix B Programmer's Sheets

B.1	Introduction . . . . .	B-3
B.2	Programmer's Sheets . . . . .	B-3

# LIST OF FIGURES

1-1	56800E Core Block Diagram . . . . .	1-4
1-2	56800E Chip Architecture with External Bus . . . . .	1-12
1-3	IPBus Bridge Interface With Other Main Components, System Side Operation . .	1-14
2-1	Option 1: Dual ADC Block Diagram . . . . .	2-4
2-2	Option 2: Two Dual ADCs with a Single Voltage Reference. . . . .	2-5
2-3	Sequential and Differential Modes of Operation of the ADC. . . . .	2-6
2-4	Simultaneous Mode Operation of the ADC. . . . .	2-7
2-5	Cyclic ADC – Top Level Block Diagram . . . . .	2-8
2-6	Typical Connections for Differential Measurements . . . . .	2-11
2-7	Result Register Data Manipulation . . . . .	2-12
2-8	Power Control Features of PSM Operation. . . . .	2-15
2-9	ADC Gain and Offset Error . . . . .	2-16
2-10	ADC Calibration References. . . . .	2-17
2-11	Code for Simultaneous Mode ADC Calibration. . . . .	2-21
2-12	Code for Sequential Mode ADC Calibration . . . . .	2-22
2-13	ADC Calibration – Sample Calculation of m and b . . . . .	2-23
2-14	ADC Calibration – Using m and b . . . . .	2-24
2-15	Equivalent Analog Input Circuit. . . . .	2-25
2-16	ADC Voltage Reference Circuit. . . . .	2-26
2-17	ADC Register Map Summary . . . . .	2-28
3-1	COP Module Block Diagram and Interface Signals . . . . .	3-3
4-1	EMI Block Diagram . . . . .	4-5
4-2	EMI Register Map Summary. . . . .	4-7
4-8	Data Bus Contention Timing Requiring MDAR Field Assertion . . . . .	4-12
4-10	External Read Cycle with Clock and RWS = 0 . . . . .	4-15
4-11	External Read Cycle with RWS = 1, RWSH = 0 and RWSS = 0 . . . . .	4-16
4-12	External Read Cycle with RWSS = RWS = 1, and RWSH = 0 . . . . .	4-17
4-13	External Read Cycle RWS = RWSH = 1 and RWSS = 0 . . . . .	4-18
4-14	External Write Cycle . . . . .	4-19
4-15	External Write Cycle with WWS = 1, WWSH = 0, and WWSS = 0 . . . . .	4-20
4-16	External Write Cycle with WWSS = 1, WWS = 0 and WWSH = 0 . . . . .	4-21
4-17	External Write Cycle with WWS = 0, WWSH = 1, WWSS = 0 . . . . .	4-22
4-18	External Write Cycle with WWSS = WWS = 1 and WWSH = 0 . . . . .	4-23
4-19	External Write Cycle with WWS = WWSH = 1 (WWSS = 0). . . . .	4-24
5-1	OCCS Block Diagram Without Relaxation Oscillator . . . . .	5-5

5-2	OCCS Block Diagram With Relaxation Oscillator . . . . .	5-6
5-3	Simplified Block Diagram, Loss of Reference Clock Detector . . . . .	5-19
5-4	External Crystal Oscillator Circuit . . . . .	5-20
5-5	External Ceramic Resonator Circuit . . . . .	5-21
5-6	Connecting an External Clock Signal Using XTAL . . . . .	5-21
5-7	OCCS Register Map Summary . . . . .	5-23
6-1	Flash Memory Block Diagram . . . . .	6-5
6-2	Flash Memory Array Small Memory Maps . . . . .	6-6
6-3	Flash Memory Array Large Memory Maps . . . . .	6-7
6-4	Example Program Algorithm . . . . .	6-13
6-5	UnBanked Flash Register Map Summary . . . . .	6-18
6-13	Banked Register Map Summary . . . . .	6-25
6-15	Protection Diagram Example . . . . .	6-27
6-17	Boot Protection Diagram Example . . . . .	6-28
6-20	Interrupt Implementation . . . . .	6-32
7-1	FlexCAN Block Diagram and Pinout . . . . .	7-4
7-2	Typical CAN System . . . . .	7-5
7-3	Extended ID Message Buffer Structure . . . . .	7-6
7-4	Standard ID Message Buffer Structure . . . . .	7-6
7-5	FlexCan Register Map Summary . . . . .	7-24
8-1	Bit-Slice View of the GPIO Logic . . . . .	8-4
8-2	GPIO Register Map Summary . . . . .	8-8
8-14	Edge Detection Circuit . . . . .	8-14
9-1	JTAG Block Diagram . . . . .	9-4
9-2	Bypass Register Diagram . . . . .	9-5
9-3	TAP Controller State Diagram . . . . .	9-8
10-1	Power Supervisor Block Diagram . . . . .	10-4
10-2	POR Vs. Low-Voltage Interrupts . . . . .	10-5
11-1	PWM Block Diagram . . . . .	11-4
11-2	Center-Aligned PWM Output . . . . .	11-5
11-3	Edge-Aligned PWM Output . . . . .	11-5
11-4	Center-Aligned PWM Period . . . . .	11-6
11-5	Edge-Aligned PWM Period . . . . .	11-6
11-6	Center-Aligned PWM Pulse Width . . . . .	11-7
11-7	Edge-Aligned PWM Pulse Width . . . . .	11-8
11-8	Complementary Channel Pairs . . . . .	11-8
11-9	Typical 3-Phase Inverter . . . . .	11-9
11-10	Deadtime Generators . . . . .	11-10



11-11	Deadtime Insertion, Center Alignment . . . . .	11-10
11-12	Deadtime at Duty Cycle Boundaries . . . . .	11-11
11-13	Deadtime and Small Pulse Widths . . . . .	11-11
11-14	Deadtime Distortion. . . . .	11-12
11-15	Current Status Sense Scheme for Deadtime Correction . . . . .	11-15
11-16	Output Voltage Waveforms. . . . .	11-16
11-17	Correction with Positive Current . . . . .	11-17
11-18	Correction with Negative Current . . . . .	11-17
11-19	Correction Logic . . . . .	11-18
11-20	PWM Polarity . . . . .	11-19
11-21	Software Output Control in Complementary Mode . . . . .	11-21
11-22	Full Cycle Reload Frequency Change . . . . .	11-22
11-23	Half Cycle Reload Frequency Change . . . . .	11-23
11-24	Full Cycle Center-Aligned PWM Value Loading . . . . .	11-23
11-25	Full Cycle Center-Aligned Modulus Loading . . . . .	11-24
11-26	Half Cycle Center-Aligned PWM Value Loading. . . . .	11-24
11-27	Half Cycle Center-Aligned Modulus Loading . . . . .	11-24
11-28	Edge-Aligned PWM Value Loading. . . . .	11-25
11-29	Edge-Aligned Modulus Loading . . . . .	11-25
11-30	PWMEN and PWM Pins in Independent Operation (OUTCTL0–5 = 0) . . . . .	11-26
11-31	PWMEN & PWM Pins in Complement Operation (OUTCTL0,2,4 = 0) . . . . .	11-26
11-32	Fault Decoder for PWM 0 . . . . .	11-27
11-33	Automatic Fault Clearing. . . . .	11-28
11-34	Manual Fault Clearing (Example 1). . . . .	11-29
11-35	Manual Fault Clearing (Example 2). . . . .	11-29
11-36	PWM Register Map Summary. . . . .	11-32
11-49	Channel Swapping . . . . .	11-46
12-1	Quadrature Decoder Block Diagram. . . . .	12-4
12-2	Quadrature Decoder Signals. . . . .	12-5
12-3	DEC Register Map Summary . . . . .	12-12
13-1	SCI Block Diagram . . . . .	13-4
13-2	SCI Data Frame Formats . . . . .	13-5
13-3	SCI Transmitter Block Diagram. . . . .	13-7
13-4	SCI Receiver Block Diagram. . . . .	13-10
13-5	Receiver Data Sampling . . . . .	13-11
13-6	Slow Data . . . . .	13-13
13-7	Fast Data. . . . .	13-14
13-8	Single-Wire Operation (LOOP = 1, RSRC = 1). . . . .	13-16

13-9	Loop Operation (LOOP = 1, RSRC = 0) . . . . .	13-17
13-10	SCI Register Map Summary . . . . .	13-19
14-1	SPI Block Diagram . . . . .	14-4
14-2	Full Duplex Master/Slave Connections . . . . .	14-5
14-3	Master with Two Slaves . . . . .	14-7
14-4	Transmission Format (CPHA = 0) . . . . .	14-11
14-5	CPHA/SS Timing . . . . .	14-11
14-6	Transmission Format (CPHA = 1) . . . . .	14-12
14-7	Transmission Start Delay (Master) . . . . .	14-13
14-8	SPRF/SPTE Interrupt Timing . . . . .	14-14
14-9	Missed Read of Overflow Condition . . . . .	14-16
14-10	Clearing SPRF When OVRF Interrupt Is Not Enabled . . . . .	14-16
14-11	SPI Register Map Summary . . . . .	14-19
14-16	SPI Interrupt Request Generation . . . . .	14-27
15-1	Temperature Sensor Block Diagram . . . . .	15-4
15-2	Temperature Sensor Output Characteristics . . . . .	15-5
16-1	TMR Module Block Diagram . . . . .	16-4
16-2	Variable PWM Waveform . . . . .	16-7
16-3	Quadrature Incremental Position Encoder . . . . .	16-11
16-4	Compare Preload Timing . . . . .	16-20
16-5	TMR Register Map Summary . . . . .	16-22
17-1	Voltage Regulator Block Diagram . . . . .	17-3

# LIST OF TABLES

2-1	ADC Scan Sequence Control . . . . .	2-13
2-2	ADC Calibration Procedure . . . . .	2-20
2-3	External Signal Properties . . . . .	2-24
2-4	ADC Memory Map . . . . .	2-27
2-5	ADC Register Summary . . . . .	2-27
2-6	ADC Input Conversion for Sample Bits . . . . .	2-35
2-7	Clock Summary . . . . .	2-47
2-8	Interrupt Summary . . . . .	2-48
3-1	COP Memory Map . . . . .	3-4
3-2	COP Register Summary . . . . .	3-4
3-3	Timeout Values . . . . .	3-7
4-1	DEC Memory Map . . . . .	4-5
4-2	EMI Register Summary . . . . .	4-6
4-3	CS Encoding of the BLKSZ Field . . . . .	4-8
4-4	CSOR Encoding BYTE_EN Values . . . . .	4-9
4-5	CSOR Encoding of Read/Write Values . . . . .	4-10
4-6	CSOR Encoding of PS/DS Values . . . . .	4-10
4-7	Operation with DRV . . . . .	4-14
5-1	Clock Choices without Relaxation Oscillator . . . . .	5-8
5-2	Clock Choices with Relaxation Oscillator . . . . .	5-9
5-3	Legal PLL Settings for 56F8300 Family . . . . .	5-13
5-4	Legal PLL Settings for 56F8100 Family . . . . .	5-15
5-5	OCCS Memory Map . . . . .	5-22
5-6	OCCS Register Summary . . . . .	5-22
5-7	Interrupt Summary . . . . .	5-32
6-2	Flash Memory Register Address Map . . . . .	6-8
6-1	Flash Memory Configuration Field . . . . .	6-8
6-3	Flash User Mode Valid Commands . . . . .	6-14
6-4	Flash Memory Map . . . . .	6-17
6-5	Unbanked Flash Registers . . . . .	6-17
6-6	BKSEL . . . . .	6-21
6-7	SEC . . . . .	6-22
6-8	Flash Memory Map . . . . .	6-24
6-9	Banked Flash Registers . . . . .	6-25
6-10	Flash CMD User Mode Commands . . . . .	6-30

6-11	Flash Memory Interrupt Sources . . . . .	6-31
7-1	Common Extended/Standard Format Frames . . . . .	7-7
7-2	Message Buffer Codes for Receive Buffers . . . . .	7-7
7-3	Message Buffer Codes for Transmit Buffers . . . . .	7-7
7-4	Extended Format Frames . . . . .	7-8
7-5	Standard Format Frames . . . . .	7-8
7-6	Examples of System Clock/CAN Bit-Rate/S-Clock . . . . .	7-16
7-7	FlexCAN Memory Map . . . . .	7-20
7-8	FlexCAN Register Summary . . . . .	7-21
7-9	Mask Examples for Normal/Extended Messages . . . . .	7-33
8-1	GPIO Registers With Default Reset Values . . . . .	8-5
8-3	GPIO Data Transfers Between I/O Pin and IPBus . . . . .	8-6
8-2	GPIO Pull-Up Enable Functionality . . . . .	8-6
8-4	GPIO Register Summary . . . . .	8-7
8-5	GPIO Interrupt Assert Functionality . . . . .	8-14
9-1	Master TAP Instructions Opcode . . . . .	9-5
9-2	Flash Erase Register . . . . .	9-7
9-3	JTAG Pin Description . . . . .	9-11
9-4	Clock Summary . . . . .	9-13
9-5	Reset Summary . . . . .	9-13
10-1	PS Memory Map . . . . .	10-6
10-2	Power Supervisor Register Summary . . . . .	10-6
11-1	PWM Value and Underflow Conditions . . . . .	11-7
11-2	Correction Method Selection . . . . .	11-13
11-3	Top/Bottom Manual Correction . . . . .	11-14
11-4	Top/Bottom Automatic Correction . . . . .	11-16
11-5	Top/Bottom Corrections Selected by ICCn Bits . . . . .	11-18
11-6	Fault Mapping . . . . .	11-27
11-7	Modes When PWM Operation is Restricted . . . . .	11-30
11-8	PWM Memory Map . . . . .	11-31
11-9	PWM Register Summary . . . . .	11-31
11-10	PWM Reload Frequency . . . . .	11-33
11-11	PWM Prescaler . . . . .	11-34
11-12	Software Output Control . . . . .	11-38
12-1	Switch Matrix for Inputs to the Timer . . . . .	12-8
12-2	DEC Memory Map . . . . .	12-10
12-3	DEC Register Summary . . . . .	12-10
12-4	Interrupt Summary . . . . .	12-21

13-1	Example 8-Bit Data Frame Formats . . . . .	13-5
13-2	Example 9-Bit Data Frame Formats . . . . .	13-5
13-3	Example Baud Rates (Module Clock = 60MHz) . . . . .	13-6
13-4	Example Baud Rates (Module Clock = 40MHz) . . . . .	13-6
13-5	Start Bit Verification. . . . .	13-11
13-6	Data Bit Recovery . . . . .	13-12
13-7	Stop Bit Recovery . . . . .	13-12
13-8	Loop Functions . . . . .	13-16
13-9	SCI Memory Map . . . . .	13-18
13-10	SCI Register Summary . . . . .	13-18
13-11	SCI Interrupt Sources . . . . .	13-26
14-1	External I/O Signals . . . . .	14-7
14-2	SPI I/O Configuration . . . . .	14-9
14-3	SPI Memory Map . . . . .	14-18
14-4	SPI Register Summary . . . . .	14-19
14-5	SPI Master Baud Rate Selection. . . . .	14-20
14-6	Data Size. . . . .	14-24
14-7	SPI Interrupts . . . . .	14-26
15-1	TSENSOR Memory Map. . . . .	15-6
15-2	TSENSOR Register Summary . . . . .	15-6
16-1	TMR Memory Map . . . . .	16-20
16-2	TMR Register Summary . . . . .	16-21
16-3	Setting Combination of Capture Mode and IPS State. . . . .	16-30
16-4	Values for Compare Preload Control 2. . . . .	16-33
16-5	Values for Compare Preload Control 1 . . . . .	16-33
16-6	Timer Interrupt Flags. . . . .	16-34
17-1	Signal Properties. . . . .	17-5

**List of Tables**



# Preface

## About This Manual

Features of the 56F8300/56F8100 Family Series of 16-bit hybrid controllers are described in this preliminary manual release. Peripheral modules are documented here. This manual is intended to be used with the *DSP56800E Reference Manual* (DSP56800ERM), describing the Central Processing Unit (CPU), programming models, and instruction set details. The *568300 Technical Data Sheet* provides electrical specifications as well as timing, pinout, packaging descriptions and chip specific details of architecture and memory map particulars.

## Audience

Information in this manual is intended to assist design and software engineers integrate the 56F8300/56F8100 digital signal processors into a design and/or while developing application software.

## Manual Organization

This manual is arranged in chapters described below:

- **Chapter 1, Overview**—provides a brief overview, describing the structure of this document, including lists of other documentation necessary to use these chips.
- **Chapter 2, Analog-to-Digital Converter (ADC)**—describes features, functions and registers of the analog-to-digital converter.
- **Chapter 3, Computer Operating Properly (COP)**—Computer Operating Properly (COP) module is used to help software recover from runaway code.
- **Chapter 4, External Memory Interface (EMI)**—provides an interface by which the 56800E core can utilize external asynchronous memory.
- **Chapter 5, On-Chip Clock Synthesis (OCCS)**—elaborates on the internal oscillator, relaxation oscillator, Phase Lock Loop (PLL), and timer distribution chain.
- **Chapter 6, Flash Memory (FM)**—describes the Program Flash, Data Flash, and Boot Flash features and registers.
- **Chapter 7, FlexCAN (FC)**—describes the implementation of the CAN protocol through the FlexCAN module.
- **Chapter 8, General Purpose Input/Output (GPIO)**—describes how peripheral pins are multiplexed with GPIO functions.

- **Chapter 9, Joint Test Action Group Port (JTAG)**—explains the Joint Test Action Group (JTAG) testing methodology and its capabilities with Test Access Port (TAP) and Enhanced OnCE (fully explained in the DSP56800E Reference Manual).
- **Chapter 10, Power Supervisor (PS)**—details how the on-chip Power Supervisor module monitors on-chip voltages.
- **Chapter 11, Pulse Width Modulator (PWM)**—describes the function, configuration and registers of the PWM.
- **Chapter 12, Quadrature Decoder (DEC)**—describes the features and registers of the Quadrature Decoder.
- **Chapter 13, Serial Communications Interface (SCI)**—presents the Serial Communications Interface (SCI), communicating with devices, such as other DSPs, microprocessors, or peripherals providing the primary data input path as codecs.
- **Chapter 14, Serial Peripheral Interface (SPI)**—describes the Serial Peripheral Interface (SPI), which communicates with external devices, such as Liquid Crystal Displays (LCDs) and Microcontroller Units (MCUs).
- **Chapter 15, Temperature Sensor System (TSENSOR)**—specifies requirements for a temperature sensor module intended to operate in conjunction with an Analog-Digital Converter.
- **Chapter 16, Quad Timer (TMR)**—outlines the internal Quad Timer devices available, including features and registers.
- **Chapter 17, Voltage Regulator (VREG)**—describes the on-chip mechanism used to regulate an external 3.3 V supply down to 2.5 V levels for use with internal logic.
- **Appendix A, Glossary**—provides definitions of terms, peripherals, acronyms and register names used in this manual.
- **Appendix B, Programmer's Sheets**—supplies concise, one-location registers and their preference tables intended to simplify programming of the devices.



## Suggested Reading

A list of DSP-related books is provided here as an aid to those who may be new to DSPs:

*Advanced Topics in Signal Processing*, Jae S. Lim and Alan V. Oppenheim (Prentice-Hall: 1988).

*Applications of Digital Signal Processing*, A. V. Oppenheim (Prentice-Hall: 1978).

*Digital Processing of Signals: Theory and Practice*, Maurice Bellanger (John Wiley and Sons: 1984).

*Digital Signal Processing*, Alan V. Oppenheim and Ronald W. Schaffer (Prentice-Hall: 1975).

*Digital Signal Processing: A System Design Approach*, David J. DeFatta, Joseph G. Lucas, and William S. Hodgkiss (John Wiley and Sons: 1988).

*Discrete-Time Signal Processing*, A. V. Oppenheim and R.W. Schaffer (Prentice-Hall: 1989).

*Foundations of Digital Signal Processing and Data Analysis*, J. A. Cadzow (Macmillan: 1987).

*Handbook of Digital Signal Processing*, D. F. Elliott (Academic Press: 1987).

*Introduction to Digital Signal Processing*, John G. Proakis and Dimitris G. Manolakis (Macmillan: 1988).

*IP Bus Specifications*, Semiconductor Reuse Standard, SRSIPB1, v 2.0, Draft 1.6.

*Multirate Digital Signal Processing*, R. E. Crochiere and L. R. Rabiner (Prentice-Hall: 1983).

*Signal Processing Algorithms*, S. Stearns and R. Davis (Prentice-Hall: 1988).

*Signal Processing Handbook*, C. H. Chen (Marcel Dekker: 1988).

*Signal Processing: The Modern Approach*, James V. Candy (McGraw-Hill: 1988).

*Theory and Application of Digital Signal Processing*, Lawrence R. Rabiner and Bernard Gold (Prentice-Hall: 1975).

# Manual Conventions

Conventions used in this manual:

- Bits within registers are always listed from Most Significant Bit (MSB) to Least Significant Bit (LSB).
- Bits within a register are formatted AA[n:0] when more than one bit is involved in a description. For purposes of description, the bits are presented as if they are contiguous within a register. However, this is not always the case. Refer to the programming model diagrams or to the programmer's sheets to see the exact location of bits within a register.
- When a bit is described as *set*, its value is set to *one*. When a bit is described as *cleared*, its value is set to *zero*.
- The word *pin* is a generic term for any pin on the chip.
- Pins or signals asserted low, made active when pulled to ground, have an over-bar above their name. For example, the  $\overline{SS0}$  pin is asserted low.
- Hex values are indicated with a dollar sign (\$) preceding the hex value, as follows: \$F1A0 is the X memory address for an Interrupt Priority Register (IPR).
- Code examples follow in a single spaced font.

---

<code>BFSET #0007,X:PCC ; Configure:</code>	line 1
<code>; MIS00, MOSI0, SCK0 for SPI master</code>	line 2
<code>; ~SS0 as PC3 for GPIO</code>	line 3

---

- Pins or signals listed in code examples asserted as low have a tilde in front of their names. In the previous example, line three refers to the  $\overline{SS0}$  pin, shown as  $\sim SS0$ .
- The word *reset* is used in three different contexts in this manual. They are described as:
  - a reset pin is always written as  $\overline{RESET}$ , in uppercase, using the over bar
  - the processor state occurs when the  $\overline{RESET}$  pin is asserted. It is always written as Reset, with a capitalized first letter
  - the word *reset* refers to the reset function. It is written in lowercase, without italics, used here only for differentiation. The word may require a capital letter as style dictates, such as in headings and captions
- The word *assert* means a high true (active high) signal is pulled high to  $V_{DD}$ , or a low true (active low) signal is pulled low to ground. The word *deassert* means a high true signal is pulled low to ground, or a low true signal is pulled high to  $V_{DD}$ .


## Pin Conventions

Signal/Symbol	Logic State	Signal State	Voltage <sup>1</sup>
$\overline{PIN}$	True	Asserted	$V_{IL}/V_{OL}$

Signal/Symbol	Logic State	Signal State	Voltage <sup>1</sup>
$\overline{\text{PIN}}$	False	Deasserted	$V_{IH}/V_{OH}$
PIN	True	Asserted	$V_{IH}/V_{OH}$
PIN	False	Deasserted	$V_{IL}/V_{OL}$

1. Values for  $V_{IL}$ ,  $V_{OL}$ ,  $V_{IH}$ , and  $V_{OH}$  are defined by individual product specifications.

Throughout the manual, registers and tables displaying a grayed area designate reserved bits.

 = Reserved or not implemented bit, write or read as zero for future compatibility



---

# Chapter 1

## Overview

## Document Revision History for [Chapter 1, Overview](#)

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 5.0	Added FLASH feature entry for 8100 devices on page 16 and converted to Freescale design Reorder information for reading ease

## 1.1 Introduction to the 56800E Core

The 56F8300 device is a member of the 16-bit controller 56800E core-based family. Its processing power combined with the functionality of a microcontroller with a flexible set of peripherals, creates an extremely cost-effective solution provided on a single chip. Because of its low cost, configuration flexibility, and compact program code, the device is well-suited for many applications.

The 56800E core is based on a Harvard-style architecture consisting of three execution units operating in parallel, allowing as many as six operations per instruction cycle. The MCU-style programming model and optimized instruction set allow straightforward generation of efficient, compact 16-bit control code. The instruction set is also highly efficient for C/C++ Compilers, enabling rapid development of optimized control applications.

### 1.1.1 56800E Core Enhancements

The 56800E core architecture extends the 56800 family architecture. It is source-code compatible with 56800 devices, adding these features:

- Byte and long data types, supplementing the word data type of the 56800
- 24-bit data memory address space
- 21-bit program memory address space
- Two additional 24-bit address registers
- Two additional 36-bit accumulator registers
- Full-precision integer multiplication
- 32-bit logical and shifting operations
- Second read in dual read instruction can access off-chip memory
- Loop Count (LC) register extended to 16 bits
- Support for nested DO looping through additional loop address and count registers
- Loop address and hardware stack extended to 24 bits
- Three additional interrupt levels with a software interrupt for each level
- Enhanced On-Chip Emulation (EOnCE) with three debugging modes:
  - non-intrusive real-time debugging
  - minimally intrusive real-time debugging
  - Breakpoint and Step modes (core is halted)

## 1.1.2 56800E Core

The 56800E core is composed of several independent functional units. The program controller, Address Generation Unit (AGU), and data Arithmetic Logic Unit (ALU) contain their own register sets and control logic, allowing them to operate independently and in parallel, increasing throughput. There is also an independent bit manipulation unit, enabling efficient bit manipulation operations. Each functional unit interfaces with the other units, memory, and the memory-mapped peripherals over the core's internal address and data buses. A block diagram of the 56800E core architecture is illustrated in [Figure 1-1](#).

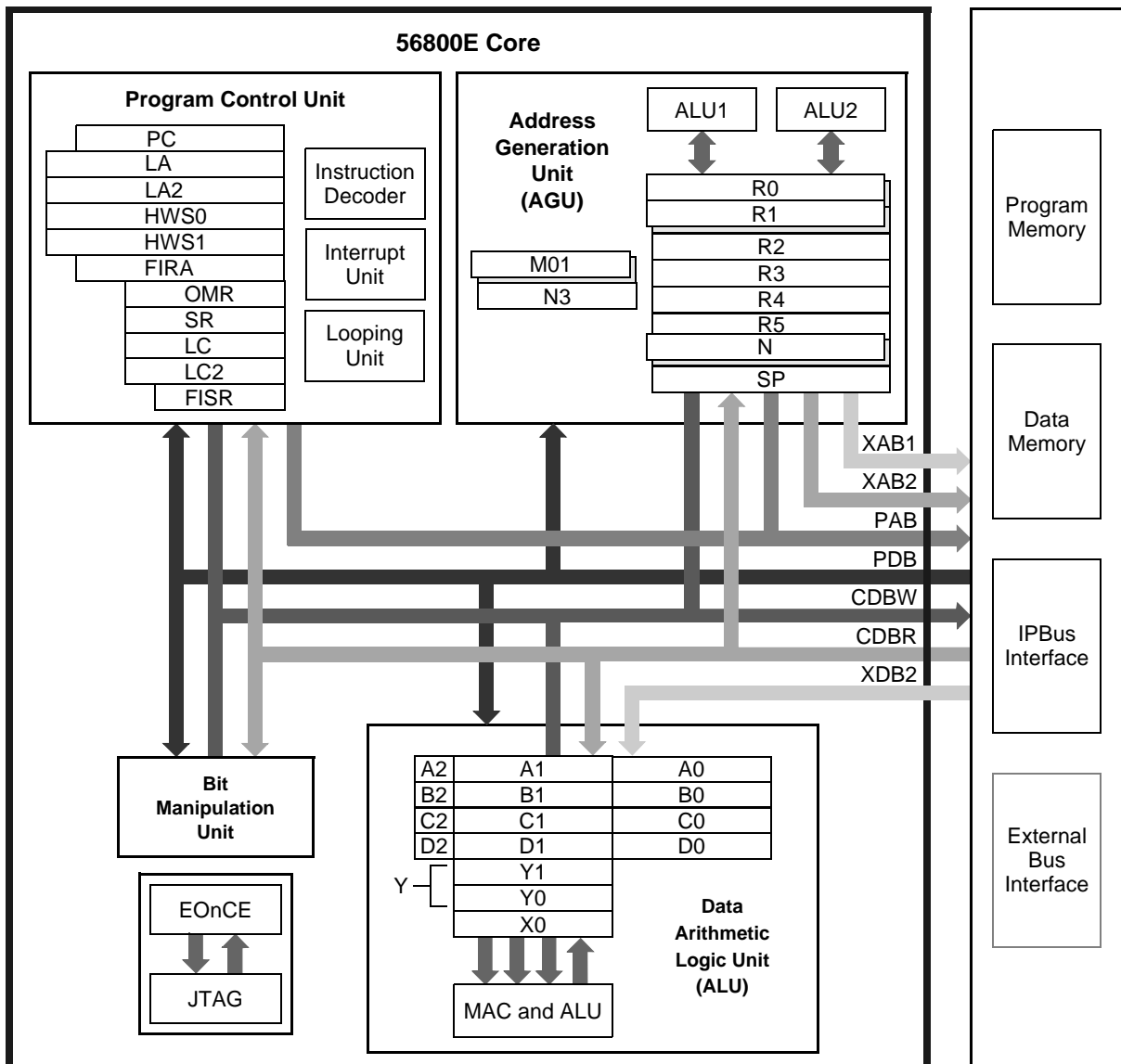


Figure 1-1. 56800E Core Block Diagram



Instruction execution is pipelined to take advantage of the parallel units, thereby significantly decreasing execution time for each instruction. For example, within a single execution cycle, it is possible for the:

- Data ALU to perform a multiplication operation
- AGU to generate up to two addresses
- Program controller to prefetch the next instruction

The major components of the 56800E core include:

- Address buses
- Data buses
- Data Arithmetic Logic Unit (ALU)
- Address Generation Unit (AGU)
- Program controller and hardware looping unit
- Bit manipulation unit
- Enhanced OnCE debugging module
- Clock generation
- Reset circuitry

### 1.1.2.1 Address Buses

The core contains three address buses:

1. Program Memory Address Bus (PAB)
2. Primary Data Address Bus (XAB1)
3. Secondary Data Address Bus (XAB2)

The program address bus is 21 bits wide and is used to address (16-bit) words in program memory. The two 24-bit data address buses allow for two simultaneous accesses to data (X) memory. The XAB1 bus can address byte, word, and long data types. The XAB2 bus is limited to (16-bit) word accesses.

All three buses address on-chip memory. They can also address off-chip memory on devices containing an external bus interface unit.

### 1.1.2.2 Data Buses

Data transfers inside the chip occur over these buses:

- Two unidirectional 32-bit buses:
  - Core Data Bus for Reads (CDBR)

- Core Data Bus for Writes (CDBW)
- Two unidirectional 16-bit buses:
  - Secondary X Data Bus (XDB2)
  - Program Data Bus (PDB)
- IPBus interface

Data transfers between the data ALU and data memory use the CDBR and CDBW when a single memory read or write is performed. When two simultaneous memory reads are performed, the transfers use the CDBR and XDB2 buses. All other data transfers to core blocks occur using the CDBR and CDBW buses. Peripheral transfers occur through the IPBus interface. Instruction word fetches occur over the PDB.

This bus structure supports up to three simultaneous 16-bit transfers. Any one of the following can occur in a single clock cycle:

- One instruction fetch
- One read from data memory
- One write to data memory
- Two reads from data memory
- One instruction fetch and one read from data memory
- One instruction fetch and one write to data memory
- One instruction fetch and two reads from data memory

An instruction fetch will take place on every clock cycle, although it is possible for data memory accesses to be performed without an instruction fetch. Such accesses typically occur when a hardware loop is executed and the repeated instruction is only fetched on the first loop iteration.

### 1.1.2.3 Data Arithmetic Logic Unit (Data ALU)

The data Arithmetic Logic Unit (ALU) performs all of the arithmetic, logical, and shifting operations on data operands. The data ALU contains the following components:

- Three 16-bit data registers (X0, Y0, and Y1)
- Four 36-bit accumulator registers (A, B, C, and D)
- One Multiply-Accumulator (MAC) unit
- A single-bit accumulator shifter
- One arithmetic and logical multi-bit shifter
- One MAC output limiter
- One data limiter

The data ALU can perform multiplication, multiply-accumulation (with positive or negative accumulation), addition, subtraction, shifting, and logical operations in a single cycle. Division and normalization operations are provided by iteration instructions. Signed and unsigned multiple precision arithmetic is also supported. All operations are performed using two's-complement fractional or integer arithmetic.

Data ALU source operands can be 8, 16, 32, or 36 bits in size and can be located in memory, in immediate instruction data, or in the data ALU registers. Arithmetic operations and shifts can have 16-, 32-, or 36-bit results. Logical operations are performed on 16- or 32-bit operands and yield results of the same size. The results of data ALU operations are stored either in one of the data ALU registers or directly in memory.

#### 1.1.2.4 Address Generation Unit (AGU)

The Address Generation Unit (AGU) performs all of the calculations of effective addresses for data operands in memory. It contains two address ALUs, allowing up to two 24-bit addresses to be generated every instruction cycle:

- One for either the Primary Data Address Bus (XAB1) or the Program Address Bus (PAB)
- One for the Secondary Data Address Bus (XAB2)

The address ALU can perform both linear and modulus address arithmetic. The AGU operates independently of the other core units, minimizing address-calculation overhead.

The AGU can directly address  $2^{24}$  (16M) words on the XAB1 and XAB2 buses. It can access  $2^{21}$  (2M) words on the PAB. The XAB1 bus can address byte, word, and long data operands. The PAB and XAB2 buses can only address words in memory.

The AGU consists of the following registers and functional units:

- Seven 24-bit address registers (R0–R5 and N)
- Four 24-bit shadow registers (for R0, R1, N, and M01)
- A 24-bit dedicated Stack Pointer (SP) register
- Two offset registers (N and N3)
- A 16-bit modifier register (M01)
- A 24-bit adder unit
- A 24-bit modulus arithmetic unit

Each of the address registers, R0–R5, can contain either data or an address. All of these registers can provide an address for the XAB1 and PAB address buses; addresses on the XAB2 bus are provided by the R3 register. The N offset register can be used either as a general-purpose address register, or as an offset, or update value for the addressing modes supporting those values. The second 16-bit offset register, N3, is used only for offset or update values. The modifier register, M01, selects between linear and modulus address arithmetic.

### 1.1.2.5 Program Controller and Hardware Looping Unit

The program controller is responsible for instruction fetching and decoding, interrupt processing, hardware interlocking, and hardware looping. Actual instruction execution takes place in the other core units, such as in the data ALU, AGU, or bit manipulation unit.

The program controller contains the following:

- An instruction latch and decoder
- The hardware looping control unit
- Interrupt control logic
- A Program Counter (PC)
- Two special registers for Fast Interrupts:
  - Fast Interrupt Return Address (FIRA) register
  - Fast Interrupt Status (FISR) register
- Seven user-accessible Status and Control registers
  - two-level deep Hardware Stack (HWS)
  - Loop Address (LA) register
  - Loop Address 2 (LA2) register
  - Loop Count (LC) register
  - Loop Count 2 (LC2) register
  - Status Register (SR)
  - Operating Mode Register (OMR)

The Operating Mode Register (OMR) is a programmable register controlling the operation of the 56800E core, including the memory map configuration. The initial operating mode is typically latched on reset from an external source; it can subsequently be altered under program control.

The Loop Address (LA) register and Loop Count (LC) register work in conjunction with the Hardware Stack (HWS) to support no-overhead hardware looping. The hardware stack is an internal Last-In-First-Out (LIFO) buffer consisting of two 24-bit words to store the address of the first instruction of a hardware DO loop. When executing the DO instruction, the address of the first instruction in the loop is pushed onto the HWS. When a loop finishes normally or an ENDDO instruction is encountered, the value is popped from the HWS. This process allows for one hardware DO loop to be nested inside another.

### 1.1.2.6 Bit Manipulation Unit

The bit manipulation unit performs bit field operations on data memory words, peripheral registers, and registers within the 56800E core. It is capable of testing, setting, clearing, or

inverting individual or multiple bits within a 16-bit word. The bit manipulation unit can also test bytes for branch-on-bit field instructions.

### 1.1.2.7 Enhanced On-Chip Emulation (EOnCE) Module

The Enhanced On-Chip Emulation (EOnCE) module allows interaction in a debug environment with the 56800E core and its peripherals. Its capabilities include:

- Examining registers
- Accessing memory or on-chip peripherals
- Setting breakpoints in memory
- Stepping or tracing instructions

The EOnCE module provides simple, inexpensive, and speed independent access to the 56800E core for sophisticated debugging and economical system development. The JTAG port allows access to the EOnCE module and through the 56F8300 device to its target system, retaining debug control without sacrificing other user accessible on-chip resources. This technique eliminates the costly cabling and access to processor pins required by traditional emulator systems. The EOnCE interface is fully described in the 56800E Reference manual.

### 1.1.3 System Bus Controller

The 56F8300 System Bus Controller (SBC) provides an interface between the 56800E core and other modules on the system bus. The SBC is composed of a set of buffers for the address and control signals originating at the core, and a separate set of multiplexers, routing data from each memory-mapped block back to the core.

The 56F8300 architecture includes two separate bus models:

1. System bus
2. IPBus

Internal memories, the external memory interface and the core are located on the system buses. All peripherals connect to the IPBus. Access to the IPBus by the 16-bit controller core is facilitated by the IPBus bridge. The system bus controller does not participate in IPBus transactions. Within this document, all descriptions of bus operations pertain only to the system bus.

For performance reasons, all system bus signals in the 56F8300 architecture have a single driver, as opposed to the more common three-state bus configurations. Read data from each memory-mapped device is multiplexed to avoid contention. Since the 16-bit controller core is the only system bus master, there is no need for multiplexers on the address, control or write data buses.

## 1.1.4 Operation Method

The 56800E system utilizes a pipelined memory architecture and separate program and data buses. Each memory cycle is completed in three or more system clock cycles. During the first of these cycles, the core presents an address, along with control signals, indicating the type of memory cycle being initiated. This clock cycle is referred to as the address phase of a memory cycle. The following cycle is an intermediate step not involving bus activity related to the memory cycle in progress. Finally, the data phase occurs. During this phase data is transferred to or from the master, depending upon the type of cycle initiated during the address phase.

Memory cycles can overlap in each clock cycle. A new address phase can begin while the data phase for a preceding memory cycle occurs. In certain cases, memory devices or the 16-bit controller core may require additional time to complete operations. When this occurs, clock edges to other modules are withheld by the clock generation circuitry.

## 1.2 Introduction to 56F8300/56F8100 Devices

### 1.2.1 Applications

The 56F8x00 family of products includes many peripherals useful for applications such as:

- Motion control
- Smart appliances
- Steppers
- Encoders
- Tachometers
- Limit switches
- Power supply and control
- Automotive control
- Engine management
- Noise suppression
- Remote utility metering
- Industrial control of
  - power
  - lighting
  - automation
- Power line modem

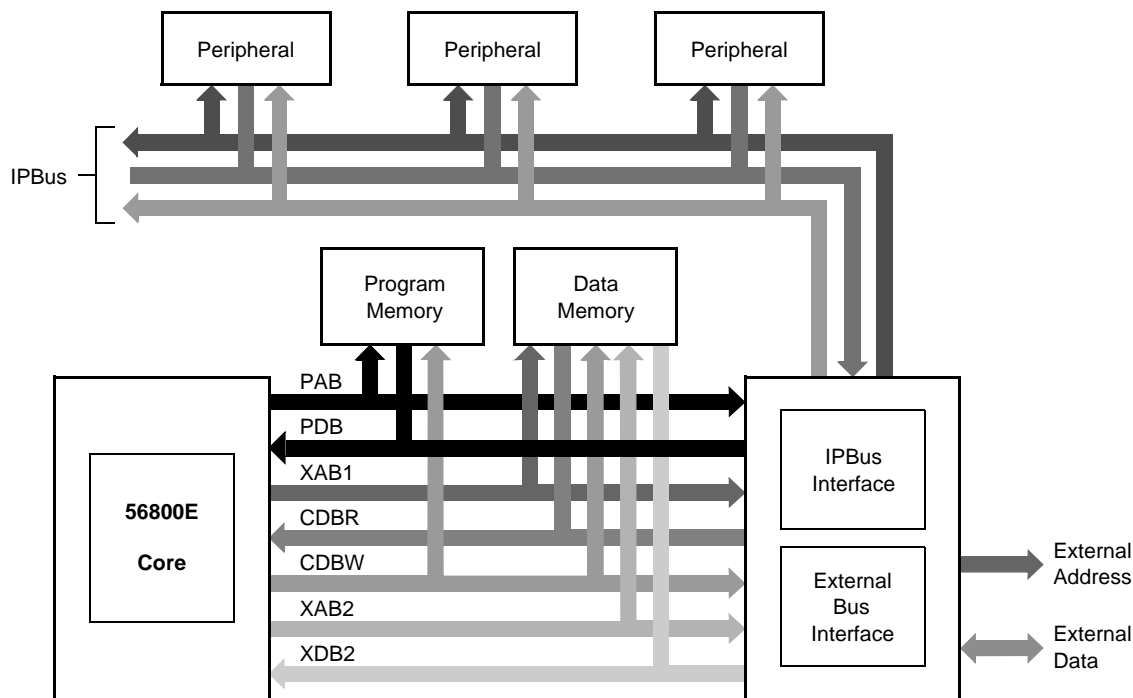
## 1.3 Features

The device enhances performance, reducing application cost, and promoting an ease of product development. Features making these benefits possible include:

- Program Flash
- Program RAM
- Data Flash
- Data RAM
- Boot Flash
- External Memory Interface (EMI)
- 12-bit ADCs
- Temperature Sensor
- Quadrature Decoders
- FlexCAN module
- Serial Communication Interfaces (SCIs)
- Serial Peripheral Interfaces (SPIs)
- Quad Timers
- Computer Operating Properly (COP)/Watchdog
- JTAG/Enhanced On-Chip Emulation (EOnCE) for debugging
- GPIO lines
- LQFP package
- Competitive cost sensitive packaging

### 1.3.1 System Architecture and Peripheral Interface

The 56800E system architecture encompasses all on-chip components, including the core, on-chip memory, peripherals, and the buses necessary to connect them. [Figure 1-2](#) illustrates the overall system architecture for a device with an external bus.



**Figure 1-2. 56800E Chip Architecture with External Bus**

The complete architecture includes the following components:

- 56800E core
- On-chip program memory
- On-chip data memory
- On-chip peripherals
- IPBus peripheral interface
- External bus interface

Some 56800E devices might not implement an external bus interface. Regardless of the implementation, all peripherals communicate with the 56800E core via the IPBus interface. The program memory buses are not connected to peripherals.

### 1.3.2 IPBus Bridge (IPBB)

The IPBus Bridge (IPBB) provides a means for communication between the high speed core and the low-bandwidth devices on the IP peripheral bus. Among other functions, the bridge is responsible for maintaining an orderly and synchronized communication between devices on both sides running at two different clock frequencies.

The IPBus architecture supports a variety of on-chip peripherals:



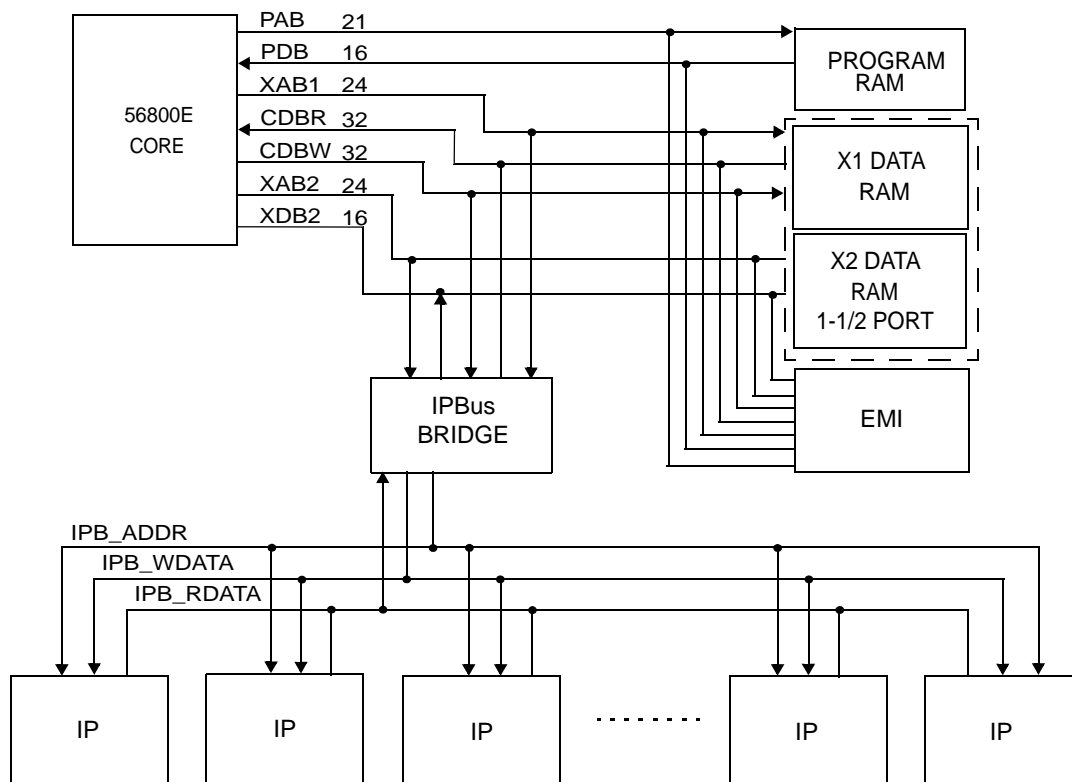
- Phase-Locked Loop (PLL) module
- 16-bit Quad Timer (TMR) modules
- Computer Operating Properly (COP) module
- Serial Peripheral Interface (SPI) modules
- Serial Communication Interface (SCI/UART) modules
- Programmable General-Purpose I/O (GPIO) modules
- Quad Decoder (DEC)
- Pulse Width Module (PWM)
- Analog-to-Digital Converters (ADC)

**Figure 1-3** denotes the position and interface of the IPBus Bridge with other main blocks within the chip.

Other connections in the figure not pertaining to the primary function of the bridge are omitted for clarity; nevertheless, they will be discussed as appropriate. A brief description of the bridge's interface with various main components on both sides is also provided.

### 1.3.2.1 System Side Operation

On the system side, the IPBus Bridge operates at 16-bit controller core frequency and fully supports pipelined communication with the core. The bridge acts as a slave device on this bus. The bridge is responsible for initiating IPBus transactions only per requests initiated by the 16-bit controller core.



**Figure 1-3. IPBus Bridge Interface With Other Main Components, System Side Operation**

### 1.3.2.2 Peripheral Side Operation

On the peripheral side, the IPBus Bridge accesses various devices through a standard non-pipelined IPBus interface. Separate bus lines are used for read and write transactions.

### 1.3.3 Peripheral Interrupts/Interrupt Controller Module

The peripherals on the 56F8300 use the interrupt channels found on the 56800E core. Each peripheral has its own interrupt vector (often more than one interrupt vector for each peripheral), and can selectively be enabled or disabled via the Interrupt Priority Register (IPR) found in the Interrupt Controller (ITCN) module. Design includes these distinctive features:

- Programmable priority levels for each IRQ
- Two programmable Fast Interrupts
- Notification to the SIM module to restart clocks out of Wait and Stop modes

**Note:** Please see the device Data Sheet for detailed information about this module.

### 1.3.3.1 System Integration Module (SIM)

The SIM module is a system catchall for the glue logic tying together the system-on-chip. It controls distribution of resets and clocks and provides a number of control features. The system integration module is responsible for the following functions:

- Reset sequencing
- Clock control & distribution
- STOP/WAIT control
- Pull-up enables for selected peripherals
- System status registers
- Registers for software access to the JTAG ID of the chip
- Enforcing Flash security

**Note:** Please see the device Data Sheet for detailed information about this module.

### 1.3.4 Peripheral Features

#### 1.3.4.1 Analog-to-Digital Converter (ADC)

- 12-bit resolution
- Maximum ADC clock frequency is 5MHz with 200ns period
- Sampling rate up to 1.66 million samples per second<sup>1</sup>
- Single conversion time of 8.5 ADC clock cycles ( $8.5 \times 200\text{ns} = 1.7\mu\text{s}$ )
- Additional conversion time of 6 ADC clock cycles ( $6 \times 200\text{ns} = 1.2\mu\text{s}$ )
- Eight conversions in 26.5 ADC clock cycles ( $26.5 \times 200\text{ns} = 5.3\mu\text{s}$ ) using simultaneous mode
- ADC can be synchronized to the PWM via the SYNC signal
- Simultaneous or sequential sampling
- Internal multiplexer to select two of eight inputs
- Ability to sequentially scan and store up to eight measurements
- Ability to simultaneously sample and hold two inputs
- Optional interrupts at the end of a scan, if an out-of-range limit is exceeded, or at zero crossing
- Optional sample correction by subtracting a pre-programmed offset value
- Signed or unsigned result
- Single-ended or differential inputs

---

1. Once in Loop mode, the time between each conversion is six ADC Clock cycles (2.4 $\mu\text{s}$ ). Samples per second is calculated according to 2.4 $\mu\text{s}$  per sample or 416666 samples per second.

### 1.3.4.2 Computer Operating Properly (COP)

- Programmable timeout period =  $(1024 \times (CT + 1))$  oscillator clock cycles, where CT can be from \$0000 to \$FFFF
- Programmable Wait and Stop modes
- COP timer is disabled while host controller is in Debug mode

### 1.3.4.3 External Memory Interface (EMI)

- Can convert any internal bus memory request to a request for external memory
- Can manage multiple internal bus requests for external memory access
- Has up to eight  $\overline{CS}_n$  configurable outputs for external device decoding
  - each  $\overline{CS}$  can be configured for Program space, Data space, both Program and Data space, or neither (disabled)
  - each  $\overline{CS}$  can be configured for Read only, Write only, or Read/Write access
  - each  $\overline{CS}$  can be configured for the number of Wait states required for device access
  - each  $\overline{CS}$  can be configured for the size and location of its activation
  - each  $\overline{CS}$  is independently configured for setup and hold timing controls for both read and write
- Supports disabling external P-space access if Flash Security mode is enabled on a chip

### 1.3.4.4 On-Chip Clock Synthesis (OCCS or CLKGEN)

- Oscillator can be crystal controlled or driven from an external clock generator
- Two-bit prescaler can divide oscillator output by 1, 2, 4, or 8 prior to its use as the PLL source clock
- Two-bit postscaler provides similar control for the PLL output
- Ability to power down the internal PLL
- Provides 2x master clock frequency and OSC\_CLK signals
- Safety shutdown feature available in the event the PLL reference clock disappears
- Internal relaxation oscillator (not available on all parts)

### 1.3.4.5 Flash Memory (FM)

- Program Flash Memory is interleaved. Please see the part Data Sheet for specifics.
- 8k/16k bytes (chip dependent) of Boot Flash Memory constructed from a single 4k/8k by 16-bit block
- 8k bytes of Data Flash Memory constructed from a single 4k by 16-bit block
- 60MHz single cycle operation for Program Flash access due to interleaved blocks. 30 MHz operation for Boot and Data accesses. All accesses are at 150°C ( $T_J$ ) and 2.25V.

- 8100 family provides 40MHz single cycle operation for Program Flash access due to interleaved blocks. 20 MHz operation for Boot and Data accesses. All accesses are at 105°C (T<sub>J</sub>) and 2.25V.
- Automated program and erase operation
- Read-while-write capability
- Interrupts on command completion, command buffer empty and access error
- Fast page erase
- Single power supply program and erase
- Security feature prohibits flash access from external devices
- Protection feature prohibits accidental program/erase
- Sector protection system for Program, Boot, and Data Flash
- Flash supports byte, word, and long word Data Flash read operations by the host core

#### 1.3.4.6 FlexCAN (FC)

- FlexCAN provides a full CAN interface
- IP Interface Architecture
- Full implementation of the CAN protocol specification - Version 2.0
  - standard data and remote frames (up to 109 bits long)
  - extended data and remote frames (up to 127 bits long)
  - 0-8 bytes data length
  - programmable Bit Rate up to 1Mbit/sec
- Up to 16 flexible Message Buffers (MBs) of 0-8 bytes Data Length, each configurable as CANRx or CANTx, all support Standard and Extended Messages
- Listen-Only mode capability
- Content-related addressing
- No read/write semaphores
- Three programmable mask registers
  - global (for MBs 0-13)
  - special for MB14
  - special for MB15
- Programmable transmit-first 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. External transceiver is assumed

- Open network architecture
- Multimaster concept
- Short latency time for high-priority messages
- Low power Sleep mode, with programmable Wake up on bus activity

#### 1.3.4.7 General Purpose Input/Output (GPIO)

- Individual control for each pin to be in either Normal or GPIO modes
- Individual direction control for each pin in GPIO mode
- Individual pull-up enable control for each pin in either Normal or GPIO modes
- Optimized for use with a keypad interface with push-pull I/O
- Ability to monitor pad logic values even when GPIO are not enabled by using the GPIO\_X\_RAWDATA register
- Interrupt Assert Capability

#### 1.3.4.8 Power Supervisor (PS)

- Power-On Reset is generated until  $V_{DD}$  core exceeds 1.8V
- Low Voltage Interrupt (LVI) is generated when the 2.5V rail drops below 2.2V
- Low Voltage Interrupt (LVI) is generated when the 3.3V rail drops below 2.7V

#### 1.3.4.9 Pulse Width Modulator (PWM)

- Three complementary PWM signal pairs, or six independent PWM signals
- Features of complementary channel operation
  - deadtime insertion
  - separate top and bottom pulse width correction via current status inputs or software
  - separate top and bottom polarity control
- Edge-aligned or center-aligned PWM signals
- 15-bits of resolution
- Half-cycle reload capability
- Integral reload rates from one to 16
- Individual software controlled PWM output
- Programmable fault protection
- Polarity control
- 10/16 mA current source/sink capability on PWM pins
- Write protected registers

#### 1.3.4.10 Quadrature Decoder (DEC)

- Includes logic to decode quadrature signals

- Configurable digital filter for inputs
- 32-bit position counter
- 16-bit position difference register
- Maximum count frequency equals the peripheral clock rate
- Position counter can be initialized by software or external events
- Preload capable 16-bit revolution counter
- Inputs can be connected to a general purpose timer to aid low speed velocity measurements
- Quadrature Decoder filter can be bypassed
- A Watchdog Timer to detect a non-rotating shaft condition

#### **1.3.4.11 Serial Communications Interface (SCI)**

- Full-duplex or single wire operation
- Standard mark/space Non-Return-to-Zero (NRZ) format
- 13-bit baud rate selection
- Programmable 8-bit or 9-bit data format
- Separately enabled transmitter and receiver
- Separate receiver and transmitter 16-bit controller interrupt requests
- Programmable polarity for transmitter and receiver
- Two receiver wake up methods:
  - idle line
  - address mark
- Interrupt-driven operation with seven flags:
  - transmitter empty
  - transmitter idle
  - receiver full
  - receiver overrun
  - noise error
  - framing error
  - parity error
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

#### 1.3.4.12 Serial Peripheral Interface (SPI)

- Full-duplex operation
- Master and Slave modes
- Double-buffered operation with separate transmit and receive registers
- Programmable length transmissions (2 to 16 bits)
- Programmable transmit and receive shift order (MSB first or last bit transmitted)
- Eight master mode frequencies (maximum = module clock  $\div$  2)
- Maximum Slave mode frequency = module clock  $\div$  2
- Clock ground for reduced Radio Frequency (RF) interference
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts
  - SPI Receiver Full (SPRF)
  - SPI Transmitter Empty (SPTE)
- Mode fault error flag interrupt capability
- Wired OR mode functionality enabling connection to multiple SPIs

#### 1.3.4.13 Temperature Sensor (TSENSOR)

- Operating range: -40 to 150°C junction temperature (inclusive, 150°C is a legal value)
- Operation will continue in a linear fashion above 150°C, although the device is only guaranteed to this point
- Supply Voltage: 3.3V  $\pm$ 0.3V
- Power down feature is an optional feature on the IC. Connection to ADC input is done in several different ways and is part and package dependent. The connection methods are
  - internal connection of the sensor output to an ADC input in the package
  - bringing the sensor voltage out to an output pin permitting optional external connection to an ADC input
- Monotonic with temperature
- Resolution is better than 1°C/bit over a 10-bit range from 0 to 3.6 V

#### 1.3.4.14 Quad Timer (TMR)

- Four 16-bit counters/timers
- Capable of counting up and down
- Counters will cascade
- Count modulus can be programmed
- Maximum count rate equals peripheral clock/2 for external clocks
- Maximum count rate equals peripheral clock for internal clocks



- Will count once or repeatedly
- Counters can be preloaded
- Counters can share available input pins
- Separate prescaler for each counter
- Each counter has capture and compare capability

#### **1.3.4.15 Voltage Regulator (VREG)**

- Provide a 2.5V  $\pm$ 10 percent accuracy
- Provide an average current of at least 250mA for the larger regulator
- Provide an average current of at least 1mA for the smaller regulators

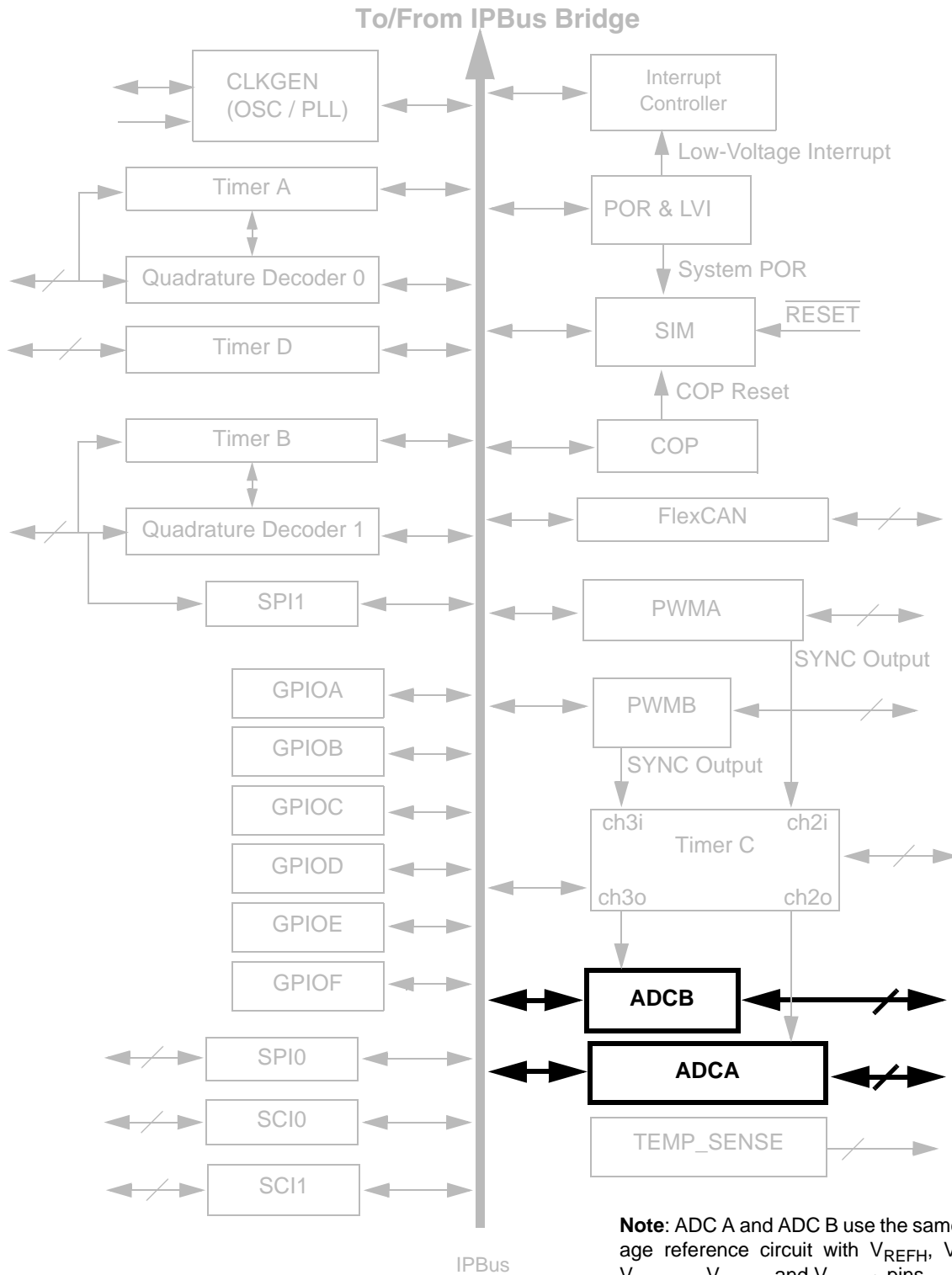
### **1.4 Energy Information**

- Fabricated in high-density CMOS with 5V tolerant, TTL-compatible digital inputs
- On-board 3.3V down to 2.5V voltage regulator for powering internal logic and memories
- On-chip regulators for digital and analog circuitry to lower cost and reduce noise
- Wait and Stop modes available
- ADC smart power management
- Each peripheral can be individually disabled to save power



# Chapter 2

## Analog-to-Digital Converter (ADC)



**Note:** ADC A and ADC B use the same voltage reference circuit with  $V_{REFH}$ ,  $V_{REFP}$ ,  $V_{REFMID}$ ,  $V_{REFN}$ , and  $V_{REFLO}$  pins.

## Document Revision History for **Chapter 2, Analog-to-Digital Converter (ADC)**

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 3.0	<p>Added references to the Device Data Sheet, "Peripheral Subsystems" Figure on pages 6 and 29            Correcting Sections 2.12.13.2 and 2.12.13.4 on page 46 to <math>V_{CAL\_L}</math> and <math>V_{CAL\_H}</math> replacing <math>V_{REFH}</math>            and <math>V_{REFLO}</math></p> <p>Corrected "three" configurations in Introduction to "two" different configurations.            Deleted last bullet on page 2-4. Redundant to the second bullet above.            Corrected Section 2.4 by adding ", rising edge," to the 1st line, 2nd paragraph</p>
Rev 4.0	<p>Corrected 4095 possible states to 4096 on page 2-10.</p> <p>Change ADCR1 and ADCR2 to ADCTL1 and ADCTL2 for consistency.</p> <p>Corrected High limit to <math>0x7FF8</math> in section 2.12.10</p> <p>Corrected description of CRS0/1 in the ADC_CAL register to reference  <math>V_{CAL\_H}/V_{CAL\_L}</math> instead of <math>V_{REFH}/V_{REFLO}</math></p> <p>Added Section 2.10.2 to explain calibration correction factors.</p> <p>Made changes to Table 2-2 and Figure 2-12 to incorporate this into the example code.</p> <p>Grammar edits throughout chapter</p> <p>Corrected names of sections 2.12.12.8 and 2.12.12.9 to            ADC Converter One and Zero</p>
Rev 5.0	<p>Corrected code example in line 6 on page 23</p> <p>Added 8100 clock frequency statement on page 34</p> <p>Added 8100 characteristic statement to table 2-7</p> <p>Added <i>56F8300/56F8100</i> to the second paragraph of Section 2.13.1</p> <p>Added to Section 2.12.6, page 36,...(ADRSLTn) "either by software or the debugger."</p> <p>Corrected equation on page 10 from <math>V_{REMID}</math> to <math>V_{REFLO}</math></p> <p>Chapter conversion to Freescale design</p>
Rev 6.0	Corrected $SAMPLE_n$ to $m$ in Figure 2-4
Rev 7.0	Corrected <b>Section 2.6.1.2</b> , Differential Samples
Rev 8.0	<p>Changed the multiplier in the equation in <b>Section 2.6.1.2</b> from 2048 to 2047.</p> <p>Minor formatting edits.</p>
Rev. 9	Defined the acronym ESR (equivalent series resistance) in Figure 2-16.
Rev. 10	<p>Updated the axis labels and values in <b>Figure 2-9</b>.</p> <p>Clarified the last calculation example in <b>Section 2.10.1</b>.</p>

## 2.1 Introduction

There are two different configurations of the The Analog-to-Digital Converter (ADC) module(s) in this device:

1. A single dual, 12-bit ADC module allowing both Analog-to-Digital Converters share a common voltage reference and control block
2. Two dual, 12-bit ADCs (a total of four converters) where all four ADCs share a common voltage reference, but each dual contains its own control block. For simplicity's sake, this will be referred to as the *quad* throughout this document. However, it is important to stress it is implemented as two dual ADCs, sharing a reference.

**Figure 2-1** illustrates option one while **Figure 2-2** depicts option two, which is simply two instances of option one.

This document only describes the control logic associated with above option one. Option two simply instantiates that same logic a second time for the second dual ADC.

## 2.2 Features

The Analog-to-Digital Converter (ADC) consists of two separate and complete ADCs, each with their own sample and hold circuits. A common digital control module configures and controls the functioning of both ADCs. ADC features are:

- 12-bit resolution
- Maximum ADC clock frequency is 5MHz with 200ns period
- Sampling rate up to 1.66 million samples per second<sup>1</sup>
- Single conversion time of 8.5 ADC clock cycles ( $8.5 \times 200\text{ns} = 1.7\mu\text{s}$ )
- Additional conversion time of 6 ADC clock cycles ( $6 \times 200\text{ns} = 1.2\mu\text{s}$ )
- Eight conversions in 26.5 ADC clock cycles ( $26.5 \times 200\text{ns} = 5.3\mu\text{s}$ ) using Simultaneous mode
- ADC conversions can be synchronized by both the PWM and the TMR
- Simultaneous or sequential sampling with additional text
- Ability to simultaneously sample and hold of two inputs
- Ability to sequentially scan and store of up to eight measurements
- Internally multiplex to select two of eight inputs
- Power savings modes allow automatic shutdown/startup of all or part of ADC

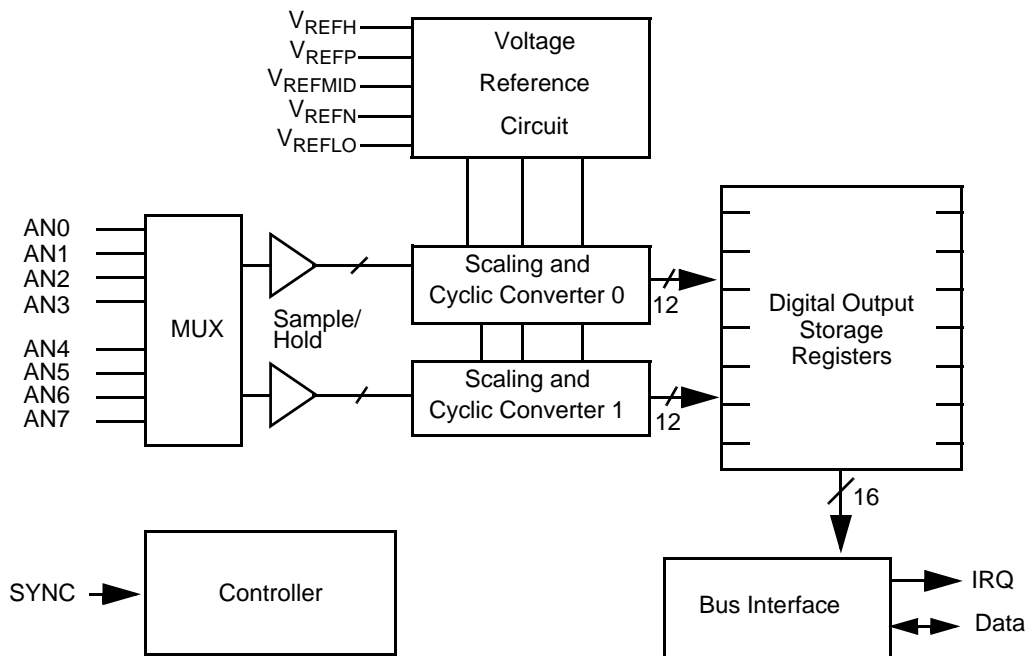
---

1. Once in Loop mode, the time between each conversion is six ADC Clock cycles ( $2.4\mu\text{s}$ ). Samples per second is calculated according to  $2.4\mu\text{s}$  per sample or 416666 samples per second.

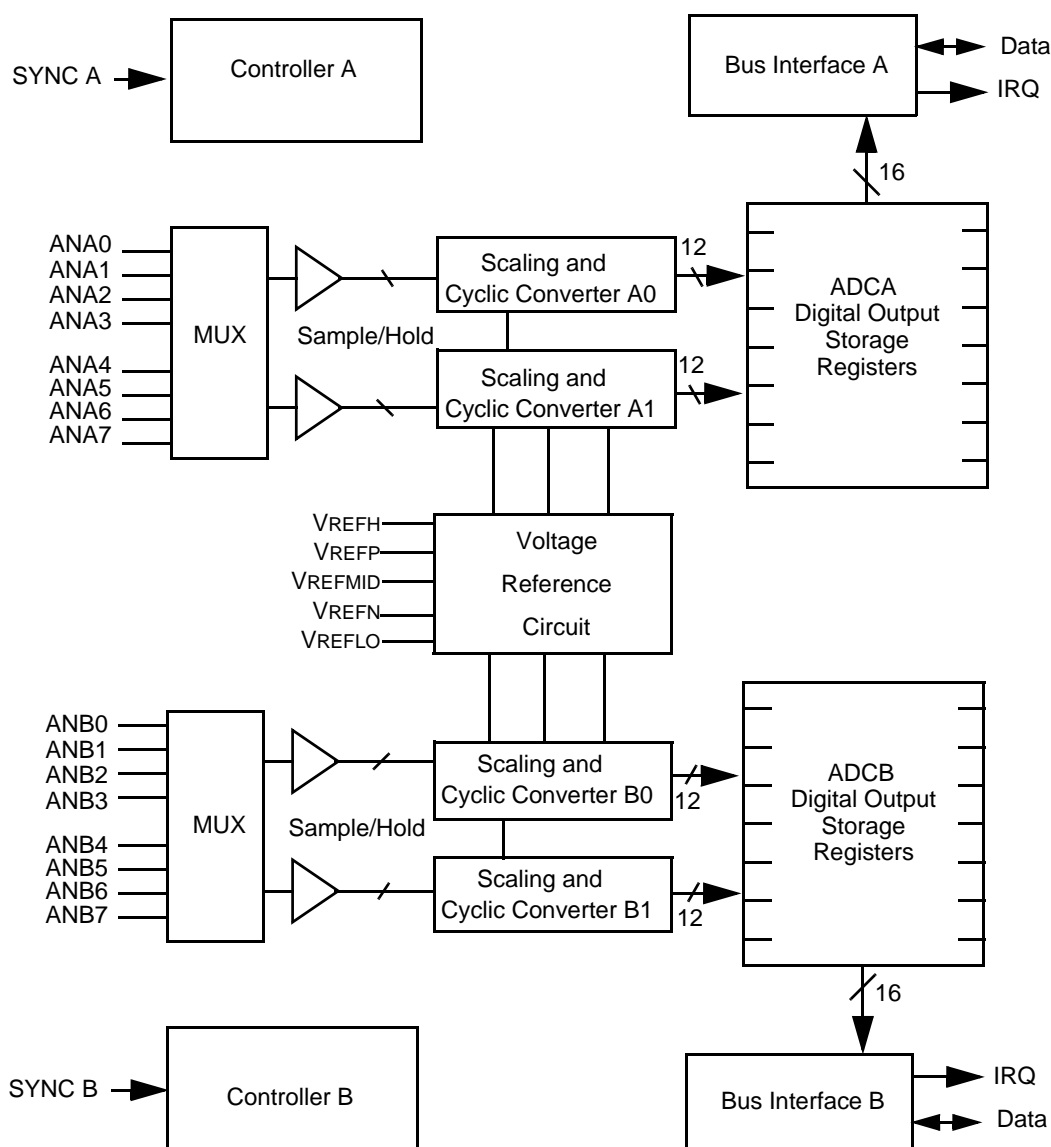
- Built-in calibration using on-chip input voltage network
- Unselected inputs tolerate injected/sourced current without affecting ADC performance, supporting operation in noisy industrial environments. See data sheet's *ADC Parameters* table for more information on current limits.
- Optional interrupts at the end of a scan, if an out-of-range limit is exceeded, (high or low) or at zero crossing
- Optional sample correction by subtracting a pre-programmed offset value
- Signed or unsigned result
- Single ended or differential inputs for all input pins with support for an arbitrary mix of input types

## 2.3 Block Diagram

**Figure 2-1** illustrates the dual ADC configuration, option one, while **Figure 2-2** illustrates quad ADC configuration option.



**Figure 2-1. Option 1: Dual ADC Block Diagram**



**Figure 2-2. Option 2: Two Dual ADCs with a Single Voltage Reference**

## 2.4 Functional Description

The ADC function, illustrated in [Figure 2-1](#), consists of an eight-channel input select function and two independent Sample and Hold (S/H) circuits feeding separate 12-bit ADCs. The two separate converters store their results in an accessible buffer, awaiting further processing by the internal functions.

The conversion process is either initiated by a SYNC signal, rising edge, from one of the on-chip timer channels (see device data sheet, *Figure Peripheral Subsystem*, for the specific timer channel

associated with the ADC) or by a write to the ADC Control Register (ADCR1) START bit. Please see [Section 2.12.1](#).

The rising edge of the SYNC signal or the setting of the START bit begins a sequence of conversions (a scan). A conversion, or scan, can sample and convert up to eight unique single ended channels, up to four differential channel pairs, or any combination of these.

The ADC can be configured to perform a single scan and halt, perform a scan whenever triggered, or perform the programmed scan sequence repeatedly until manually stopped via the STOP bit. Please see [Section 2.12.1.2](#).

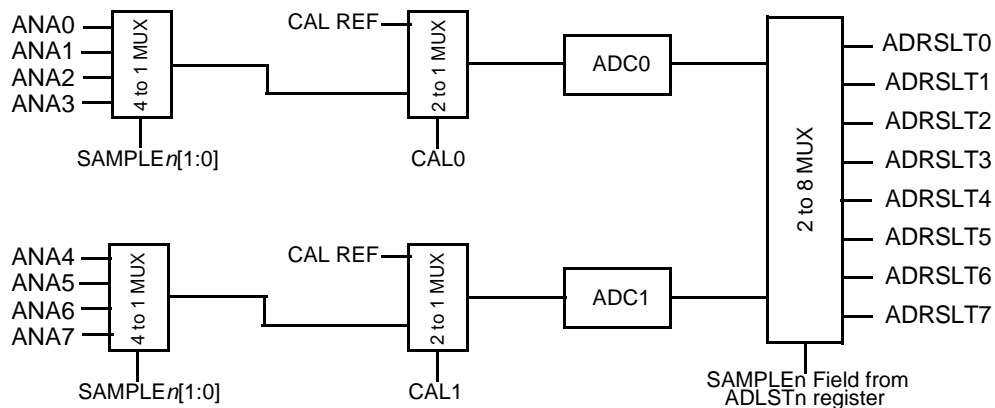
The Dual ADC can be configured for either *sequential* or *simultaneous* conversion. When configured for sequential configuration, up to eight channels, single ended connection, can be sampled and stored in any order specified by the Channel List register. [Figure 2-3](#) illustrates the functional operation of the ADC during a sequential conversion. Notice both ADCs may be required during a scan, depending on the ANA $n$  inputs to be sampled.

[Figure 2-4](#) illustrates the functional operation of the ADC during simultaneous operation. During a simultaneous conversion, both S/H circuits are used to *capture two different channels at the same time*. This configuration requires that a single channel may not be sampled by both S/H circuits simultaneously.

The Channel List register (ADLST1 and ADLST2) is programmed with the scan order of the desired channels. Please see [Section 2.12.4](#).

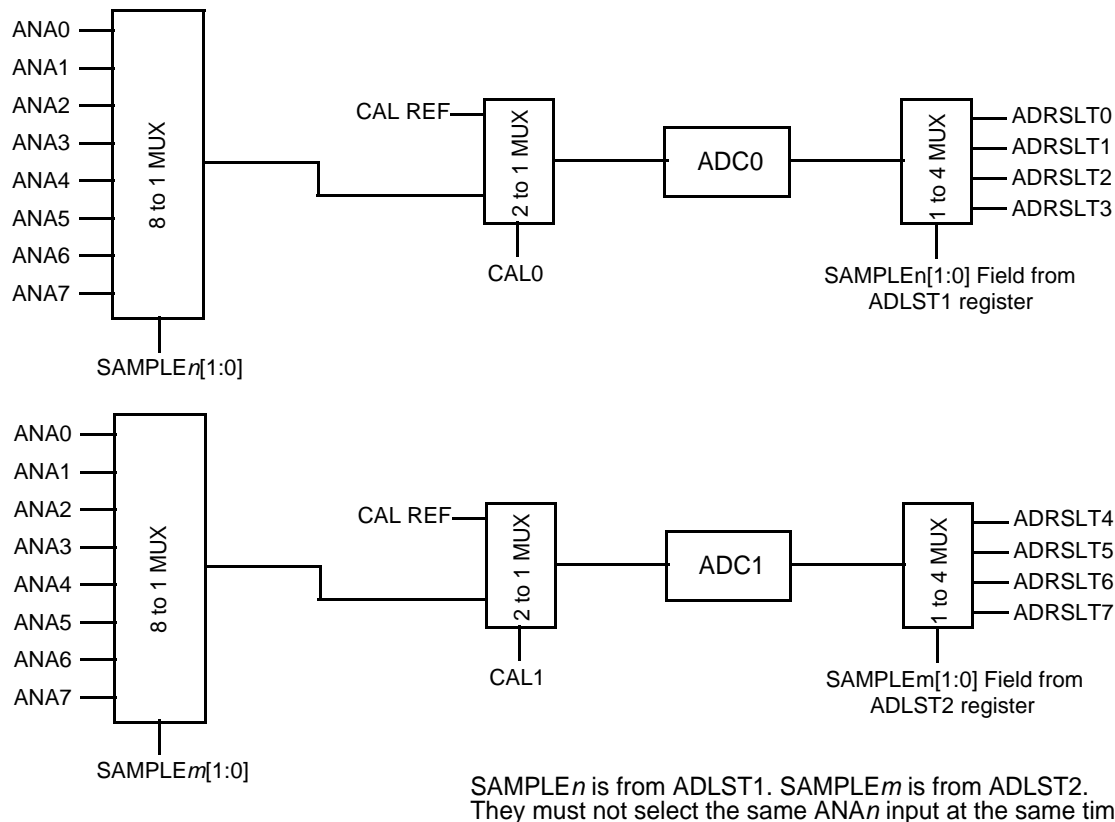
Optional interrupts can be generated at the end of the scan sequence. Optional interrupts will occur if a channel is out of range (measures below the low threshold limit or above the high threshold limit set in the limit registers) or at several different zero crossing conditions.

To understand the operation of the ADC it is important to understand the features and limitations of each of the functional parts.



**Figure 2-3. Sequential and Differential Modes of Operation of the ADC**





**Figure 2-4. Simultaneous Mode Operation of the ADC**

## 2.5 Input MUX Function

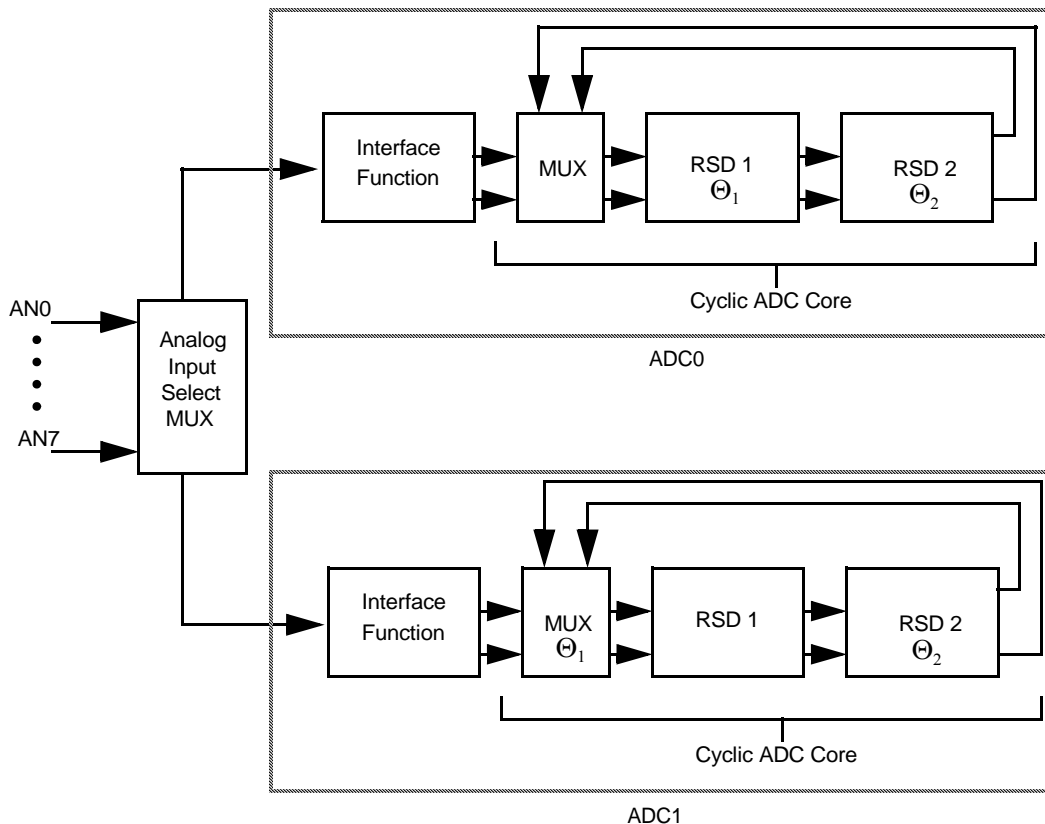
The Input MUX function is illustrated in [Figure 2-3](#) and [Figure 2-4](#). This MUX is able to take eight single ended inputs and provide them to either of the two output channels for ADC processing. The two output channels must never both select the same input or conversation accuracy will suffer. The four functional modes of the MUX and the associated restrictions are described in the following paragraphs.

1. MUXing for Sequential, Single Ended Mode Conversions—[Figure 2-3](#) illustrates either ADC can be used depending on the input selected. If AN0-3 is selected AD0 does the conversion. If AN4-7 is selected ADC1 does the conversion. The output of the appropriate ADC is routed to the Offset, Limit and Result registers.  $(V_{REFH} - V_{REFLO})/2$  is selected for the differential input to both ADCs.
2. MUXing for Sequential, Differential Mode Conversions—[Figure 2-3](#) illustrates the MUX operation as in the single ended case with the even number input routed to the ADC input and the odd numbered input used as the differential input to the ADC.

3. MUXing for Simultaneous, Single Ended Mode Conversions—**Figure 2-4** illustrates the process during the conversion cycle. Any input can be used by ADC0 and any other input can be selected for use by ADC1.  $(V_{REFH}-V_{REFLO})/2$  is selected for the differential input to both ADCs.
4. MUXing for Simultaneous, Differential Mode Conversions—**Figure 2-3** illustrates in this mode AN0 or AN2 is selected as the input to ADC0 with AN1 or AN3 respectively selected as its differential input. Similarly, AN4–AN7 are restricted to ADC1.

## 2.6 ADC Sample Conversion Operation Modes

The ADC consists of a cyclic, algorithmic architecture using two recursive sub-ranging sections (RSD1 and RSD2), illustrated in **Figure 2-5**. Each sub-ranging section resolves a single bit in each conversion clock, resulting in an overall conversion rate of two bits per clock cycle. Each sub-ranging section is designed to run at a maximum clock speed of 5MHz so a complete 12-bit conversion can be accommodated in 1.2 $\mu$ s, not including sample or post processing time.



**Figure 2-5. Cyclic ADC – Top Level Block Diagram**

## 2.6.1 Normal Operating Mode

The ADC has two modes of normal operation. The mode of operation for a given pair of analog inputs is determined by the corresponding bit of CHNCFG field of the ADCR1. Please see [Section 2.12.1.9](#). The two operating modes are:

1. Single Ended Mode (CHNCFG bit = 0)—In the Single Ended mode, input mux of the ADC selects one of the eight analog inputs and directs it to the plus terminal of the ADC core. The minus terminal of the ADC core is connected to the  $V_{REFLO}$  reference during this mode. The ADC measures the voltage of the selected analog input and compares it against the  $(V_{REFH} - V_{REFLO})$  reference voltage range.
2. Differential Mode (CHNCFG bit = 1)—In the Differential mode, the ADC measures the voltage difference between two analog inputs and compares that against the  $(V_{REFH} - V_{REFLO})$  voltage range. The input is selected as an input pair: AN0/1, AN2/3, AN4/5 or AN6/7. In this mode, the plus terminal of the ADC core is connected to the even analog input while the minus terminal is connected to the odd analog input.

A mix and match combination of single ended and differential configurations may exist. For example:

- AN0 and AN1 differential, AN2 and AN3 single ended
- AN4 and AN5 differential, and AN6 and AN7 single ended

### 2.6.1.1 Single Ended Samples

The ADC module performs a ratio metric conversion. For single ended measurements, the digital result is proportional to the ratio of the analog input to the reference voltage in the following diagram.

$$\text{Single Ended Value} = \text{round}\left(\frac{V_{IN} - V_{REFLO}}{V_{REFH} - V_{REFLO}} \times 4095\right) \times 8$$

$V_{IN}$  = Applied voltage at the input pin

$V_{REFH}$  = Voltage Reference, High  
=  $V_{DD}$  (in most cases)

$V_{REFLO}$  = Voltage Reference, Low  
=  $V_{SS}$  (In most cases)

**Note:** The 12-bit result is rounded to the nearest LSB.

**Note:** The ADC is a 12-bit function with 4096 possible states. However, the 12 bits have been left shifted three bits on the 16-bit data bus so its magnitude, as read from the data bus, is now 32760.

### 2.6.1.2 Differential Samples

For differential measurements, the digital result is proportional to the ratio of the difference in the inputs to the difference in the reference voltages ( $V_{REFH}$  and  $V_{REFLO}$ ). [Figure 2-6](#) shows typical configurations for differential inputs.

When converting differential measurements, the following formula is useful:

$$\text{Differential Value} = \text{round}\left\{\left(\frac{V_{IN+} - V_{IN-}}{V_{REFH} - V_{REFLO}} * 2047\right) + 2048\right\} * 8$$

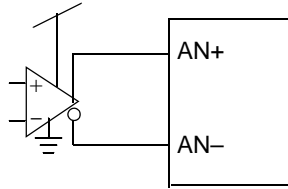
$V_{IN}$  = Applied voltage at the input pin

$V_{REFH}$  and  $V_{REFLO}$  = Voltage at the external reference pins on the device  
(typically  $V_{REFH} = V_{SSA}$  and  $V_{REFLO} = V_{DDA}$ )

**Note:** The 12-bit result is rounded to the nearest LSB.

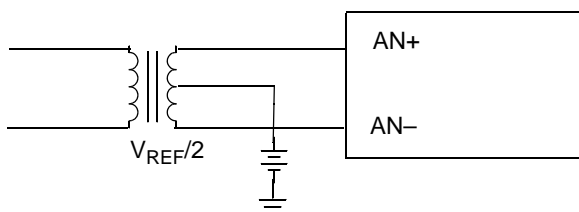
**Note:** The ADC is a 12-bit function with 4096 possible states. However, the 12 bits have been left shifted three bits on the 16-bit data bus so its magnitude, as read from the data bus, is now 32760.

$V_{REFH}$  Potential



Differential Buffer will center about mid point.

**Note:** Normally,  $V_{REFLO}$  is set to  $V_{SSA}=0V$ .



Center tap held at  $(V_{REFH} + V_{REFLO}) / 2$

**Figure 2-6. Typical Connections for Differential Measurements**

## 2.7 ADC Data Processing

As shown in [Figure 2-7](#), the result of the ADC conversion process is normally sent to an adder for offset correction. The adder subtracts the ADC Offset (ADOFS) register value from each sample and the resultant value is then stored in an ADC Result (ADCRSLT1-7) register. At the same time, the raw ADC value and the ADRSLT values are checked for limit violations and zero-crossing, as shown. Appropriate interrupts are asserted, if enabled.

The result value sign is determined from the ADC unsigned result minus the respective offset register value (ADCOFS1-7). If the offset register is programmed with a value of zero, the result register value is unsigned and equals the cyclic converter unsigned result. The range of the ADRSLT register is \$0000–\$7FF8 assuming the ADOFS $_n$  register is set to all zeros. This is equal to the raw value of the ADC core.

Each result register may only be modified by the processor when the ADC is in Stop or Power-Down mode; that is, when the STOP bit in ADCR1 is set to one or *both* PD0 and PD1 are set to one in the ADC power (ADCPOWER) register. This write operation is treated as if it came from the ADC analog core; therefore, the limit checking, zero crossing, and the offset registers function identically as when in normal operation. For example, if the STOP bit is set to one and the processor writes to ADRSLT5, the data written to the ADRSLT5 is muxed to the ADC digital logic inputs, processed and stored into ADRSLT5 as if the analog core had provided the data.

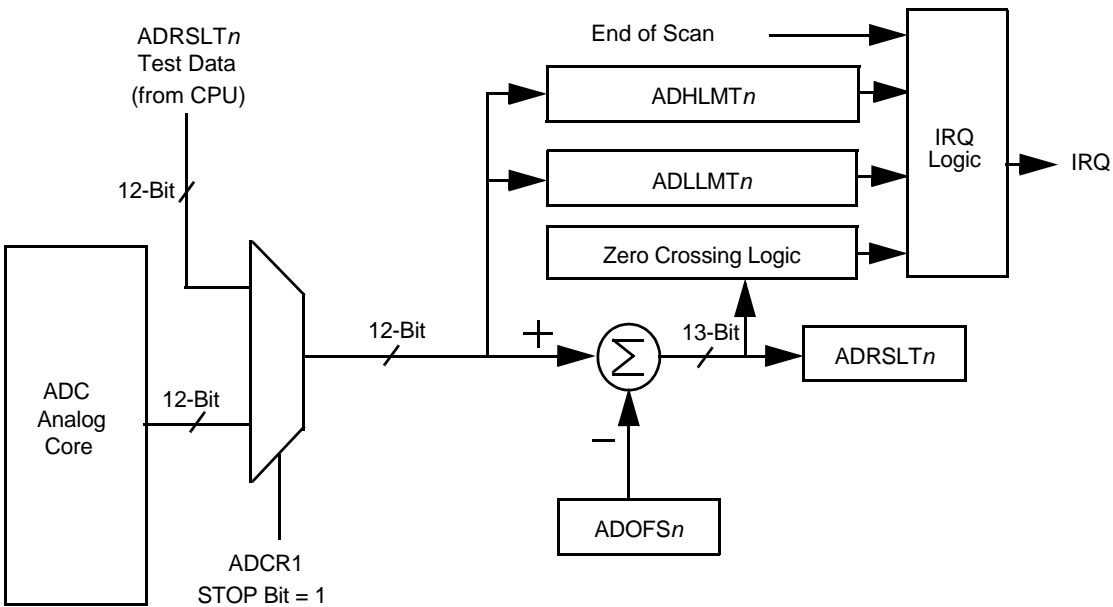


Figure 2-7. Result Register Data Manipulation

## 2.8 Sequential Vs. Simultaneous Sampling

The ADC can be configured for either sequential or simultaneous conversion. When configured for sequential conversion only one of the two ADCs operates at any time. A scan sequence of up to eight conversions can be specified. Each of the conversions can select any of the eight input channels for conversion, with the result stored in the consecutive result registers. Each conversion can be either single ended or differential.

When configured for simultaneous conversion both of the ADCs operate in parallel. A scan sequence of up to four conversion can be specified. Each of the conversions can select any of the eight inputs as long as both ADCs are not sampling the same channel. The conversion results are stored, two at a time, in consecutive result register pairs. The first conversion is stored in ADRSLT0 and ADRSLT4. The next conversion pair is stored in ADRSLT1 and ADRSLT5, and so on. Each conversion can be either single ended or differential, in any combination.

## 2.9 Scan Sequencing

The conversion process is either initiated by a SYNC signal from one of the on-chip timer channels or by a write to the ADCR1 START bit. See the chip's data sheet for the timer channel associated with the ADC SYNC input(s). Once a conversion process is initiated, a sequence of up to eight analog-to-digital conversion(s) is begun.

If the scan parameter indicates more than one conversion, a second conversion is immediately initiated following the completion of the first conversion. The number of conversions in each scan is controlled via the Disable Sample (DS) field of the ADSDIS register, illustrated in the first row of [Table 2-1](#). The sequence of channels to convert is controlled by the SAMPLE<sub>n</sub> fields of the ADLST1 and ADLST2 registers.

If a scan is initiated while another scan is in process, the start signal is ignored until the active conversion scan is complete, i.e. until the Conversion In Process, (CIP), bit in ADC Status (ADSTAT) register has changed from 1 to 0.

**Table 2-1. ADC Scan Sequence Control**

Scan Feature	Control	Sequential Mode	Simultaneous Mode
Scan Length Control	Disable Sample (DS) Field of ADSDIS Register	Starting with DS0, each bit is checked in sequence, and a conversion is started, until a 1 bit is found. This stops the conversion scan. If all eight bits are enabled (0) the scan is completed after eight analog-to-digital conversions.	Starting with DS0/4, each pair of bits is checked in sequence, and a conversion is started, until a 1 bit is found in either bit of the corresponding part of the ADSDIS register. This stops the conversion scan. If all eight bits are enabled (0) the scan is completed after four analog-to-digital conversions.
Scan Sequencing	SAMPLE <sub>n</sub> Field of ADLST1 and ADLST2 Registers	Starting with SAMPLE0, assuming DS0=0, read SAMPLE0 to determine which channel (channel pair if differential conversion) to route to the analog-to-digital input. Proceed with SAMPLE1 through SAMPLE7, as indicated by the ADSDIS register.	Starting with SAMPLE0 and SAMPLE4, assuming DS0=0 and DS4=0, read SAMPLE0 and SAMPLE4 to determine which channel (channel pair if differential conversion) to route to the analog-to-digital inputs. Proceed with SAMPLE1/5 through SAMPLE3/7, as indicated by the ADSDIS register.
Scan Looping	SMODE Field of ADCR1	Scan once, continuously (loop), or when triggered	
Interrupts	EOSIE, ZCIE, LLMTIE, HLMTIE Bits in the ADCR1	End of Scan Interrupt — except in Loop Continuously mode Zero Crossing Interrupt — + to -, - to +, or both Low Limit Interrupt — out of range on low side High Limit Interrupt — out of range on high side	

## 2.9.1 Low Power Operating Mode

Power-down mode is manually controlled via PD[2:0] and PUDELAY. Please see [Section 2.12.12](#). Power Savings mode provides an automated way for the chip to dynamically control ADC power-up/down status. Only one of Power-down or Power Savings modes can be active at a given time.

### 2.9.1.1 Power-Down

The analog core of the ADC can be shut down for reduced power consumption when the ADC module is not being used. In the Low Power mode current of the ADC is  $<1\mu\text{A}$ . The ADC analog core can be powered down by setting the PD0, PD1, and PD2 bits of the ADPOWER register. Please see [Section 2.12.12](#). When set to one, the PD $n$  bits power-down the specified portion of the analog core of the ADC.

Any conversions in progress by the affected ADC converter will be aborted when PD0/PD1/PD2 are asserted.

**Note:** Affected result registers may be corrupted if the power-down operation occurs late in the conversion sequence, when those registers are normally updated. Otherwise they will retain their previous value.

In a dual ADC configuration, if all of PD0, PD1 and PD2 are set, then the voltage reference is also powered down. In a quad ADC configuration, these bits must be set for both dual ADCs in order for the voltage reference to be powered down.

**Note:** A wait of at least 25msec is required after powering up a voltage reference prior to initiating a data conversion. Failure to do so will result in incorrect conversion results.

**Note:** A delay of PUDELAY ADC clocks is required after powering up an ADC converter prior to initiating a data conversion. The PUDELAY is built into the ADC state machine, providing freedom from concern other than being aware of the timing.

When the ADC reference is powered down the output reference voltages are set to Low ( $V_{SSA}$ ) and the ADC data output is driven low. The ADC analog core is powered up (PD $n$  = 0) on reset.

Any attempt to perform conversions while a converter is powered down will result in undefined behavior. It is possible to read ADC Result registers after converter power-down for results calculated before power-down.

Power-Down modes are cleared by resetting PD0, PD1 and/or PD2 to zero. Allow an amount of time equivalent to 13 ADC clocks or greater (specified in the PUDELAY field) after clearing PD0 and/or PD1 before beginning conversions again (assuming the voltage reference was not powered down).

Coming out of Low Power mode, the ADC does not retain any history of the last conversion completed before power-down. A new scan sequence must be started with a SYNC pulse or a write to the START bit before valid results are available.

**Note:** Powering up the ADC voltage references usually takes much longer than powering up the ADC. Additional delay must be provided, allowing  $V_{REFP}$ ,  $V_{REFMID}$ ,  $V_{REFN}$  to stabilize. If the bypass capacitors on these pins are not fully charged, ADC accuracy is



adversely affected. This delay is usually longer than the number of ADC\_CLK cycles specified by PUDELAY for ADC power-up.

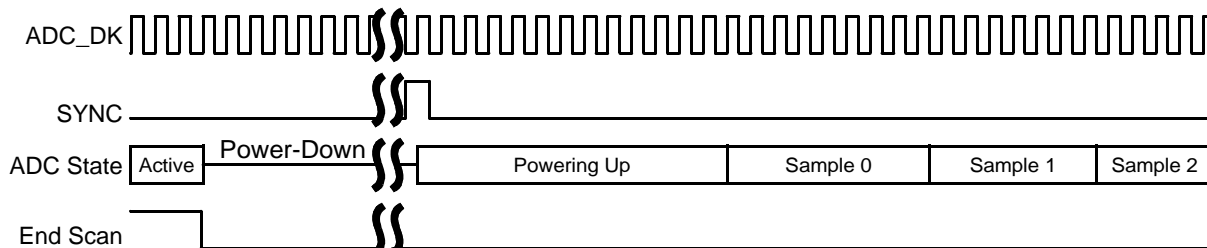
Asserting any of PD0, PD1 or PD2 will clear Power Saving Modes (PSM). Please see [Figure 2-8](#). If PSM was at a logic 1 and only one of PD0 or PD1 are asserted, then the other converter will power-up after a delay specified by PUDELAY.

### 2.9.1.2 Power Savings Mode

The Power Savings Mode (PSM) allows for automatic power-down/up of the ADC core, on an as needed basis, illustrated in [Figure 2-8](#). The PSM feature can yield significant power savings when used correctly.

When an ADC conversion scan is completed the ADC core is automatically powered down. When the next SYNC pulse (or write of 1 to the START bit of the ADCR1) occurs, the ADC core is automatically powered up. ADC clock cycles are counted to provide for this power-up delay and then another ADC scan sequence is started. SYNC pulses and writes to Start during a conversion sequence are ignored.

ADCs are powered up in this mode only if needed for the programmed conversion. If only one ADC converter is required, then only that ADC converter is powered up.



**Figure 2-8. Power Control Features of PSM Operation**

The voltage reference circuit remains powered up in Power Savings mode. This is because it normally requires 25msec for the reference voltages to stabilize after power-up.

If PSM is asserted while a conversion is in progress, that conversion is unaffected and the ADC will wait to enter PSM until the conversion is complete. Contrast this to Power-Down mode, where conversions in progress on the affected converter(s) are aborted when PD0/1 are asserted.

PSM does not apply to Loop modes. Because the ADC does not stop once Loop mode is started, PSM is useful only in Once/Trigger mode.

## 2.9.2 ADC STOP Operating Mode

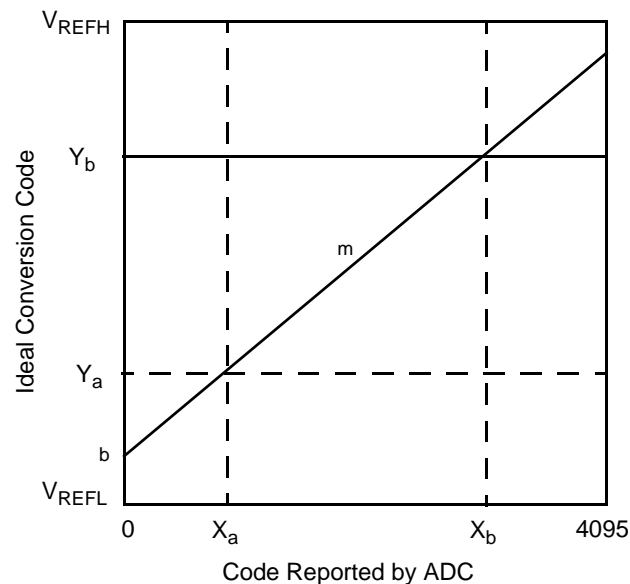
Any conversion sequence in progress can be stopped by setting the ADC STOP bit in the ADCR1. Any further SYNC pulses or writes to the START bit are ignored until the ADC STOP bit is cleared. After the ADC is in ADC Stop mode, the results registers can be modified by writes from the processor. Any write to the result register in the ADC Stop mode is treated as if the analog core supplied the data. Therefore, limit checking, zero crossing, and associated interrupts can occur if enabled.

## 2.10 Calibration

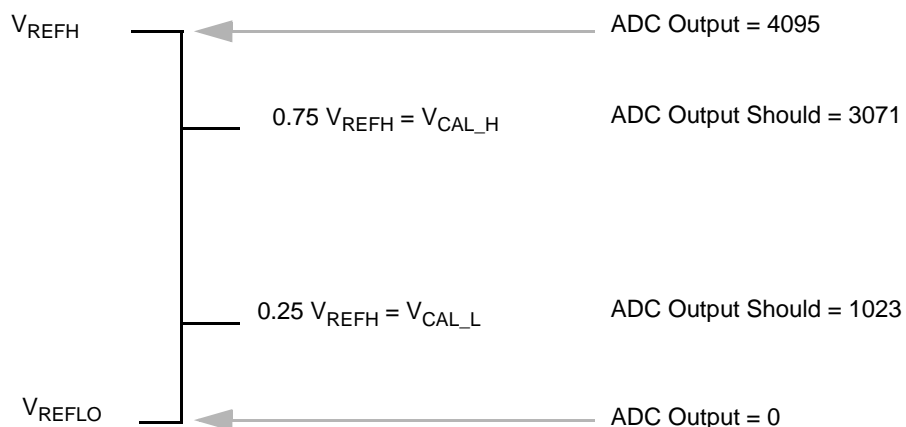
### 2.10.1 Calibration Overview

Any ADC will typically incorporate a gain and offset error that will affect its accuracy. This is illustrated in [Figure 2-9](#). To compensate for this, the ADC has a built in calibration feature which provides either of two known inputs to be applied to the ADC. This is illustrated in [Figure 2-10](#), where the ideal conversion codes are indicated. Based on the ADC results after converting these two reference input voltages, the gain and offset errors can be determined. With appropriate software action, these errors can be compensated for in the user code.

**Note:** The calibration described here can only correct for on-chip errors. It does NOT compensate for any errors due to the off-chip design.



**Figure 2-9. ADC Gain and Offset Error**



**Figure 2-10. ADC Calibration References**

Calibration is typically used to remove the effects of gain and offset from a specific reading. The transfer function relating an ideal code to a measured code, gained and offset, follows:

$$Y = mX + b$$

Where  $X$  is the measured code corresponding to a specific input voltage,  $m$  is the transfer gain of the ADC,  $Y$  is the ideal code corresponding to this input voltage, and  $b$  is the code offset.

The transfer equations for the resulting conversion code for node  $V_{CAL\_H}$  and node  $V_{CAL\_L}$  are:

$$Y_{CAL\_H} = m \times X_{CAL\_H} + b \quad \text{and} \quad Y_{CAL\_L} = m \times X_{CAL\_L} + b$$

Using these two equations, the ADC transfer gain is:

$$m = \frac{Y_{CALH} - Y_{CALL}}{X_{CALH} - X_{CALL}}$$

The offset code is:

$$b = Y_{CAL\_H} - m \times X_{CAL\_H} \quad \text{or} \quad b = Y_{CAL\_L} - m \times X_{CAL\_L}$$

**Note:** Using either equation should result in the same answer.

Once  $m$  and  $b$  are determined, any ADC measurement can be corrected to the ideal value.

**Note:** To obtain good accuracy, when converting the calibration node voltages, make multiple measurements, averaging the result.

Example:

The voltage at nodes  $V_{CAL\_H}$  and  $V_{CAL\_L}$  were converted by the ADC and found to be 3020 and 980. The ideal codes resulting from the conversion of the voltage are illustrated in [Figure 2-10](#).

Therefore the transfer gain  $m$  is:

$$m = \frac{Y_{CALH} - Y_{CALL}}{X_{CALH} - X_{CALL}} = \frac{3071 - 1023}{3010 - 990} = \frac{2048}{2020} = 1.0138$$

The code offset  $b$  is:  $3071 - (3010) \times 1.01386 = 19.0$

When this ADC is converting an input signal, it produces a code of 2000. Correcting this result, using the offset and gain derived above yields:

$$Y = m \times X + b = 1.01386 \times 2000 + 19 = 2047$$

## 2.10.2 Calibration Correction Factors

The following is the ADC calibration equation used for these devices:

$$Y = (m + cf1) \times X + (b + cf2)$$

where  $cf1$  is a correction factor for the transfer gain of the ADC and  $cf2$  is a correction factor for the code offset of the ADC. The equation above is a variation of the equation for a straight line,  $Y = mX + b$ . Correction factors  $cf1$  and  $cf2$  have been introduced into the equation to account for the slight difference in voltage which the ADC sees at its reference input depending on whether it is using the internal 1/4-scale, 3/4-scale reference (special calibration mode), or the external voltage reference (normal operating mode). The parasitic resistances associated with the internal and external voltage references are not exactly the same. Therefore, the values of  $m$  and  $b$  (as calculated above) which were based on conversions using the internal reference are not the same values that would be obtained if the calculations of  $m$  and  $b$  were based on conversions using an external reference. The two correction factors ensure that the equation above yields the correct result for a conversion done in the ADC's normal operating mode (using the external reference) even though the determination of  $m$  and  $b$  was based on conversions done in the ADC's calibration mode (using the internal reference).

The values of  $cf1$  and  $cf2$  are unique to each chip and can be found in the ADC Parameters Table of the device Data Sheet.

See Figure for example code incorporating the correction factors into the  $m$  and  $b$  values.

### 2.10.3 Calibration Procedure

ADC calibration can be accomplished in either Simultaneous or Sequential mode. [Figure 2-3](#) and [Figure 2-4](#) show the calibration mux to illustrate where the calibration references are used in the ADC design. The calibration procedure is described in [Table 2-2](#), with coding examples shown in [Figure 2-11](#) and [Figure 2-12](#) respectively.

There are a couple of subtle differences between sequential mode and simultaneous mode calibration that the user should be aware of. Both are documented below. Simultaneous mode is recommended since it takes a total of two conversion times; whereas sequential mode calibration will required a total of four conversion times.

Once the calibration data has been obtained, [Figure 2-13](#) show how this data may be used to determine  $m$  and  $b$ . Note the example code does not average the calibration data as suggested. This is left as an exercise. [Figure 2-14](#) illustrates how subsequent ADC data is adjusted based on the calibration data obtained in [Figure 2-13](#).

**Table 2-2. ADC Calibration Procedure**

<b>STEP</b>	<b>Simultaneous Operation (See Figure 2-11)</b>	<b>Sequential Operation (See Figure 2-12)</b>
1	Save appropriate registers so they can be redefined for calibration and then restored.	
2	Stop the current ADC operation and configure for single ended simultaneous operation.	Stop the current ADC operation and configure for single ended sequential operation.
3	Set the calibration bit (CAL $n$ ) in the calibration register. Also configure high/low reference for calibration.	
4	Set the sample disable register to enable samples 0 and 4.	Set the sample disable register to enable samples 0 and 1. The programming of the ADLST1 register controls the analog-to-digital that is calibrated. Assuming that ADC0 and then ADC1 are to be calibrated (in that order), the ADLST1 register must have sample 0 programmed for channel 0-3 and sample 1 programmed for channel 4-7.
5	Start the ADC so a calibration result is obtained.	
6	Wait for the ADC conversion to complete.	
7	Clear the End of Scan (EOSI) flag	
8	Read the ADC_0 calibration result from the ADRSLT0 register. Read the ADC_1 calibration result from the ADRSLT4 register.	Read the ADC_0 calibration result from the ADRSLT0 register. Read the ADC_1 calibration result from the ADRSLT1 register.
9	Change the CAL register so the other calibration reference inputs are selected.	
10	Start the ADC so a calibration result is obtained.	
11	Wait for the ADC conversion to complete.	
12	Clear the End of Scan (EOSI) flag	
13	Read the ADC_0 calibration result from the ADRSLT0 register. Read the ADC_1 calibration result from the ADRSLT4 register.	Read the ADC_0 calibration result from the ADRSLT0 register. Read the ADC_1 calibration result from the ADRSLT1 register.
14	Set the CAL register to normal operation is restored.	
15	Restore the ADC registers to their state when we started. We assume a Hardware sync signal will trigger the next ADC conversion. We also assume that our task completed before the Hardware sync occurs.	

```

    UInt16 low_ADC_cal_0;    // Define calibration reading variables in global
    UInt16 low_ADC_cal_1;    scope
    UInt16 high_ADC_cal_0;
    UInt16 high_ADC_cal_1;

Step | static void Calibrate ADC_simul(void)
    | {
    |     UInt16     save_ADCR1, save_ADSDIS;
    |
    |     1     save_ADCR1 = getReg( ADCA_ADCR1 );
    |           save_ADSDIS = getReg( ADCA_ADSDIS );
    |
    |     2     /* ADCA_ADCR1: ??=0,STOP=1,START=0,SYNC=0,EOSIE=0,ZCIE=0,LLMTIE=0,HLMTIE=0,
    |           CHNCFG=0,??=0,SMODE=1 */
    |           setReg(ADCA_ADCR1,0x4001);    // stop the current ADC operation,
    |                                           // set single ended mode,
    |                                           // once simultaneous operation
    |
    |           clrRegBit( ADCA_ADCR1, STOP );    // clear the stop bit so we can do
    |                                           //calibration
    |
    |     3     setReg( ADCA_CAL, 0x7 );    // cal_0 high, cal_1 low reference
    |
    |     4     setReg( ADCA_ADSDIS, 0xEE );    // enable Sample0 and Sample4, only
    |
    |     5     setRegBit( ADCA_ADCR1, START ); // single ended conversions
    |
    |     6     while ( !getRegBit( ADCA_ADSTAT, EOSI ) ); // wait for conversion to
    |           // complete
    |
    |     7     setReg(ADCA_ADSTAT,0x0800);    /* Clear EOSI flag */
    |     8     high_ADC_cal_0 = getReg( ADCA_ADRSLT0 );
    |           low_ADC_cal_1 = getReg( ADCA_ADRSLT4 );
    |
    |     9     setReg( ADCA_CAL, 0xD );    // cal_0 low, cal_1 high reference
    |
    |    10     setRegBit( ADCA_ADCR1, START ); // single ended conversions
    |
    |    11     while ( !getRegBit( ADCA_ADSTAT, EOSI ) ); // wait for conversion to
    |           // complete
    |
    |    12     setReg(ADCA_ADSTAT,0x0800);    /* Clear EOSI flag */
    |    13     low_ADC_cal_0 = getReg( ADCA_ADRSLT0 );
    |           high_ADC_cal_1 = getReg( ADCA_ADRSLT4 );
    |
    |    14     setReg( ADCA_CAL, 0 );    // return to normal ADC operation
    |
    |    15     setReg( ADCA_ADSDIS, save_ADSDIS ); // restore registers to previous mode of
    |           setReg( ADCA_ADCR1, save_ADCR1 ); // operation
    | }

```

**Figure 2-11. Code for Simultaneous Mode ADC Calibration**

```

    UInt16 low_ADC_cal_0; // Define calibration reading variables in global
    UInt16 low_ADC_cal_1; scope
    UInt16 high_ADC_cal_0;
    UInt16 high_ADC_cal_1;

static void CalibrateADC_sequential( void )
{
    UInt16 save_ADCR1, save_ADSDIS, save_ADLST1;

1   save_ADCR1 = getReg( ADCA_ADCR1 );
    save_ADSDIS = getReg( ADCA_ADSDIS );
    save_ADLST1 = getReg( ADCA_ADLST1 );

2   /* ADCA_ADCR1: ??=0,STOP=1,START=0,SYNC=0,EOSIE=0,ZCIE=0,LLMTIE=0,
        HLMTIE=0,CHNCFG=0,??=0,SMODE=0 */
    setReg(ADCA_ADCR1,0x4000); //stop the current ADC operation
                                // set single ended mode,
                                // once sequential operation

    clrRegBit( ADCA_ADCR1, STOP ); // clear the stop bit so we can do
                                    calibration

3   setReg( ADCA_CAL, 0x7 ); // cal_0 high, cal_1 low reference

4   setReg( ADCA_ADSDIS, 0xFC ); // enable Sample0 and Sample1, only

    setReg( ADCA_ADLST1, 0x40 ); //for ADC 0 and then ADC 1

5   setRegBit( ADCA_ADCR1, START ); // single ended conversions

6   while ( !getRegBit( ADCA_ADSTAT, EOSI ) ); // wait for conversion to
        // complete

7   setReg(ADCA_ADSTAT,0x0800); /* Clear EOSI flag */
8   high_ADC_cal_0 = getReg( ADCA_ADRSLT0 );
    low_ADC_cal_1 = getReg( ADCA_ADRSLT1 );

9   setReg( ADCA_CAL, 0xD ); // cal_0 low, cal_1 high reference

10  setRegBit( ADCA_ADCR1, START ); // single ended conversions

11  while ( !getRegBit( ADCA_ADSTAT, EOSI ) ); // wait for conversion to
        // complete

12  setReg(ADCA_ADSTAT,0x0800); /* Clear EOSI flag */
13  low_ADC_cal_0 = getReg( ADCA_ADRSLT0 );
    high_ADC_cal_1 = getReg( ADCA_ADRSLT1 );

14  setReg( ADCA_CAL, 0 ); // return to normal ADC operation

15  setReg( ADCA_ADSDIS, save_ADSDIS ); // restore registers to previous
    setReg( ADCA_ADCR1, save_ADCR1 ); // mode of operation
    setReg(ADCA_ADLST1, save_ADLST1);
}

```

**Figure 2-12. Code for Sequential Mode ADC Calibration**



```

#define Ya    (1023 << 3)
#define Yb    (3071 << 3)

//For "m" correction, need fractional representation of data sheet value / 2
#define CF1_CORRECTION ( DATA_SHEET_PARAMETER_CF1 * 16384 / 32768 )
// For "b" correction, shift data sheet parameter by 3 bits to align with RESULT reg.
#define CF2_CORRECTION ( (INT) ( DATA_SHEET_PARAMETER_CF2 * 8 ) )

extern UInt16  low_ADC_cal_0;      // Reference calibration reading variables
extern UInt16  low_ADC_cal_1;      //      in global scope.
extern UInt16  high_ADC_cal_0;
extern UInt16  high_ADC_cal_1;

UInt16 cal_m_0, cal_m_1, cal_b_0, cal_b_1; // Define calibration parameter variables
                                           //      in global scope
static void CalibrateADC_m_and_b( void )
{
    Int16 Xa_ADC0, Xb_ADC0, Xa_ADC1, Xb_ADC1;

    Xa_ADC0 = low_ADC_cal_0;
    Xb_ADC0 = high_ADC_cal_0;

    Xa_ADC1 = low_ADC_cal_1;
    Xb_ADC1 = high_ADC_cal_1;

    // Ideal slope is 1.0 which is the limit of what can be represented with a fractional
    // number. We need to be able to represent something larger than 1.0, so we shift
    // the numerator down the bus by 1 bit, which moves the decimal point of the result
    // of the division. This allows us to represent a slope of m < 2.0
    //
    // cal_m_0 = ( Yb - Ya ) / ( Xb_ADC0 - Xa_ADC0 );
    cal_m_0 = (UInt16) div_s( ( Yb - Ya ) >> 1, ( Xb_ADC0 - Xa_ADC0 ) );
    cal_m_0 = cal_m_0 + CF1_CORRECTION;

    // cal_m_1 = ( Yb - Ya ) / ( Xb_ADC1 - Xa_ADC1 );
    cal_m_1 = (UInt16) div_s( ( Yb - Ya ) >> 1, ( Xb_ADC1 - Xa_ADC1 ) );
    cal_m_1 = cal_m_1 + CF1_CORRECTION;

    // cal_b_0 = Ya - cal_m_0 * Xa_ADC0;
    cal_b_0 = Ya - (UInt16) ( ( (UInt32) cal_m_0 * (UInt32) Xa_ADC0 ) >> 15 );
    cal_b_0 = cal_b_0 + CF2_CORRECTION;

    // cal_b_1 = Ya - cal_m_1 * Xa_ADC1;
    cal_b_1 = Ya - (UInt16) ( ( (UInt32) cal_m_1 * (UInt32) Xa_ADC1 ) >> 15 );
    cal_b_1 = cal_b_1 + CF2_CORRECTION;
}

```

**Figure 2-13. ADC Calibration – Sample Calculation of *m* and *b***

```

extern UInt16 cal_m_0, cal_b_0;      // Reference calibration parameter variables
extern UInt16 cal_m_1, cal_b_1;      //      in global scope.

static void CalibrateADC_m_and_b( void )
{
    UInt16 ADC_Val_1, ADC_Val_2;

    // Now, process the previous info we just measured.
    ADC_Val = getReg( ADCA_ADRSLT0 );

    if ( cal_b_0 )      // Have we done calibration yet?
    {
        // First, multiply by m/2, and shift the result to give the correct
        // bus position.
        ADC_Val_1 = (UInt16) ( ((UInt32) cal_m_0 * (UInt32) ADC_Val ) >> 14 );

        // Then, add the expected offset.
        ADC_Val_2 = ( ADC_Val_1 + cal_b_0 ) << 1;
        ADC_Val = (word) ADC_Val_2;
    }
}

```

**Figure 2-14. ADC Calibration – Using *m* and *b***

## 2.11 Pin Descriptions

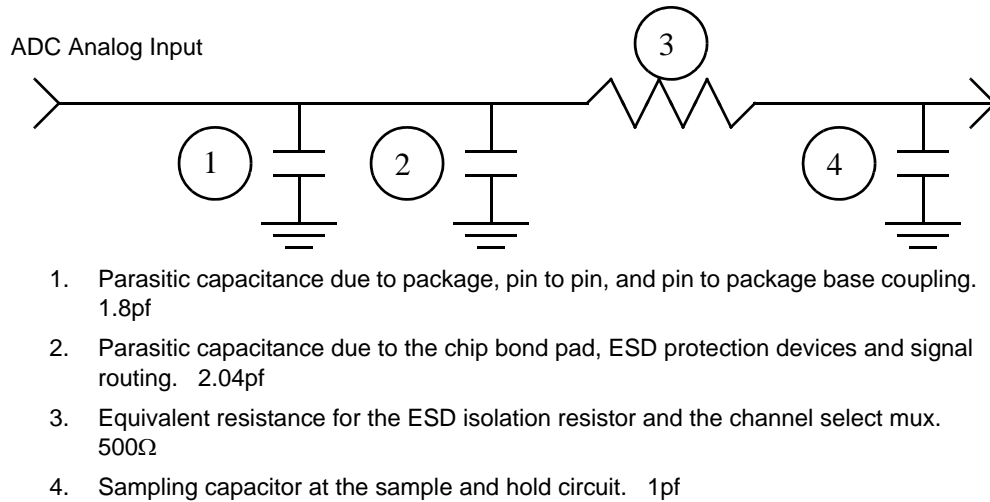
**Table 2-3. External Signal Properties**

Name	I/O Type	Function	Reset State	Notes
AN0- AN7	Input	Analog Input Pins	n/a	In the quad option, these will be ANA0-ANA7 and ANB0-ANB7.
V <sub>REFH</sub>	Input	Voltage Reference Pin	n/a	May be connected to VDDA. Must have bypassed cap to V <sub>SSA_ADC</sub>
V <sub>REFP</sub>	In/Out	Voltage Reference Pin	n/a	Must have bypass cap to V <sub>SSA_ADC</sub>
V <sub>REFMID</sub>	In/Out	Voltage Reference Pin	n/a	Must have bypass cap to V <sub>SSA_ADC</sub>
V <sub>REFN</sub>	In/Out	Voltage Reference Pin	n/a	Must have bypass cap to V <sub>SSA_ADC</sub>
V <sub>REFLO</sub>	In/Out	Voltage Reference Pin	n/a	Nominally tied to V <sub>SSA_ADC</sub>
V <sub>DDA</sub>	Supply	ADC Power	n/a	In the quad option, these will be V <sub>DDA_ADCA</sub> for ADCA and V <sub>DDA_ADCB</sub> for ADCB
V <sub>SSA</sub>	Supply	ADC Ground	n/a	—

### 2.11.1 AN0-AN7 — Analog Input Pins

Each ADC module has up to eight analog input pins. Each of the pins is connected to an internal multiplexer. The multiplexer selects the analog voltage to be converted. For simultaneous

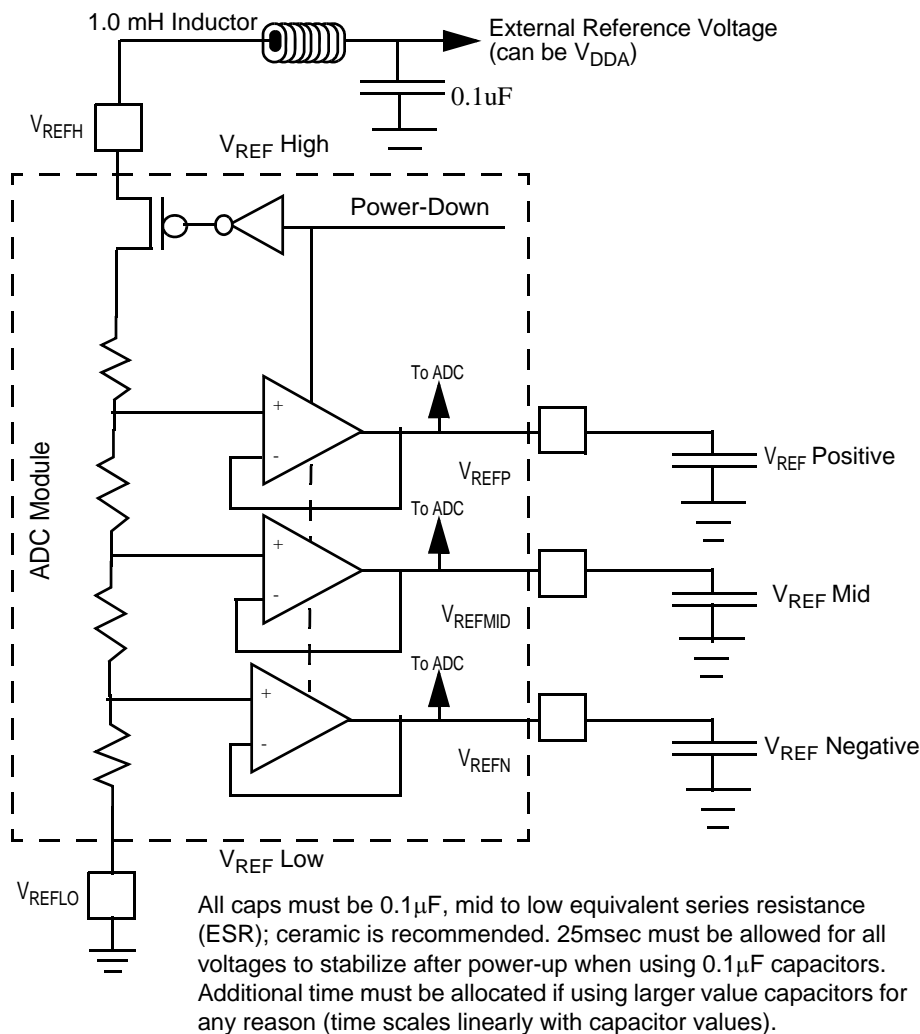
sampling of single ended inputs the multiplexer can switch any input pin to any sample slot in the ADC's scan. Differential signals are switched in pairs. Simultaneous sampling selects a pair of signals (either single ended or differential), sending each signal to one of the ADCs. Please refer to [Figure 2-3](#) and [Figure 2-4](#). An equivalent circuit for an analog input is illustrated in [Figure 2-15](#).



**Figure 2-15. Equivalent Analog Input Circuit**

### 2.11.2 Voltage Reference Pins—( $V_{REFH}$ , $V_{REFP}$ , $V_{REFMID}$ , $V_{REFN}$ , and $V_{REFLO}$ )

The voltage difference between  $V_{REFH}$  and  $V_{REFLO}$  provides the reference voltage all analog inputs are measured against.  $V_{REFH}$  is nominally set to  $V_{DDA}$ .  $V_{REFLO}$  is nominally set to  $V_{SSA}$ . The reference voltage should be provided from a low noise filtered source. The reference source should be capable of providing up to 1mA of reference current. source. The reference source should be capable of providing up to 1mA of reference current.



**Figure 2-16. ADC Voltage Reference Circuit**

When tying  $V_{REFH}$  to the same potential as  $V_{DDA}$  relative measurements are being made with respect to the amplitude of  $V_{DDA}$ . It is imperative special precautions be taken assuring the voltage applied to  $V_{REFH}$  be as noise free as possible. Any noise residing on the  $V_{REFH}$  voltage is directly transferred to the digital result.

**Figure 2-16** illustrates the internal workings of the ADC voltage reference circuit.

**Note:** This can be powered down if all associated ADCs are also powered down.

**Note:**  $V_{REFH}$  must be noise filtered - a minimum configuration is shown in the figure. In addition,  $V_{REFP}$ ,  $V_{REFMID}$ , and  $V_{REFN}$  must all be bypassed to  $V_{SSA\_ADC}$  with 0.1µF ceramic capacitors.

When using 0.1 $\mu$ F ceramic capacitors (recommended), it is essential to wait at least 25msec after power-up to begin conversions. If capacitors are greater than 0.1 $\mu$ F, additional time must be allocated (scale the delay linearly with the capacitor value.)

## 2.12 Register Definitions

**Table 2-4. ADC Memory Map**

Device	Peripheral	Address
8100/8300	ADCA_BASE	\$00F200
	ADCB_BASE	\$00F240

The address of a register is the sum of a base address and an address offset. The base address given each register will be either ADCA\_BASE or ADCB\_BASE depending on the ADC being used. Please see the chip's data sheet for the definition of ADCA\_BASE and ADCB\_BASE.

**Table 2-5. ADC Register Summary**

Address Offset	Register Acronym	Register Name	Access Type	Register Location
Base + \$0	ADCTL1	Control Register 1	Read/Write	<a href="#">Section 2.12.1</a>
Base + \$1	ADCTL2	Control Register 2	Read/Write	<a href="#">Section 2.12.2</a>
Base + \$2	ADZCC	Zero Crossing Control Register	Read/Write	<a href="#">Section 2.12.3</a>
Base + \$3	ADLST1	Channel List Register 1	Read/Write	<a href="#">Section 2.12.4</a>
Base + \$4	ADLST2	Channel List Register 2	Read/Write	
Base + \$5	ADSDIS	Sample Disable Register	Read/Write	<a href="#">Section 2.12.5</a>
Base + \$6	ADSTAT	Status Register	Read/Write	<a href="#">Section 2.12.6</a>
Base + \$7	ADLSTAT	Limit Status Register	Read/Write	<a href="#">Section 2.12.7</a>
Base + \$8	ADZCSTAT	Zero Crossing Status Register	Read/Write	<a href="#">Section 2.12.8</a>
Base + \$9-10	ADRSLT0-7	Result Registers 0-7	Read/Write	<a href="#">Section 2.12.9</a>
Base + \$11-18	ADLLMT0-7	Low Limit Registers 0-7	Read/Write	<a href="#">Section 2.12.10</a>
Base + \$19-20	ADHLMT0-7	High Limit Registers 0-7	Read/Write	
Base + \$21-28	ADDFS0-7	Offset Registers 0-7	Read/Write	<a href="#">Section 2.12.11</a>
Base + \$29	ADPOWER	Power Control Register	Read/Write	<a href="#">Section 2.12.12</a>
Base + \$2A	ADCAL	Calibration Register	Read/Write	<a href="#">Section 2.12.13</a>

Bit fields of each of the 31 registers are summarized in [Figure 2-17](#). Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
\$0	ADCTL1	R	0	STOP	0	SYNC	EOSIE	ZCIE	LLMTIE	HLMTIE	CHNCFG				0	SMODE			
		W			START														
\$1	ADCTL2	R	0	0	0	0	0	0	0	0	0	0	0	DIV					
		W																	
\$2	ADZCC	R	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0		
		W																	
\$3	ADLST1	R	0	SAMPLE3			0	SAMPLE2			0	SAMPLE1			0	SAMPLE0			
		W																	
\$4	ADLST2	R	0	SAMPLE7			0	SAMPLE6			0	SAMPLE5			0	SAMPLE4			
		W																	
\$5	ADSDIS	R	TEST		0	0	0	0	0	0	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0	
		W																	
\$6	ADSTAT	R	CIP	0	0	0	EOSI	ZCI	LLMTI	HLMT	RDY7	RDY6	RDY5	RDY4	RDY3	RDY2	RDY1	RDY0	
		W																	
\$7	ADLSTAT	R	HLS7	HLS6	HLS5	HLS4	HLS3	HLS2	HLS1	HLS0	LLS7	LLS6	LLS5	LLS4	LLS3	LLS2	LLS1	LLS0	
		W																	
\$8	ADZCSTAT	R	0	0	0	0	0	0	0	0	ZCS7	ZCS6	ZCS5	ZCS4	ZCS3	ZCS2	ZCS1	ZCS0	
		W																	
\$9	ADRSLT0	R	SEXT	RSLT											0	0	0		
		W																	
\$A	ADRSLT1	R	SEXT	RSLT											0	0	0		
		W																	
\$B	ADRSLT2	R	SEXT	RSLT											0	0	0		
		W																	
\$C	ADRSLT3	R	SEXT	RSLT											0	0	0		
		W																	
\$D	ADRSLT4	R	SEXT	RSLT											0	0	0		
		W																	
\$E	ADRSLT5	R	SEXT	RSLT											0	0	0		
		W																	
\$F	ADRSLT6	R	SEXT	RSLT											0	0	0		
		W																	
\$10	ADRSLT7	R	SEXT	RSLT											0	0	0		
		W																	
\$11	ADRLLMT0	R	0	LLMT											0	0	0		
		W																	
\$12	ADRLLMT1	R	0	LLMT											0	0	0		
		W	0																
\$13	ADRLLMT2	R	0	LLMT											0	0	0		
		W	0																
\$14	ADRLLMT3	R	0	LLMT											0	0	0		
		W	0																
\$15	ADRLLMT4	R	0	LLMT											0	0	0		
		W	0																
\$16	ADRLLMT5	R	0	LLMT											0	0	0		
		W	0																
\$17	ADRLLMT6	R	0	LLMT											0	0	0		
		W	0																
Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
\$18	ADRLLMT7	R	0	LLMT											0	0	0		
		W	0																

Figure 2-17. ADC Register Map Summary

\$19	ADHLMT0	R	0									HLMT			0	0	0
		W	0												0	0	0
\$1A	ADHLMT1	R	0									HLMT			0	0	0
		W	0												0	0	0
\$1B	ADHLMT2	R	0									HLMT			0	0	0
		W	0												0	0	0
\$1C	ADHLMT3	R	0									HLMT			0	0	0
		W	0												0	0	0
\$1D	ADHLMT4	R	0									HLMT			0	0	0
		W	0												0	0	0
\$1E	ADHLMT5	R	0									HLMT			0	0	0
		W	0												0	0	0
\$1F	ADHLMT6	R	0									HLMT			0	0	0
		W	0												0	0	0
\$20	ADHLMT7	R	0									HLMT			0	0	0
		W	0												0	0	0
\$21	ADOF0S0	R	0									OFFSET			0	0	0
		W	0												0	0	0
\$22	ADOF0S1	R	0									OFFSET			0	0	0
		W	0												0	0	0
\$23	ADOF0S2	R	0									OFFSET			0	0	0
		W	0												0	0	0
\$24	ADOF0S3	R	0									OFFSET			0	0	0
		W	0												0	0	0
\$25	ADOF0S4	R	0									OFFSET			0	0	0
		W	0												0	0	0
\$26	ADOF0S5	R	0									OFFSET			0	0	0
		W	0												0	0	0
\$27	ADOF0S6	R	0									OFFSET			0	0	0
		W	0												0	0	0
\$28	ADOF0S7	R	0									OFFSET			0	0	0
		W	0												0	0	0
\$29	ADPOWER	R	0	0	0	PSTS2	PSTS1	PSTS0	PUDELAY				PSM	PD2	PD1	PD0	
		W															
\$2A	ADCAL	R	0	0	0	0	0	0	0	0	0	0	0	CRS1	CAL1	CRS0	CAL0
		W															

R	0	Read as 0
W		Reserved

Figure 2-17. ADC Register Map Summary (Continued)

### 2.12.1 ADC Control Register 1 (ADCTL1)

Base + \$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	STOP	0	SYNC	EOSIE	ZCIE	LLMTIE	HLMTIE	CHNCFG				0	SMODE		
Write			START													
Reset	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1

Figure 2-18. ADC Control Register 1 (ADCTL1)

### 2.12.1.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 2.12.1.2 Stop (STOP)—Bit 14

When the STOP bit is selected, the current conversion process is stopped. Any further sync pulses or modifications to the START bit are ignored until the STOP bit has been cleared. After the ADC is in Stop mode, the results registers can be modified by the processor. Any changes to the results register while in Stop mode are treated as if the analog core supplied the data. Therefore, limit checking, zero crossing, and associated interrupts can occur when authorized.

**Note:** This is not the same as Stop mode for the whole chip.

- 0 = Normal operation
- 1 = Stop command issued

### 2.12.1.3 Start Conversion (START)—Bit 13

The conversion process is started by a modification to the *write-only* START bit. Once a start command is issued, the conversion process is synchronized and begun on the positive edge of the ADC clock. Setting the START bit again is ignored while the ADC is in a conversion cycle. The ADC must be idle CIP bit is 0 for it to recognize a new start-up.

- 0 = No action
- 1 = Start command is issued

The ADCs required for conversion must be in a stable power mode prior to writing the START bit. Do not assert this bit during the power-up sequence controlled by the ADCPOWER register. If both ADCs are required for conversion, *both must* be completely ON, or *both must* be disabled in Power Savings Mode (PSM). In the latter case, asserting Start will begin the power-up sequence for both ADCs. START and SYNC write signals occurring during the Power-Down mode via PD0/1 will be ignored.

### 2.12.1.4 SYNC Select (SYNC)—Bit 12

A conversion can be initiated by the SYNC input or by a write to the START bit. The timer channel providing the SYNC signal is shown in the device data sheets, Figure "Peripheral Subsystem". A SYNC pulse restarted while the ADC is in a conversion cycle is ignored. The ADC must be idle (CIP bit is 0) for it to recognize a new start-up.

- 0 = Conversion is initiated by a write to START bit only
- 1 = Use the sync input or START bit to initiate a conversion



SYNC signals must obey power mode restrictions similar to the START bit above. ADCs must be in a stable power mode prior to SYNC signal assertion. If both ADCs are required for conversion, *both must* be completely ON, or *both must* be disabled in Power Savings Mode (PSM). In the latter case, asserting a SYNC signal will begin the power-up sequence for both ADCs. A SYNC signal will have no effect on result registers when both converters are in the Power-Down mode.

#### 2.12.1.5 End-of-Scan Interrupt Enable (EOSIE)—Bit 11

This bit enables the generation of an interrupt upon completion of any scan and convert sequence, except in Loop Sequential or Loop Simultaneous modes. This bit is ignored when the ADC is configured for either Loop modes. (There is no *end* of a scan in Loop mode.)

- 0 = Interrupt disabled
- 1 = Interrupt enabled

#### 2.12.1.6 Zero Crossing Interrupt Enable (ZCIE)—Bit 10

This bit enables the generation of an interrupt if the current result value has a sign change from the previous result, configured by the ADZCC register.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

#### 2.12.1.7 Low Limit Interrupt Enable (LLMTIE)—Bit 9

This bit enables the generation of an interrupt when the current result value is less than the Low Limit register value. The raw result (i.e. without offset) is compared to the ADC Low Limit (ADLLMT $n$ ) register corresponding to the sample.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

#### 2.12.1.8 High Limit Interrupt Enable (HLMTIE)—Bit 8

This bit enables the generation of an interrupt if the current result value is greater than the High Limit register value. The raw result (i.e. without offset) value is compared to the ADC High Limit (ADHLMT $n$ ) register corresponding to the sample.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

#### 2.12.1.9 Channel Configure (CHNCFG)—Bits 7–4

Each bit in the four bit field determines the configuration of an analog pin input pair as either single ended or differential.

- xxx1 = Inputs AN0–AN1 configured as differential input (AN0 is + and AN1 is –)
- xxx0 = Inputs AN0–AN1 configured as single ended inputs
- xx1x = Inputs AN2–AN3 configured as differential input (AN2 is + and AN3 is –)
- xx0x = Inputs AN2–AN3 configured as single input
- x1xx = Inputs AN4–AN5 configured as differential inputs (AN4 is + and AN5 is –)
- x0xx = Inputs AN4–AN5 configured as single ended inputs
- 1xxx = Inputs AN6–AN7 configured as differential inputs (AN6 is + and AN7 is –)
- 0xxx = Inputs AN6–AN7 configured as single ended inputs

### 2.12.1.10 Reserved—Bit 3

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 2.12.1.11 ADC Mode Control (SMODE)—Bits 2–0

Begin a conversion sequence by asserting the ADCR1 START bit, or by a SYNC pulse. A conversion sequence can take up to eight unique samples. These eight sample slots are supported by the ADSDIS register. A sample sequence is terminated when the first disabled sample is encountered. Analog input channels are assigned to each of the eight sample slots by the ADLST1 and ADLST2 registers. A conversion sequence can be run through just once every time a trigger occurs, or it can loop continuously. Samples may be taken one at a time (sequential) or two at a time (simultaneous). A scan occurs once for each START/SYNC (Once mode), loops continuously after start (Loop mode), or can be triggered by a START/SYNC (Triggered mode).

- 000 = Once Sequential  
Upon start, or a first sync signal, samples are taken one at a time starting with SAMPLE0 until a first disabled sample is encountered. If no disabled sample is encountered in the ADSDIS register conversion concludes after the SAMPLE7.
- 001 = Once Simultaneous  
Upon start or a first sync signal, samples are taken two at a time, in the order: SAMPLE0/4, SAMPLE1/5, SAMPLE2/6, and SAMPLE3/7. The sample sequence will stop at SAMPLE3/7, or as soon as any disabled sample slot is encountered. For example, if SAMPLE0 or SAMPLE4 were disabled, no samples would be taken at all.

**Note:** In the once (single) sequential 000 and once (single) simultaneous 001 modes, provided a sampling sequence has completed, a start pulse will always initiate another once (single) sequence, either sequential or simultaneous. A sync pulse (from the appropriate timer) must be re-armed via another write to SMODE in ADCR1 in order for a second sampling sequence to occur.

- 010 = Loop Sequential  
Upon start, or a first sync pulse, samples are taken one at a time until a disabled sample is encountered. The process repeats perpetually until the STOP bit is set in ADCR1. For example, if samples zero, one, and five are enabled, the loop would look like SAMPLE0, SAMPLE1, SAMPLE0, SAMPLE1, and so on because SAMPLE 2 is disabled. While a Loop mode is running, any additional start commands or sync pulses are ignored.
- 011 = Loop Simultaneous  
Like *Once Simultaneous (001)*, samples are taken two at a time. The loop restarts as soon as a sample slot is encountered with either, or both disabled samples. For example, if DS[7:0] = [11001000] enabling samples zero, one, two, four, and five, the loop would look like SAMPLE0/4, SAMPLE1/5, SAMPLE0/4, and SAMPLE1/5, and so on. SAMPLE6 was disabled; therefore, no data was taken in the SAMPLE2/6 slot even though SAMPLE2 was enabled.
- 100 = Triggered Sequential  
A single, sequential sampling begins with every recognized start command or sync pulse. Samples are taken sequentially as described above.
- 101 = Triggered Simultaneous (default)  
A single, simultaneous sampling begins with every recognized start command or sync pulse. Samples are taken simultaneously as previously described.
- 110 = Reserved use
- 111 = Reserved use

## 2.12.2 ADC Control Register 2 (ADCTL2)

Base + \$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	DIV				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Figure 2-19. ADC Control Register 2 (ADCTL2)

### 2.12.2.1 Reserved—Bits 15–4

This bit field is reserved or not implemented. Each bit is read as 0 and they cannot be modified by writing.

### 2.12.2.2 Clock Divisor Select (DIV)—Bits 4–0

ADC\_CLK must be run at a slower rate than the system clock. The divider circuit provides a clock of period  $2N$  of system clock where  $N = \text{DIV}[3:0] + 1$ . A divisor must be selected so the ADC clock is within specified limits. For an IPBus clock frequency of 60MHz and a desired ADC\_CLK frequency of 5MHz, a DIV minimum value of 5 is required; 5 is the default value for DIV.

For an IPBus clock frequency of 40 MHz and a desired ADC\_CLK frequency of 5 MHz, a DIV minimum value of 3 is required; 5 is the default value which will result in an ADC\_CLK frequency of 3.33MHz.

$$\text{Clock Divisor Select Value} = N = \text{DIV} + 1$$

$$F_{\text{ADC}} = (F_{\text{IPR}}) / 2N$$

$F_{\text{ADC}}$  = Analog-to-Digital Converter Frequency

$F_{\text{IPR}}$  = Interface Peripheral Bus Clock Frequency

### 2.12.3 ADC Zero Crossing Control Register (ADZCC)

The ADC Zero Crossing Control (ADZCC) register provides the ability to monitor the selected channels and determine the direction of Zero Crossing triggering an optional interrupt. Zero Crossing logic monitors only the sign change between current and previous sample. ZCE0 monitors the sample stored in ADRSLT0 and ZCE7 monitors ADRSLT7. When the Zero Crossing is disabled for a selected result register, sign changes are not monitored and updated in the ADC Zero Crossing Status (ADZCSTAT) register, but the captured sample is properly stored in the respective result register.

Base + \$2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0	
Write	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-20. ADC Zero Crossing Control Register (ADZCC)

#### 2.12.3.1 Zero Crossing Enable $n$ (ZCEn)—Bits 15–0

Representing the channel number,  $n$ , setting the ZCEn field allows detection of the indicated zero crossing condition.

- 00 = Zero crossing disabled
- 01 = Zero crossing enabled for positive to negative sign change
- 10 = Zero crossing enabled for negative to positive sign change

- 11 = Zero crossing enabled for any sign change

## 2.12.4 ADC Channel List Registers (ADLST1 and ADLST2)

The Channel List register contains an ordered list of the analog input channels to be converted when the next scan is initiated. If all samples are enabled in the ADSDIS register, a simultaneous scan of inputs proceeds in order of SAMPLE0 through SAMPLE7. If one of the simultaneous sampling modes is selected instead, the sampling order is SAMPLE0 and SAMPLE4, SAMPLE1 and SAMPLE5, SAMPLE2 and SAMPLE6, and finally SAMPLE3 and SAMPLE7

**Table 2-6. ADC Input Conversion for Sample Bits**

SAMPLE $n$ [2:0]	Single Ended <sup>1</sup>	Differential <sup>1</sup>
000	AN0	AN0+, AN1-
001	AN1	AN0+, AN1-
010	AN2	AN2+, AN3-
011	AN3	AN2+, AN3-
100	AN4	AN4+, AN5-
101	AN5	AN4+, AN5-
110	AN6	AN6+, AN7-
111	AN7	AN6+, AN7-

1. Configured by CHNCFG bits in ADCR1.

In any of the sequential sampling modes there are no restrictions on assigning an analog channel, AN0-AN7, to a sample. Any sample slot may contain any channel assignment.

Base + \$3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SAMPLE3			0	SAMPLE2			0	SAMPLE1			0	SAMPLE0		
Write																
Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

**Figure 2-21. ADC Channel List Register 1 (ADLST1)**

Base + \$4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SAMPLE7			0	SAMPLE6			0	SAMPLE5			0	SAMPLE4		
Write																
Reset	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0

**Figure 2-22. ADC Channel List Register 2 (ADLST2)**

#### 2.12.4.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 2.12.4.2 SAMPLE3 and SAMPLE7—Bits 14–12

Channel select, where  $n$  is the sample number, zero through seven. This is a binary representation of the analog input to be converted.

#### 2.12.4.3 Reserved—Bit 11

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 2.12.4.4 SAMPLE2 and SAMPLE6—Bits 10–8

Channel select, where  $n$  is the sample number, 0 through 7. This is a binary representation of the analog input to be converted.

#### 2.12.4.5 Reserved—Bit 7

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 2.12.4.6 SAMPLE1 and SAMPLE5—Bits 6–4

Channel select, where  $n$  is the sample number, 0 through 7. This is a binary representation of the analog input to be converted.

#### 2.12.4.7 Reserved—Bit 3

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 2.12.4.8 SAMPLE0 and SAMPLE4—Bits 2–0

Channel select, where  $n$  is the sample number, 0 through 7. This is a binary representation of the analog input to be converted.

### 2.12.5 ADC Sample Disable Register (ADSDIS)

This register is an extension to the ADLST1 and ADLST2, providing the ability to enable only the desired samples programmed in the SAMPLE0–SAMPLE7. The  $DS_n$  bits are scanned from LSB to MSB, enabling samples for each 0 encountered until a 1 is found. In simultaneous mode, the  $DS[7:4]$  bits are scanned simultaneously with the  $DS[3:0]$  bits. For example, if in sequential mode and  $DS5$  is set to one, SAMPLE0 through SAMPLE4 are sampled. However, if Simultaneous mode is selected and  $DS5$  or  $DS1$  is set to one, only SAMPLE0 and SAMPLE4 are sampled. At reset all samples are enabled.

Base + \$5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 2-23. ADC Sample Disable Register (ADSDIS)**

### 2.12.5.1 Reserved—Bits 15–8

These bits are reserved or not implemented. They can be neither read nor modified by writing.

### 2.12.5.2 Disable Sample7–0 (DS)—Bits 7–0

The respective  $SAMPLE_n$  can be enabled or disabled where  $n = 0–7$ .

- 0 = Enable  $SAMPLE_n$
- 1 = Disable  $SAMPLE_n$

## 2.12.6 ADC Status Register (ADSTAT)

This register provides the current status of the ADC module.  $RDY_n$  bits are cleared by reading their corresponding result registers ( $ADRSLT_n$ ) either by software or the debugger.  $HLMTI$  and  $LLMTI$  bits are cleared by writing 1 to all asserted bits in the Limit Status register,  $ADLSTAT$ . The  $ZCI$ , bit is cleared by writing ones to all asserted bits in the Zero Crossing Status ( $ADZCST$ ) register.

The  $EOSI$  bit is cleared by writing 1 to it. That is, a write of \$0800 to  $ADSTAT$  will clear the  $EOSI$  bit.  $ADSTAT$  bits are sticky. Once set to one state, sticky bits require some specific action to clear them. They are not cleared automatically on the next scan sequence. These bits cannot be set by software.

Base + \$6	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CIP	0	0	0	EOSI	ZCI	LLMTI	HLMTI	RDY7	RDY6	RDY5	RDY4	RDY3	RDY2	RDY1	RDY0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 2-24. ADC Status Register (ADSTAT)**

### 2.12.6.1 Conversion in Progress (CIP)—Bit 15

This bit indicates when a scan is in progress.

- 0 = Idle state
- 1 = A scan cycle is in progress. The ADC will ignore all sync pulses or start commands

### 2.12.6.2 Reserved—Bits 14–12

This bit field is reserved or not implemented. Each bit is read as 0 and cannot be modified by writing.

### 2.12.6.3 End of Scan Interrupt (EOSI)—Bit 11

This bit indicates if a scan of analog inputs is completed since the last read of the status register, or since a reset. The EOSI bit is cleared by writing 1 to it. This bit cannot be set by software.

- 0 = A scan cycle is not completed, no end of scan IRQ pending
- 1 = A scan cycle is completed, end of scan IRQ pending

### 2.12.6.4 Zero Crossing Interrupt (ZCI)—Bit 10

If the respective Offset register has a value greater than 0000h, Zero Crossing checking is enabled. If the Offset register is programmed with 7FF8h, the result will always be less than, or equal to zero. On the other hand, if 0000h is programmed into the Offset register, the result will always be greater than, or equal to zero. No zero crossing can occur because the sign of the result will not change.

- 0 = No ZCI IRQ
- 1 = Zero crossing encountered, IRQ pending if ZCIE is set

### 2.12.6.5 Low Limit Interrupt (LLMTI)—Bit 9

If the respective Low Limit register is enabled by having a value other than 0000h, low limit checking is enabled.

- 0 = No low limit IRQ
- 1 = Low limit exceeded, IRQ pending if LLMTIE is set

### 2.12.6.6 High Limit Interrupt (HLMTI)—Bit 8

If the respective High Limit register is enabled by having a value other than 7FF8h, high limit checking is enabled.

- 0 = No high limit IRQ
- 1 = High limit exceeded, IRQ pending if HLMTIE is set



### 2.12.6.7 Ready Sample7–0 (RDY<sub>n</sub>)—Bits 7–0

These bits indicate whether samples seven through zero are ready to be read. These bits are cleared after a read from the respective results register.

- 0 = Sample not ready or has been read
- 1 = Sample ready to be read

### 2.12.7 ADC Limit Status Register (ADLSTAT)

The Limit Status register latches in the result of the comparison between the result of the sample and the respective limit register, ADHLMT0-7 and ADLLMT0-7. For an example, if the result for the channel programmed in SAMPLE0 is greater than the value programmed into the High Limit register zero, then the HLS0 bit is set to one. An interrupt is generated if the HLMTIE bit is set in ADCR1. A bit may only be cleared by writing 1 to that specific bit. These bits are sticky. Once set, the bits require a specific modification to clear them. They are not cleared automatically by subsequent conversions.

Base + \$7	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HLS7	HLS6	HLS5	HLS4	HLS3	HLS2	HLS1	HLS0	LLS7	LLS6	LLS5	LLS4	LLS3	LLS2	LLS1	LLS0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-25. ADC Limit Status Register (ADLSTAT)

### 2.12.8 ADC Zero Crossing Status Register (ADZCSTAT)

The ADZCSTAT register latches in the result of the comparison between the current result of the sample and the previous result of the same sample register. For example, if the result for the channel programmed in SAMPLE0 changes sign from the previous conversion and the respective ZCE bit in the ADZCC register is set to 11b, any edge change, then the ZCS bit is set to one. An interrupt is generated if the ZCIE bit is set in ADCR1. A bit can only be cleared by writing 1 to that specific bit. These bits are sticky. Once set, they require a write to clear them. They are not cleared automatically by subsequent conversions.

Base + \$8	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	ZCS7	ZCS6	ZCS5	ZCS4	ZCS3	ZCS2	ZCS1	ZCS0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-26. ADC Zero Crossing Status Register (ADZCSTAT)

### 2.12.8.1 Reserved—Bits 15–8

This bit field is reserved or not implemented. Each bit is read as 0 and cannot be modified by writing.

### 2.12.8.2 Zero Crossing Status (ZCS<sub>n</sub>)—Bits 7–0

The zero crossing condition is determined by examining the ADC value after it is adjusted by the offset for the Result register. Please see [Figure 2-7](#). Each bit of the register is cleared by writing 1 to the register bit.

- 0 = a. A sign change did not occur in a comparison between the current channel<sub>x</sub> result and the previous channel<sub>n</sub> result, or  
b. Zero Crossing Control is disabled for the channel in the ADZCC register.
- 1 = In a comparison between the current channel result and the previous channel result a sign change condition occurred as defined in the ADZCC register.

### 2.12.9 ADC Result Registers (ADRSLT0–7)

The eight Result registers contain the converted results from a scan. The SAMPLE0 result is loaded into ADRSLT0, SAMPLE1 result in ADRSLT1, and so on. In a Simultaneous Scan mode, the first channel pair designated by SAMPLE0 and SAMPLE4 in register ADLST1/2 are stored in ADRSLT0 and ADRSLT4, respectively.

**Note:** When writing to this register, the TEST\_DATA is used in place of any ADC Analog Core result, shown in [Figure 2-7](#). This value is then modified by the OFFSET and the result of the subtraction is read back as SEXT and RSLT[11:0].

ADC Result Register 0—Address: ADC\_BASE + \$9  
 ADC Result Register 1—Address: ADC\_BASE + \$A  
 ADC Result Register 2—Address: ADC\_BASE + \$B  
 ADC Result Register 3—Address: ADC\_BASE + \$C  
 ADC Result Register 4—Address: ADC\_BASE + \$D  
 ADC Result Register 5—Address: ADC\_BASE + \$E  
 ADC Result Register 6—Address: ADC\_BASE + \$F  
 ADC Result Register 7—Address: ADC\_BASE + \$10

Base + \$9-10	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SEXT	RSLT											0	0	0	
Write		TEST_DATA														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 2-27. ADC Result Registers (ADRSLT0-7)**

### 2.12.9.1 Sign Extend (SEXT)—Bit 15

SEXT is the sign-extend bit of the result. SEXT set to one implies a negative result. Set to zero, the implication is a positive result. If positive results are required, then the respective offset register must be set to a value of zero.

The interpretation of the numbers programmed into the Limit and Offset registers, ADLLMT, ADHLMT, and ADOFS should match the interpretation of the Result register.

### 2.12.9.2 Digital Result of the Conversion (RSLT)—Bits 14–3

ADRSLT can be interpreted as either a signed integer or a signed fractional number. As a signed fractional number, the ADRSLT can be used directly. As a signed integer, it is an option to right shift with sign extend (ASR) three places and interpret the number, or accept the number as presented, knowing there are missing codes. The lower three bits are always going to be zero.

Negative results (SEXT = 1) are always presented in two's complement format. If it is a requirement of application the result registers always be positive, the offset registers must always be set to zero.

### 2.12.9.3 Test Data (TEST\_DATA)—Bits 14–3

Please refer to [Section 2.7](#) for a description of this field and how/when it can be used.

### 2.12.9.4 Reserved—Bits 2–0

This bit field is reserved or not implemented. Each bit is read as 0 and cannot be modified by writing.

## 2.12.10 ADC Low and High Limit Registers (ADHLMT0-7) and (ADLLMT0-7)

Limit registers are programmed with the value the result is compared against. The High Limit register is used for the comparison of *Result* > *High Limit*. The Low Limit register is used for the comparison of the comparison of *Result* < *Low Limit*. The limit checking can be disabled by programming the respective Limit register with \$7FF8 for the High Limit and \$0000 for the Low Limit register. The power-up default is limit checking disabled.

ADC Low Limit Register 0—Address:ADC\_BASE + \$11  
 ADC Low Limit Register 1—Address:ADC\_BASE + \$12  
 ADC Low Limit Register 2—Address:ADC\_BASE + \$13  
 ADC Low Limit Register 3—Address:ADC\_BASE + \$14  
 ADC Low Limit Register 4—Address:ADC\_BASE + \$15  
 ADC Low Limit Register 5—Address: ADC\_BASE + \$16  
 ADC Low Limit Register 6—Address:ADC\_BASE + \$17  
 ADC Low Limit Register 7—Address:ADC\_BASE + \$18

Base + \$11-18	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	LLMT												0	0	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 2-28. ADC Low Limit Registers (ADLLMT0-7)**

ADC High Limit Register 0—Address:ADC\_BASE + \$19  
 ADC High Limit Register 1—Address:ADC\_BASE + \$1A  
 ADC High Limit Register 2—Address:ADC\_BASE + \$1B  
 ADC High Limit Register 3—Address:ADC\_BASE + \$1C  
 ADC High Limit Register 4—Address:ADC\_BASE + \$1D  
 ADC High Limit Register 5—Address:ADC\_BASE + \$1E  
 ADC High Limit Register 6—Address:ADC\_BASE + \$1F  
 ADC High Limit Register 7—Address:ADC\_BASE + \$20

Base + \$19-20	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	HLMT												0	0	0
Write																
Reset	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0

**Figure 2-29. ADC High Limit Registers (ADHLMT0-7)**

### 2.12.11 ADC Offset Registers (ADOFS0–7)

Value of the Offset register is used to correct the ADC result before it is stored in the ADRSLT registers.

ADC Offset Register 0—Address:ADC\_BASE + \$21  
 ADC Offset Register 1—Address:ADC\_BASE + \$22  
 ADC Offset Register 2—Address:ADC\_BASE + \$23  
 ADC Offset Register 3—Address:ADC\_BASE + \$24  
 ADC Offset Register 4—Address:ADC\_BASE + \$25  
 ADC Offset Register 5—Address:ADC\_BASE + \$26  
 ADC Offset Register 6—Address:ADC\_BASE + \$27  
 ADC Offset Register 7—Address:ADC\_BASE + \$28

Base + \$21-28	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	OFFSET												0	0	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 2-30. ADC Offset Registers (ADOFS0-7)**

Offset value is subtracted from the ADC result. In order to obtain unsigned results, the respective offset register should be programmed with a value of \$0000, thus giving a result range of \$0000 to \$7FF8.

### 2.12.12 ADC Power Control Register (ADCPOWER)

This register provides mechanisms for powering down individual ADC converters manually or automatically. Automatic restarting of the ADCs is triggered by the sync signal or a write to the ADCR1 START bit. The features controlled by this register can yield significant power savings when used correctly.

Features controlled by this register can yield significant power savings when used correctly.

Any conversions in progress by the affected ADC converter are aborted when PD0-2 are asserted. Affected results registers may be corrupted if the power-down operation occurs late in the conversion sequence when those registers are normally updated, otherwise they will retain their previous value.

In a dual ADC configuration, if all of PD0-2 are set, the voltage reference is also powered down. In a quad ADC configuration, these bits must be set for both dual ADCs in order for the voltage reference to be powered down. It is necessary to wait a minimum of 25msec after powering up a voltage reference and PUDELAY ADC clocks after powering up an ADC converter prior to initiating a data conversion. Failure to do so will result in incorrect conversion results.

**Note:** PUDELAY is built into the ADC state machine. PUDELAY automatically occurs.

**Note:** Power-Down mode is manually controlled via PD[2:0] and PUDELAY. Power Savings mode provides an automated way for the chip to dynamically control ADC power-up/down status. Only one of Power-Down or Power Savings modes can be active at a given time.

Base + \$29	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	PSTS2	PSTS1	PSTS0	PUDELAY						PSM	PD2	PD1	PD0
Write																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0

**Figure 2-31. ADC Power Control Register (ADCPOWER)**

#### 2.12.12.1 Reserved—Bits 15-13

This bit field is reserved or not implemented. Each bit is read as 0 and cannot be modified by writing.

### 2.12.12.2 Voltage Reference Power Status (PSTS2)—Bit 12

This *read-only* bit does not have a non-zero fixed ADC clock assertion time unlike PSTS0 and PSTS1. It simply reflects the current state of the control bit to the voltage reference circuit. Application software is responsible to ensure sufficient time is allocated for the reference to stabilize prior to beginning conversions.

- 0 = Voltage reference circuit is currently powered up
- 1 = Voltage reference circuit is currently powered down

### 2.12.12.3 ADC Converter 1 Power Status (PSTS1)—Bit 11

This *read-only* bit is deasserted immediately following a write of one to PD1. It asserts PUDELAY cycles after a write of zero to PD1. Consequently, this bit *can be read only* as a status bit to determine when the ADC is ready for operation.

- 0 = ADC Converter 1 is currently powered up
- 1 = ADC Converter 1 is currently powered down

### 2.12.12.4 ADC Converter 0 Power Status (PSTS0)—Bit 10

This *read-only* bit is deasserted immediately following a write of one to PD0. It asserts PUDELAY cycles after a write of zero to PD0. Consequently, this bit *can be read only* as a status bit to determine when the ADC is ready for operation.

- 0 = ADC Converter 0 is currently powered up
- 1 = ADC Converter 0 is currently powered down

### 2.12.12.5 Power-Up Delay (PUDELAY)—Bits 9–4

This bit field determines the number of ADC clocks required for an ADC converter to exit from Power-Down mode or begin conversions after START/SYNC in Power Savings mode. In the latter case, it determines the delay in terms of ADC clocks between power-up initiated by a write to START or SYNC and initiation of an ADC conversion cycle. The default value is 13 ADC clocks. The binary value in this field is used to initialize a 6-bit count down counter. Counting begins when the START/SYNC occurs; the ADC conversion commences when the count hits zero.

### 2.12.12.6 Power Savings Mode (PSM)—Bit 3

This bit powers down both ADCs when written to a value of one. It modifies the operation of the SYNC and START bits in ADCTL1 in the following way: instead of simply initiating a new conversion sequence, writing to START or a SYNC signal will power-up the ADC, wait a number of ADC clock cycles as determined by the PUDELAY bits and then initiate a conversion

sequence equivalent to that done when PSM is not active. At the end of the last conversion, the ADCs will be powered down again. A new SYNC or write to START will then begin the sequence over. SYNC pulses and writes to START during a conversion sequence are ignored.

ADC converters are powered up in this mode only if needed for the programmed conversion. If only one ADC converter is required, only that ADC converter is powered up.

The voltage reference circuit remains powered up in Power Savings mode. This is because it normally requires 25 msec for the reference voltages to stabilize after power-up.

- 0 = Power Savings mode is not active
- 1 = Power Savings mode is active. This bit cannot be set if any of PD0, PD1, or PD2 are asserted. When written to one, the bit is cleared.

**Note:** If PSM is asserted while a conversion is in progress, that conversion is unaffected and the ADC will wait to enter PSM only after the conversion is complete. Contrast this to Power-Down mode in which conversions in progress on the affected converter(s) are aborted when PD0-1 are asserted.

**Note:** PSM does not apply to Loop modes. It is useful only in Once/Trigger mode.

#### 2.12.12.7 Power-Down Voltage Reference (PD2)—Bit 2

This bit can be used to power-down the voltage reference.

- 0 = ADC voltage reference is powered up
- 1 = Power-down voltage reference *if* all associated ADC converters are also powered down

#### 2.12.12.8 Power-Down ADC Converter One (PD1)—Bit 1

This bit can be used to force ADC1 to power-down.

- 0 = ADC1 is powered up
- 1 = Power-down ADC1

#### 2.12.12.9 Power-Down ADC Converter Zero (PD0)—Bit 0

This bit can be used to force ADC0 to power-down.

- 0 = ADC0 is powered up
- 1 = Power-down ADC0

### 2.12.13 ADC Calibration Register (ADC\_CAL)

The ADC contains on-chip references used to calibrate the gain and offset of each ADC. **Figure 2-10** illustrates the ideal ADC and the outputs expected.

**Note:** The calibration described here can only correct errors inherent in the sample/hold and ADC itself. Errors from off-chip sources and the on-chip input muxing cannot be corrected with this feature.

Base + \$2A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	CRS1	CAL1	CRS0	CAL0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 2-32. ADC Calibration Register (ADC\_CAL)**

#### 2.12.13.1 Reserved—Bits 15–4

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 2.12.13.2 Calibration Reference Select ADC1(CRS1)—Bit 3

This bit selects which reference to choose during ADC calibration.

- 0 =  $V_{CAL\_L}$  is selected as the calibrate input to ADC1
- 1 =  $V_{CAL\_H}$  is selected as the calibrate input to ADC1

#### 2.12.13.3 Calibrate ADC1 (CAL1)—Bit 2

When this bit is set, ADC1 enters Calibrate mode and VREFLO or VREFH is routed to the ADC input.

- 0 = ADC 1 normal operation
- 1 = ADC 1 is place in Calibrate mode

#### 2.12.13.4 Calibration Reference Select (CRS0)—Bit 1

This bit selects which reference to choose during ADC calibration.

- 0 =  $V_{CAL\_L}$  is selected as the calibrate input to ADC0
- 1 =  $V_{CAL\_H}$  is selected as the calibrate input to ADC0



### 2.12.13.5 Calibrate ADC0 (CAL0)—Bit 0

When this bit is set, ADC0 enters Calibrate mode and  $V_{REFLO}$  or  $V_{REFH}$  is routed to the ADC input as selected by the CRS0 bit.

- 0 = ADC0 normal operation
- 1 = ADC0 is placed in Calibrate mode

## 2.13 Clocks

The ADC has only one clock input from the IPBus Bridge.

**Table 2-7. Clock Summary**

Clock	Priority	Source	Characteristics
IPCLK	1	IPBus Bridge	60MHz (normal for 8300 family) 40MHz (normal for 8100 family)

### 2.13.1 Clock Operation Description

The ADC internal clock used for sampling is calculated using the IPBus Clock and the clock divisor bits within the ADC Control Register 2. Please see [Section 2.12.2](#). The maximum frequency the internal clock can run is 5MHz with 200ns period. The ADC clock is active while in Loop Sequential and Simultaneous modes. It is also active during all<sup>1</sup> ADC power-up sequences for a period of time determined by the PUDELAY field in the ADC Power Control register. If a conversion is being initiated in Power Savings mode, then the ADC clock continues until the conversion sequence is complete.

When 56F8300/56F8100 are powered up, and in Once and Triggered modes, the clock is left in an off condition (logic zero) until a SYNC pulse or a write to the START bit is used to initiate a conversion. The ADC clock then continues to run until all conversions are complete at which point it shuts down until the next cycle of conversions begins. This feature makes it possible to synchronize conversions between a number of device parts running in parallel. The timing error between devices running in parallel is  $\pm 1$  IPBus cycle ( $1/(60\text{MHz})$  at max clock rate). This is a significant improvement over the 80x family, where the ADC clocks were free-running. In that family, the maximum synchronization error was  $\pm 1$  ADC clock cycle ( $1/(5\text{MHz})$  at max clock rate).

1.From either Power-Down or Power Savings modes.

## 2.14 Interrupts

**Table 2-8. Interrupt Summary**

Interrupt	Source	Description
ADC_ERR_INT	—	Zero Crossing, Low Limit, and High Limit Interrupt
ADC_CC_INT	—	Conversion Complete Interrupt

1. A Zero Crossing—occurs if the current result value has a sign change from the previous result as configured by the ADZCC register.
2. Low Limit Exceeded Error—occurs when the current result value is less than the Low Limit register value. The raw result value is compared to ADLLMT, the Low Limit register, bits LLMT, before the offset register value is subtracted.
3. High Limit Exceeded Error—is asserted if the current result value is greater than the High Limit register value. The raw result value is compared to ADHLMT, the High Limit register, bits HLMT, before the offset register value is subtracted.

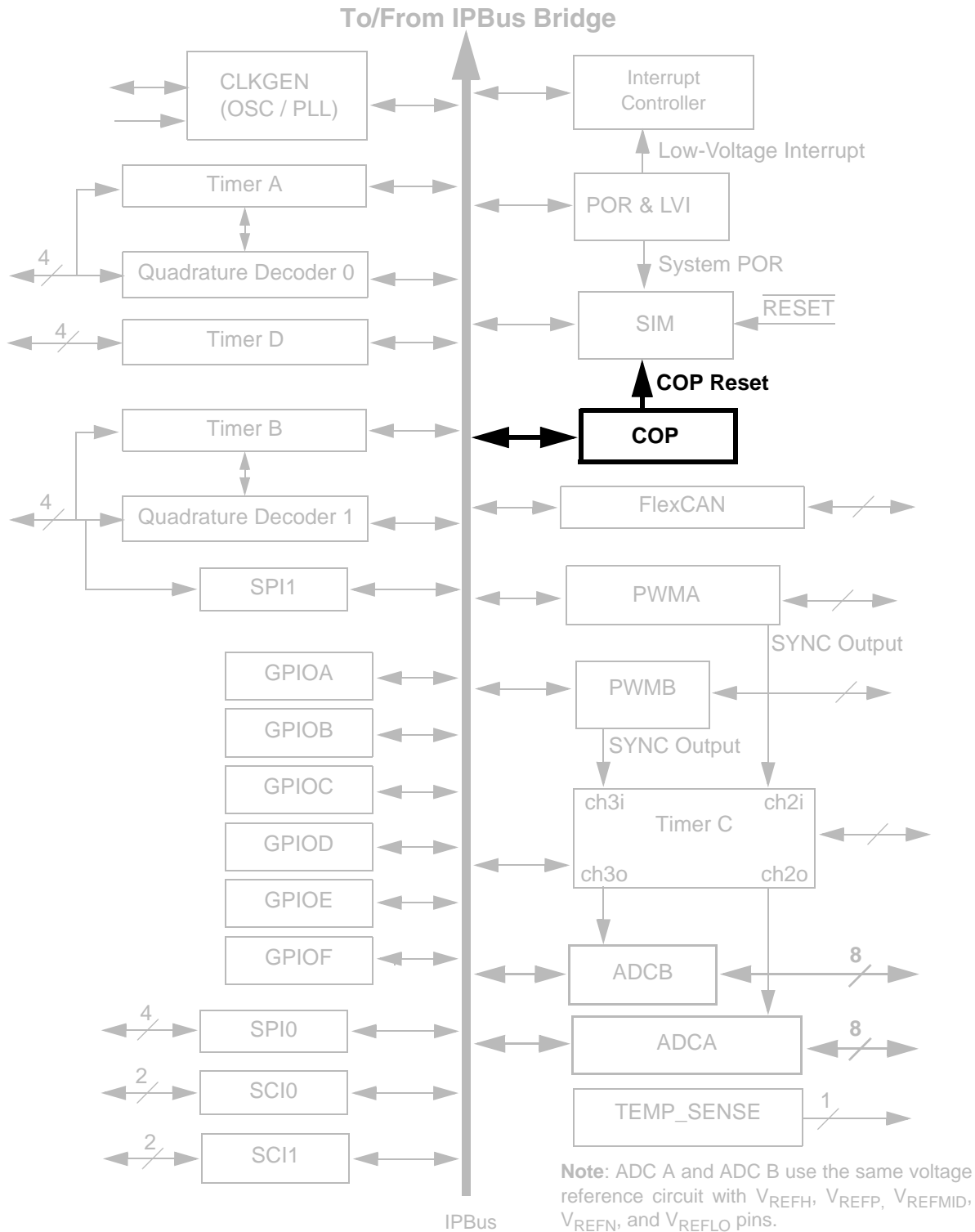
All of these interrupts are optional and enabled through the Control register.





# Chapter 3

## Computer Operating Properly (COP)



**Document Revision History for Chapter 3, Computer Operating Properly (COP)**

<b>Version History</b>	<b>Description of Change</b>
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 5.0	Converted chapter to Freescale design standards

### 3.1 Introduction

The Computer Operating Properly (COP) module assists software recover from runaway code. The COP is a free-running down counter. Once enabled, it is designed to generate a reset when reaching zero. Software must periodically service the COP to clear the counter and prevent a reset.

### 3.2 Features

The COP module design includes these characteristics:

- Programmable timeout period =  $(1024 \times (CT + 1))$  oscillator clock cycles, where CT can be from \$0000 to \$FFFF
- Programmable Wait and Stop modes
- COP timer is disabled while the host is in Debug mode

### 3.3 Block Diagram

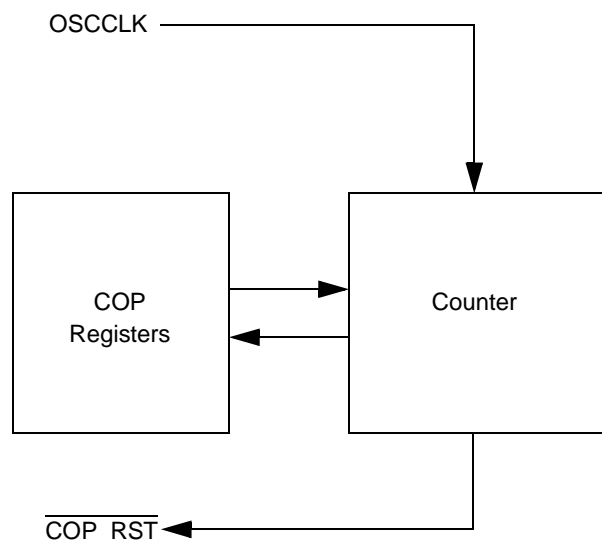


Figure 3-1. COP Module Block Diagram and Interface Signals

### 3.4 Functional Description

When the COP is enabled, each positive edge of OSCCLK causes the counter to decrease by one. If the count reaches a value of \$0000, the  $\overline{\text{COP\_RST}}$  signal is asserted and the chip is reset. In order for the 16-bit controller to show it is operating properly, it must perform a service routine prior to the count reaching \$0000. The service routine consists of writing \$5555 followed by \$AAAA to COP Counter (COPCTR) register.

**Note:** The COP timer is stopped while the processor is in Debug mode (i.e. using CodeWarrior IDE). If the COP is enabled, the timer will resume counting upon exiting Debug. The CEN bit in the COP Control (COPCTL) register always reads as zero when in Debug mode, even when it has a value of one.

## 3.5 Register Definitions

**Table 3-1. COP Memory Map**

Device	Peripheral	Address
8100/8300	COP_BASE	\$00F2C0

A register address is the sum of a base address and an address offset. The base address is defined at the system level and the address offset is defined at the module level. The COP module has three registers.

**Table 3-2. COP Register Summary**

Address Offset	Address Acronym	Register Name	Chapter Location
Base + \$0	COPCTL	Control Register	<a href="#">Section 3.5.1</a>
Base + \$1	COPTO	Timeout Register	<a href="#">Section 3.5.2</a>
Base + \$2	COPCTR	Counter Register	<a href="#">Section 3.5.3</a>

Bit fields of each of the three registers are illustrated in [Figure 3-1](#). Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$0	COPCTL	R	0	0	0	0	0	0	0	0	0	0	0	BYPS	CSEN	CWEN	CEN	CWP
		W																
\$1	COPTO	R	TIMEOUT															
		W																
\$2	COPCTR	R	COUNT															
		W	SERVICE															

R	0	Read as 0
W		Reserved

**Figure 3-2. COP Register Map Summary**



### 3.5.1 Control Register (COPCTL)

Base + \$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	BYPS	CSEN	CWEN	CEN	CWP
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-3. Control Register (COPCTL)

#### 3.5.1.1 Reserved—Bits 15–5

These bits are reserved or not implemented. The bits are read as 0 and cannot be modified by writing.

#### 3.5.1.2 Bypass OSCCLK (BYPS)—Bit 4

Setting this bit allows testing of the COP to be increased by routing the IPBus clock to the counter instead of the OSCCLK. This bit should not be set during normal operation of the chip because it will cause the COP Timeout to occur sooner. If this bit is utilized, it should *only* be changed while CEN = 0. This bit can only be changed when CWP is set to zero.

- 0 = Counter uses OSCCLK (default)
- 1 = Counter uses IPBus clock

#### 3.5.1.3 COP Stop Mode Enable (CSEN)—Bit 3

This bit controls the operation of the COP counter in Stop mode. This bit can only be changed when CWP is set to zero.

- 0 = COP counter will stop in Stop mode (default)
- 1 = COP counter will run in Stop mode when CEN is set to one

#### 3.5.1.4 COP Wait Mode Enable (CWEN)—Bit 2

This bit controls the operation of the COP counter in Wait mode. This bit can only be changed when CWP is set to zero.

- 0 = COP counter will stop in Wait mode (default)
- 1 = COP counter will run in Wait mode if CEN is set to one

### 3.5.1.5 COP Enable (CEN)—Bit 1

This bit controls the operation of the Counter (COPCTR). It can only be changed when CWP is set to zero. This bit always reads as zero when the chip is in Debug mode.

- 0 = COP counter is disabled (default)
- 1 = COP counter is enabled

### 3.5.1.6 COP Write Protect (CWP)—Bit 0

This bit controls the write protection feature of both the COPCTL and Timeout (COPTO) registers. Once set, this bit can only be cleared by resetting the module.

- 0 = COPCTL and COPTO can be read and modified by writing (default)
- 1 = COPCTL and COPTO are *read-only*

## 3.5.2 Timeout Register (COPTO)

Base + \$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TIMEOUT															
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 3-4. Timeout Register (COPTO)

### 3.5.2.1 Timeout Period (TIMEOUT)—Bits 15–0

The value in this register determines the timeout period of the COP counter. TIMEOUT should be written before enabling COP. Once COP is enabled, the recommended procedure for changing TIMEOUT is to disable the COP, write to COPTO, then re-enable the COP. This procedure ensures the new TIMEOUT is loaded into the counter. Alternatively, the 16-bit controller can write to COPTO prior to writing the proper patterns to COPCTR, thereby causing the counter to reload with the new TIMEOUT value. The COPCTR is not reset by a write to COPTO. Changing TIMEOUT while the COP is enabled results in a timeout period differing from the expected value. These bits can be changed only when CWP is set to zero.

### 3.5.3 Count Register (COPCTR)

Base + \$2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COUNT															
Write	SERVICE															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 3-5. Counter Register (COPCTR)

#### 3.5.3.1 Count (COUNT)

This is the current value of the COP counter as it counts down from the timeout value to zero. A reset is issued when this count reaches zero.

#### 3.5.3.2 Service (SERVICE)

When enabled, the COP requires a service sequence be performed periodically in order to clear its counter and prevent a reset from being issued. This routine consists of writing \$5555 to the COPCTR followed by writing \$AAAA before the timeout period expires. The writes to COPCTR must be performed in the correct order, but any number of other instructions (and writes to other registers) may be executed between the two writes.

## 3.6 Timeout Specifications

The COP uses a 16-bit counter clocked by the PLL input clock MSTR\_OSC prescaled by 1024. The COP\_PRESCALER for the family devices is 1024.

Table 3-3. Timeout Values

Count	4MHz	6MHz	8MHz
\$0000	256 $\mu$ sec	170.7 $\mu$ sec	128 $\mu$ sec
\$FFFF	16.77 sec	11.18 sec	8.39 sec

**Table 3-3** presents the range of timeout values supported as a function of oscillator frequency. For a crystal operating at 8MHz, the clock to the COP counter will be 7.81kHz. The value of the COPTO register can be programmed from 0 to 65535 giving a timeout period range from 128 $\mu$ sec minimum to 8.4sec maximum.

## 3.7 COP After Reset

COPCTL is cleared out of reset. Therefore, the counter is disabled by default. In addition, COPTO is set to its maximum value of \$FFFF during reset so the counter is loaded with a maximum timeout period when reset is released.

## 3.8 Clocks

The COP timer base is the PLL input clock MSTR\_OSC divided by a fixed prescaler value of 1024.

## 3.9 Interrupts

The COP module does not generate interrupts. It does, however, generate the  $\overline{\text{COP\_RST}}$  signal when the counter reaches a value of \$0000 causing a chip wide reset.

## 3.10 Resets

Any system reset forces all registers to their reset state, clearing the  $\overline{\text{COP\_RST}}$  signal if it is asserted. The counter will be loaded with its maximum value of \$FFFF but will not start when reset is released because CEN is disabled by default.

## 3.11 Wait Mode Operation

When entering Wait mode with both CEN and CWEN set to one, the COPCTR continues to count down. If either CEN or CWEN is set to zero when Wait mode is entered, the counter is disabled and reloads using the value in the COPTO register.

## 3.12 Stop Mode Operation

When entering Stop mode with both CEN and CSEN set to one, the COPCTR continues to count down. If either CEN or CSEN is set to zero when Stop mode is entered, the counter is disabled and reloads using the value in the COPTO register.

## 3.13 Debug Mode Operation

The COP counter is not allowed to count when the chip is in Debug mode. Further, the CEN bit in the COPCTL always reads as 0 when the chip is in Debug. The actual value of CEN is unaffected by Debug, however. The CEN resumes its previously set value when exiting Debug.

---

# Chapter 4

## External Memory Interface (EMI)

## Document Revision History for **Chapter 4, External Memory Interface (EMI)**

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 4.0	Complete re-write of this chapter Clarification in first paragraph of Sec. 4.6.3, page. 4-11, changed word <i>space</i> to <i>device</i> Added Flash security reference to Feature Section 4-2
Rev 5.0	Added bullet to Section 4.2 "Disables program space access to external memory when flash security is enabled. Please see Flash Chapter, Section 6.5.4 for additional information." Added ellipses indicating additional possible chip select lines to Table 4-2 and Figure 4-2. Corrected statement in third bullet of Section 4.2 from four CSn pins to "Has up to eight CSn configurable outputs (CS0 – CS7) for external device decoding" Converted chapter to Freescale design standards
Rev. 6.0	Corrected $\overline{CS}$ to CSBAR where appropriate
Rev. 7.0	Corrected Reset Values on Chip Select Timing Control Registers CS0, CS1 and Chip Select Option Registers CS0 and CS1. Default reset vlaues were corrected on Bus Control Register.

## 4.1 Introduction

The External Memory Interface (EMI) provides an interface allowing 56800E core to utilize external asynchronous memory. The EMI for the 56800E core operates from the system bus.

The 56800E core EMI is implemented as a core bus peripheral. Data can be transferred through the EMI to the core directly.

The EMI described in this document is intended to be interfaced to 16-bit wide external memory. External arrays may be implemented either using single 16-bit wide parts or pairs of 8-bit wide memories. An external data space memory interface to the 56800E core accommodating single 8-bit wide external data memories could be implemented (at a substantial performance penalty) with appropriate programming of the CSOR register(s).

## 4.2 Features

The External Memory Interface supports the following general characteristics:

- Can convert any internal bus memory request to a request for external memory
- Can manage multiple internal bus requests for external memory access
- Has up to eight  $\overline{CS}_n$  configurable outputs ( $\overline{CS}_0 - \overline{CS}_7$ ) for external device decoding
  - each  $\overline{CS}$  can be configured for Program or Data space
  - each  $\overline{CS}$  can be configured for Read only, Write only, or Read/Write access
  - each  $\overline{CS}$  can be configured for the number of wait states required for device access
  - each  $\overline{CS}$  can be configured for the size and location of its activation
  - each  $\overline{CS}$  is independently configured for setup and hold timing controls for both read and write
- Disables program space access to external memory when flash security is enabled. Please see Flash Chapter, Section 6.5.4 for additional information.

## 4.3 Functional Description

The 56800E core architecture contains three separate buses for access to memory/peripherals. The EMI attaches to all of these buses and provides an interface to external memory over a single external bus. The EMI serializes these internal requests to external memory in a manner avoiding conflicts and contention.

### 4.3.1 Core Interface Detail

Managing the core access to the external memory consists of four issues: (Please refer to [Figure 4-1](#).)

1. Any of the three buses can request external access at any time. This means the EMI can potentially have three requests it must complete before the core can proceed. The EMI must hold-off further execution of the core until it can serialize the requests over the external bus. This provides *simultaneous* data for all buses to the core for read operations.
2. There may be a mixture of read and write requests on the core buses. For instance, the program memory bus may request a read operation while the primary data bus (XAB1) is requesting a write operation.
3. The primary data bus (XAB1) may request an 8-bit transfer. This request must access the appropriate external byte.
4. The primary data bus (XAB1) may request a 32-bit transfer. This request requires two accesses of the external bus. The EMI must hold-off further core execution until all 32 bits have been transferred. This action may happen in conjunction with item one above.

## 4.4 Block Diagram

A simplified block diagram illustrating the connections to the EMI is illustrated in [Figure 4-1](#). The left side of the figure shows connections to the 56800E core buses and clocks. All available external EMI signals are shown on the right side of the figure. In some cases, pin count restrictions may limit the number of EMI signals brought out of the package.



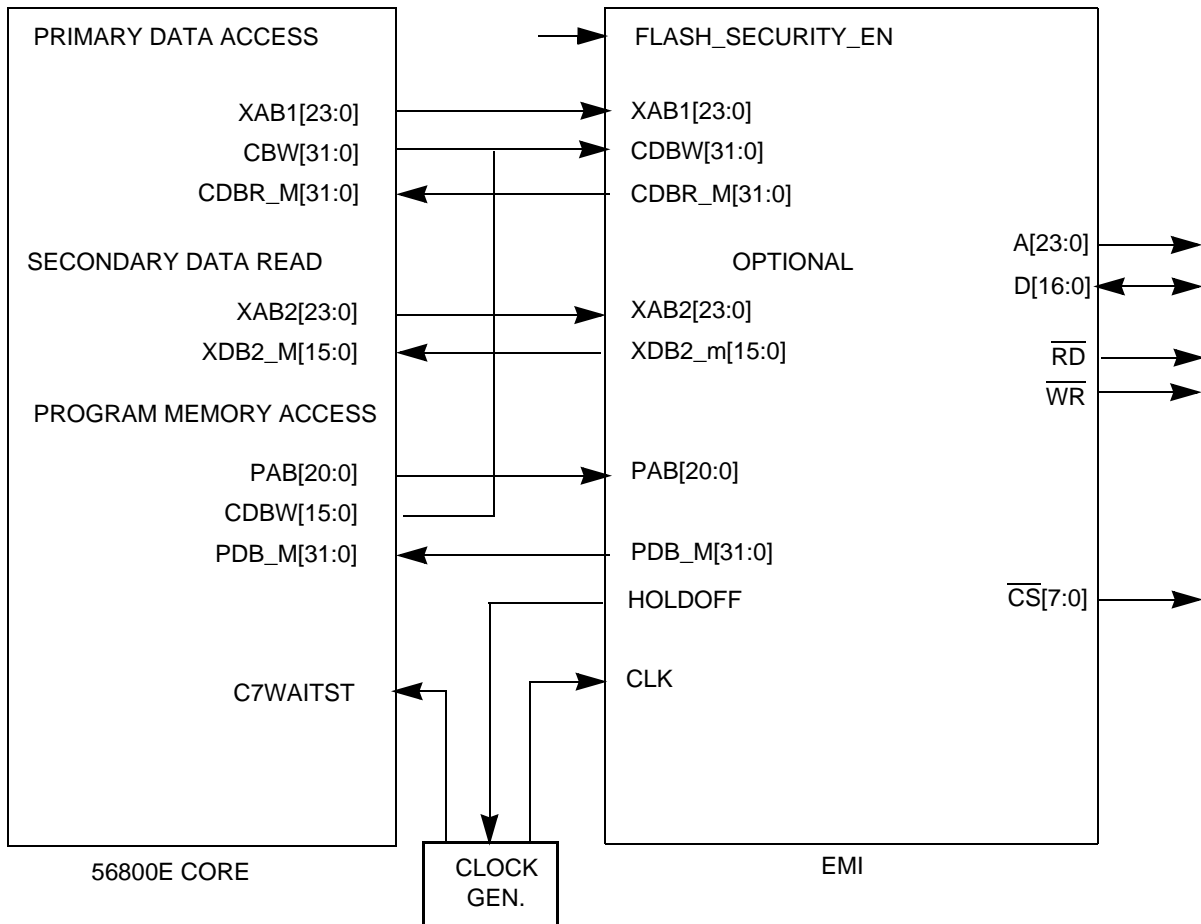


Figure 4-1. EMI Block Diagram

## 4.5 Register Definitions

Table 4-1. DEC Memory Map

Device	Peripheral	Address
8100/8300	EMI_BASE	\$00F020

The address of a register is the sum of a base address and an address offset. The base address is defined at the device level. Registers are summarized in [Table 4-2](#) and [Figure 4-2](#).

**Table 4-2. EMI Register Summary**

Address Offset	Register Acronym	Register Name	Chapter Location
Base + \$0	$\overline{\text{CS0}}$	Chip Select Base Address Register 0	<a href="#">Section 4.6.1</a>
Base + \$1	$\overline{\text{CS1}}$	Chip Select Base Address Register 1	
Base + \$2	$\overline{\text{CS2}}$	Chip Select Base Address Register 2	
Base + \$3	$\overline{\text{CS3}}$	Chip Select Base Address Register 3	
⋮	⋮	Note: Depending on the device, there may be additional $\text{CS}_n$ registers available.	
Base + \$8	CSOR0	Chip Select Option Register 0	<a href="#">Section 4.6.2</a>
Base + \$9	CSOR1	Chip Select Option Register 1	
Base + \$A	CSOR2	Chip Select Option Register 2	
Base + \$B	CSOR3	Chip Select Option Register 3	
⋮	⋮	Note: Depending on the device, there may be additional $\text{CSOR}_n$ registers available.	
Base + \$10	CSTC0	Chip Select Timing Control Register 0	<a href="#">Section 4.6.3</a>
Base + \$11	CSTC1	Chip Select Timing Control Register 1	
Base + \$12	CSTC2	Chip Select Timing Control Register 2	
Base + \$13	CSTC3	Chip Select Timing Control Register 3	
⋮	⋮	Note: Depending on the device, there may be additional $\text{CSTC}_n$ registers available.	
Base + \$18	BCR	Bus Control Register	<a href="#">Section 4.6.4</a>

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$0	CSBAR0	R W	ADR 23	ADR 22	ADR 21	ADR 20	ADR 19	ADR 18	ADR 17	ADR 16	ADR 15	ADR 14	ADR 13	ADR 12	BLKSZ			
\$1	CSBAR1	R W	ADR 23	ADR 22	ADR 21	ADR 20	ADR 19	ADR 18	ADR 17	ADR 16	ADR 15	ADR 14	ADR 13	ADR 12	BLKSZ			
\$2	CSBAR2	R W	ADR 23	ADR 22	ADR 21	ADR 20	ADR 19	ADR 18	ADR 17	ADR 16	ADR 15	ADR 14	ADR 13	ADR 12	BLKSZ			
\$3	CSBAR3	R W	ADR 23	ADR 22	ADR 21	ADR 20	ADR 19	ADR 18	ADR 17	ADR 16	ADR 15	ADR 14	ADR 13	ADR 12	BLKSZ			
⋮	⋮		Note: Depending on the device, there may be additional CSBAR <sub>n</sub> registers available.															
\$8	CSOR0	R W	RWS				BYTE_EN		R/W	PS/DS		WWS						
\$9	CSOR1	R W	RWS				BYTE_EN		R/W	PS/DS		WWS						
\$A	CSOR2	R W	RWS				BYTE_EN		R/W	PS/DS		WWS						
\$B	CSOR3	R W	RWS				BYTE_EN		R/W	PS/DS		WWS						
⋮	⋮		Note: Depending on the device, there may be additional CSOR <sub>n</sub> registers available.															
\$10	CSTC0	R W	WWSS		WWSH		RWSS		RWSH		0	0	0	0	0	MDAR		
\$11	CSTC1	R W	WWSS		WWSH		RWSS		RWSH		0	0	0	0	0	MDAR		
\$12	CSTC2	R W	WWSS		WWSH		RWSS		RWSH		0	0	0	0	0	MDAR		
\$13	CSTC3	R W	WWSS		WWSH		RWSS		RWSH		0	0	0	0	0	MDAR		
⋮	⋮		Note: Depending on the device, there may be additional CSTC <sub>n</sub> registers available.															
\$18	BCR	R W	DRV	BMDAR			0	0	BWWS				BRWS					

R	0	Read as 0
W		Reserved

Note: Please see the chip specific Data Sheet for the base address.

**Figure 4-2. EMI Register Map Summary**

## 4.6 Register Descriptions

### 4.6.1 Chip Select Base Address Registers 0–3 (CSBAR0–CSBAR3)

The  $\overline{CS}$  registers are defined in [Figure 4-3](#). It determines the active address range of a given  $\overline{CS}_n$ . The Block Size (BLKSZ) field determines the size of the memory map covered by the  $\overline{CS}_n$ . This field also determines which of the address bits to use when specifying the base address of the  $\overline{CS}_n$ . This encoding is detailed in [Table 4-3](#). When the active bits match the address and the constraints specified in the CSOR are also met, the  $\overline{CS}_n$  is asserted.

The chip-select address compare logic uses only the most significant bits to match an address within a block. The value of the base address must be an integer multiple of the block size (i.e.,

the base address can be at block size boundaries only). For example, for a block size of 64k, the base address can be 64k, 128k, 192k, 256k, 320k, etc.

**Note:** The default reset value for  $\overline{CSn}$  will enable a 32K block of external memory starting at address zero. This may be defined to be something else for a specific chip in which case the chip user manual will detail the specific reset value.

Base + \$0 - \$3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	ADR23	ADR22	ADR21	ADR20	ADR19	ADR18	ADR17	ADR16	ADR15	ADR14	ADR13	ADR12	BLKSZ			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

**Figure 4-3. Chip Select Base Address Registers 0–3 (CSBAR0–CSBAR3)**

**Table 4-3.  $\overline{CS}$  Encoding of the BLKSZ Field**

BLKSZ	Block Size	Address Lines Compared
0000	4K	X: ADR[23:12], P: ADR[20:12]
0001	8K	X: ADR[23:13], P: ADR[20:13]
0010	16K	X: ADR[23:14], P: ADR[20:14]
0011	32K	X: ADR[23:15], P: ADR[20:15]
0100	64K	X: ADR[23:16], P: ADR [20:16]
0101	128K	X: ADR[23:17], P: ADR[20:17]
0110	256K	X: ADR[23:18], P: ADR[20:18]
0111	512K	X: ADR[23:19], P: ADR [20:19]
1000	1M	X: ADR[23:20], P: ADR[20:20]
1001	2M	X: ADR[23:21]. All program address space decoded.
1010	4M	X: ADR[23:22]. No program address space decoded.
1011	8M	X: ADR[23:23]. No program address space decoded.
1100	16M	All data address space decoded. No program address space decoded.
1101	Reserved	No data address space decoded. No program address space decoded.
1110	Reserved	No data address space decoded. No program address space decoded.
1111	Reserved	No data address space decoded. No program address space decoded.

#### 4.6.2 Chip Select Option Registers 0–3 (CSOR0–CSOR3)

A Chip Select Option Register is required for every chip select. This register specifies the mode of operation of the chip select and the timing requirements of the external memory.

**Note:** The  $\overline{CSn}$  logic can be used to define external memory wait states even if the  $\overline{CSn}$  pin is used as GPIO.

**Note:** The  $\overline{CSn}$  output can be disabled by setting either the PS/DS, R/W or BYTE\_EN fields to zero.

Base + \$8 - \$B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RWS					BYTE_EN		R/W		PS/DS		WWS				
Write	RWS					BYTE_EN		R/W		PS/DS		WWS				
Reset	0	1	0	1	1	1	1	1	1	1	0	0	1	0	1	1

Figure 4-4. Chip Select Option Register 0 (CSOR0)

Base + \$8 - \$B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RWS					BYTE_EN		R/W		PS/DS		WWS				
Write	RWS					BYTE_EN		R/W		PS/DS		WWS				
Reset	0	1	0	1	1	1	1	1	1	0	1	0	1	0	1	1

Figure 4-5. Chip Select Option Register 1 (CSOR1)

Base + \$8 - \$B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RWS					BYTE_EN		R/W		PS/DS		WWS				
Write	RWS					BYTE_EN		R/W		PS/DS		WWS				
Reset	1	0	1	1	1	0	0	0	0	0	0	1	0	1	1	1

Figure 4-6. Chip Select Option Registers 2-3 (CSOR2–CSOR3)

#### 4.6.2.1 Read Wait States (RWS)—Bits 15–11

The RWS field specifies the number of additional system clocks, 0-30 (31 is invalid) to delay for read access to the selected memory. The value of RWS should be set as indicated in [Section 4.7.1](#).

#### 4.6.2.2 Upper/Lower Byte Option (BYTE\_EN)—Bits 10–9

This field specifies whether the memory is 16 bits wide or one byte wide. If the memory is byte wide, the option of upper or lower byte of a 16-bit word is selectable. [Table 4-4](#) provides the encoding of this field.

Table 4-4. CSOR Encoding BYTE\_EN Values

Value	Meaning
00	Disable
01	Lower Byte Enable
10	Upper Byte Enable
11	Both Bytes Enable

### 4.6.2.3 Read/Write Enable (R/W)—Bits 8–7

This field determines the read/write capabilities of the associated memory as shown in [Table 4-5](#).

**Table 4-5. CSOR Encoding of Read/Write Values**

Value	Meaning
00	Disable
01	Write-Only
10	Read-Only
11	Read / Write

### 4.6.2.4 Program/Data Space Select (PS/DS)—Bits 6–5

The mapping of a chip select to program and/or data space is shown in [Table 4-6](#).

**Table 4-6. CSOR Encoding of PS/DS Values**

Value	Meaning	
	Flash_Security_Enable = 0	Flash_Security_Enable = 1
00	Disable	Disable
01	DS Only	DS Only
10	PS Only	Disable
11	Both PS and DS	DS Only

### 4.6.2.5 Write Wait States (WWS)—Bits 4–0

The WWS field specifies the number of additional system clocks 0-30 (31 is invalid) to delay for write access to the selected memory. The value of WWS should be set as indicated in [Section 4.7.2](#).

## 4.6.3 Chip Select Timing Control Registers 0–3 (CSTC0–CSTC3)

A Chip Select Timing Control (CSTC) register is required for every chip select. This register specifies the detailed timing required for accessing devices in the selected memory map. At reset, these registers are configured for minimal timing in the external access waveforms. Therefore, these registers need only be adjusted if required by slower memory/peripheral devices.

Base + \$10 - \$13	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	WWSS		WWSH		RWSS		RWSH		0	0	0	0	0	MDAR		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

**Figure 4-7. Chip Select Timing Control Registers 0–3 (CSTC0–CSTC3)**

#### 4.6.3.1 Write Wait States Setup Delay (WWSS)—Bits 15–14

This field affects the write cycle timing diagram, illustrated in [Figure 4-18](#). Additional time (clock cycles) is provided between the assertion of  $\overline{CSn}$  and address lines and the assertion of  $\overline{WR}$ . The value of WWSS should be set as indicated in [Section 4.7.2](#).

#### 4.6.3.2 Write Wait States Hold Delay (WWSH)—Bits 13–12

This field affects the write cycle timing diagram, illustrated in [Figure 4-19](#). The WWSH field specifies the number of additional system clocks to hold the address, data, and  $\overline{CSn}$  signals after the  $\overline{WR}$  signal is deasserted. The value of WWSH should be set as indicated in [Section 4.7.2](#).

#### 4.6.3.3 Read Wait States Setup Delay (RWSS)—Bits 11–10

This field affects the read cycle timing diagram, illustrated in [Figure 4-12](#). Additional time (clock cycles) is provided between the assertion of  $\overline{CSn}$  and address lines and the assertion of  $\overline{RD}$ . The value of RWSS should be set as indicated in [Section 4.7.1](#).

#### 4.6.3.4 Read Wait States Hold Delay (RWSH)—Bits 9–8

This field affects the read cycle timing diagram, illustrated in [Figure 4-13](#). The RWSH field specifies the number of additional system clocks to hold the address, data, and  $\overline{CSn}$  signals after the  $\overline{RD}$  signal is deasserted. The value of RWSH should be set as indicated in [Section 4.7.1](#).

**Note:** If both, the RWSS and RWSH fields are set to zero the EMI read timing is set for consecutive mode. In this mode the  $\overline{RD}$  signal will remain active during back-to-back reads from the same  $\overline{CSn}$  controlled memory space.

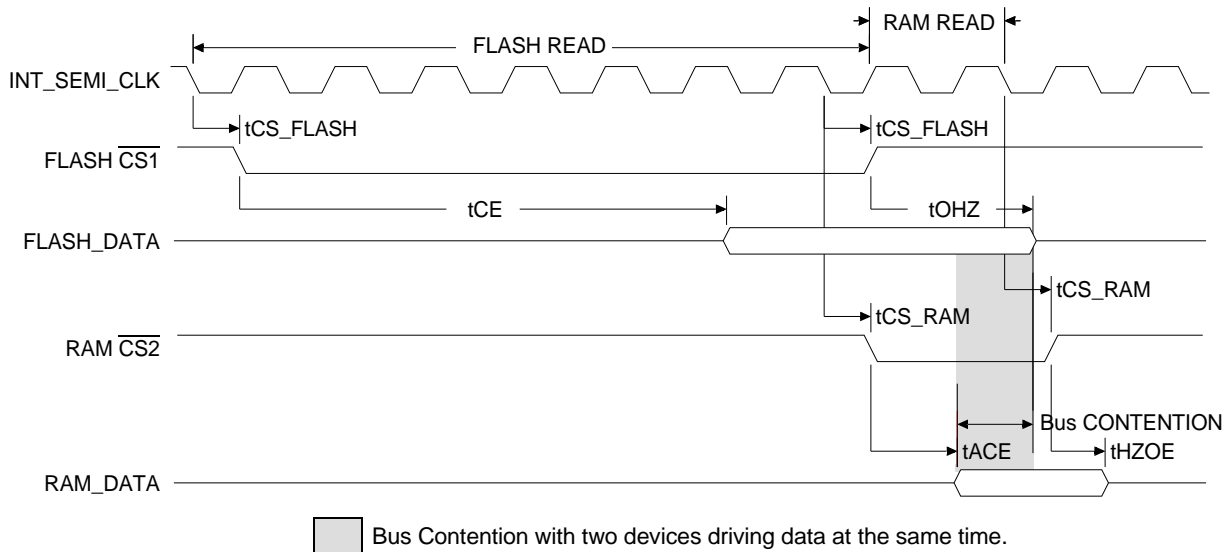
#### 4.6.3.5 Reserved—Bits 7–3

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 4.6.3.6 Minimal Delay After Read (MDAR)—Bits 2–0

This field specifies the number of system clocks to delay between reading from memory in a  $\overline{CSn}$  controlled device and reading from another device. Since a write to the device implies activating the DSP on the bus, this is also considered a read from another device.

[Figure 4-8](#) illustrates the timing issue requiring the introduction of the MDAR field. In this diagram,  $\overline{CS1}$  is assumed to operate a slow flash memory in P-space while  $\overline{CS2}$  is operating a faster RAM in X-space. In some bus contention cases, it is possible to encounter data integrity problems where the contention is occurring at the time the data bus is sampled.



**Figure 4-8. Data Bus Contention Timing Requiring MDAR Field Assertion**

### 4.6.4 Bus Control Register (BCR)

The BCR register defines the default read timing for external memory accesses to addresses not covered by the CSBAR/CSOR/CSTC registers. The timing specified by the BCR register applies to both program and data space accesses because the PS and DS control signals are not directly available on the chip pinouts.

**Note:** Any of the  $\overline{CS}_n$  signals can be configured to mirror the  $\overline{PS}$  and/or  $\overline{DS}$  function, but then the associated  $\overline{CS}_n$  configuration registers control the timing.

Base + \$18	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Read	DRV	BMDAR				0	0	BWWS					BRWS				
Write						0	0										
Reset	0	0	0	0	0	0	0	1	0	1	1	0	1	0	1	1	

**Figure 4-9. Bus Control Register (BCR)**

#### 4.6.4.1 Drive (DRV)—Bit 15

The Drive (DRV) control bit is used to specify what occurs on the external memory port pins when no external access is performed (whether the pins remain driven or are placed in tri-state). [Table 4-7](#) summarizes the action of the EMI when the DRV bit is cleared or is set. DRV bit is cleared on hardware reset, but should be set in most customer applications.





**Table 4-7. Operation with DRV**

800E Core Operating State	DRV	Pins		
		A23:A0	$\overline{RD}$ , $\overline{WR}$ , $\overline{CS}_n$	D15:D0
EMI is Between External Memory Accesses	0	Tri-stated	Tri-stated	Tri-stated
Reset Mode		Tri-stated	Pulled High Internally	Tri-stated
EMI is Between External Memory Accesses	1	Driven	Driven ( $\overline{RD}$ , $\overline{WR}$ , $\overline{CS}_n$ are Deasserted)	Tri-stated
Reset Mode		Tri-stated	Pulled High Internally	Tri-stated

#### 4.6.4.2 Base Minimal Delay After Read (BMDAR)—Bits 14–12

This bit field specifies the number of system clocks to delay after reading from memory *not* in  $\overline{CS}$  controlled space. Since a write to the device implies activating the DSP on the bus, this is also considered a read from another device, therefore activating the BMDAR timing control. Please see the description of the MDAR field of the CSTC registers for a discussion of the function of this control.

#### 4.6.4.3 Reserved—Bits 11–10

This bit field is reserved or not implemented. It is read as 0 and it cannot be modified by writing.

#### 4.6.4.4 Base Write Wait States (BWWS)—Bits 9–5

This bit field specifies the number of additional system clocks 0-30 (31 is invalid) to delay for write access to the selected memory when the memory address does *not* fall within  $\overline{CS}$  controlled range. The value of BWWS should be set as indicated in [Section 4.7](#).

#### 4.6.4.5 Base Read Wait States (BRWS)—Bits 4–0

This bit field specifies the number of additional system clocks 0-30 (31 is invalid) to delay for read access to the selected memory when the memory address does *not* fall within  $\overline{CS}$  controlled range. The value of BRWS should be set as indicated in [Section 4.7](#).

## 4.7 Timing Specifications

### 4.7.1 Read Timing

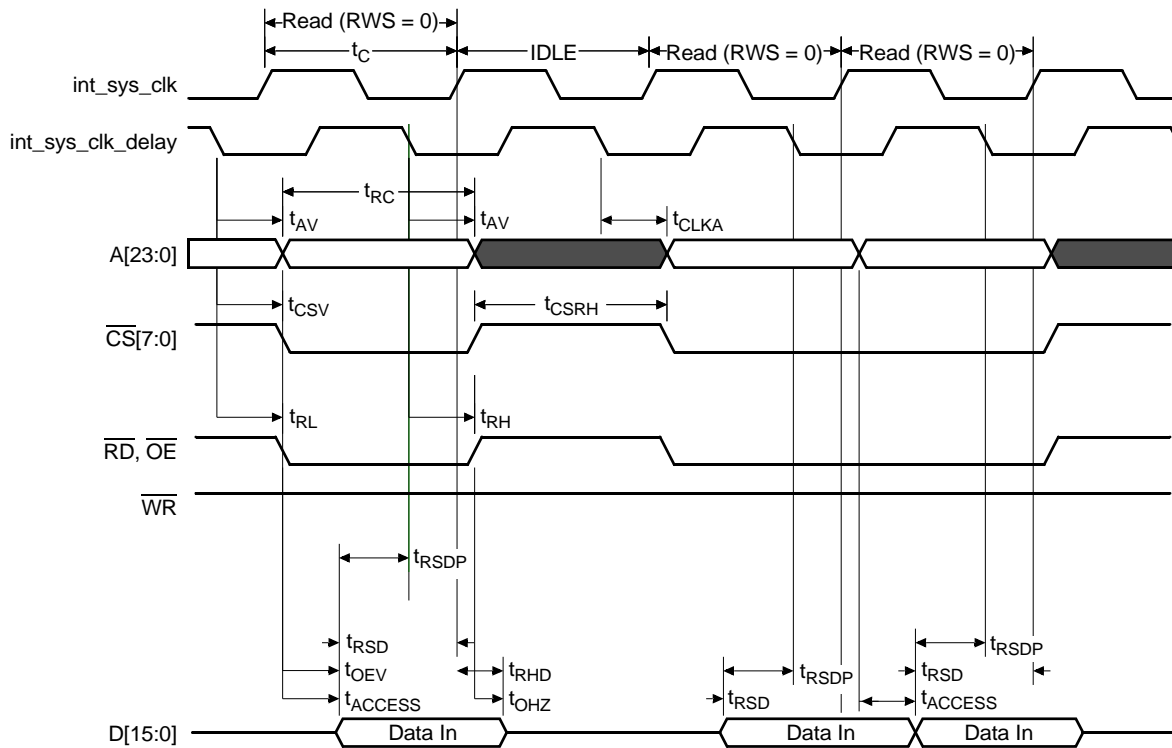
#### 4.7.1.1 Consecutive Mode Operation

[Figure 4-10](#) illustrates the read timing for external memory access. For comparison, a single read cycle is illustrated followed by a null cycle and then a back-to-back read.

[Figure 4-10](#) assumes zero wait states are required for the access. [Figure 4-11](#) illustrates a timing diagram with one wait state added.

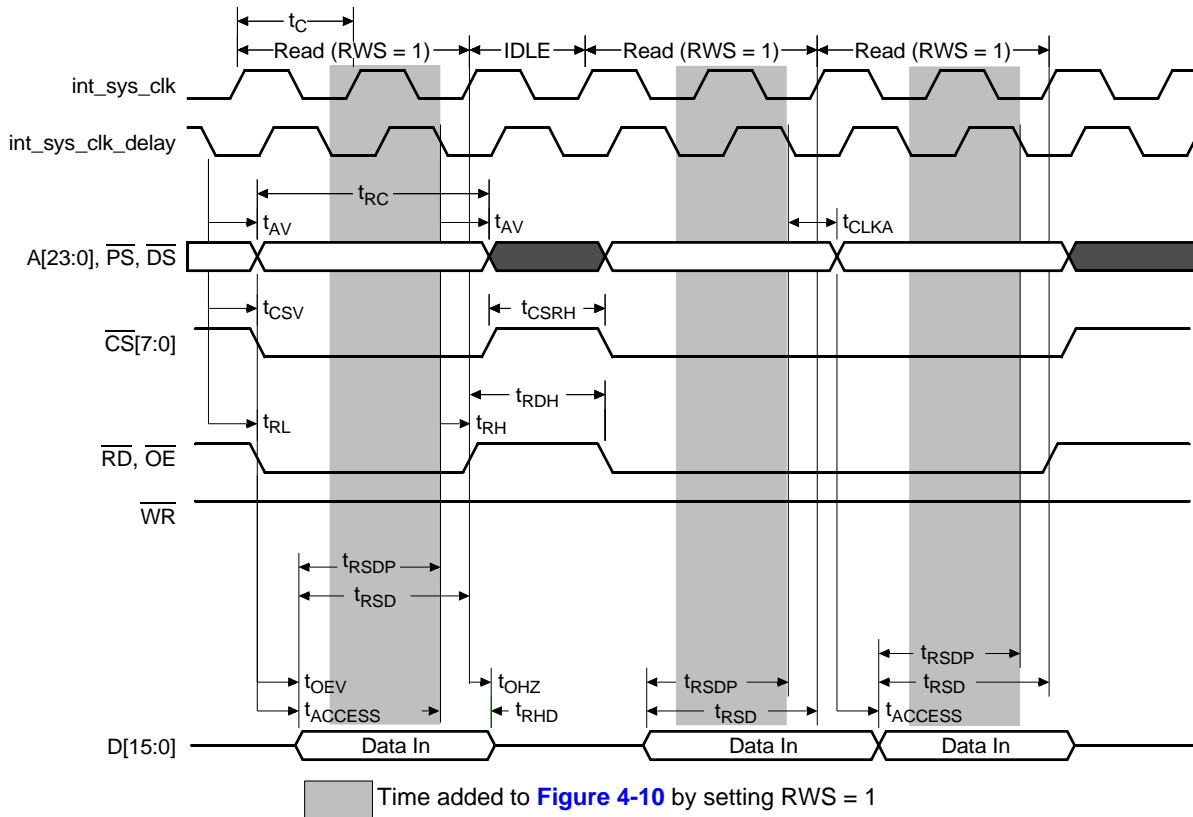
There are two read setup timing parameters for each read cycle. The core will latch the data on the rising edge of the internal clock while  $t_{RSD}$  indicates the core setup time. The external timing of the address and controls is adjusted so they may be changing at this time. Therefore, a data latch is introduced to capture the data (at the pin) a quarter clock earlier, on the rising edge of the internal delayed clock. The setup time required for this latch is illustrated by  $t_{RSDP}$  in the diagrams. For slow clock speeds,  $t_{RSDP}$  is more critical, while  $t_{RSD}$  may be harder to meet for faster clock rates.

**Note:** During back-to-back reads,  $\overline{RD}$  remains low to provide the fastest read cycle time.



**Figure 4-10. External Read Cycle with Clock and RWS = 0**

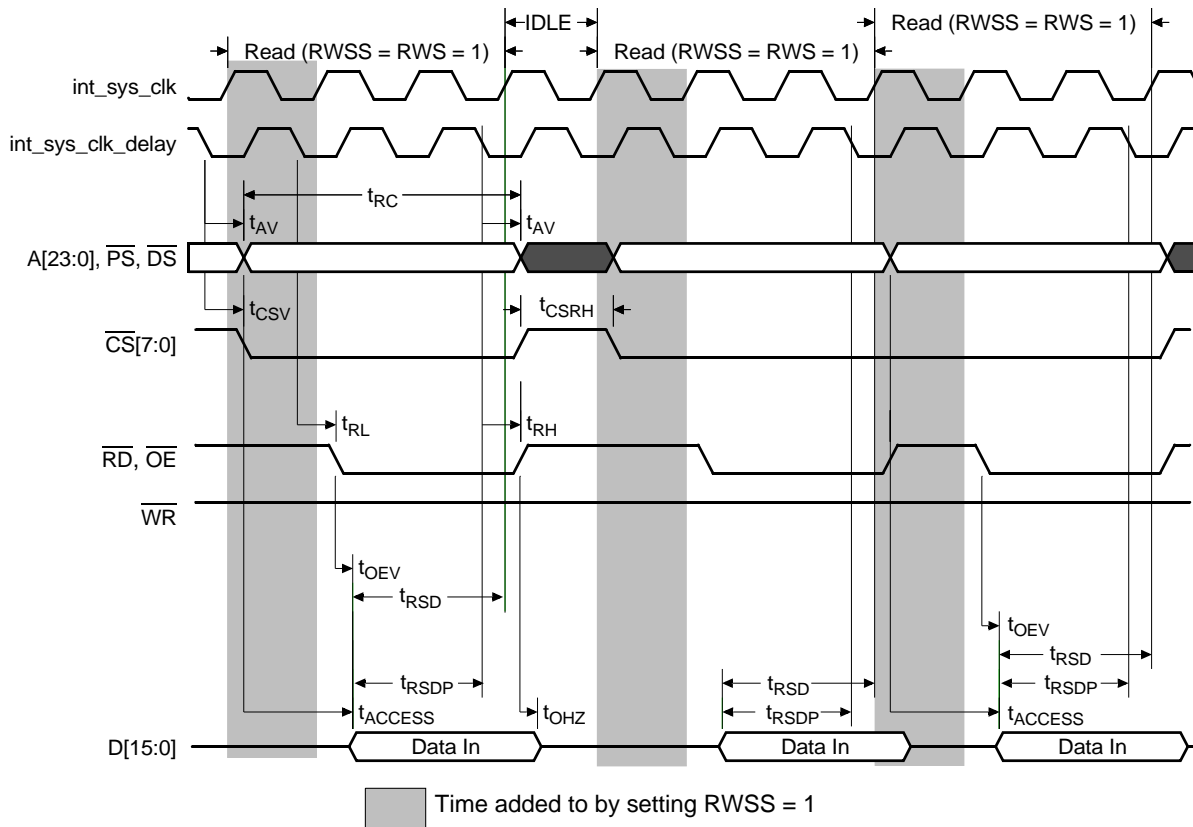
**Note:** INT\_SYS\_CLK is the internal system clock from which everything is referenced.



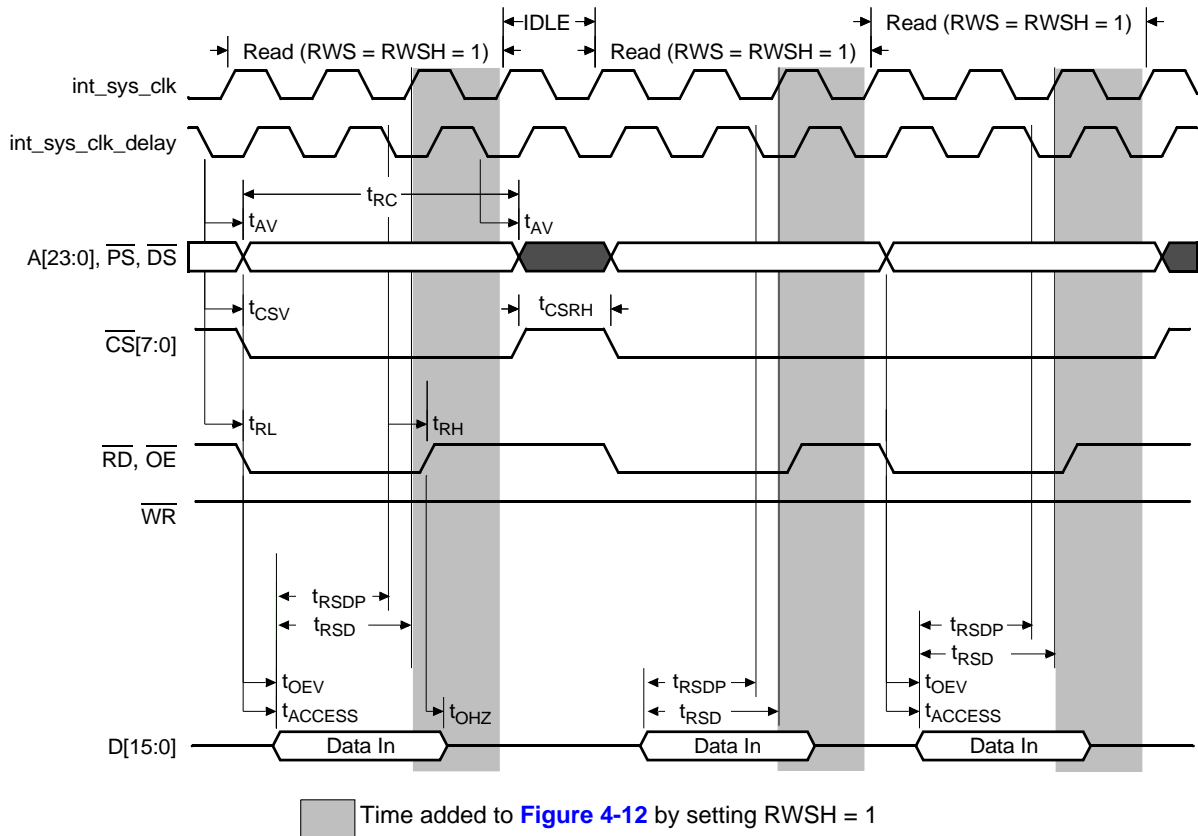
**Figure 4-11. External Read Cycle with RWS = 1, RWSH = 0 and RWSS = 0**

#### 4.7.1.2 Read Setup and Hold Timing

Although most memory devices can perform consecutive reads by holding the  $\overline{CS}_n$  and  $\overline{RD}(\overline{OE})$  signals in the active state and changing the address, there are peripheral devices that require  $\overline{RD}(\overline{OE})$  to transition to the inactive state between reads of certain registers. This timing can be accommodated with the Read Setup (RWSS) and/or Read Hold (RWSH) control fields illustrated in [Figure 4-12](#) and [Figure 4-13](#).



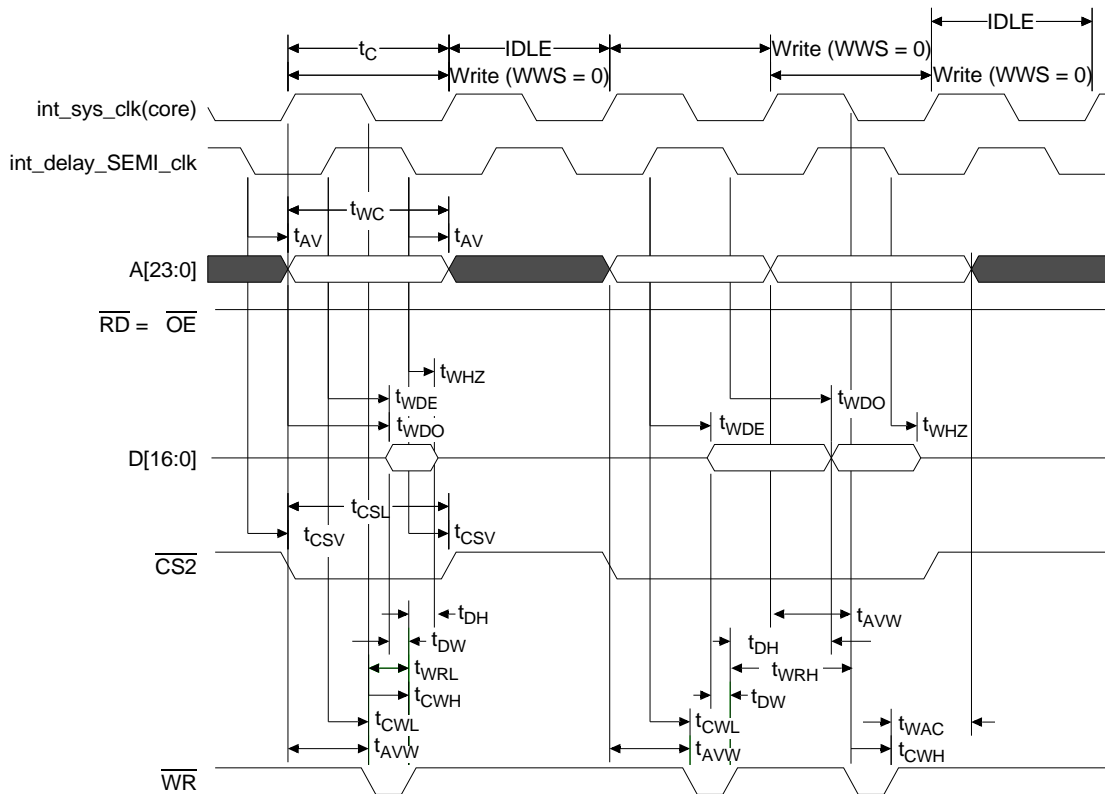
**Figure 4-12. External Read Cycle with  $RWSS = RWS = 1$ , and  $RWSH = 0$**



**Figure 4-13. External Read Cycle RWS = RWSH = 1 and RWSS = 0**

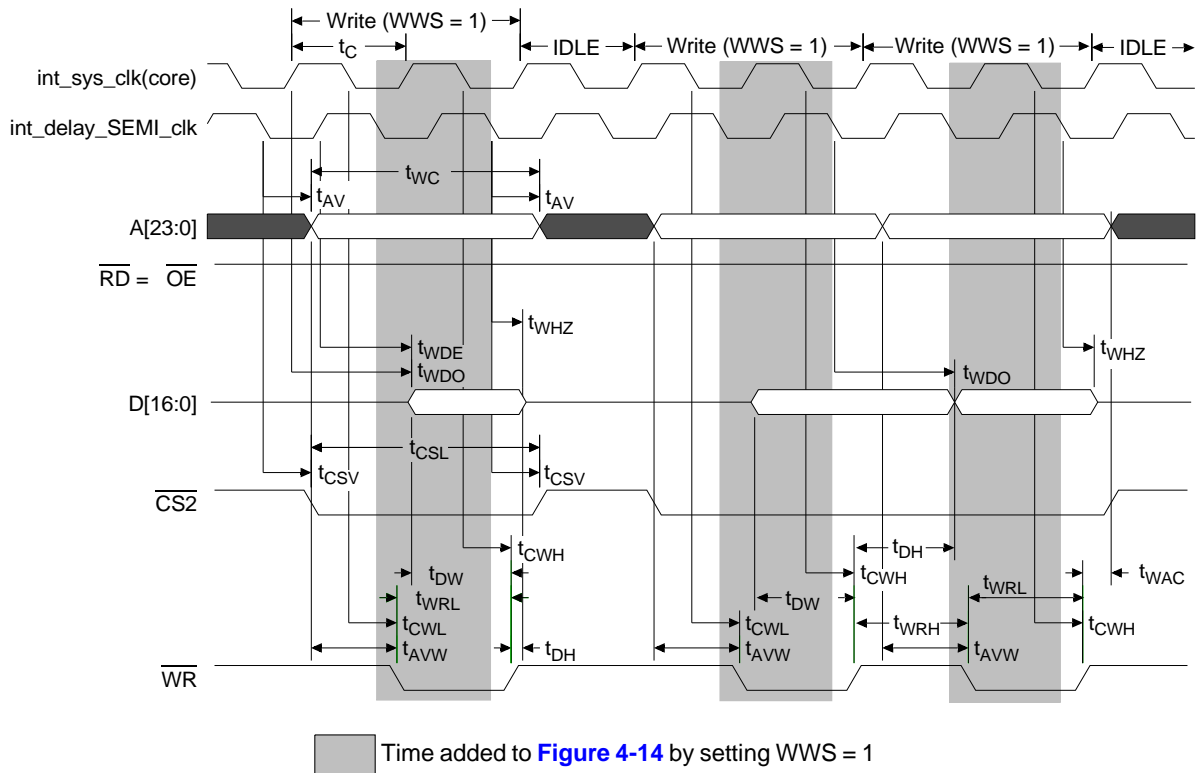
### 4.7.2 Write Timing

[Figure 4-14](#) shows the write timing for external memory access. For comparison, a single write cycle is shown followed by a null cycle and then a back-to-back write. This figure assumes zero wait states are required for the access.



**Figure 4-14. External Write Cycle**

**Note:** When  $\text{WWS} = 0$  the timing of the  $\overline{\text{WR}}$  strobe is generated from different clock edges than when it is set to some other value. This change in timing allows the possibility of single cycle write operation, but reduces the pulse width of  $\overline{\text{WR}}$  to one quarter clock. This may make it difficult to meet write timing requirements for most devices when operating at normal clock rates.



**Figure 4-15. External Write Cycle with WWS = 1, WWSH = 0, and WWS = 0**

### 4.7.2.1 Write Setup and Hold Timing

Since the timing of the strobes is different when WWS = 0 than it is when WWS > 0, two sets of timing diagrams are illustrated in [Figure 4-16](#), [Figure 4-17](#), [Figure 4-18](#), and [Figure 4-19](#).

### 4.7.2.2 WWS = 0

Although most memory devices require a zero setup and hold time, there are some peripheral devices where a setup/hold time is required. The WWS and WWSH field of the CSTC register provides the ability to allow for a write setup and/or hold time requirement as shown in [Figure 4-16](#) and [Figure 4-17](#).



Write (WWS=0,WWSH=1)

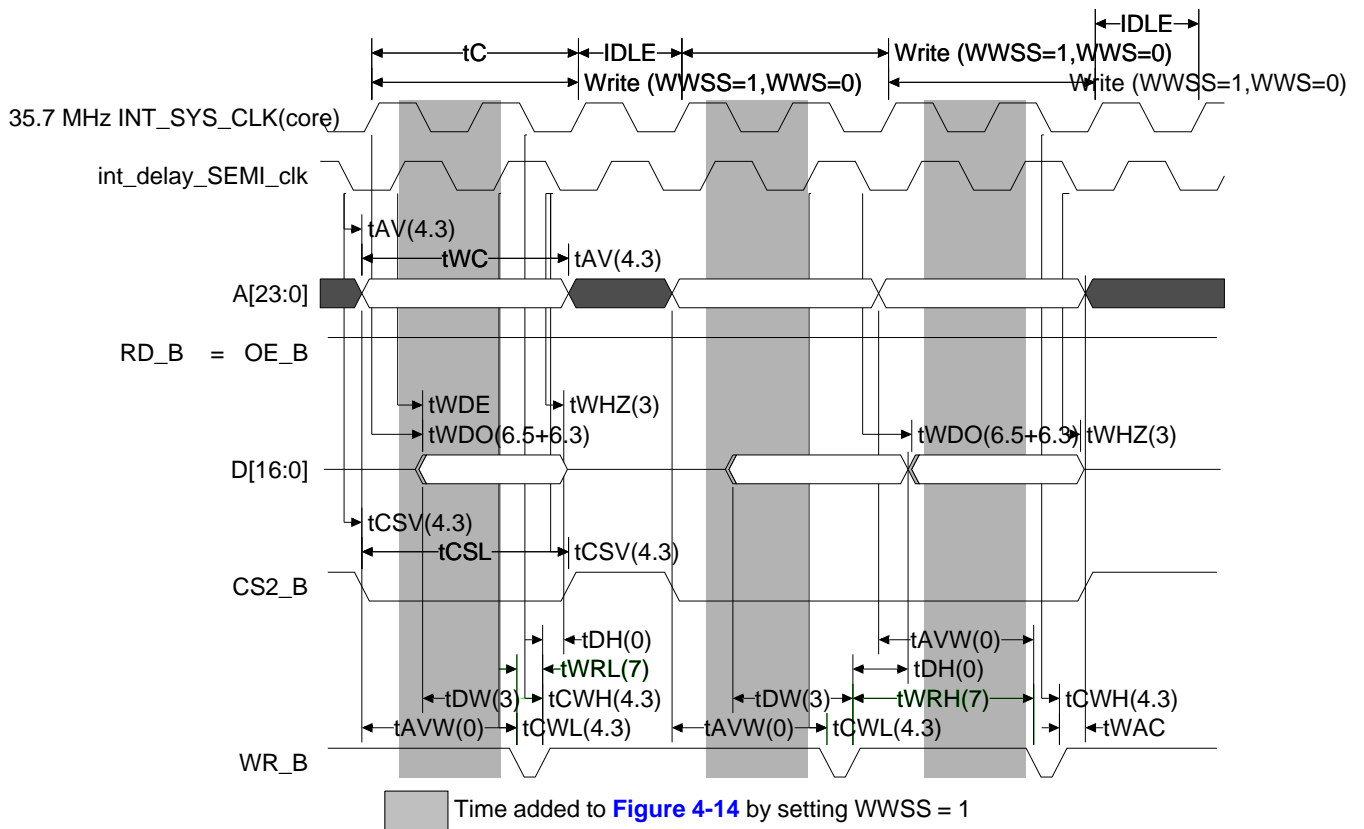
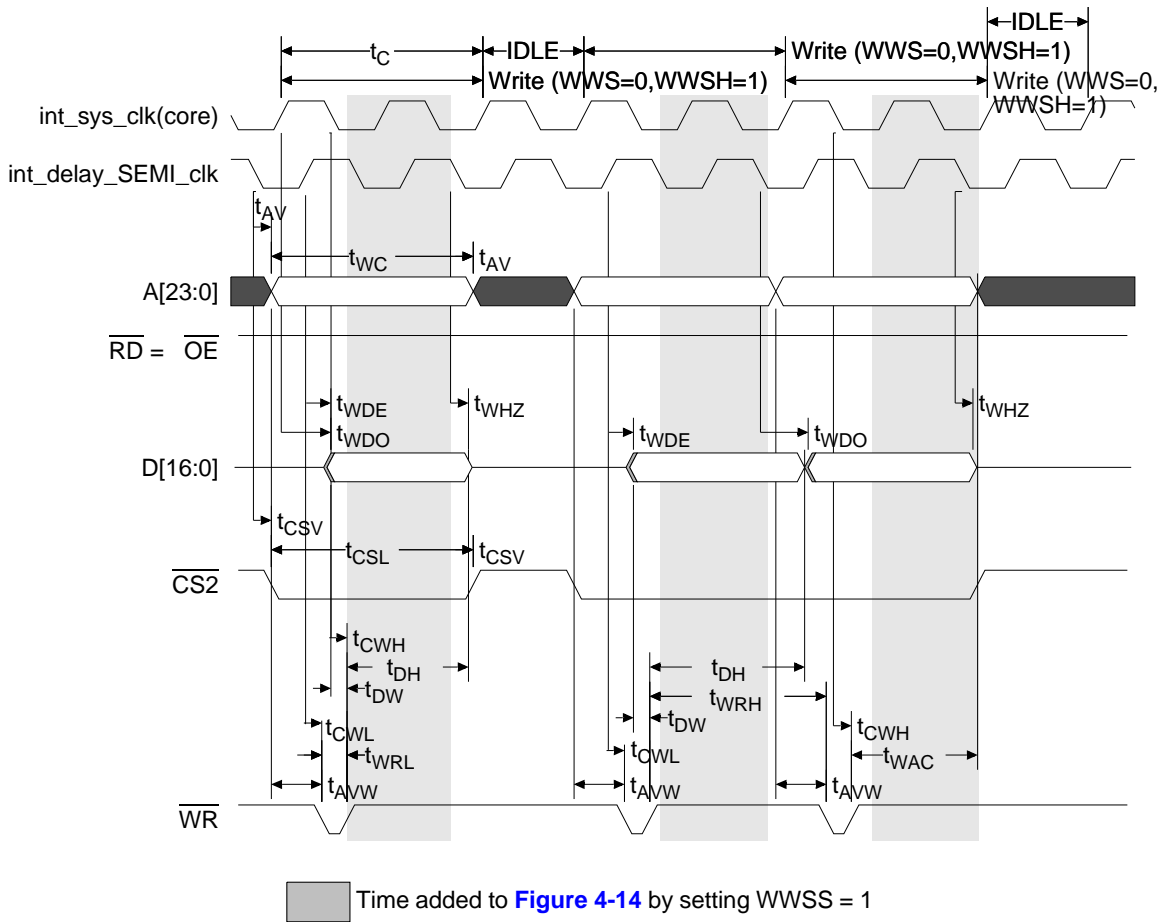


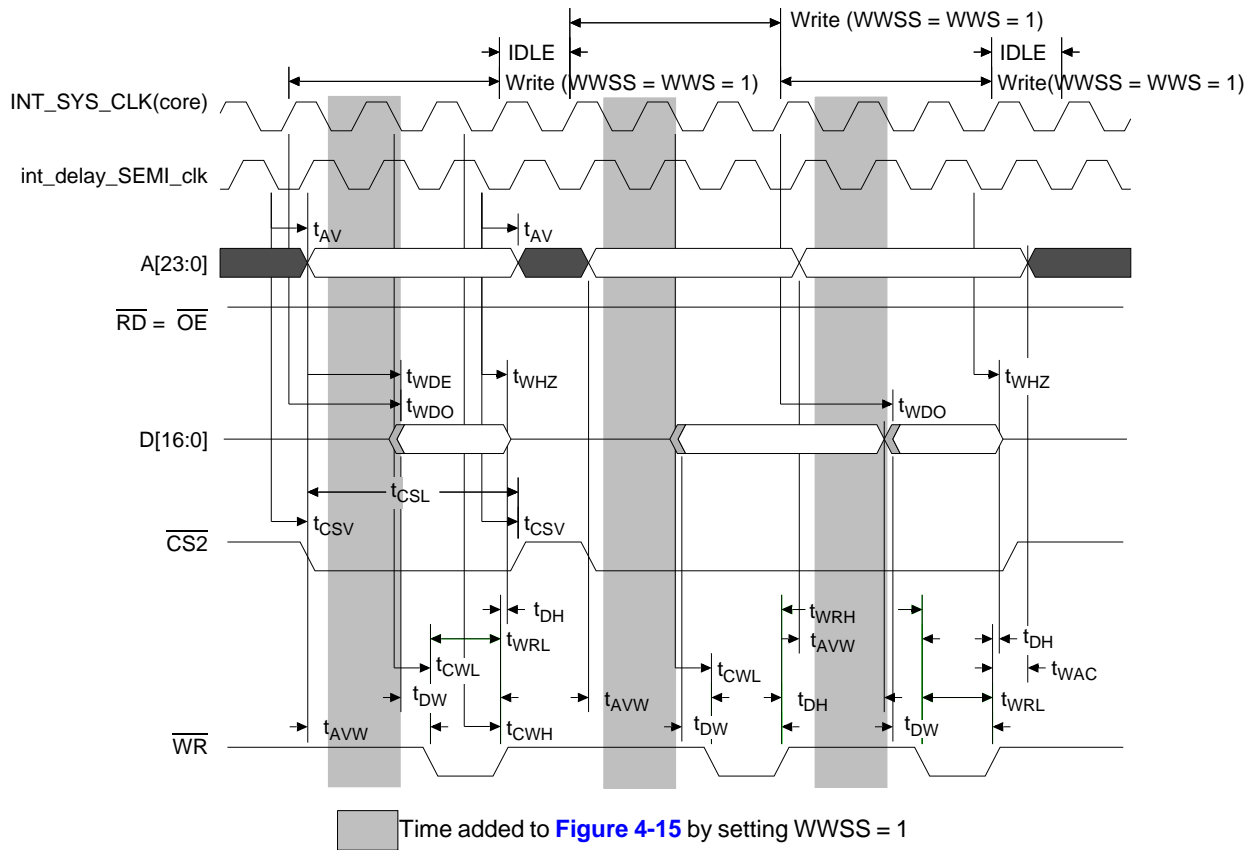
Figure 4-16. External Write Cycle with WWSS = 1, WWS = 0 and WWSH = 0



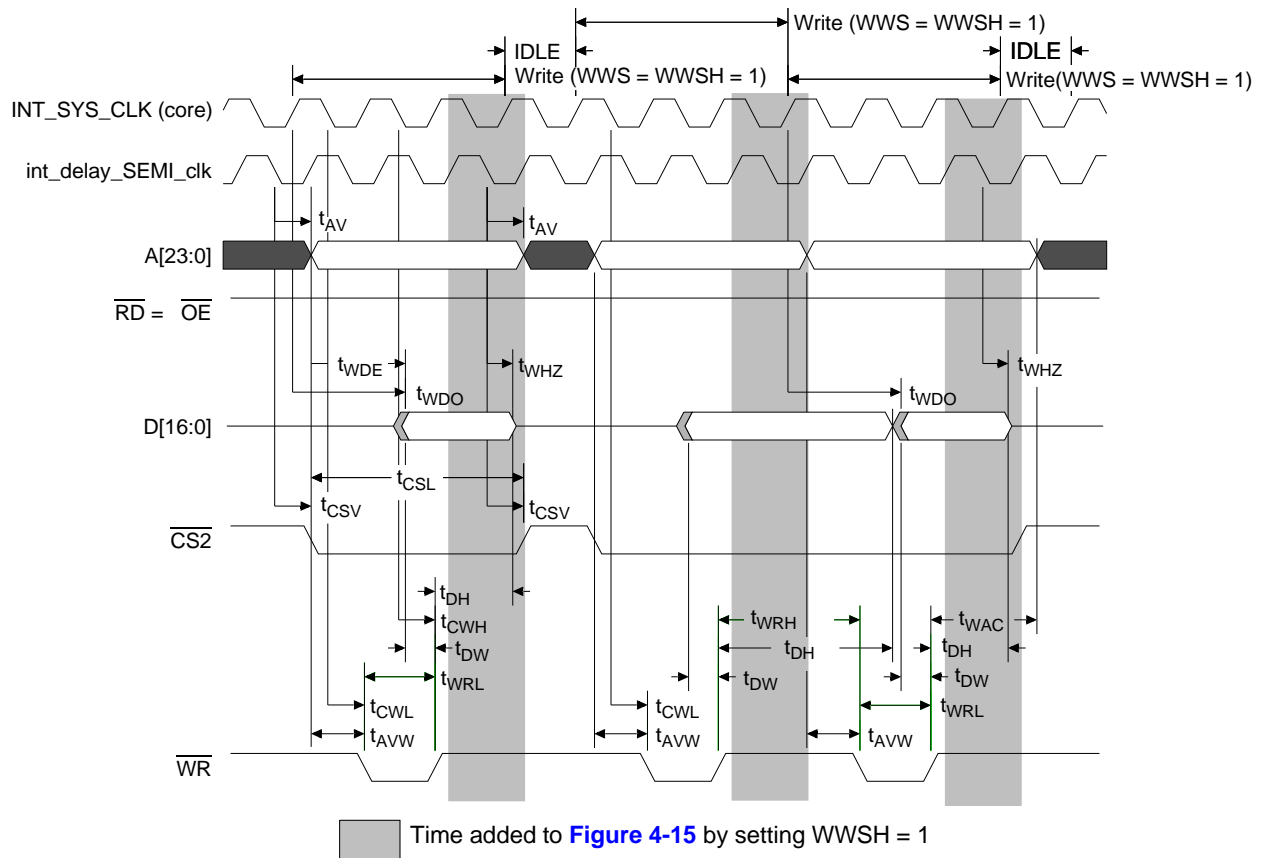
**Figure 4-17. External Write Cycle with WWS = 0, WWSH = 1, WWSS = 0**

### 4.7.2.3 WWS > 0

Although most memory devices require a zero setup and hold time, there are some peripheral devices where a setup/hold time is required. The WWS and WWSH field of the CSTC register provides the ability to allow for a write setup and/or hold time requirement as shown in [Figure 4-18](#) and [Figure 4-19](#) respectively.



**Figure 4-18. External Write Cycle with WWSS = WWS = 1 and WWSH = 0**



**Figure 4-19. External Write Cycle with WWS = WWSH = 1 (WWSS = 0)**

## 4.8 Clocks

The EMI operates from clocks internal to the chip and does not require/provide clocks external to the chip.

## 4.9 Interrupts

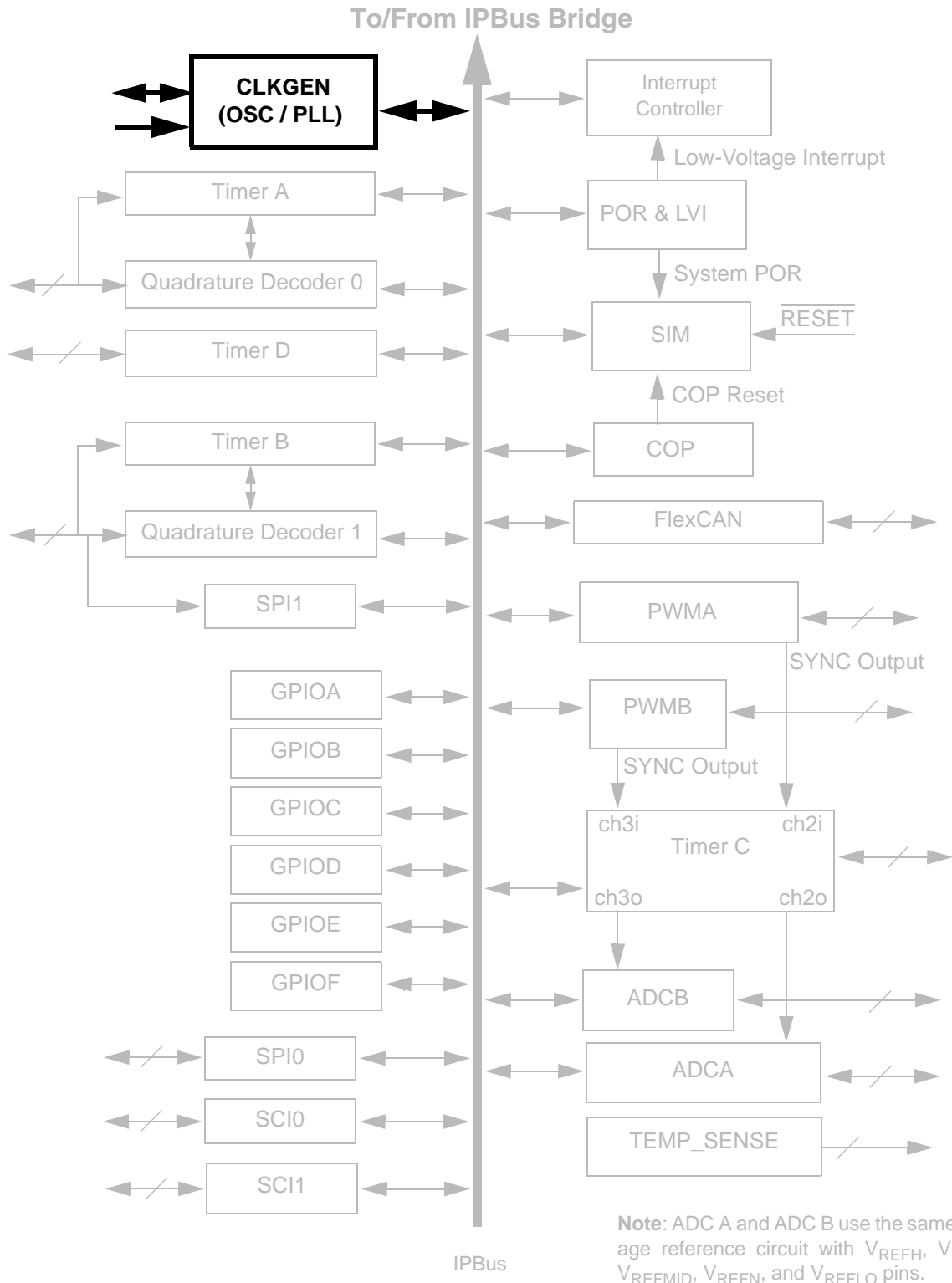
There are no interrupts generated by this module.

## 4.10 Resets

All reset types are equivalent for the EMI and therefore have the same effect. The EMI outputs during reset are controlled by the DRV bit of the BCR. During reset this bit is set to zero. Therefore, [Table 4-7](#) defines the reset state of all EMI pins.

# Chapter 5

## On-Chip Clock Synthesis (OCCS)



## Document Revision History for **Chapter 5, On-Chip Clock Synthesis (OCCS)**

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 4.0	Added note in Section 5.11.3.6 (LCK0)
Rev 5.0	Converted chapter to Freescale design standards Added reference to 56F8100 devices being 40MHz to introductory paragraph Corrected shaded areas in Table 5-3 Added Table 5-4 reflecting 56F8100 devices legal PLL settings Added footnotes to both tables 5-3 and 5-4 Added note to Section 5.11.2.5 referencing the 8100 family of devices
Rev 8.0	Minor layout edits.
Rev. 9	Deleted the step involving a change to low- or high-power mode from Table 5-1. In Section 5.5, replaced the sentence " <i>If a crystal is used on the board the power level of this oscillator should be lowered to prevent over driving the crystal and to reduce overall power consumption</i> " with " <i>When a crystal is used, the lower-power setting can be used when the crystal's equivalent series resistance (ESR) is less than 40Ω</i> ". Revised OSCTL[COHL] bit descriptions.



## 5.1 Introduction

The On-Chip Clock Synthesis (OCCS) module allows product design using inexpensive 8MHz crystals to run the 56F8300 at frequencies up to 60Mhz while 56F8100 devices are guaranteed to run at frequencies up to only 40MHz. This module provides the  $2 \times$  system clock frequency to the System Integration Module (SIM), using it to generate the various chip clocks. This module also produces the OSC\_CLK signal. For chip-specific details related to the OCCS module, please refer to the relevant data sheet.

## 5.2 Features

The OCCS module interfaces with the oscillator and PLL, as well as having an on-chip prescaler and relaxation oscillator. The OCCS module characteristics include:

- Oscillator can be crystal controlled or driven from an external clock generator
- Two-bit prescaler can divide oscillator output by 1, 2, 4, or 8 prior to its use as the PLL source clock
- Two-bit postscaler provides similar control for the PLL output
- Ability to power-down the internal PLL
- Provides 2x master clock frequency (SYS\_CLK\_x2) and oscillator clock (MSTR\_OSC) signals to the SIM module<sup>1</sup>
- Safety shutdown feature available in the event the PLL reference clock disappears
- Internal relaxation oscillator (not available on all parts)

The clock generation module provides the programming interface for both the PLL and on and off-chip oscillators and the on-chip relaxation oscillator.

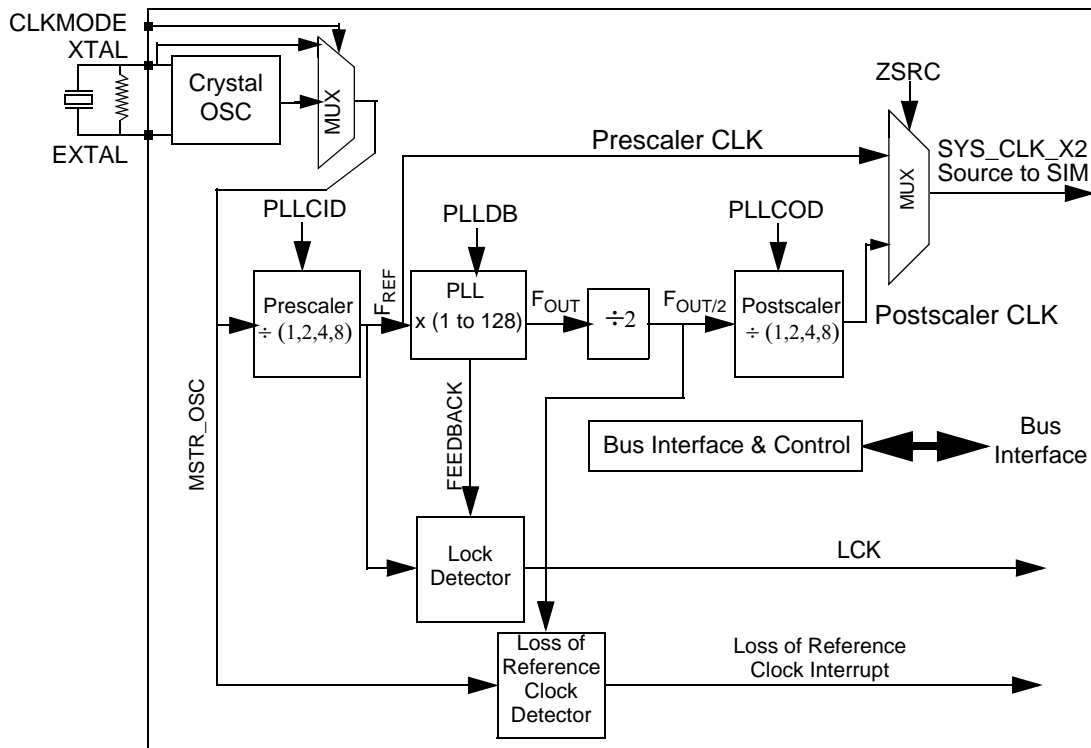
## 5.3 Block Diagram

**Figure 5-1** illustrates the clock generation module for those chips without an internal relaxation oscillator.

---

1. See the relevant data sheet for description of the SIM module.





**Figure 5-1. OCCS Block Diagram Without Relaxation Oscillator**

**Figure 5-2** illustrates the clock generation module for those chips with an internal relaxation oscillator.

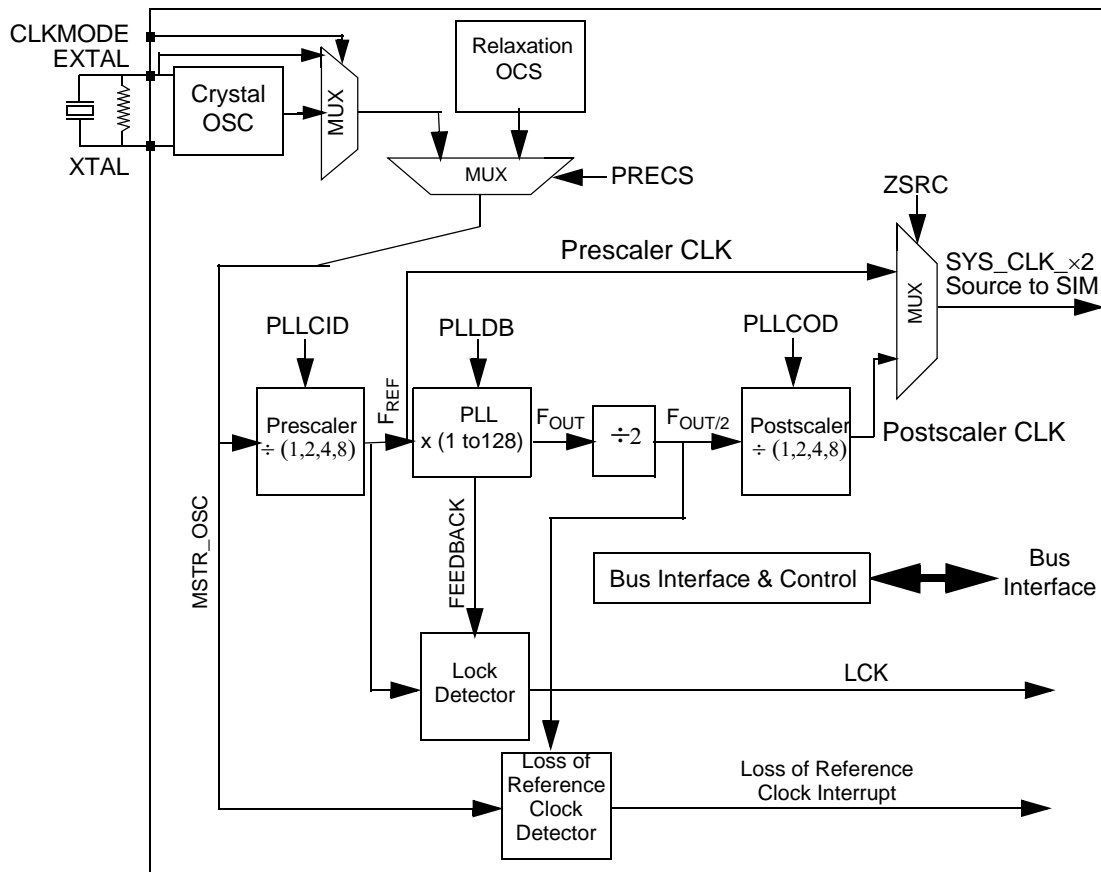


Figure 5-2. OCCS Block Diagram With Relaxation Oscillator

## 5.4 Functional Description

A block diagram of the OCCS module is illustrated in [Figure 5-1](#) or [Figure 5-2](#), depending on whether the relaxation oscillator is included on-chip. Possible clock source choices are:

- Internal relaxation oscillator (on some chips)
- External ceramic resonator
- External crystal
- External clock source

Each of these clock sources can be selected to drive the remainder of the clock generation circuitry. This circuitry allows direct use of the clock, taken from the prescaler output or the clock can be used as an input to the PLL. The PLL will generate a higher frequency clock for use within the chip. This allows two final clock output selections:

- Prescaler output

- Postscaler output

The clock multiplexer (ZSRC MUX) selects the prescaler clock on power-up. A different clock source can be selected by writing to the PLL Control Register (PLLCR), discussed in [Section 5.11.1](#). Once a new clock source is selected, the new clock will be activated within four clock periods of the new clock after the clock selection request is re-clocked by the current IPBus clock.

*Transitions to/from prescaler to postscaler frequencies are guaranteed to be glitch free. Changes from one prescaler value to another are guaranteed to be glitch-free.* Before changing the divide-by value for the PLL, the system clock must be first switched to the prescaler clock. After the PLL has locked, the hybrid controller core clock can be switched back to the PLL by writing to the ZSRC bits in the PLLCR.

The PLL Status Register (PLLSR), discussed in [Section 5.11.3](#), illustrates the status of the hybrid controller core clock source. Because the synchronizing circuit changes modes to avoid any glitches, the PLLSR ZCLOCK Source (ZSRC) will show overlapping modes as an intermediate step.

Frequencies going into and out of the PLL are controlled by the prescaler, postscaler, and the divide-by ratio within the PLL. For proper operation of the PLL, the Voltage Controlled Oscillator (VCO) within the PLL must be kept in its operational range of 160–260MHz, the output of the VCO is depicted as  $F_{OUT}$  in [Figure 5-1](#). Prescaling the input frequency as well as adjusting the divide-by ratio determines the frequency at which the VCO is running. The input frequency divided by the prescaler ratio, multiplied by the divide-by ratio, is the frequency at which the VCO is running.

The PLL lock time is 10ms or less when coming from a powered down state to a power-up state. It is recommended when changing the prescaler ratio or the divide-by ratio, powering down, or powering up, the PLL be deselected as the clocking source. Only after lock is achieved should the PLL be used as a valid clocking source.

[Table 5-1](#) provides possible clock sources and configurations for chips not including the on-chip relaxation oscillator. If this oscillator is included, use [Table 5-2](#).

**Table 5-1. Clock Choices without Relaxation Oscillator**

Clock Source	Selected Clock	Configuration Steps (After Reset)
Ceramic Resonator	Prescaler	Default—No action required 1. CLKMODE pin should be tied to $V_{SS}$ 2. Change PLLCID if desired
Ceramic Resonator	Postscaler	1. CLKMODE pin should be tied to $V_{SS}$ 2. Change PLLCID and PLLCOD if desired 3. Configure the PLL Divide-By (PLLDB) field for the desired output clock 4. Enable the PLL (PLLPB = 0) 5. Wait for PLL lock (LCK1 = 1 and LCK0 = 1) 6. Change ZSRC to select the postscaler clock (ZSRC = 10)
Crystal	Prescaler	1. CLKMODE pin should be tied to $V_{SS}$ 2. Change PLLCID if desired
Crystal	Postscaler	1. CLKMODE pin should be tied to $V_{SS}$ 2. Change PLLCID and PLLCOD if desired 3. Configure the PLLDB field for the desired output clock 4. Enable the PLL (PLLPD = 0) 5. Wait for PLL lock (LCK1 = 1 and LCK0 = 1) 6. Change ZSCR to select the postscaler clock (ZSRC = 10)
External Clock Source	Prescaler	1. CLKMODE pin should be tied to $V_{DD}$ No configuration change required 2. Change PLLCID if desired
External Clock Source	Postscaler	1. CLKMODE pin should be tied to $V_{DD}$ 2. Configure the PLLDB field for the desired output clock 3. Change PLLCID and PLLCOD if desired 4. Enable the PLL (PLLPD = 0) 5. Wait for PLL lock (LCK1 = 1 and LCK0 = 1) 6. Change ZSRC to select the postscaler clock (ZSRC = 10)

**Table 5-2. Clock Choices with Relaxation Oscillator**

Clock Source	Selected Clock	Configuration Steps (After Reset)
Relaxation Oscillator	Prescaler	Default—No action required 1. To obtain the desired clock rate change TRIM as required 2. Change PLLCID if desired
Relaxation Oscillator	Postscaler	1. To obtain the desired clock rate change TRIM as required 2. Change PLLCID and PLLCOD if desired 3. Configure the PLLDB field for the desired output clock 4. Enable the PLL (PLLPD = 0) 5. Wait for PLL lock (LCK1 = 1 and LCK0 = 1) 6. Change ZSRC to select the postscaler clock (ZSRC = 10)
Ceramic Resonator	Prescaler	1. The crystal oscillator must be power up (CLKMODE = 0 in OSCTL register) 2. Wait for crystal oscillator to stabilize (up to 10ms) 3. The clock source should be changed to the crystal oscillator (PRECS = 1) 4. To conserve power, the relaxation oscillator should be powered down (ROPD = 1) 5. Change PLLCID if desired
Ceramic Resonator	Postscaler	1. The crystal oscillator must be power up (CLKMODE = 0 in OSCTL register) 2. Wait for crystal oscillator to stabilize (up to 10ms) 3. The clock source should be changed to the crystal oscillator (PRECS = 1) 4. To conserve power, the relaxation oscillator should be powered down (ROPD = 1) 5. Change PLLCID and PLLCOD if desired 6. Configure the PLLDB field for the desired output clock frequency 7. Enable the PLL (PLLPD = 0) 8. Wait for PLL lock (LCK1 = 1 and LCK0 = 1) 9. Change ZSCR to select the postscaler clock (ZSRC = 10)
Crystal	Prescaler	1. The crystal oscillator must be power up (CLKMODE = 0 in OSCTL register) 2. Wait for crystal oscillator to stabilize (up to 10ms) 3. The clock source should be changed to the crystal oscillator (PRECS = 1) 4. To conserve power, the relaxation oscillator should be powered down (ROPD = 1) 5. Change PLLCID if desired
Crystal	Postscaler	1. The crystal oscillator must be power up (CLKMODE = 0 in OSCTL register) 2. Wait for crystal oscillator to stabilize (up to 10ms) 3. The clock source should be changed to the crystal oscillator (PRECS = 1) 4. To conserve power, the relaxation oscillator should be powered down (ROPD = 1) 5. Change PLLCID and PLLCOD if desired 6. Configure the PLLDB field for the desired output clock frequency 7. Enable the PLL (PLLPD = 0) 8. Wait for PLL lock (LCK1 = 1 and LCK0 = 1) 9. Change ZSRC to select the postscaler clock (ZSRC = 10)

**Table 5-2. Clock Choices with Relaxation Oscillator (Continued)**

Clock Source	Selected Clock	Configuration Steps (After Reset)
External Clock Source	Prescaler	<ol style="list-style-type: none"> <li>1. The clock source should be changed to the crystal oscillator (PRECS =1)</li> <li>2. To conserve power, the relaxation oscillator should be powered down (ROPD = 1)</li> <li>3. Change PLLCID if desired</li> </ol>
External Clock Source	Postscaler	<ol style="list-style-type: none"> <li>1. The clock source should be changed to the crystal oscillator (PRECS =1)</li> <li>2. To conserve power, the relaxation oscillator should be powered down (ROPD = 1)</li> <li>3. Configure PLLDB field for the desired output clock frequency</li> <li>4. Change PLLCID and PLLCOD if desired</li> <li>5. Enable the PLL (PLLPD = 0)</li> <li>6. Wait for PLL lock (LCK1 = 1 and LCK0 = 1)</li> <li>7. Change ZSRC to select the postscaler clock (ZSRC = 10)</li> </ol>

## 5.5 Crystal Oscillator

The crystal oscillator is designed to operate with either an external crystal or an external ceramic resonator. The ceramic oscillator requires a larger current source from the amplifier and this is the default. When a crystal is used, the lower-power setting can be used when the crystal's equivalent series resistance (ESR) is less than 40Ω. Please see the COHL bit in the Oscillator Control (OSCTL) register, discussed in [Section 5.11.5](#).

## 5.6 Relaxation Oscillator

### 5.6.1 Trimming Frequency on the Internal Relaxation Oscillator

The Internal Relaxation Oscillator frequency will vary as much as  $\pm 20$  percent due to process, temperature, and voltage dependencies. These dependencies are in the voltage and current references, the offset of the comparators, and the internal capacitor. Voltage and temperature dependencies are designed to be a maximum of approximately two percent error. The process dependencies account for the rest.

Fortunately for an individual part, process dependencies are constant. An individual part can operate at approximately two percent variance from its adjusted operating point over the entire specification range of the application. If the adjusted operating point can be changed, the entire variance can be limited to two percent.

The method of changing the adjusted operating point is by changing the size of the capacitor. This capacitor value is controlled by the Trim Factor (TRIM) in the OSCTL. The default value for TRIM is \$200. Each unit added or removed will adjust the output period by about 0.078% of the unadjusted period (adding to TRIM will increase the clock period, decreasing the frequency). With TRIM containing 10-bits, the clock period of the Relaxation Oscillator clock can be

changed to  $\pm 40\%$  of its unadjusted value, which is enough to cancel the process variability mentioned before.

The best way to trim the Internal Clock is to use the timer to measure the width of an input pulse on an input capture pin. This pulse must be supplied by the application and should be as long, or wide as possible. Considering the prescale value of the timer and the theoretical (zero error) frequency of the bus, the error can be calculated. This error, expressed as a percentage, can be divided by resolution of the trim, i.e. 0.078 percent and the resultant factor added or subtracted from TRIM. This process should be repeated to eliminate any residual error.

**Note:** Parts with the internal oscillator will have a TRIM value recorded in Flash at the factory. The customer's code needs only to copy it across to the TRIM register.

## 5.6.2 Switching Clock Sources

To robustly switch between the Internal Relaxation Oscillator Clock and the External Oscillator Clock, the changeover switch assumes the clocks are completely asynchronous, so a synchronizing circuit is required to make the transition. When the Prescaler Clock Select (PRECS) is changed, the switch will continue to operate off the original clock for between one and two cycles as the select input is transitioned through one side of the synchronizer. Please see [Section 5.11.1.11](#) for a discussion about PRECS. Next, the output will be held low for between one and two cycles of the new clock as the select input transitions through the other side. Then the output starts switching at the new clock's frequency. This transition guarantees no glitches will be seen on the output even though the select input may change asynchronously to the clocks. The unpredictability of the transition period is a necessary result of the asynchronicity. The switch automatically selects Internal Relaxation Oscillator Clock during hardware reset.

Switching from the Internal Relaxation Oscillator Clock to the Crystal Oscillator Clock source or vice-versa requires both clock sources to be enabled and stable. A simple flow requires:

- If switching to the crystal oscillator, be certain it was enabled via GPIO and is powered up (CLKMODE= 0), discussed in [Section 5.11.6.4](#)
- If switching to the relaxation oscillator, be certain it is powered up (ROPD = 0), discussed in [Section 5.11.6.1](#)
- Wait for a few cycles for the clock to become active
- Switch clocks
- Execute 4 NOP instructions
- Disable previous clock source, e.g. power-down the relaxation oscillator if the crystal is selected

The key point to remember in this flow is the clock source should not be switched unless the desired clock is on and stable.

When a new controller core clock is selected, the clock generation module will synchronize the request and select the new clock. Since the synchronizing circuit changes modes as to avoid any glitches, the ZSRCS bits in PLLSR will show overlapping modes as an intermediate step.

## 5.7 Phase Locked Loop

### 5.7.1 PLL Recommended Range of Operation

The Voltage Controlled Oscillator (VCO) within the PLL has a characterized operating range extending from 160MHz to 260MHz. The output of the PLL,  $F_{OUT}$  in [Figure 5-1](#), [Figure 5-2](#), and [Table 5-3](#) is divided by two then fed to the input of the postscaler. The PLL is programmable via a divide by  $n+1$  register, capable of taking on values varying between one and 128. The seven PLL Divide-By (PLLDB) bits determine this value, referenced in [Section 5.11.2](#). For higher values of  $n$ , PLL lock time becomes an issue. It is recommended to avoid values of  $n$  resulting in the VCO frequency greater than 260MHz or less than 160MHz. [Table 5-3](#) shows all legal combinations of PLL settings. The PLL is optimized for an  $F_{OUT}$  value of 240MHz.

**Note:** Values down to  $SYS\_CLK = 0.5\text{MHz}$  can be achieved directly from an 8MHz crystal simply by using the prescaler clock directly (with the PLL disabled).

The equations below were used to create the data in [Table 5-3](#).

$$\text{Prescaler} = 2^{(\text{PLLCID})}$$

Given a 8MHz input clock,  $F_{OUT}$  is determined by:

$$F_{OUT} = \frac{\text{PLLDB} + 1}{\text{Prescaler}} \times 8\text{MHz} = \frac{\text{PLLDB} + 1}{2^{\text{PLLCID}}} \times 8\text{MHz}$$

The system clock is determined from  $F_{OUT}$ :

$$SYS\_CLK\_x2 = \frac{1}{\text{Postscaler}} \times \frac{F_{OUT}}{2} = \frac{1}{2^{\text{PLLCOD}}} \times \frac{F_{OUT}}{2} = \frac{F_{OUT}}{2^{(\text{PLLCOD}+1)}}$$



Substituting for  $F_{OUT}$ ,

$$SYS\_CLK\_x2 = \frac{1}{2^{(PLL\text{COD}+1)}} \times \frac{PLLDB+1}{2^{PLL\text{CID}}} \times 8\text{MHz}$$

$$SYS\_CLK\_x2 = \frac{PLLDB+1}{2^{(PLL\text{COD}+PLL\text{CID})}} \times 4\text{MHz}$$

$$SYS\_CLK = \frac{SYS\_CLK\_x2}{2}$$

This is the value shown in [Table 5-3](#).

**Table 5-3. Legal PLL Settings for 56F8300 Family<sup>1</sup>**

PLL Input Config.		PLLDB <sup>2</sup>	PLL F <sub>OUT</sub> (MHz)	SYS_CLK Operating Frequency (MHz)			
PLLCID	Prescaler			Postscaler=1 (PLLCOD=0)	Postscaler=2 (PLLCOD=1)	Postscaler=4 (PLLCOD=2)	Postscaler=8 (PLLCOD=3)
0	1	19	160	40.000	20.000	10.000	5.000
0	1	20	168	42.000	21.000	10.500	5.250
0	1	21	176	44.000	22.000	11.000	5.500
0	1	22	184	46.000	23.000	11.500	5.750
0	1	23	192	48.000	24.000	12.000	6.000
0	1	24	200	50.000	25.000	12.500	6.250
0	1	25	208	52.000	26.000	13.000	6.500
0	1	26	216	54.000	27.000	13.500	6.750
0	1	27	224	56.000	28.000	14.000	7.000
0	1	28	232	58.000	29.000	14.500	7.250
0	1	29	240	<b>60.000</b>	30.000	15.000	7.500
0	1	30	248		31.000	15.500	7.750
0	1	31	256		32.000	16.000	8.000
1	2	39	160	40.000	20.000	10.000	5.000
1	2	40	164	41.000	20.500	10.250	5.125
1	2	41	168	42.000	21.000	10.500	5.250
1	2	42	172	43.000	21.500	10.750	5.375
1	2	43	176	44.000	22.000	11.000	5.500

Table 5-3. Legal PLL Settings for 56F8300 Family<sup>1</sup>

PLL Input Config.		PLLDDB <sup>2</sup>	PLL F <sub>OUT</sub> (MHz)	SYS_CLK Operating Frequency (MHz)			
PLLCID	Prescaler			Postscaler=1 (PLLCOD=0)	Postscaler=2 (PLLCOD=1)	Postscaler=4 (PLLCOD=2)	Postscaler=8 (PLLCOD=3)
1	2	44	180	45.000	22.500	11.250	5.625
1	2	45	184	46.000	23.000	11.500	5.750
1	2	46	188	47.000	23.500	11.750	5.875
1	2	47	192	48.000	24.000	12.000	6.000
1	2	48	196	49.000	24.500	12.250	6.125
1	2	49	200	50.000	25.000	12.500	6.250

Note: Shaded area reflects recommended setting

1	2	50	204	51.000	25.500	12.750	6.375
1	2	51	208	52.000	26.000	13.000	6.500
1	2	52	212	53.000	26.500	13.250	6.625
1	2	53	216	54.000	27.000	13.500	6.750
1	2	54	220	55.000	27.500	13.750	6.875
1	2	55	224	56.000	28.000	14.000	7.000
1	2	56	228	57.000	28.500	14.250	7.125
1	2	57	232	58.000	29.000	14.500	7.250
1	2	58	236	59.000	29.500	14.750	7.375
1	2	59	240	<b>60.000</b>	30.000	15.000	7.500
1	2	60	244		30.500	15.250	7.625
1	2	61	248		31.000	15.500	7.750
1	2	62	252		31.500	15.750	7.875
1	2	63	256		32.000	16.000	8.000
1	2	64	260		32.500	16.250	8.125

Note: Shaded area reflects recommended setting

1. Assumes 8MHz input clock
2. Only values yielding operational F<sub>OUT</sub> frequencies are shown.

Table 5-4. Legal PLL Settings for 56F8100 Family<sup>1</sup>

PLL Input Config.		PLLDDB <sup>2</sup>	PLL F <sub>OUT</sub> (MHz)	SYS_CLK Operating Frequency (MHz)			
PLLCID	Prescaler			Postscaler=1 (PLLCOD=0)	Postscaler=2 (PLLCOD=1)	Postscaler=4 (PLLCOD=2)	Postscaler=8 (PLLCOD=3)
0	1	19	160	40.000	20.000	10.000	5.000
0	1	20	168		21.000	10.500	5.250
0	1	21	176		22.000	11.000	5.500
0	1	22	184		23.000	11.500	5.750
0	1	23	192		24.000	12.000	6.000
0	1	24	200		25.000	12.500	6.250
0	1	25	208		26.000	13.000	6.500
0	1	26	216		27.000	13.500	6.750
0	1	27	224		28.000	14.000	7.000
0	1	28	232		29.000	14.500	7.250
0	1	29	240		30.000	15.000	7.500
0	1	30	248		31.000	15.500	7.750
0	1	31	256		32.000	16.000	8.000
1	2	39	160	40.000	20.000	10.000	5.000
1	2	40	164		20.500	10.250	5.125
1	2	41	168		21.000	10.500	5.250
1	2	42	172		21.500	10.750	5.375
1	2	43	176		22.000	11.000	5.500
1	2	44	180		22.500	11.250	5.625
1	2	45	184		23.000	11.500	5.750
1	2	46	188		23.500	11.750	5.875
1	2	47	192		24.000	12.000	6.000
1	2	48	196		24.500	12.250	6.125
1	2	49	200		25.000	12.500	6.250
1	2	50	204		25.500	12.750	6.375
1	2	51	208		26.000	13.000	6.500
1	2	52	212		26.500	13.250	6.625
1	2	53	216		27.000	13.500	6.750
1	2	54	220		27.500	13.750	6.875
1	2	55	224		28.000	14.000	7.000
1	2	56	228		28.500	14.250	7.125
1	2	57	232		29.000	14.500	7.250
1	2	58	236		29.500	14.750	7.375
1	2	59	240		30.000	15.000	7.500
1	2	60	244		30.500	15.250	7.625
1	2	61	248		31.000	15.500	7.750
1	2	62	252		31.500	15.750	7.875
1	2	63	256		32.000	16.000	8.000
1	2	64	260		32.500	16.250	8.125

Table 5-4. Legal PLL Settings for 56F8100 Family<sup>1</sup>

PLL Input Config.		PLLD <sup>2</sup>	PLL F <sub>OUT</sub> (MHz)	SYS_CLK Operating Frequency (MHz)			
PLLCID	Prescaler			Postscaler=1 (PLLCOD=0)	Postscaler=2 (PLLCOD=1)	Postscaler=4 (PLLCOD=2)	Postscaler=8 (PLLCOD=3)
2	4	79	160	40.000	20.000	10.000	5.000
2	4	80	162		20.250	10.125	5.0625
2	4	81	164		20.500	10.250	5.125
2	4	82	166		20.750	10.375	5.1875
2	4	83	168		21.000	10.500	5.250
2	4	84	170		21.250	10.625	5.3125
2	4	85	172		21.500	10.750	5.375
2	4	86	174		21.750	10.875	5.4375
2	4	87	176		22.000	11.000	5.500
2	4	88	178		22.250	11.125	5.5625
2	4	89	180		22.500	11.250	5.625
2	4	90	182		22.750	11.375	5.6875
2	4	91	184		23.000	11.500	5.750
2	4	92	186		23.250	11.625	5.8125
2	4	93	188		23.500	11.750	5.875
2	4	94	190		23.750	11.875	5.9375
2	4	95	192		24.000	12.000	6.000
2	4	96	194		24.250	12.125	6.0625
2	4	97	196		24.500	12.250	6.125
2	4	98	198		24.750	12.375	6.1875
2	4	99	200		25.000	12.500	6.250
2	4	100	202		25.250	12.625	6.3125
2	4	101	204		25.500	12.750	6.375
2	4	102	206		25.750	12.875	6.4375
2	4	103	208		26.000	13.000	6.500
2	4	104	210		26.250	13.125	6.5625
2	4	105	212		26.500	13.250	6.625
2	4	106	214		26.750	13.375	6.6875
2	4	107	216		27.000	13.500	6.750
2	4	108	218		27.250	13.625	6.8125
2	4	109	220		27.500	13.750	6.875
2	4	110	222		27.750	13.875	6.9375
2	4	111	224		28.000	14.000	7.000
2	4	112	226		28.250	14.125	7.0625
2	4	113	228		28.500	14.250	7.125
2	4	114	230		28.750	14.375	7.1875
2	4	115	232		29.000	14.500	7.250
2	4	116	234		29.250	14.625	7.3125
2	4	117	236		29.500	14.750	7.375

**Table 5-4. Legal PLL Settings for 56F8100 Family<sup>1</sup>**

PLL Input Config.		PLLDB <sup>2</sup>	PLL F <sub>OUT</sub> (MHz)	SYS_CLK Operating Frequency (MHz)			
PLLCID	Prescaler			Postscaler=1 (PLLCOD=0)	Postscaler=2 (PLLCOD=1)	Postscaler=4 (PLLCOD=2)	Postscaler=8 (PLLCOD=3)
2	4	118	238		29.750	14.875	7.4375
2	4	119	240		30.000	15.000	7.500
2	4	120	242		30.250	15.125	7.5625
2	4	121	244		30.500	15.250	7.625
2	4	122	246		30.750	15.375	7.6875
2	4	123	248		31.000	15.500	7.750
2	4	124	250		31.250	15.625	7.8125
2	4	125	252		31.500	15.750	7.875
2	4	126	254		31.750	15.875	7.9375
2	4	127	256		32.000	16.000	8.000

**Note:** Shaded area reflects recommended setting

1. Assumes 8MHz input clock
2. Only values yielding operational F<sub>OUT</sub> frequencies are shown.

## 5.7.2 PLL Lock Time Specification

In many applications, the lock time of the PLL is the most critical PLL design parameters. Proper use of the PLL ensures the highest stability and lowest lock time.

### 5.7.2.1 4.3.2.1 Lock Time Definition

Typical control systems refer to the lock time as the reaction time within specified tolerances of the system to a step input. In a PLL, the step input occurs when the PLL is turned on or when it suffers a noise hit. The tolerance is usually specified as a percent of the step input or when the output settles to the desired value plus or minus a given percent of the frequency change. Therefore, the reaction time is constant in this definition, regardless of the size of the step input.

When the PLL is coming from a powered down state, (PLL\_PDN high<sup>1</sup>), to a powered up condition, (PLL\_PDN low), the maximum lock time, with a divide-by count of 16 or less, is 10ms. Other systems refer to lock time as the time the system takes to reduce the error between the actual output and the desired output to within specified tolerances. Therefore, the lock time varies according to the original error in the output. Minor errors may be shorter or longer in many cases.

<sup>1</sup> PLL Power-Down Status (PLL\_PDN) – [Section 5.11.3.7](#)

### 5.7.2.2 Parametric Influences on Reaction Time

Lock time is designed to be as short as possible while still providing the highest possible stability. The reaction time is not constant, however. Many factors directly and indirectly affect the lock time.

The most critical parameter affecting the reaction time of the PLL is the reference frequency,  $F_{REF}$ , illustrated in [Figure 5-1](#).

$$F_{REF} = \frac{\text{Input Clock}}{2^{PLL\text{CID}}} = \frac{8\text{MHz}}{2^{PLL\text{CID}}}$$

This frequency is the input to the phase detector, controlling how often the PLL makes corrections. To assure stability, it is desirable for corrections to be small and frequent. Therefore, a higher reference frequency provides optimal performance; 8MHz is preferred. Thus  $PLL\text{CID} = 0$  is preferred over other values.

## 5.8 PLL Frequency Lock Detector Block

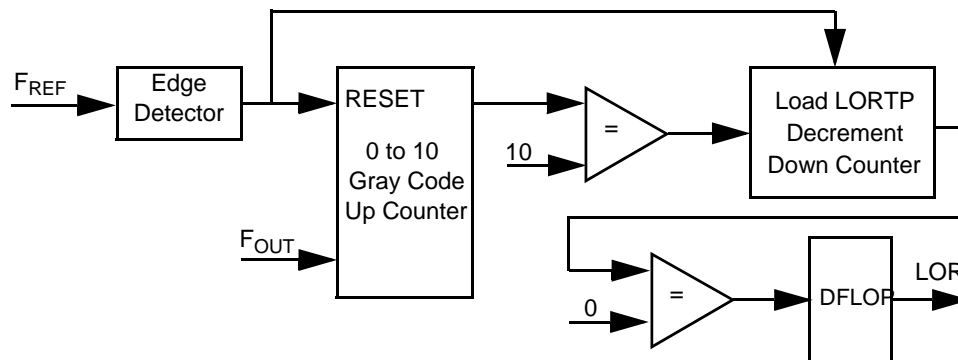
This digital block (please see [Figure 5-1](#)) monitors the VCO output clock, setting the LCK0 and LCK1 bit fields in the PLLSR based on its frequency accuracy. The lock detector is enabled with the LCKON bit of the PLLCR, as well as the PLLPD bit of PLLCR. Once enabled, the detector starts two counters whose outputs are periodically compared. The input clocks to these counters are the VCO output clock divided by  $(PLL\text{DB} + 1)$ , called FEEDBACK, and the PLL input clock. The period of the pulses being compared cover one whole period of each clock. This is due to the feedback clock not guaranteeing a 50 percent duty cycle. The design of this block was accomplished with the assumption the feedback clock transitions high on the rising edge of  $F_{REF}$ . Feedback and  $F_{REF}$  clocks are compared after 16, 32, and 64 cycles. If, after 32 cycles, the clocks match, the LCK0 bit is set to one. If, after 64 cycles of  $F_{REF}$ , there are the same number of  $F_{REF}$  clocks and feedback clocks, the LCK1 bit is also set. The LCK bit remains set until:

- Clocks fail to match
- When a new value is written to the PLLDB-factor
- On reset caused by LCKON, PLLPD
- Chip level reset

When the circuit sets the LCK1, the two counters are reset and start the count again. The lock detector is designed so if LCK1 is reset to zero because clocks did not match, LCK0 can stay high. This provides the processor the accuracy of the two clocks with respect to each other.

## 5.9 Loss of Reference Clock Detector

The Loss of Reference Clock Detector is designed to generate an interrupt when the reference clock to the PLL is interrupted. For instance, this might occur if the external crystal is suddenly physically damaged. An LOR interrupt should occur after  $LORTP \times 10 \times \text{PLL-clock-time-periods}$ . **Figure 5-3** illustrates the general operation of the LOR detector, relying on the phase locked loop can continue running for a time after its reference clock has been disturbed. This provides time for detection of the problem and an orderly system shutdown.



**Figure 5-3. Simplified Block Diagram, Loss of Reference Clock Detector**

## 5.10 Operating Modes

Either an external crystal, external ceramic resonator, or an external frequency source can be used for the 2x System Clock (SYS\_CLK\_X2). In some parts, an internal relaxation oscillator can also be used to provide this clock.

The 2x System Clock source output from the OCCS can be described with one of the following two equations:

if ZSRC = 01;  $\text{SYS\_CLK\_X2} = (\text{oscillator frequency})/\text{prescaler}$

if ZSRC = 10;  $\text{SYS\_CLK\_X2} = (\text{oscillator frequency} \times (\text{PLLDB} + 1))/(\text{2} \times \text{prescaler} \times \text{postscaler})$

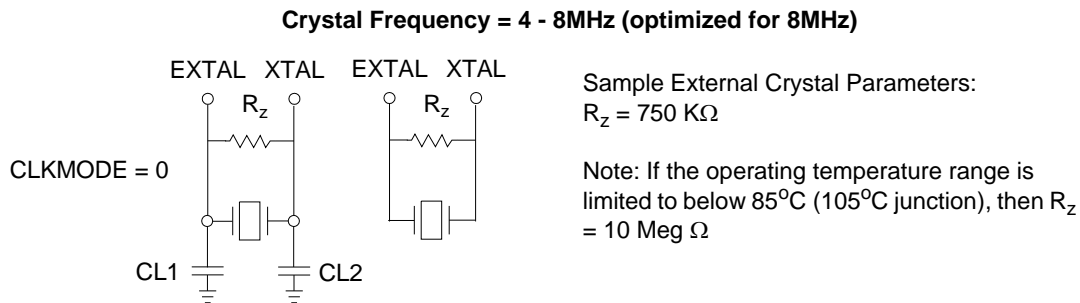
Where:

- PLLDB = the PLL divide-by ratio
- Prescaler = 1, 2, 4 or 8 PLL input frequency divider =  $2^{\text{PLLCID}}$
- Postscaler = 1, 2, 4 or 8 PLL output divider =  $2^{\text{PLLCOD}}$

The SIM is responsible for further dividing these frequencies by two, ensuring a 50 percent duty cycle in the system clock output.

### 5.10.1 Crystal Oscillator

The Internal Crystal Oscillator circuit is designed to interface with a parallel resonant crystal resonator. The frequency range, specified for the external crystal, is 4-8MHz. **Figure 5-4** illustrates a typical crystal oscillator circuit. Follow the crystal supplier's recommendations when selecting a crystal, because crystal parameters determine the component values required to provide maximum stability and reliable start-up. The load capacitance values used in the oscillator circuit design should include all stray layout capacitances. The crystal and associated components should be mounted as close as possible to the EXTAL and XTAL pins to minimize output distortion and start-up stabilization time.

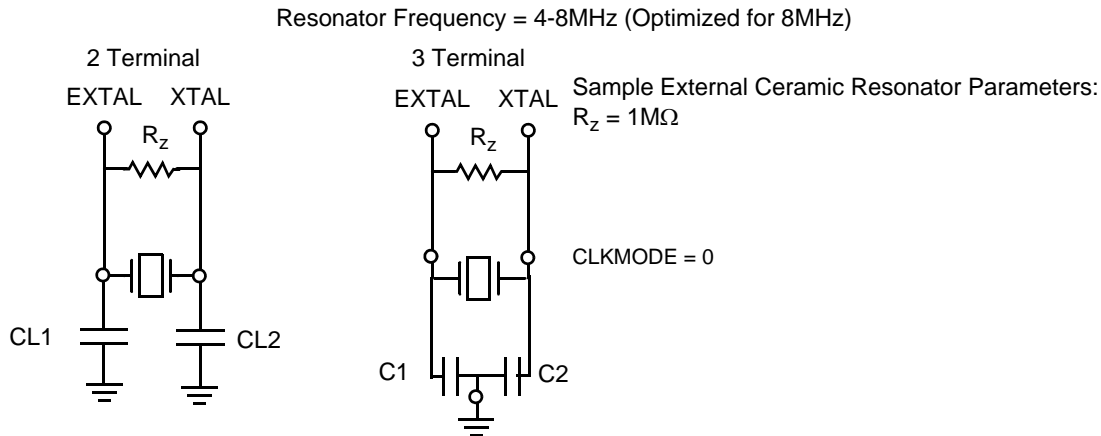


**Figure 5-4. External Crystal Oscillator Circuit**

### 5.10.2 Ceramic Resonator

The Internal Crystal Oscillator circuit is also designed to interface with a ceramic resonator in the frequency range of 4-8MHz. **Figure 5-5** illustrates the typical two and three terminal ceramic resonators and their circuits. Follow the resonator supplier's recommendations when selecting a resonator, since these parameters determine the component values required to provide maximum stability and reliable start-up. The load capacitance values used in the resonator circuit design should include all stray layout capacitances. The resonator and associated components should be mounted as close as possible to the EXTAL and XTAL pins to minimize output distortion and start-up stabilization time.

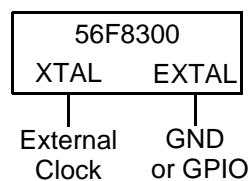




**Figure 5-5. External Ceramic Resonator Circuit**

### 5.10.3 External Clock Source

The recommended method of connecting an External Clock is illustrated in [Figure 5-6](#). The External Clock Source is connected to XTAL and the EXTAL pin is grounded. The External Clock input must be generated using a relatively low impedance driver, as the XTAL pin is actually the output pin of the oscillator, containing a very weak driver.



**Note:** When using an external clocking source with this configuration, CLKMODE should be high and COHL = 1 in the OSCTL register.

**Figure 5-6. Connecting an External Clock Signal Using XTAL**

### 5.10.4 Internal Clock Source

Some chips will have an internal relaxation oscillator which can be used as a clock source when clock accuracy is not critically important. The internal oscillator has very little variability with temperature and voltage, but it does vary as much as  $\pm 20$  percent as a function of wafer fabrication process. It also is very fast (well under one  $\mu\text{sec}$ ) in reaching a stable frequency. During the reset sequence, the Internal Oscillator will be enabled by default on these chips. Application code can then switch to the Crystal Oscillator/External Source and power-down the Internal Oscillator if desired.

## 5.11 Register Definitions

**Table 5-5. OCCS Memory Map**

Device	Peripheral	Address
8100/8300	CLKGEN_BASE	\$00F2D0

A register address is the sum of a base address and an address offset. The base address<sup>1</sup> is defined at the system level and the address offset is defined at the module level. The OCCS has five usable registers. [Table 5-6](#) summarizes these five registers. The definitions of the PLLCR and OSCTL registers depend on whether the chip contains an on-chip Relaxation Oscillator.

**Table 5-6. OCCS Register Summary**

Address Offset	Address Acronym	Register Name	Access Type	Chapter Location
Base + \$0	PLLCR	Control Register	Read/Write	<a href="#">Section 5.11.1</a>
Base + \$1	PLLDB	Divide-By Register	Read/Write	<a href="#">Section 5.11.2</a>
Base + \$2	PLLSR	Status Register	Read/Write	<a href="#">Section 5.11.3</a>
		Reserved		
Base + \$4	SHUTDOWN	Shutdown Register	Read/Write	<a href="#">Section 5.11.4</a>
Base + \$5	OSCTL	Oscillator Control Register	Read/Write	<a href="#">Section 5.11.5</a>

Bit fields of each of the five registers are illustrated in [Figure 5-7](#). Details of each follow.

---

1. Base Address: See the data sheet for the OCCS module base address.

Addr. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$0 (w/o)	PLLCR	R	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	CHPMPTRI	0	PLLPD	0	0	ZSRC	
\$0 (w/)	PLLCR	R	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	CHPMPTRI	0	PLLPD	0	PRECS	ZSRC	
\$1	PLLDDB	R	LORTP				PLLCOD		PLLCID		0	PLLDDB						
\$2	PLLSR	R	LOLI1	LOLI0	LOCI	0	0	0	0	0	0	LCK1	LCK0	PLL PDN	0	0	ZSRCS	
		W	RESERVED															
\$4	SHUTDOWN	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W	SHUTDOWN															
\$5 (w/o)	OSCTL	R	0	0	COHL	0	0	0	0	0	0	0	0	0	0	0	0	0
		W	RESERVED															
\$5 (w/)	OSCTL	R	ROPD	0	COHL	CLK MODE	0	0	TRIM									
		W	RESERVED															

R	0	Read as 0
W		Reserved

Figure 5-7. OCCS Register Map Summary

### 5.11.1 PLL Control Register (PLLCR)

Base + \$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	CHPMPTRI	0	PLLPD	0	0	ZSRC	
Write	RESERVED															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Figure 5-8. PLL Control Register (PLLCR)–Without Relaxation Oscillator

Base + \$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	CHPMPTRI	0	PLLPD	0	PRECS	ZSRC	
Write	RESERVED															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Figure 5-9. PLL Control Register (PLLCR)–With Relaxation Oscillator

### 5.11.1.1 PLL Interrupt Enable 1 (PLLIE1)—Bits 15–14

An interrupt can be generated when the Fine PLL Lock (LCK1) status bit in the PLLSR changes:

- 00 = Disable interrupt
- 01 = Enable interrupt on any rising edge of LCK1
- 10 = Enable interrupt on falling edge of LCK1
- 11 = Enable interrupt on any edge change of LCK1

### 5.11.1.2 PLL Interrupt Enable 0 (PLLIE0)—Bits 13–12

An interrupt can be generated if the Coarse PLL Lock (LCK0) status bit in the PLLSR changes:

- 00 = Disable interrupt
- 01 = Enable interrupt on any rising edge of LCK0
- 10 = Enable interrupt on falling edge of LCK0
- 11 = Enable interrupt on any edge change of LCK0

### 5.11.1.3 Loss of Reference Clock Interrupt Enable (LOCIE)—Bit 11

Loss of the reference clock circuit monitors the output of the selected clock source. In the event of reference clock loss, an interrupt can be generated.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

### 5.11.1.4 Reserved—Bits 10–8

This bit field is reserved or not implemented. It is read and written as 0.

### 5.11.1.5 Lock Detector On (LCKON)—Bit 7

- 0 = Lock detector disabled
- 1 = Lock detector enabled

### 5.11.1.6 Charge Pump Tri-State (CHPMPTRI)—Bit 6

During normal chip operation the CHPMPTRI bit should be set to a value of zero. In the event of loss of reference clock, the CHPMPTRI bit must be set to a value of one.

- 0 = Normal operation
- 1 = Isolates the charge pump from the loop filter allowing the PLL output to slowly drift, thereby providing enough time to shutdown the chip. Activating this bit will render the PLL inoperable and should not be executed during standard operation of the chip.

### 5.11.1.7 Reserved—Bit 5

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 5.11.1.8 PLL Power-Down (PLLPD)—Bit 4

The PLL can be turned off by setting the PLLPD bit. There is a four IPBus clock delay from changing the bit to signaling the PLL. When the PLL is powered down, the gear shifting logic automatically switches to ZSRC = 1, preventing loss of reference clock to the core.

- 0 = PLL enabled
- 1 = PLL powered down

### 5.11.1.9 Reserved (w/o Relaxation Oscillator)—Bits 3–2

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 5.11.1.10 Reserved (w/ Relaxation Oscillator)—Bit 3

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 5.11.1.11 Prescaler Clock Select (PRECS) (w/ Relaxation Oscillator)—Bit 2

The clock source to the Prescaler can be selected to be either the Internal Relaxation Oscillator or Crystal Oscillator.

- 0 = Relaxation Oscillator selected (reset value)
- 1 = Crystal Oscillator selected. This bit should only be set if XTAL and EXTAL pin functions are enabled in the appropriate GPIO control register.

### 5.11.1.12 Clock Source (ZSRC)—Bits 1–0

The Clock Source (ZSRC) determines the SYS\_CLK\_X2 source to the SIM module. In turn the SIM module generates divided down versions of this signal for use by memories and the IPBus. ZSRC is automatically set to one during Stop mode, or when PLLPD is set, preventing loss of the reference clock to the core. For the 568300 family parts, ZSRC may have the following values. The operational value of ZSRC is indicated in the ZSRCS field of the PLL Status Register (PLLSR). See [Section 5.11.1.12](#) for additional information.

- 00 = Reserved
- 01 = Prescaler output
- 10 = Postscaler output
- 11 = Reserved

## 5.11.2 PLL Divide-By Register (PLLDB)

Base + \$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LORTP				PLLCOD		PLLCID		0	PLLDB						
Write																
Reset	0	0	1	0	0	0	0	0	0	0	0	1	1	1	0	1

Figure 5-10. PLL Divide-By Register (PLLDB)

### 5.11.2.1 Loss of Reference Timer Period (LORTP)—Bits 15-12

This bit field controls the amount of time required for the loss of reference clock interrupt to be generated. This failure detection time is  $LORTP \times 10 \times \text{PLL-clock-time-period}$ , where  $\text{PLL-clock-time-period} = 2/F_{OUT}$ .

*Loss of Reference Clock Detector* block counts  $F_{OUT}/2$  clocks continuously as illustrated in [Figure 5-1](#) and [Figure 5-2](#). The MSTR\_OSC clock input resets this counter. If the counter ever exceeds  $LORTP \times 10$  the *loss of reference clock interrupt* is generated.

**Note:** When programming the PLLDB field be sure to not zero out the LORTP field. A zero value for LORTP will preemptively set the LOCI bit in the PLL Status register, rendering the Loss of Clock Interrupt useless.

### 5.11.2.2 PLL Clock Out Divide or Postscaler (PLLCOD)—Bits 11-10

The PLL output clock can be divided down by a 2-bit postscaler. The output of the postscaler is a selectable clock source for the controller core as determined by the ZSRC bits in the PLLCR. Legal combinations of PLL settings are located in [Table 5-3](#).

- 00 = Divide by one
- 01 = Divide by two
- 10 = Divide by four
- 11 = Divide by eight

### 5.11.2.3 PLL Clock In Divide or Prescaler (PLLCID)—Bits 9-8

The PLL input clock (MSTR\_OSC), illustrated in [Figure 5-1](#), can be divided down by a 2-bit prescaler.

- 00 = Divide by one
- 01 = Divide by two
- 10 = Divide by four

- 11 = Divide by eight

#### 5.11.2.4 Reserved—Bit 7

This bit is a reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 5.11.2.5 PLL Divide-By (PLLDB)—Bits 6–0

In part, the output frequency of the PLL is controlled by the PLLDB register. The value written to this register, plus one, is used by the PLL to directly multiply the input frequency and present it at its output. For example, if the input frequency is 8MHz and the PLLDB register is set to default 29, the PLL output frequency is 240MHz ( $= F_{OUT} = ((PLLDB + 1) \times 8MHz)$ ). This yields a 120MHz SYS\_CLK\_2X ( $F_{OUT}/2$ ). Dividing this formula by two (hard coded into SIM) results in a 60MHz system clock assuming both prescaler and postscaler are set to one.

Before changing the divide-by value, the core clock must first be switched to the prescaler clock. Legal combinations of PLL settings are located in [Table 5-3](#).

**Note:** The lock detect circuit is reset upon writing to the PLLDB register.

**Note:** For the 8100 family of devices, the PLLDB should be changed from the default such that the chip operates at 40MHz instead of 60MHz. For example, assuming an 8MHz input clock; PLLDB should be set to 19. This yields a PLL frequency of 160MHz with an 80MHz SYS\_CLK\_2X ( $F_{OUT}/2$ ) and a SYS\_CLK frequency of 40MHz.

### 5.11.3 PLL Status Register (PLLSR)

Base + \$2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LOLI1	LOLI0	LOCI	0	0	0	0	0	0	LCK1	LCK0	PLLPDN	0	0	ZSRCS	
Write																
Reset	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1

**Figure 5-11. PLL Status Register (PLLSR)**

Loss of Lock Interrupt is cleared by writing one to the respective LOLI bit in the PLLSR. Loss of Reference Clock Interrupt is cleared by writing one to the LOCI bit in the PLLSR or disabling the interrupt in the PLLCR. A PLL interrupt is generated if any of the LOLI or LOCI bits are set and the respective interrupt enable is set in the PLLCR.

#### 5.11.3.1 PLL Loss of Lock Interrupt 1 (LOLI1)—Bit 15

LOLI1 displays the status of the lock detector state from LCK1 circuit. The interrupt is cleared by writing 1 to LOLI1.

- 0 = PLL locked
- 1 = PLL unlocked

### 5.11.3.2 PLL Loss of Lock Interrupt 0 (LOLI0)—Bit 14

LOLI0 shows the status of the lock detector state from LCK0 circuit. The interrupt is cleared by writing 1 to LOLI0.

- 0 = PLL locked
- 1 = PLL unlocked

### 5.11.3.3 Loss of Reference Clock Interrupt (LOCI)—Bit 13

LOCI shows the status of the reference clock detection circuit.

The interrupt can be cleared by writing 1 to LOCI, but this is not recommended. A set LOCI bit indicates a loss of reference clock. Nominally the application should immediately set the Charge Pump Tri-State (CHPMPTRI) bit in the Control register and put the device into a safe mode (peripheral shutdown followed by STOP) using the remaining good clocks cycles provided.

- 0 = Oscillator clock normal
- 1 = Lost oscillator clock

### 5.11.3.4 Reserved—Bits 12–7

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 5.11.3.5 PLL Fine Lock (LCK1)—Bit 6

- 0 = PLL is unlocked
- 1 = PLL is locked (fine)

### 5.11.3.6 PLL Coarse Lock (LCK0)—Bit 5

- 0 = PLL is unlocked
- 1 = PLL is locked (coarse)

**Note:** Either LCK0 or LCK1 may be set first to indicate a loss of lock. There is no guaranteed sequence.

### 5.11.3.7 PLL Power-Down Status (PLLPDN)—Bit 4

PLL power-down status is delayed by four IPBus clocks from the PLLPD bit in the PLLCR.



- 0 = PLL not powered down
- 1 = PLL powered down

#### 5.11.3.8 Reserved—Bits 3–2

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 5.11.3.9 Clock Source Status (ZSRCS)—Bit 1–0

ZSRCS indicates the current SYS\_CLK\_X2 clock source. Since the synchronizing circuit switches the system clock source, ZSRCS takes more than one IPBus clock to indicate the new selection.

- 00 = Synchronizing in progress
- 01 = Prescaler output
- 10 = Postscaler output
- 11 = Synchronizing in progress

#### 5.11.4 Shutdown Register (SHUTDOWN)

This register can be used to shutdown all system clocks, including SYS\_CLK\_X2, SYS\_CLK, and SYS\_CLK\_DIV2.

**Note:** *This is a terminal condition. The only recovery is to assert the  $\overline{\text{RESET}}$ .*

This function is intended to place the device in a known state in the following specific circumstance:

- The Loss of Reference (LOR) interrupt is asserted
- The CHPMPTRI bit in the PLLCR has been set
- The value 0xDEAD is written to this register

This occurs when the chip clock source or crystal input dies, an LOR interrupt is issued, and the PLL is placed in a mode where it ignores the reference clock input and continues to run open loop. The PLL is designed to run at the target frequency for at least 1000 cycles. During this time, the LOR routine can shutdown the system in a controlled manner, terminating in actually shutting down the system clocks as well.

A write of 0xDEAD, or any other value, to this register will have no effect unless the first two conditions above have been met. If the LOR is clear, or CHPMPTRI is not set, no action is taken and the register is not written.

Base + \$4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Write	SHUTDOWN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 5-12. Shutdown Register (SHUTDOWN)**

Clock shutdown may take several cycles, so the write to this register should be followed by an infinite loop, simply branching to itself.

### 5.11.5 Oscillator Control Register w/o Relaxation Oscillator (OSCTL)

This register controls aspects of both the Internal Relaxation Oscillator and the Crystal/Resonator Oscillator. If the Relaxation Oscillator is not included on the chip only bit 13 is defined, illustrated in [Figure 5-13](#) while [Figure 5-14](#) illustrates the OSCTL configuration for those chips with a relaxation oscillator.

Base + \$5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	COHL	0	0	0	0	0	0	0	0	0	0	0	0	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 5-13. Oscillator Control Register (OSCTL) w/o Relaxation Oscillator**

#### 5.11.5.1 Reserved (w/o Relaxation Oscillator)—Bits 15–14

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 5.11.5.2 Crystal Oscillator High/Low Power Level (COHL)—Bit 13

This bit controls the power usage of the crystal oscillator. It is reset to the high power state, allowing use of either a crystal or resonator.

- 0 = High power mode
- 1 = Low power mode. This mode can be used with a crystal if the crystal's ESR is less than 40Ω.

#### 5.11.5.3 Reserved (w/o Relaxation Oscillator)—Bits 12–0

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 5.11.6 Oscillator Control Register w/ Relaxation Oscillator (OSCTL)

Base + \$5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	ROPD	0	COHL	CLKMODE	0	0	TRIM									
Write																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

Figure 5-14. Oscillator Control Register (OSCTL) w/ Relaxation Oscillator

#### 5.11.6.1 Relaxation Oscillator Power-Down (ROPD)—Bit 15

This bit is used to power-down the relaxation oscillator if the crystal oscillator is being used. To prevent loss of clock to the core or the PLL, set this bit only if the prescaler clock source has been changed to the crystal oscillator by setting the PRECS bit in PLLCR.

- 0 = Relaxation oscillator enabled
- 1 = Relaxation oscillator powered down

#### 5.11.6.2 Reserved—Bit 14

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 5.11.6.3 Crystal Oscillator High/Low Power Level (COHL)—Bit 13

This bit is used to control the power of the crystal oscillator. It is reset to the high power state, allowing either a crystal or resonator to be used.

- 0 = High Power mode is required when a resonator is used
- 1 = Low Power mode is desired when a crystal is used

#### 5.11.6.4 Crystal Oscillator Clock Mode (CLKMODE)—Bit 12

This bit is used to control the Crystal/Resonator Clock selection.

- 0 = Crystal oscillator enabled
- 1 = Direct clock mode. Setting this bit shuts down the crystal oscillator, allowing an external clock source on the XTAL pin to drive the clock input to the chip directly.

#### 5.11.6.5 Reserved—Bits 11–10

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 5.11.6.6 Internal Relaxation Oscillator (TRIM)—Bits 9–0

This bit field changes the size of the internal capacitor used by the Internal Relaxation Oscillator. By testing the frequency of the internal clock and incrementing or decrementing this factor accordingly, the accuracy of the internal clock can be improved by 40 percent. Incrementing these bits by one decreases the frequency by 0.078 percent of the unadjusted value. Decrementing this register by one increases the frequency by 0.078 percent. Reset sets these bits to \$200, centering the range of possible adjustment.

## 5.12 Interrupts

The interrupts listed in [Table 5-7](#) are ORed into a single processor core interrupt.

**Table 5-7. Interrupt Summary**

Interrupt	Source	Description	Reference
LOLI1	PLLSR	Lock 1 Interrupt	<a href="#">Section 5.11.3.1</a>
LOLI0	PLLSR	Lock 2 Interrupt	<a href="#">Section 5.11.3.2</a>
LOCI	PLLSR	Loss of Reference Clock Interrupt	<a href="#">Section 5.11.3.3</a>

---

# Chapter 6

## Flash Memory (FM)

## Document Revision History for **Chapter 6, Flash Memory (FM)**

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 3.0	On top of page 6-22, last sentence in the first paragraph, text correct to say, Bits six through zero <b>cannot</b> be modified once written.
Rev 4.0	Corrected Figure 6-2 and 6-3 with correct boot flash start address, updated Table 6-5 to clarify the reserved area and grammar edits throughout. Corrected dated part numbers in FMOPT1 row of Fig 6-5, page 6-19 Corrected statement made in Section 6.5.4 re: P space
Rev 5.0	Added statement of difference between 8300 and 8100 in Section 6.1 Introduction Added 8100 device features to Section 6.2 Added 8100 system clock frequency example on page 6-12 Replaced second sentence of Section 6.5.4 to: "While this feature is enabled application code (P space) is restricted to the internal memory of the 16-bit controller." Converted chapter to Freescale design standards

## 6.1 Introduction

The Flash Memory (FM) module is a non-volatile memory module, operating with the Program, Primary and Secondary Data, and IPBuses. It consists of three categories of Flash Memory:

1. Program
2. Data
3. Boot

Please refer to the chip's data sheet for actual Program Flash size. As an example, the Program Flash is constructed from multiple 32k by 16-bit interleaved blocks. Interleaved means each 32k block consists of one 16k block dedicated to even addresses and one 16k block dedicated to odd addresses. This design allows for single cycle access to consecutive addresses allowing access rates up to 60MHz, equal to the core's maximum operating frequency. The 8100 devices are guaranteed only to 40MHz.

The Data and Boot Flashes are constructed from single non-interleaved blocks. Non-interleaved means a single Flash is used for both even and odd addresses. This design requires a two-cycle access to consecutive addresses allowing access rates up to 30MHz, half the core's maximum operating frequency. The Data Flash and Boot Flash are constructed from a single non-interleaved block. Please refer to the chip's data sheet for actual Data and Boot Flash sizes.

These Flash arrays serve as electrically erasable and programmable memories. The Flash EEPROM is ideal for program and data storage for single-chip applications, allowing for field reprogramming without requiring external programming voltage sources.

The programming voltage required to program and erase the Flash is generated internally by on-chip charge pumps. Program and erase operations are performed by a command driven interface from the 16-bit controller, using an internal state machine. All Flash blocks can be programmed or erased at the same time; however, it is not possible to read from a physical Flash block while the same block is being erased or programmed. However, it is possible to perform *read-while-write* operations when using independent blocks. For example, an application executing from the Program Flash block may program or erase the Data Flash block.

Flash also contains security and protection mechanisms with the potential of being enabled to prohibit flash access from external devices and accidental program/erase respectively.

## 6.2 Features

- Program Flash Memory is interleaved. Please see the Data Sheet for the specifics
- 8k/16k bytes (chip dependent) of boot flash memory constructed from a single 4k/8k by 16-bit block
- 8k bytes of data flash memory constructed from a single 4k by 16-bit block

- The 8300 60MHz single cycle operation for Program Flash access due to interleaved blocks; 30 MHz operation for boot and data accesses; all accesses are at 150°C (T<sub>J</sub>) and 2.25V.
- The 8100 40MHz single cycle operation for Program Flash access due to interleaved blocks; 20 MHz operation for boot and data accesses; all accesses at 105°C (T<sub>J</sub>) and 2.25V.
- Automated program and erase operation
- Read-while-write capability
- Interrupts on command completion, command buffer empty and access error
- Fast page erase
- Single power supply program and erase
- Security feature prohibits flash access from external devices
- Protection feature prohibits accidental program/erase
- Sector protection system for program, boot, and data flash
- Flash supports byte, word, and long word data flash read operations by the host 16-bit controller

### 6.3 Block Diagram

The Flash Memory (FM) demonstrated in [Figure 6-1](#) contains Flash array blocks, buses, IP Interface Control, Flash Interface, and register blocks. The Flash blocks are arrays of electrically erasable and programmable, non-volatile memory for use with the Program, Primary, and Secondary Data buses.

Program Flash read access is through the Program bus and is interleaved (Odd and Even address locations) providing no wait states unless the same block is accessed in succession. A penalty of one wait state will occur if the same block is accessed in succession (e.g. if the same address is read twice in a row).

Data Flash read access is through the Primary and Secondary data buses. Consecutive 16-bit access incurs a penalty of one wait state. A 32-bit access develops a minimum penalty of two wait states. Only 16-bit aligned write access is permitted.

Boot Flash read access is through the Program bus. Write access can only be aligned and 16 bits wide. Read access will always require a penalty of one wait state on a consecutive access.

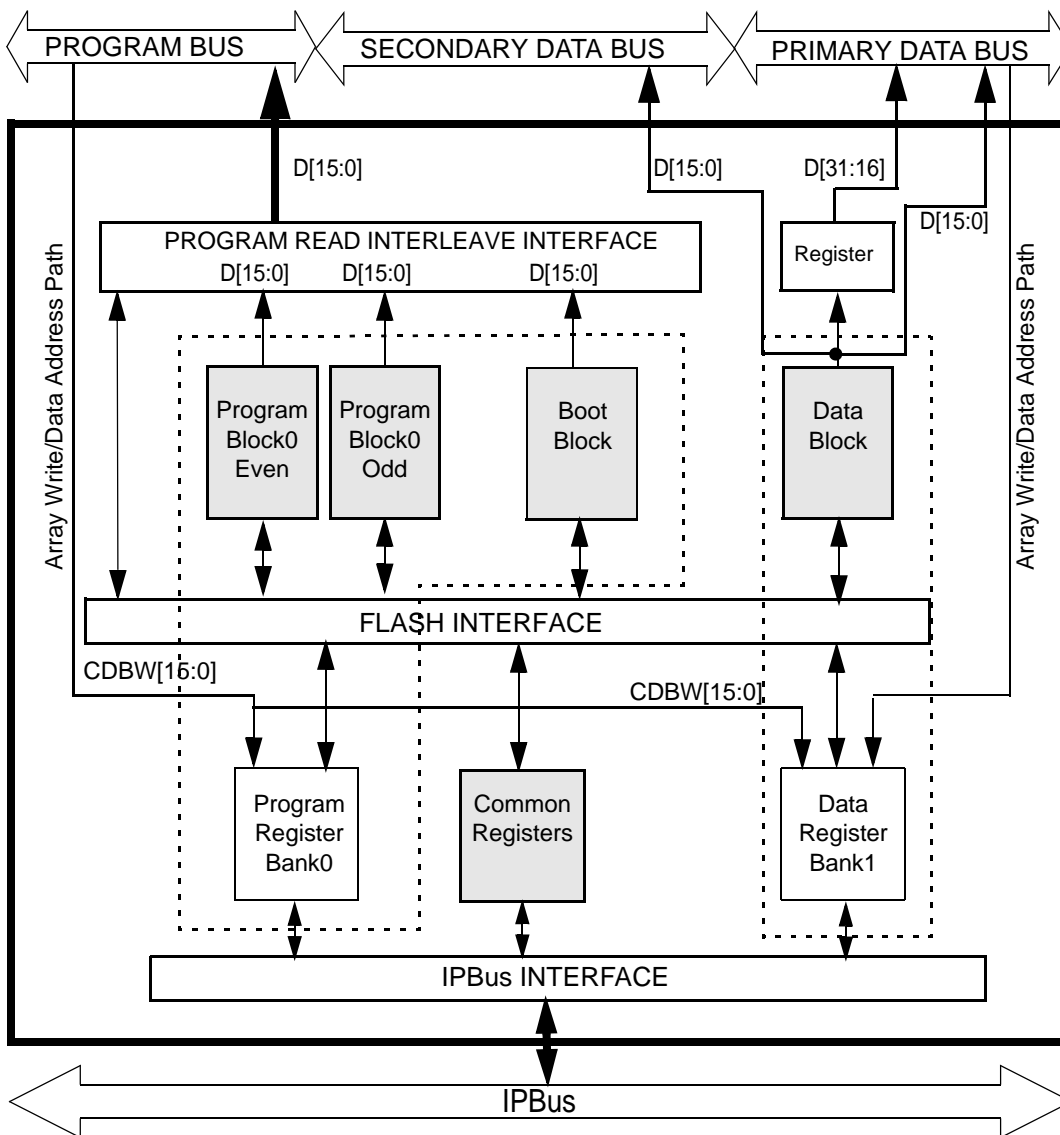
Flash memory must be erased before it can be written. The smallest memory block able to be erased is a page, organized as eight rows of 32 words or 512 bytes. This represents the smallest piece of memory possible to be erased in Boot and Data Flash. Since Program Flash is two blocks



of interleaved memory, as erase operation is performed on both blocks simultaneously. Therefore, the smallest piece of Program Flash possible to be erased is 1024 bytes. The erase operation also supports mass erase of each block, erasing the entire block.

**Note:** An erased bit reads *one* and a programmed bit reads *zero*.

Program, Boot, and Data Flash memories require a banked set of control registers to control program and erase operation. **Figure 6-1** illustrates a Program, Boot, and Data Flash configured with two sets of banked registers.



**Figure 6-1. Flash Memory Block Diagram**

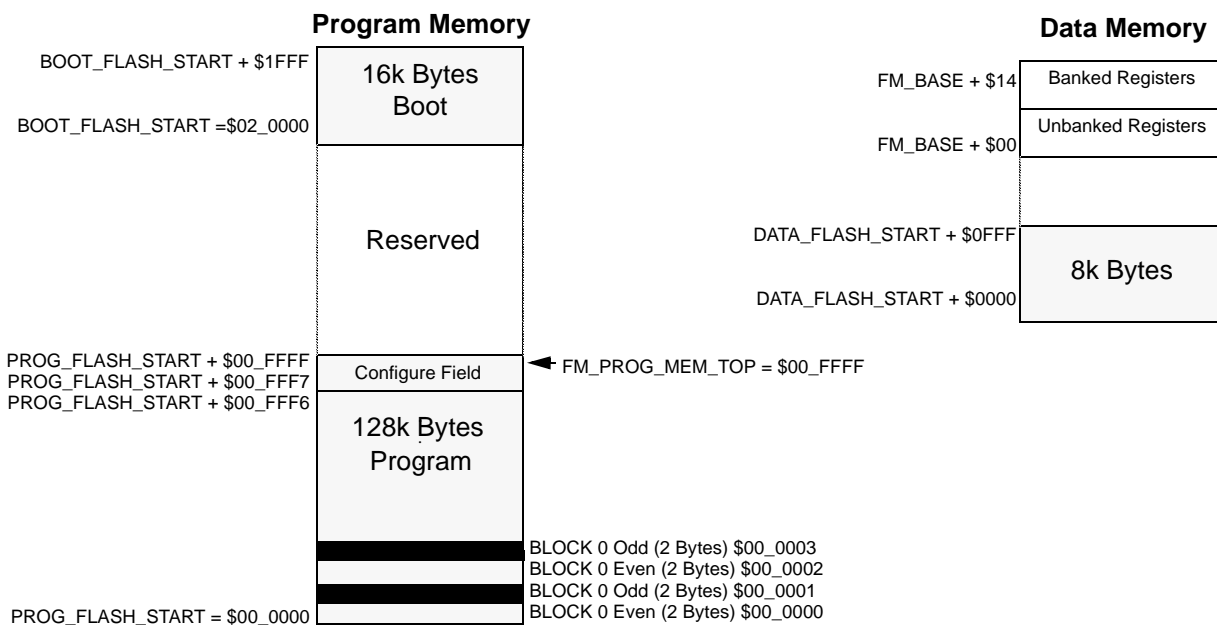
## 6.4 Memory Map

**Figure 6-2** and **Figure 6-3** illustrate the memory map of the FM on the system bus. The chip specific data sheet will identify which figure is correct. This is determined by the amount of the program Flash Memory contained on the chip. If the chip has less than 128k bytes of program Flash, refer to **Figure 6-2**. If more memory is included on the chip, **Figure 6-3** is correct.

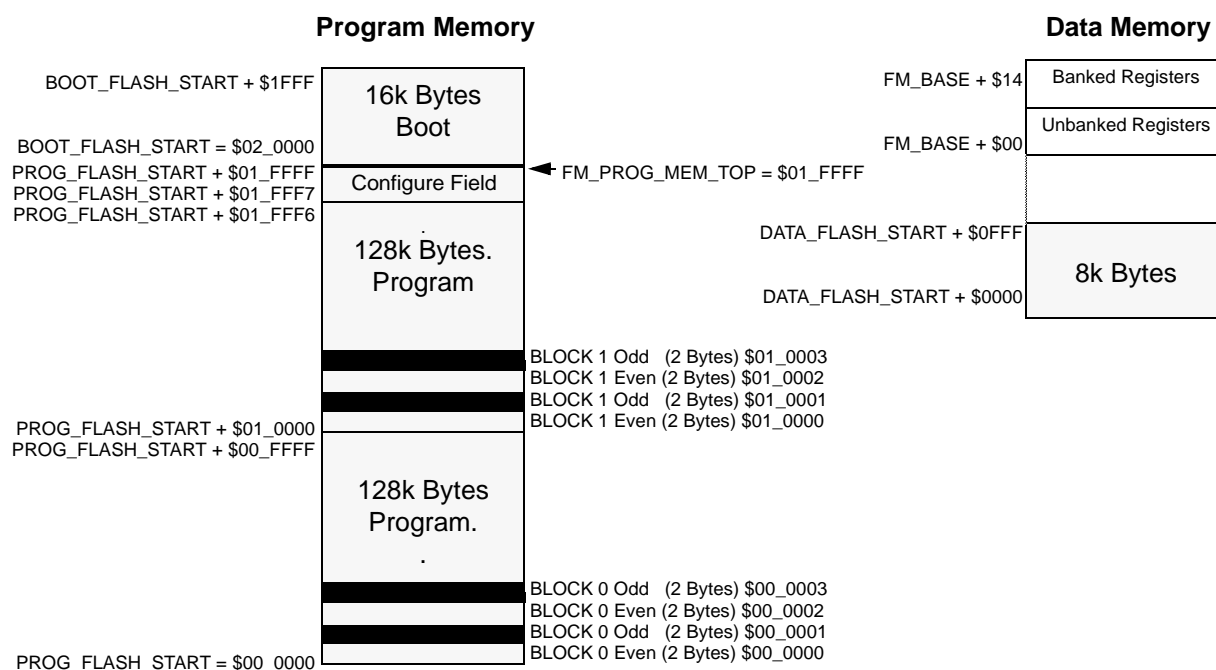
Both figures illustrate FM is divided into three functional blocks. The Program and Boot Memories reside on the Program Memory buses. They are controlled by one set of banked registers. Data Memory Flash resides on the Data Memory buses and is controlled separately, having its own set of banked registers.

The top nine words of the Program Memory Flash are treated as special memory locations. The content of these words is used to control the operation of the Flash Controller. Because these words are part of the FM content, their state is maintained during power down and reset. During chip initialization the content of these memory locations is loaded into FM control registers, detailed in **Table 6-1**. Because this FM Configuration Field is located at the top of the Program Flash, its specific address within the memory map is determined by the size of the Program Flash Block in each chip.

The Program Flash interleaves odd and even address locations using 32k × 16 bit blocks (Block\_n\_odd), covering all odd locations for the Program Flash address range, and another block (Block\_n\_even), covering all even address ranges. This architecture allows the Program Memory to operate at twice the speed of the other FM blocks.



**Figure 6-2. Flash Memory Array Small Memory Maps**



**Figure 6-3. Flash Memory Array Large Memory Maps**

The FM Configuration Field is composed of 18 bytes of reserved memory space within the FM Program Flash array, containing information determining the module's protection and access restriction scheme out of reset. The top four words contain the Back Door Access Key. This key must be entered correctly to insecure the 16-bit controller. The next three words contain the protection bytes loaded into the Protection registers upon reset. The PROT\_BANK1\_VALUE, PROT\_BANK0\_VALUE, and PROTB\_VALUE words contain the protection values for Data, Program, and Boot flash blocks respectively. The security words (SECH\_VALUE, SECL\_VALUE) are loaded into the Security registers upon reset. The SECH\_VALUE contains information to enable/disable the Back Door Access scheme. The SECL\_VALUE contains information to enable/disable the Security feature permitting the 16-bit controller to prevent intrusive access to the FM content. A description of each byte used in the FM Configuration Field is provided in [Table 6-1](#). A description of the security scheme is provided in [Section 6.5.4](#).

**Table 6-1. Flash Memory Configuration Field**

P-Space Memory Address	Size (Bytes)	Description	Word Name
FM_PROG_MEM_TOP -0	2	Backdoor Comparison Key 3	BACK_KEY_3_VALUE
FM_PROG_MEM_TOP -1	2	Backdoor Comparison Key 2	BACK_KEY_2_VALUE
FM_PROG_MEM_TOP -2	2	Backdoor Comparison Key 1	BACK_KEY_1_VALUE
FM_PROG_MEM_TOP -3	2	Backdoor Comparison Key 0	BACK_KEY_0_VALUE
FM_PROG_MEM_TOP -4	2	Protection Bytes for Bank1 (Data)	PROT_BANK1_VALUE
FM_PROG_MEM_TOP -5	2	Protection Bytes for Bank0 (Program)	PROT_BANK0_VALUE
FM_PROG_MEM_TOP -6	2	Protection Bytes for Bank0 (Boot)	PROTB_VALUE
FM_PROG_MEM_TOP -7	2	Security Word Upper (See <a href="#">Section 6.7.3</a> )	SECH_VALUE
FM_PROG_MEM_TOP -8	2	Security Word Lower	SECL_VALUE

Flash Memory also contains a set of control and status registers located at the FM register's base address. A summary of these registers is provided in [Table 6-2](#). Flash Memory contains two sets of banked registers between offsets \$10 and \$14. The active register bank is selected via the BKSEL bits in the unbanked FMCR register. Bank0 corresponds to Program and Boot Flash. It is the default bank after reset. Bank1 corresponds to Data Flash.

**Table 6-2. Flash Memory Register Address Map**

Offset From Register Base Address <sup>1</sup>	Bits 15-8	Bits 7-0	Banked Register <sup>2</sup>
FM_BASE + \$1D-\$3C	RESERVED/NOT IMPLEMENTED		
FM_BASE + \$1C		FMOPT2	NO
FM_BASE + \$1B		FMOPT1	NO
FM_BASE + \$1A		FMOPT0	NO
FM_BASE + \$15-\$19	RESERVED/NOT IMPLEMENTED		
FM_BASE + \$14	RESERVED	FMCMD	YES
FM_BASE + \$13	RESERVED	FMUSTAT	YES
FM_BASE + \$12	RESERVED3		
FM_BASE + \$11	RESERVED	FMPROTB <sup>3</sup>	YES
FM_BASE + \$10	FMPROT		YES
FM_BASE + \$05	RESERVED2		
FM_BASE + \$04	FMSECL		NO
FM_BASE + \$03	FMSECH		NO
FM_BASE + \$02	RESERVED1		
FM_BASE + \$01	FMCR		NO
FM_BASE + \$00	RESERVED	FMCLKD	NO

NOTE:

1. Absolute address is equal to the Offset plus the base address. See the chip specific Data Sheet for the base address.
2. Light shaded registers are banked. Visibility of banked registers is controlled by the BKSEL bit field in the FMCR register.
3. FMPROTB is not present for Data Flash.

## 6.5 Functional Description

This section details Flash modes of operation, security and protection features.

### 6.5.1 Read Operation

A valid read operation occurs whenever an address on the Program Bus, Primary Data Bus, or Secondary Data Bus is equal to an address within the valid range of their respective FM space and the read/write control indicates a read cycle.

### 6.5.2 Write Operation

A valid write operation can occur from one of two sources:

1. Program Address Bus
2. Primary Data Address Bus

Those sources occur whenever the bus select signals are active 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 sequence operation. Only 16-bit write operations are allowed to the FM space.

### 6.5.3 Program and Erase Operation

Write and read operations are both used for the program and erase algorithms described in this section. These algorithms are controlled by a state machine whose time base is derived from the 16-bit controller's system clock divided by two and subsequently input into a programmable counter.

The Command (FMCMD) register, as well as the associated address and data registers, operate as a buffer and a register (two-stage FIFO). Consequently, a new command, along with the necessary data and address, can be stored to the buffer while the previous command is still in progress. This feature increases the rate at which the FM state machine receives commands and therefore reduces command processing time and ultimately speeds up program/erase cycles. Buffer Empty as well as Command Completion are signaled by flags in the FM Status register (FMUSTAT). Interrupts will be generated if enabled.

#### 6.5.3.1 Writing the Clock Divider (FMCLKD) Register

Prior to issuing the first program or erase command, it is first necessary to write the FMCLKD register to divide the input clock to within the 150kHz to 200kHz range. The input clock is equal to the 16-bit controller's system clock divided by two. The FMCLKD register bits PRDIV8 and DIV are to be configured as follows.

For input clock frequencies greater than 12.8MHz, the PRDIV8 bit must be set in the FMCLKD register to pre-divide the input clock by eight.

If PRDIV8 = 1 then ICLK = input clock ÷ 8, else ICLK = input clock.

If ICLK, is divisible by 200kHz, then

$$\text{DIV} = (\text{ICLK} \div 200\text{kHz}) - 1$$

else

$$\text{DIV} = \text{INT}(\text{InputClock} \div 200\text{kHz}).$$

**Note:** INT means rounding towards zero. For example,  $\text{INT}(950\text{kHz} \div 200\text{kHz}) = 4$ .

**Note:** See 8100 family devices example on the following page.

For example, for a system clock frequency of 60MHz, input clock is 30MHz. Since input clock > 12.8MHz, the PRDIV8 = 1 and ICLK = input clock ÷ 8:

$$\text{DIV} = \text{INT}(\text{ICLK} \div 200\text{kHz}) = \text{INT}(3750\text{kHz} \div 200\text{kHz}) = 18$$

$$\text{FCLK} = (\text{input clock}) \div (\text{DIV} + 1) = 3750\text{kHz} \div 19 = 197.4\text{kHz}$$

FMCLKD register should be set to  $0 \times 52$ . The resulting FCLK (Flash timing control clock) is 197.4kHz. As a result, the Flash algorithm programming times are increased over the 200kHz optimal target speed by:

$$(200 - 197.4) \div 200 \times 100\% = 1.3\%$$

**Note:** Command execution time will increase proportionally with the period of FCLK.

**WARNING:**

Setting FMCLKD to a value so FCLK <150kHz can destroy the Flash due to overstress. Setting FMCLKD to a value so FCLK >200kHz can result in incomplete programming or erasure of the memory array cells.

The DIVLD bit is set automatically if the FMCLKD register is written. If this bit is zero, the register has not been written since the last reset. Program and erase commands will not be executed when this register has not been written. Please see [Section 6.5.3.4](#) for additional information on illegal operations.

**Note:** 8100 devices example for a system clock frequency of 40MHz input clock is 20MHz. Since input clock > 12.8MHz, the PRDIV8 = 1 and ICLK = input clock ÷ 8 = 2500kHz:

$$\text{DIV} = \text{INT}(\text{ICLK} \div 200\text{kHz}) = \text{INT}(2500\text{kHz} \div 200\text{kHz}) = 12$$

$$\text{FCLK} = (\text{input clock}) \div (\text{DOV} + 1) = 2500\text{kHz} \div 13 = 192.3\text{kHz}$$

FMCLKD register should be set to 0 × 4D. The resulting FCLK (Flash timing control clock) is 192.3kHz. As a result, the Flash algorithm programming times are increased over the 200kHz optimal target speed by:

$$(200 - 192.3) \div 200 \times 100\% = 3.85\%$$

### 6.5.3.2 Program and Erase Sequences

A command state machine is used to supervise the write sequencing for program and erase. Before the write sequence is started, it is necessary to write the Bank Select (BKSEL) bit field in the Flash Configuration Register (FMCR) to select the banked set of registers associated with the Flash blocks to be programmed or erased. Please refer to [Figure 6-1](#) for additional details. The Command Buffer Empty Interrupt Flag (CBEIF) bit in the Flash User Status Register (FMUSTAT) should be tested, ensuring the address, data, and command buffers are empty in preparation for a program or erase. If the CBEIF flag is set, the Program/Erase Command-Write sequence can be started.

Strictly adhere to the following three-step Command-Write sequence with no intermediate writes to the FM permitted between the three steps. The Command-Write sequence:

1. a. Write the 16-bit data word to be programmed to the desired FM address space (Program/Data/Boot). Please refer to the FM memory map in the device Data Sheet. The address and data will be stored in internal buffers to be used by the command state machine.
1. b. If programming a Program Flash block, two words may be programmed at once due to its interleaved design. The first data value in step 1a should be written to an Even address and an additional data value may be written to the complementary Odd address. Both values will be written concurrently when the program command is written in the following sequence. For Program operation, all address bits are valid. For Mass Erase, the address can be anywhere in the available address space. For Page Erase, the address must be within the page boundary. The value of the data word is ignored for an erasure.
2. Write the Program or Erase Command to the Command Buffer. These commands are described in the next section and listed in [Table 6-3](#).

3. Clear the CBEIF flag by writing one to launch the command. After the CBEIF flag is cleared the CCIF flag will be cleared by hardware, indicating the command was successfully launched. The CBEIF flag will be set again indicating the Address, Data, and Command Buffers are ready for a new command sequence to begin.

The completion of the Command Operation is indicated by the CCIF flag setting. The Command state machine will flag errors in program or erase write sequences by means of the Access Error (ACCERR) and Protection Violation (PVIOL) flags in the FMUSTAT register. An erroneous Command-Write sequence aborts, setting the appropriate flag. If a flag is set, clear the ACCERR or PVIOL flags before commencing another Command-Write sequence.

**Note:** *By writing a zero to the CBEIF flag, the command sequence can be aborted after the word write to the Flash address space (step 1) or after writing a command to the FMCMD register and before the command is launched (step 2). The ACCERR flag will be set on aborted commands and must be cleared before a new command is launched.*

A summary of the program algorithm is illustrated in [Figure 6-4](#). Write either a mass or page erasure command to the Command register for the erase algorithm.



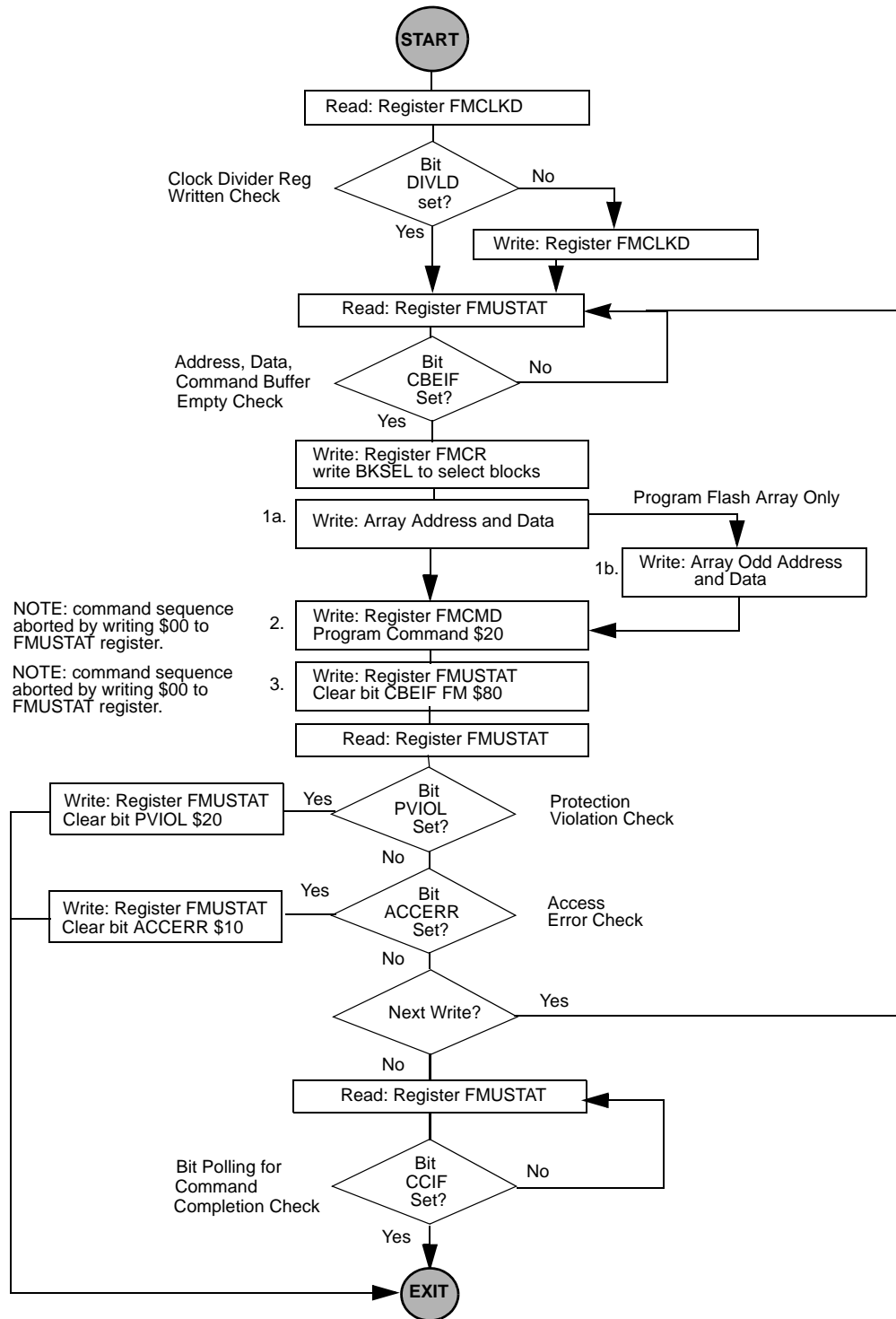


Figure 6-4. Example Program Algorithm

### 6.5.3.3 Flash User Mode Valid Commands

**Table 6-3** summarizes the valid Flash commands.

**Table 6-3. Flash User Mode Valid Commands**

CMD	Meaning	Description
\$05	Erase Verify	Verify the selected Program, Data, or Boot Flash is erased. If the Flash selected is erased, the BLANK bit will set in the FMUSTAT register ( <b>Figure 6-18</b> ) upon command completion. A valid address within the desired Flash block must be written prior to the command write.
\$20	Program	Program a 16-bit word in Program, Data, or Boot Flash; Program Flash may write a 32-bit word on a common odd/even interleave boundary.
\$40	Page Erase	Eight rows of 64 bytes (8 x 64 = 512 bytes) for all by 16-bit Flash modules. This will result in 1k bytes for interleaved blocks and 512 bytes for all other blocks being erased due to a page erase command.
\$41	Mass Erase	Erase entire Program, Data, or Boot Flash block selected. For Program Flash, all 128K bytes of Flash from two interleaving blocks are erased. For Data and Boot, the entire Flash is erased. A mass erase of any block is only possible when no protection bits for that block are set.

### 6.5.3.4 Flash User Mode Illegal Operations

The ACCERR flag is set during the Command-Write sequence if any of the following illegal operations are performed. These operations will cause the Command State Machine to immediately abort. Writes to the Flash Memory Address space refers to writes through the 16-bit controller core buses, not the IP register bus.

1. Writing to the Flash Memory (FM) address space before initializing FMCLKD
2. Writing a byte or long word to the FM address space with only 16-bit writes are allowed
3. Writing to the FM address space while CBEIF is not set
4. Writing a second word to the FM address space considered not a valid odd address complement to a previous even program address and then executing a valid command. For example, write address \$1 followed by address \$3 then execute the program command.
5. Writing an invalid user command to the FMCMD register
6. Writing to any FM register other than FMCMD after writing a word to the FM address space
7. Writing a second command to the FMCMD register before executing the previously written command
8. Writing to any FM register other than FMUSTAT (to clear CBEIF) after writing to the command register, FMCMD
9. The part enters Stop or Wait modes and a program or erase command is in progress; the command is aborted

10. Aborting a Command sequence by writing zero to the CBEIF flag after the word *write* to the Flash address space; or after writing a command to the FMCMD register, and before the command is launched

The Protection Violation (PVIOL) flag will be set during the Command-Write sequence after the word *write* to the FM address space if any of the following illegal operations are performed. Such operations will cause the Command state machine to immediately abort:

1. Attempting to program in a protected area
2. Attempting to page erase in a protected area
3. Attempting to mass erase while any protection is enabled for that block. Please see [Section 6.8.1](#).

If a Flash block is read during execution of an algorithm on that block, for example CCIF is low, the read will return non valid data and the ACCERR flag will not be set.

### 6.5.3.5 Wait/Stop Modes

If a command is active (CCIF = 0) when the 16-bit controller enters Wait or Stop mode, the command will be aborted, and the data being programmed or erased is lost. The high voltage circuitry to the Flash will be switched off and a pending command (CBEIF = 0) will not be executed once the 16-bit controller exits Wait or Stop mode. The CCIF and ACCERR flags will be set if a command is active when the 16-bit controller enters Wait or Stop mode.

#### **WARNING:**

As active commands are immediately aborted when the 16-bit controller enters Wait mode, it is *strongly* recommended not to execute the Wait instruction during program and erase execution.

## 6.5.4 Flash Security Operation

The Flash Security feature is intended to prevent unauthorized users from reading the contents of the Flash arrays (Program, Data, and Boot). While this feature is enabled application code (P space) is restricted to the internal memory of the 16-bit controller.

The FM Configuration Field, located at the top of the Program Flash address range, contains information in the SECL\_VALUE to configure the security feature of the 16-bit controller (see [Table 6-1](#)). This word is read by the 16-bit controller automatically after each reset and stored in the FM Security Low (FMSECL) register. If the SECL\_VALUE is equal to 0xE70A, the Security Status (SECSTAT) bit in the FM Security High (FMSECH) is set, indicating the 16-bit controller is secured.

After reset, the Flash module may be unsecured through one of two methods:

1. Executing a back door access scheme
2. Mass Erase Program Flash

These mechanisms will have to be implemented in the application code executing from internal memory.

#### 6.5.4.1 Back Door Access

It is possible to temporarily bypass the security through a back door access scheme, using an 8-byte long key. Upon successful completion of the back door access sequence, the SECSTAT bit is cleared, indicating the chip is not secured. The following sequence may be followed to bypass security using the back door access scheme:

1. If the KEYEN bit is set, back door access is enabled.
2. Set the KEYACC bit in the FMCR to enable access key entry.
3. Write the correct 8-byte Back Door Access Key to the FM Configuration Field. These writes must be composed of four, 16-bit sequential writes to the top four words in the Program Flash address space, beginning at address FM\_PROG\_MEM\_TOP-3 and ending at address FM\_PROG\_MEM\_TOP (Please see [Table 6-1](#)). For example on 56835, the addresses would be: \$01FFFC, \$01FFFD, \$01FFFE, and \$01FFFF. The four write cycles can be separated by any number of bus cycles. See the 16-bit controller specific data sheet memory map for the definition of FM\_PROG\_MEM\_TOP.
4. Clear the KEYACC bit to disable access key entry.

If all eight bytes written match the Flash content, the security is bypassed until the next reset.

**Note:** The security and protection of the Flash is not changed by insecuring the device via the back door access scheme. The configuration of these features are restored at reset from the FM Configuration Field in the Program Flash.

After the next reset sequence, the device is secured again and the same back door access key is in effect unless the FM Configuration Field was changed by program or erase.

#### 6.5.4.2 Mass Erase Program Flash

A secured module can be unsecured by performing a Mass Erase on the Program Flash. After the next reset sequence, the security state of the Flash module is determined by the Flash Security word. Since the word no longer contains the security enable word (0xE70A), the Flash will be unsecured. Any sensitive information should be erased from Data and Boot Flash prior to insecuring the chip through this option.

## 6.5.5 Flash Protection Operation

The Flash may be protected from program/erase on a sector by sector basis. The FM Configuration Field contains the default protection values for the Data, Program, and Boot Flash blocks in the PROT\_BANK1\_VALUE, PROT\_BANK0\_VALUE, and PROTB\_VALUE memory locations, respectively. See [Table 6-1](#). These values are loaded at reset into the Protection Registers, at addresses shown in [Table 6-2](#).

After reset, the Protection Register values may be modified if the LOCK bit in the Flash Configuration Register (FMCR) is clear. Once the LOCK bit has been set, the protection bits may no longer be changed until the 16-bit controller has been reset thereby clearing the LOCK bit.

## 6.6 Pin Definitions

The Flash Memory module contains no signals connected off-chip.

## 6.7 Unbanked Register Definition

An unbanked register operates and controls all Flash blocks. It is also referenced as a *common register*.

**Table 6-4. Flash Memory Map**

Device	Peripheral	Address
8100/8300	FM_BASE	\$00F400

**Table 6-5. Unbanked Flash Registers**

Address + Offset	Address Acronym	Register Name	Chapter Location
FM_BASE + \$0	FMCLKD	Clock Divider Register	<a href="#">Section 6.7.1</a>
FM_BASE + \$1	FMCR	Module Configuration Register	<a href="#">Section 6.7.2</a>
		Reserved	
FM_BASE + \$3	FMSECH	Security High Half Register	<a href="#">Section 6.7.3</a>
FM_BASE + \$4	FMSECL	Security Low Half Register	<a href="#">Section 6.7.3</a>
		Reserved	
		Banked Registers, See <a href="#">Table 6-9</a>	
		Reserved	
FM_BASE + \$1A	FMOPT0	Flash Optional Data 0 Register	<a href="#">Section 6.7.4</a>
FM_BASE + \$1B	FMOPT1	Flash Optional Data 1 Register	
FM_BASE + \$1C	FMOPT2	Flash Optional Data 2 Register	

Bit fields of each of the seven registers are illustrated in **Table 6-5**. Details of each follow.

Addr. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
\$0	FMCLKD	R	0	0	0	0	0	0	0	0	DIVLD	PRDIV8	DIV						
		W																	
\$1	FMCR	R	0	0	0	0	0	LOCK	0	AEIE	CBEIE	CCIE	KEY ACC	0	0	0	BKSEL		
		W																	
RESERVED																			
\$3	FMSECH	R	KEYEN	SEC STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																	
\$4	FMSECL	R	SEC																
		W																	
RESERVED																			
Banked Registers, See <a href="#">Table 6-13</a>																			
RESERVED																			
\$1A	FMOPT0	R	Reserved for hot temperature ADC reading of the temperature sense. Value set during test.																
		W																	
\$1B	FMOPT1	R	Reserved for trim cap setting of the relaxation oscillator (on chips other than 56F8345-6, 56F8355-7, 56F8366-7).																
		W																	
\$1C	FMOPT2	R	Reserved for room temperature ADC reading of the temperature sense. Value set during test																
		W																	

R	0	Read as 0
W		Reserved

**Figure 6-5. UnBanked Flash Register Map Summary**

### 6.7.1 Clock Divider Register (FMCLKD)

The Flash Memory Clock Divider (FMCLKD) register is unbanked. It is used to control the period of the clock used for timed events in program and erase algorithms within the Flash Interface (FI). It runs at the 16-bit controller’s system clock frequency. Bits six through zero cannot be modified once written.

FM_Base + \$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	DIVLD	PRDIV8	DIV					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 6-6. Clock Divider Register (FMCLKD)**

### 6.7.1.1 Reserved—Bits 15–8

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 6.7.1.2 Clock Divider Loaded (DIVLD)—Bit 7

Writing to this bit has no effect.

- 0 = Register has not been written
- 1 = Register has been written to since the last reset

### 6.7.1.3 Enable Prescaler By 8 (PRDIV8)—Bit 6

- 0 = The input clock is directly fed into the FMCLKD Divider
- 1 = Enables a prescaler  $\times 8$  to divide the FM input clock before feeding it into the FMCLKD Divider

### 6.7.1.4 Clock Divider Bits (DIV)—Bits 5–0

The combination of PRDIV8 and DIV effectively divides the FM input clock down to a frequency of 150kHz - 200kHz. The input clock frequency range is  $150\text{kHz} < \text{input clock} < 102.4\text{MHz}$ .

The Clock Divider (FMCLKD) register bits PRDIV8 and DIV must be set with appropriate values before programming or erasing the FM array.

## 6.7.2 Configuration Register (FMCR)

The Flash Memory Configuration Register (FMCR) is unbanked. It is used to configure and control the operation of the FM array.

FM_Base + \$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	LOCK	0	AEIE	CBEIE	CCIE	KEYACC	0	0	0	BKSEL	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6-7. Configuration Register (FMCR)

### 6.7.2.1 Reserved—Bits 15–11

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 6.7.2.2 Write Lock Control (LOCK)—Bit 10

This bit can always be read. It may be set only once and is cleared at chip reset.

**Note:** This bit is written each time other bits in this register are written. Exercise care not to accidentally set this bit when modifying other bits.

- 0 = The FMPROT and FMPROTB can receive writing
- 1 = The FMPROT and FMPROTB are write-locked

### 6.7.2.3 Reserved—Bit 9

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 6.7.2.4 Access Error Interrupt Enable (AEIE)—Bit 8

This read/write bit enables an interrupt in case of an ACCERR flag being set.

- 0 = ACCERR interrupts disabled
- 1 = An interrupt will be requested whenever the ACCERR flag is set

### 6.7.2.5 Command Buffer Empty Interrupt Enable (CBEIE)—Bit 7

This read/write bit enables an interrupt in case of an empty command buffer in the Flash.

- 0 = Command Buffer Empty interrupts disabled
- 1 = An interrupt will be requested whenever the CBEIF flag is set

### 6.7.2.6 Command Complete Interrupt Enable (CCIE)—Bit 6

This read/write bit enables an interrupt in case of all commands being completed in the Flash.

- 0 = Command Complete interrupts disabled
- 1 = An interrupt will be requested whenever the CCIF flag is set

### 6.7.2.7 Enable Security Key Writing (KEYACC)—Bit 5

This bit can be read; however, it can only receive writing if the KEYEN bit in the FMSECH register is set.

- 0 = Flash writes are interpreted as the start of a program or erase sequence
- 1 = Writes to Flash array are interpreted as keys to open the back door

### 6.7.2.8 Reserved—Bits 4–2

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.



### 6.7.2.9 Bank Select (BKSEL)—Bits 1–0

This read/write bit field selects which register bank is addressed. Two bits are used because it is compatible with existing logic, allowing for future expansion. [Table 6-6](#) illustrates how the BKSEL bit field is used to select between Program/Boot and Data Flash.

**Table 6-6. BKSEL**

BKSEL	Flash Selected (for all program sizes)
00	Program or Boot
01	Data
10	Program or Boot
11	Data

### 6.7.3 Security Registers (FMSECH and FMSECL)

The FMSECH and FMSECL registers are unbanked. They are used to store the Flash security word. They are defined in [Figure 6-8](#) and [Figure 6-9](#).

FM_Base + \$3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	KEYEN	SECSTAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Write																
Reset	F <sup>1</sup>	F <sup>2</sup>	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset state loaded from Flash Configuration Field (SECH\_VALUE) during reset.
2. Reset state determined by value in FMSECL. If FMSECL=0xE70A, the bit is set and chip is secured.

**Figure 6-8. Security High Register (FMSECH)**

FM_Base + \$4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SEC															
Write																
Reset	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>

1. Reset state loaded from Flash Configuration Field (SECL\_VALUE) during reset.

**Figure 6-9. Security Low Register (FMSECL)**

Bits 15 and 14 of the FMSECH register illustrated in [Figure 6-8](#), and each of the 16 bits of FMSECL register, defined in [Figure 6-9](#), can be read; however, neither register can be modified by writing.

During the reset sequence the FM locations SECL\_VALUE and SECH\_VALUE are loaded into the FMSECL and FMSECH registers respectively. Please see [Table 6-5](#) for the specific address offset of SECL\_VALUE and SECH\_VALUE.

#### 6.7.3.1 Enable Back Door Key to Security (KEYEN)—Bit 15

- 0 = Back door to Flash is disabled
- 1 = Back door to Flash is enabled

#### 6.7.3.2 Security Status (SECSTAT)—Bit 14

- 0 = Flash Security is disabled
- 1 = Flash Security is enabled

#### 6.7.3.3 Reserved—Bit 13–0

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 6.7.3.4 Security (SEC)—Bits 15–0

The Security (SEC) bits define the security state of the device. [Table 6-7](#) outlines the single code enabling the security feature in the Flash Memory. This register is loaded by the 16-bit controller from the FM Configuration Field (SECL\_VALUE) at reset. It is absolutely necessary to program the FM Configuration Field (SECL\_VALUE) located in Program Flash with the required value.

**Table 6-7. SEC**

SEC	Description
0xE70A	Flash Secured <sup>1</sup>
All other combinations	Flash Unsecured

1. The 0xE70A value was selected because it represents an illegal instruction on the 56800E core, making it unlikely user compiled code accidentally programmed in the SECL\_VALUE in the FM Configuration Field location would unintentionally secure the Flash.

**WARNING:**

To perform product analysis when security is enabled, either security must be disabled by the back door key, or the array must be totally erased and verified blank.

The security function in the FM is described in [Section 6.5.4](#).

### 6.7.4 Flash Optional Data Registers (FMOPTn)

The Flash Optional Data registers are unbanked. At reset, these *read-only* registers are loaded with factory calibration values to be used by applications at run-time. Refer to the chip specific data sheet for information to determine how to use the peripheral calibration values.

FM_Base + \$1A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	TSENSOR HOT TEMP											
Write																
Reset	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>

1. Reset state loaded from Flash information block at reset.

**Figure 6-10. Flash Optional Data Register 0 (FMOPT0)**

#### 6.7.4.1 Reserved—Bits 15–12

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 6.7.4.2 TSENSOR Hot Temp—Bits 11–0

This bit field contains the ADC voltage conversion value for the Temperature Sensor at Hot Temperature (150°C). This value is referenced as  $V_{HOT}$  in the Temperature Sensor section.

FM_Base + \$1B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	RELAXATION TRIM 8MHz									
Write																
Reset	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>

1. Reset state loaded from Flash information block at reset.

**Figure 6-11. Flash Optional Data Register 1 (FMOPT1)**

### 6.7.4.3 Reserved—Bits 15–10

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 6.7.4.4 Relaxation TRIM 8MHz—Bits 9–0

This bit field contains the value required to trim the internal relaxation oscillator to 8MHz as measured at the factory. Refer to the chip specific data sheet to determine if the chip in question has an internal relaxation oscillator.

FM_Base + \$1C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	TSENSOR ROOM TEMP											
Write																
Reset	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>

1. Reset state loaded from Flash information block at reset.

**Figure 6-12. Flash Optional Data Register 2 (FMOPT2)**

### 6.7.4.5 Reserved—Bits 15–12

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 6.7.4.6 TSENSOR Room Temp—Bits 11–0

This bit field contains the ADC voltage conversion value for the Temperature Sensor at Room Temperature (25°C). This value is referenced as  $V_R$  in the Temperature Sensor section.

## 6.8 Banked Register Definition

**Table 6-8. Flash Memory Map**

Device	Peripheral	Address
8300	FM_BASE	\$00F400

Banked registers share the same register address offset as the equivalent registers for the other flash blocks. The active register bank is selected by a BKSEL bit field in the FMCR. Flash Memory contains two sets of banked registers. Bank0 is for Program and Boot Flash and Bank1 is for Data Flash.

**Table 6-9. Banked Flash Registers**

Address + Offset	Address Acronym	Register Name	Chapter Location
FM_BASE + \$10	FMPROT	Protection Register	<a href="#">Section 6.8.1</a>
FM_BASE + \$11	FMPROTB	Protection Boot Register	<a href="#">Section 6.8.2</a>
		Reserved	
FM_BASE + \$13	FMUSTAT	User Status Register	<a href="#">Section 6.8.3</a>
FM_BASE+ \$14	FMCMD	Command Buffer Register	<a href="#">Section 6.8.4</a>

Bit fields of each of the four registers are illustrated in [Table 6-13](#). Details of each follow.

Addr. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$10	FMPROT	R	PROTECT															
		W																
\$11	FMPROTB	R	0	0	0	0	0	0	0	0	0	0	0	0	PROTECT			
		W																
RESERVED																		
\$13	FMUSTAT	R	0	0	0	0	0	0	0	0	CBEIF	CCIF	PVIOL	ACC ERR	0	BLANK	0	0
		W																
\$14	FMCMD	R	0	0	0	0	0	0	0	0	CMD							
		W																

R	0	Read as 0
W		Reserved

**Figure 6-13. Banked Register Map Summary**

### 6.8.1 Protection Register (FMPROT)

This register is banked. It defines which data or Program Flash Sectors are protected against program and erase. When Bank1 is selected, BKSEL = (01 or 11), this register contains the Data Flash Protections. When Bank0 is selected, BKSEL = (00 or 10), this register contains the program Flash Protections.

FM_Base + \$10	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PROTECT															
Write																
Reset	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>

1. Reset state loaded from FM Configuration Field during reset.

**Figure 6-14. Protection Register (FMPROT)**

This register is loaded by the 16-bit controller from the FM Configuration Field at reset. The PROT\_BANK1\_VALUE defines the protection for Data Flash and the PROT\_BANK0\_VALUE defines the protection for the Program Flash.

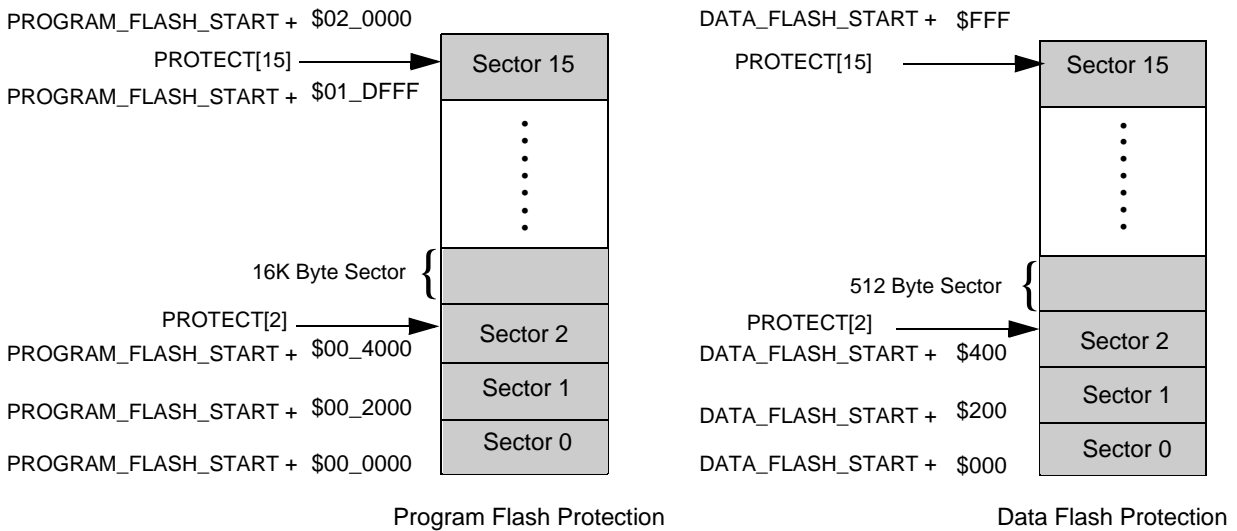
After reset, the protections may temporarily be changed by writing a new value into this register. This read/write register can only be modified when LOCK = 0 in the FMCR register.

To change the Flash Protection to be loaded on reset, the FM Configuration Field must be modified. To do this, the protection bit for the top PROGRAM FLASH SECTOR (bit15) must first be unprotected because this sector contains the FM configuration field. Then the Flash page must be erased. The Protection words (PROT\_BANK1\_VALUE, PROT\_BANK0\_VALUE and PROTB\_VALUE) must then be programmed with the desired values.

PROTECT[15:0] — Each Program or Data Flash sector can be protected from program and erase by setting PROTECT[M] bit.

- PROTECT[M] = 0: Array sector M is not protected
- PROTECT[M] = 1: Array sector M is protected

The FMPROT register is banked and controls the protection of 16 sectors within each Flash block. The size of the sector is determined by the size of the Flash Memory block; 16k byte sectors in a 256k byte section of program memory and 16 × 512 byte sectors within the 8k Data Flash. The example in [Figure 6-15](#) outlines the association between each bit in the FMPROT register banks and the corresponding Flash Memory sector within the Program and Data Flash arrays. For the actual map of the part, **please consult the Technical Data sheet** for the device being implemented.



**Figure 6-15. Protection Diagram Example**

## 6.8.2 Protection Boot Register (FMPROTB)

This register is banked. It defines which Boot Flash sectors are protected against program and erase. When Bank0 is selected, BKSEL = (00 or 10), this register contains the Boot Flash protections. When Bank1 is selected, BKSEL = (01 or 11), this register is invalid since Bank1 is used for Data Flash and its protection is defined in the FMPROT register.

FM Base + \$11	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	PROTECT			
Write													PROTECT			
Reset	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>

1. Reset state loaded from Flash Configuration Field during reset.

**Figure 6-16. Protection Boot Register (FMPROTB)**

This register is loaded by the 16-bit controller from the FM Configuration Field at reset. The PROT<sub>B</sub>\_VALUE defines the protection for Boot Flash.

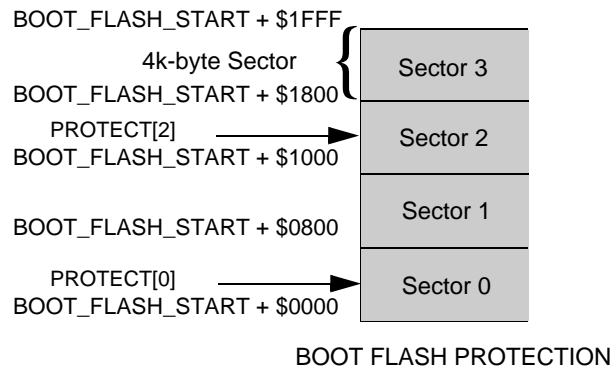
After reset, the protections may temporarily be changed by writing a new value to this register. This read/write register can only be modified when LOCK = 0 in the FMCR register.

To change the Boot Flash protection to be loaded on reset, the FM Configuration field must be modified, as described in [Table 6.8.1](#).

PROTECT[3:0] — Each Boot Flash sector can be protected from program and erase by setting PROTECT[M] bit.

- PROTECT[M] = 0: Array sector M is not protected
- PROTECT[M] = 1: Array sector M is protected

The Protection Boot register is banked. It controls the protection of four, 4k-byte sectors in a 16k-byte section of Boot Memory. The example in [Figure 6-17](#) outlines the association between each bit in the PROT B register banks and its corresponding FM sector. For the actual map of the part, **please consult the Technical Data sheet** for the device being implemented.



**Figure 6-17. Boot Protection Diagram Example**

### 6.8.3 User Status Register (FMUSTAT)

The Flash Memory User Status (FMUSTAT) register is banked. It defines the Flash state machine command status, access error, protection violation, and blank verify status. FMUSTAT register bits 7, 5, 4, and 2 are read/write. Bit 6 in this register is *read-only*.

FM_Base + \$13	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
Write																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

**Figure 6-18. User Status Register (FMUSTAT)**

**Note:** Only one bit should be cleared at a time in the FMUSTAT register because of the nature of the State Machine clearing the register bits.

#### 6.8.3.1 Reserved—Bits 15–8

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.



### 6.8.3.2 Command Buffer Empty Interrupt Flag (CBEIF)—Bit 7

The CBEIF flag indicates the address, data, and command buffers are empty so a new command sequence can be started. The CBEIF flag is cleared by writing 1 to it. Writing 0 has no effect on CBEIF; however, writing 0 to the CBEIF bit can be used to abort a command sequence. The CBEIF bit can generate an interrupt if the CBEIE bit in the Configuration register is set. While the CBEIF flag = 0 the FMCMD register cannot be written.

- 0 = Buffers are full
- 1 = Buffers are ready to accept a new command

### 6.8.3.3 Command Complete Interrupt Flag (CCIF)—Bit 6

The CCIF flag indicates there are no pending commands. The CCIF flag is set and cleared automatically upon start and completion of a command. Writing to the CCIF bit has no effect. The CCIF bit can generate an interrupt if the CCIE bit in the Configuration register is set.

- 0 = Command in progress
- 1 = All commands are completed

### 6.8.3.4 Protection Violation (PVIOL)—Bit 5

The PVIOL flag indicates an attempt was made to program, or erase an address in a protected Flash memory area. Clear the PVIOL flag by writing 1 to the bit. Writing 0 to the bit has no effect on PVIOL. While the PVIOL flag is set in any banked register, it is not possible to launch another command.

- 0 = No failure
- 1 = A protection violation has occurred

### 6.8.3.5 Access Error (ACCERR)—Bit 4

The ACCERR flag indicates an illegal access to the Flash Memory array, or registers caused by a bad program, or an erase sequence. Clear the ACCERR flag by writing 1 to the bit. Writing 0 to the ACCERR bit has no effect. While the ACCERR flag is set in any banked register, it is not possible to launch another command. Please see [Section 6.5.3.4](#) for ACCERR flag setting details.

- 0 = No failure
- 1 = Access error has occurred

### 6.8.3.6 Reserved—Bit 3

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 6.8.3.7 Block Verified Erased (BLANK)—Bit 2

The BLANK flag indicates an Erase Verify command (RDARY1) has checked the Flash block and found it to be blank. Clear the BLANK flag by writing 1 to the bit. Writing 0 to the bit has no effect.

- 0 = If an Erase Verify command is requested, the CCIF flag is set. A zero in BLANK indicates the block is not erased
- 1 = Flash block verifies as erased

### 6.8.3.8 Reserved—Bit 1–0

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

## 6.8.4 Command Buffer Register (FMCMD)

This banked register defines the Flash commands.

FM_Base + \$14	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	CMD						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6-19. Command Buffer Register (FMCMD)

### 6.8.4.1 Reserved—Bits 15–7

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 6.8.4.2 Command—Bits 6–0

Valid commands are illustrated in [Table 6-10](#). Writing a command other than those listed in [Table 6-10](#) will set the ACCERR flag in the User Status register.

Table 6-10. Flash CMD User Mode Commands

Command	Name	Description
\$05	RDARY1	Erase Verify (All Ones)
\$20	PGM	Word Program
\$40	PGERS	Page Erase
\$41	MASERS	Mass Erase

## 6.9 Interrupts

The Flash Memory module generates an interrupt when all Flash commands are completed, or when the address, data, and command buffers are empty. Interrupts can also be generated when the ACCERR bit is set.

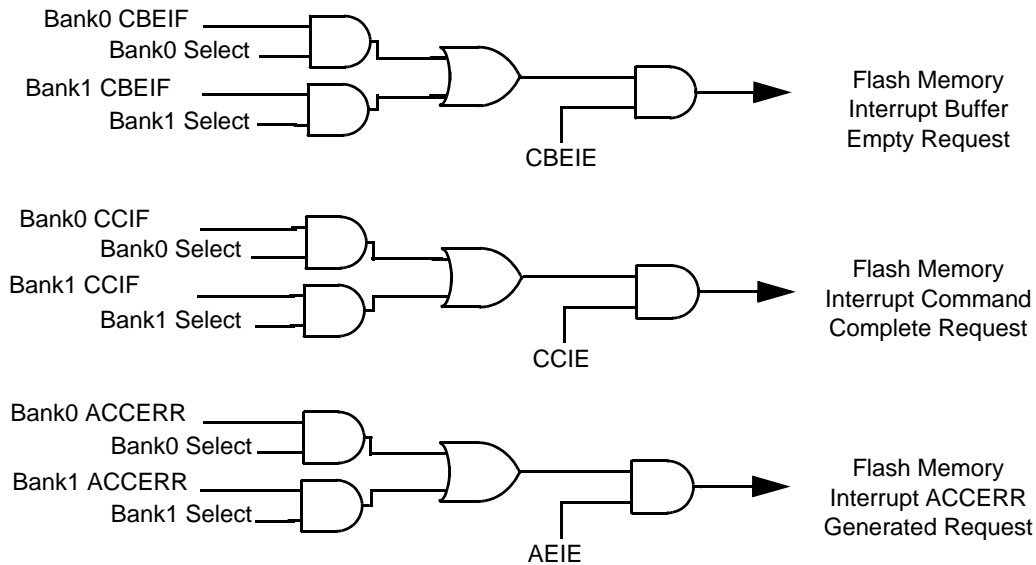
**Table 6-11. Flash Memory Interrupt Sources**

Interrupt Source	Interrupt Flag	Local Enable
Flash Command, Data and Address Buffer empty	CBEIF (FMUSTAT)	CBEIE (FMCR)
All commands are completed on Flash	CCIF (FMUSTAT)	CCIE (FMCR)
ACCERR generated	ACCERR (FMUSTAT)	AEIE (FMCR)

**Note:** Vector addresses and their relative interrupt priority are determined at the 16-bit controller level.

### 6.9.1 Interrupt Operation Description

**Figure 6-20** illustrates the operation of the interrupt request generated by this module. To enable an interrupt, this system uses the CBEIE, CCIE, and the Bank Select (BKSEL). By taking into account the selected bank, the module is prevented from generating false interrupts when the command buffer is empty in a non-selected bank.



**Figure 6-20. Interrupt Implementation**

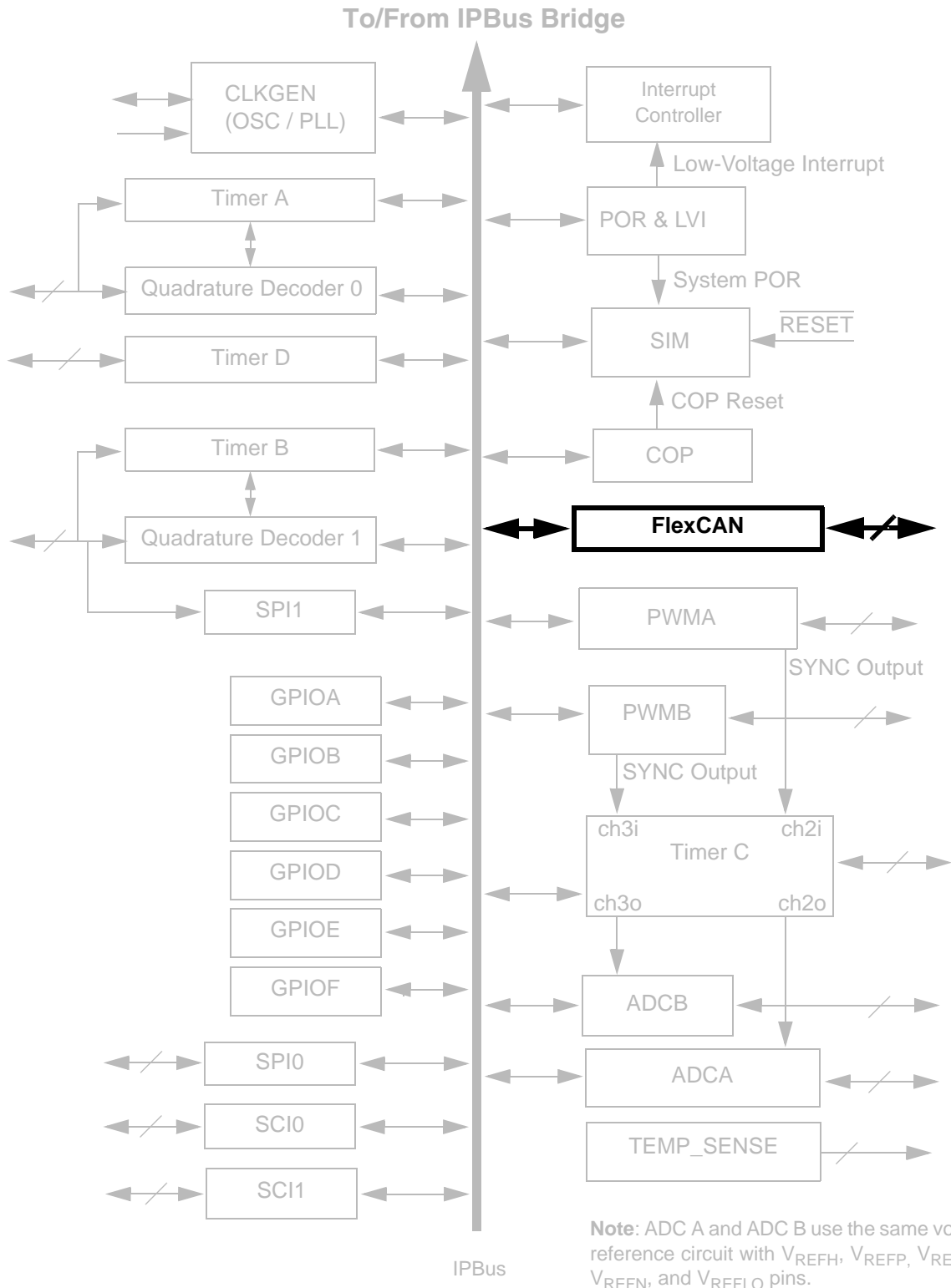
## 6.10 Resets

If a reset occurs while any command is in progress, that command will immediately be aborted. The state of the word being programmed or the page/block being erased is not guaranteed after a reset interrupt. The following registers are loaded from the FM Configuration Field in Program Flash on Reset.

- Data Protection Register (FM\_BASE + \$10 - FMPROT) in Bank1 from FM Configuration Field location PROT\_BANK1\_VALUE
- Program Protection Register (FM\_BASE + \$10 - FMPROT) in Bank0 from FM Configuration Field location PROT\_BANK0\_VALUE
- Boot Protection Register (FM\_BASE + \$11 - FMPROTB) in Bank0 from FM Configuration Field location PROTB\_VALUE
- Security High Register (FM\_BASE + \$03 - FMSECH) a common register from FM Configuration Field location SECH\_VALUE.
- Security Low Register (FM\_BASE + \$04 - FMSECL) a common register from FM Configuration Field location SECL\_VALUE

# Chapter 7

## FlexCAN (FC)



FlexCAN (FC), Rev. 10

## Document Revision History for **Chapter 7, FlexCAN (FC)**

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 3.0	<p>Section 7.6.3, third paragraph, second line, deleted (CODE=0100). Only if the Receive Code is 0100 will it examine the ID_HIGH word of that MB.</p> <p>Section 7.6.9.1, Section 7.7.1, Section 7.8.1.2, Section 7.8.1.2, and Section 7.8.1.4 have been edited for clarification</p> <p>Table 7-9, ID numbers added and table devived into "FlexCAN Configuration" and "Received Message ID"</p>
Rev 4.0	<p>Added reference to the CAN protocol in Section 7.6.9.1,</p> <p>Clarified first sentence in Section 7.7.1,</p> <p>Clarified Section 7.8.1.2 and Section 7.8.1.4,</p> <p>Corrected heading and clarified information in Table 7-9,</p> <p>Minor edit in Section 7.7.2 - added "affecting the whole chip,"</p> <p>Changed CANTX/CANRX bit to TX/RX bit in Section 7.8.10.9.</p>
Rev 5.0	Converted chapter to Freescale design standards

## 7.1 Introduction

The FlexCAN (FC) module is a communication controller implementing the Controller Area Network (CAN) protocol, an asynchronous communications protocol used in automotive and industrial control systems. It is a high speed (1Mbit/sec), short distance, priority based protocol able to communicate using a variety of mediums (fiber optic cable or an unshielded twisted pair of wires, for example). The FlexCAN module supports both the standard and extended Identifier (ID) message formats specified in the CAN protocol specification, Revision 2.0, Part B.

The CAN protocol was primarily, but not exclusively, designed to be used as a vehicle Serial Data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth. A general working knowledge of the CAN protocol Revision 2.0 is assumed in this document. For details, refer to the CAN Protocol Revision 2.0 Specification.

## 7.2 Features

- Based on and includes all existing TouCAN module features
- IP interface architecture
- Full implementation of the CAN Protocol Specification Version 2.0
  - Standard Data and Remote Frames (up to 109 bits long)
  - Extended Data and Remote Frames (up to 127 bits long)
  - 0–8 bytes Data Length
  - Programmable bit rate up to 1Mbit/sec
- Up to 16 flexible message buffers of 0–8 bytes Data Length, each configurable as Receive or Transmit, all supporting Standard and Extended messages
- Listen-Only mode capability
- Content-related addressing
- No read/write semaphores
- Three programmable Mask Registers:
  - Global (for MBs 0-13)
  - Special for MB14
  - Special for MB15
- Programmable transmit-first 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 (external transceiver is assumed)
- Open network architecture
- Multimaster bus
- High immunity to EMI
- Short latency time for high-priority messages
- Low-power Sleep mode, with programmable wake-up on bus activity

### 7.3 Block Diagram

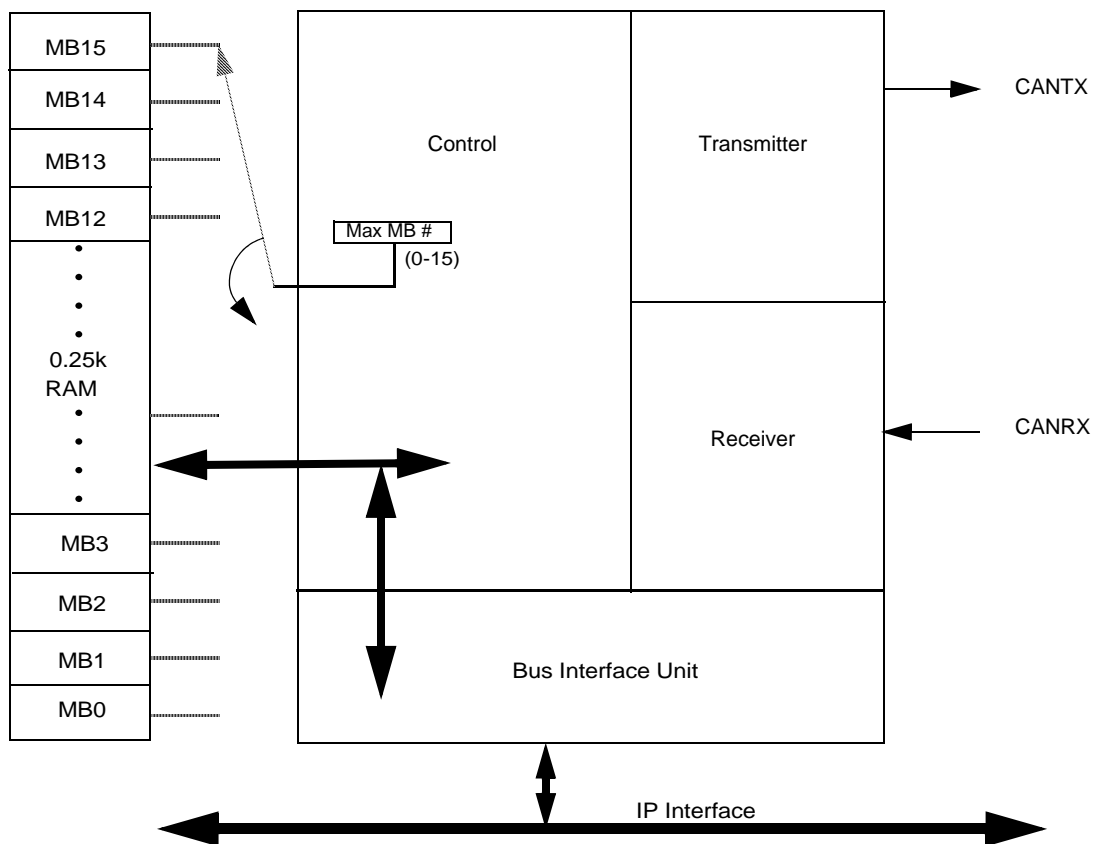
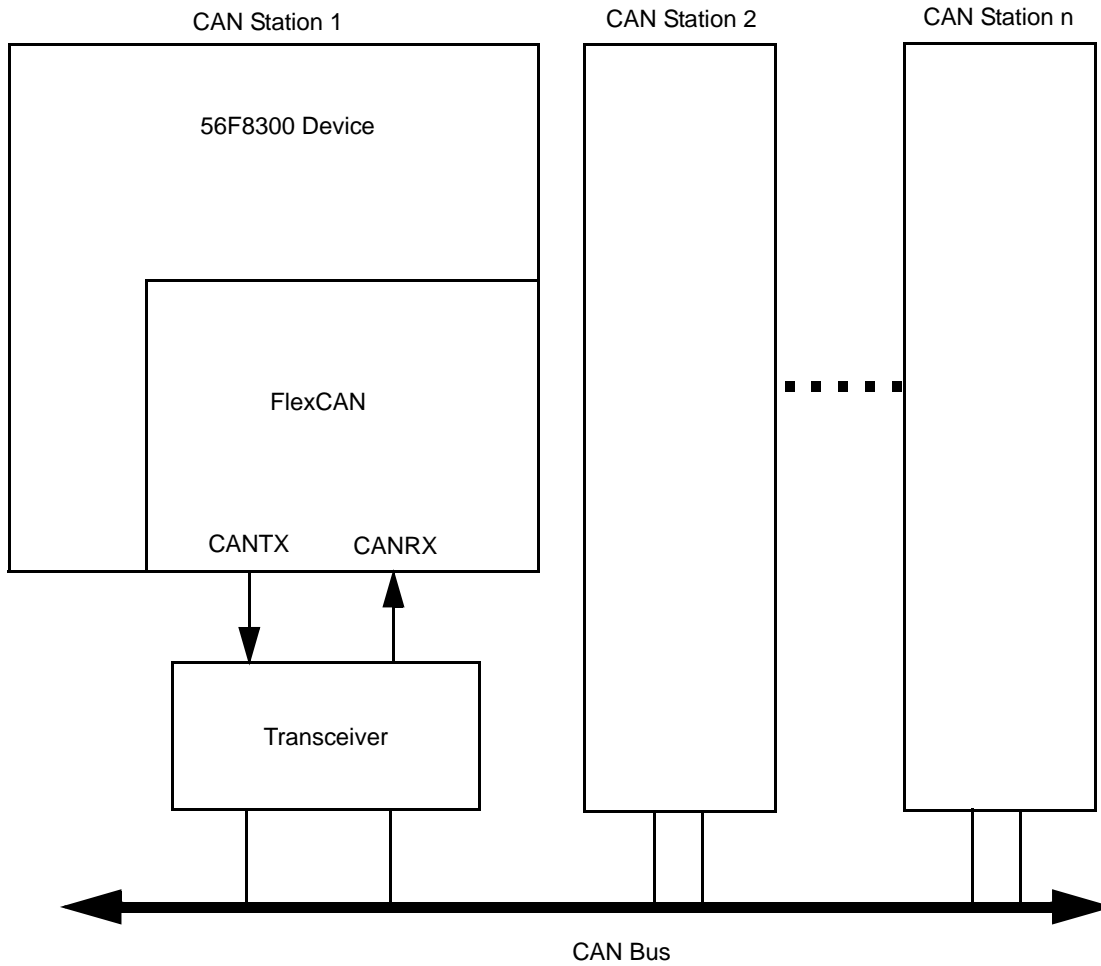


Figure 7-1. FlexCAN Block Diagram and Pinout

### 7.4 Typical CAN System Diagram

A typical CAN system is illustrated in [Figure 7-2](#).





**Figure 7-2. Typical CAN System**

Each CAN station is connected physically to the CAN bus through a transceiver. The transceiver provides the transmit drive, wave shaping, and receive/compare functions required for communicating on the CAN bus. It can also provide protection against damage to the FlexCAN module caused by a defective CAN bus or defective stations.

## 7.5 Message Buffers

### 7.5.1 Message Buffer Structure

**Figure 7-3** shows the extended (29-bit) ID message buffer structure. **Figure 7-4** displays the standard (11-bit) ID message buffer structure.

	15–8	7–4	3–0	
\$0	TIME_STAMP	CODE	LENGTH	CONTROL/STATUS
\$1	ID[28:18]		SR R	IDE
			ID[17-15]	ID_HIGH
\$2	ID[14-0]			RTR
				ID_LOW
\$3	DATA BYTE 0	DATA BYTE 1		
\$4	DATA BYTE 2	DATA BYTE 3		
\$5	DATA BYTE 4	DATA BYTE 5		
\$6	DATA BYTE 6	DATA BYTE 7		
\$7	Reserved			

**Figure 7-3. Extended ID Message Buffer Structure**

	15–8	7–4	3–0	
\$0	TIME_STAMP	CODE	LENGTH	CONTROL/STATUS
\$1	ID[28:18]		RTR	0
			0	0
			0	0
\$2	16-BIT TIME STAMP			ID_HIGH
				ID_LOW
\$3	DATA BYTE 0	DATA BYTE 1		
\$4	DATA BYTE 2	DATA BYTE 3		
\$5	DATA BYTE 4	DATA BYTE 5		
\$6	DATA BYTE 6	DATA BYTE 7		
\$7	Reserved			

**Figure 7-4. Standard ID Message Buffer Structure**

#### 7.5.1.1 Common Fields for Extended and Standard Format Frames

**Table 7-1** describes the Message Buffer (MB) fields common to both Extended and Standard Identifier Format Frames.

**Table 7-1. Common Extended/Standard Format Frames**

Field	Description
TIME STAMP	Contains a copy of the high byte of the Free-Running Timer, captured at the beginning of the identifier field of the frame on the CAN bus.
CODE	Refer to <a href="#">Table 7-2</a> and <a href="#">Table 7-3</a> .
LENGTH (Receive)	Length (in bytes) of the receive data stored in offset \$3 through \$6 of the buffer. This field is written by the FlexCAN module, copied from the Data Length Code (DLC) field of the received frame. In case the length value received in the DLC field exceeds eight, it is assumed the first eight received data bytes are stored.
LENGTH (Transmit)	Length (in bytes) of the data to be transmitted, located in offset \$3 through \$6 of the buffer. This field is written by the device, and is used as the DLC field value. If Remote Transmission Request (RTR) = 1, the frame is a Remote Frame and will be transmitted without the Data Field, regardless of the value in LENGTH.
DATA	This field can store up to eight data bytes for a frame. For Receive Frames, the data is stored as it is received from the bus. For Transmit Frames, the device provides the data to be transmitted within the frame.
Reserved	This word entry field (16 bits) should not be accessed by the device. This word is used for internal testing only. It should not be accessed in any way.

**Table 7-2. Message Buffer Codes for Receive Buffers**

RX Code Before RX New Frame	Description	RX Code After RX New Frame	Comment
0000	NOT ACTIVE — message buffer is not active	—	—
0100	EMPTY — message buffer is active and empty	0010	—
0010	FULL — message buffer is full	0110	If a device read occurs before the new frame, new receive code is 0010
0110	OVERRUN — second frame was received into a full buffer before the device read the first one		
0101 <sup>1</sup>	BUSY — message buffer is now being filled with a new receive frame. This condition will be cleared within 20 cycles	0010	An empty buffer was filled
0011 <sup>1</sup>		0110	A full buffer was filled
0111 <sup>1</sup>		0110	An overrun buffer was filled

- For transmit message buffers, upon read, the BUSY bit should be ignored.

**Table 7-3. Message Buffer Codes for Transmit Buffers**

RTR	Initial TX Code	Description	Code After Successful Transmission
X	1000	Message buffer not ready for transmit	—
0	1100	Data Frame to be transmitted once, unconditionally	1000
1	1100	Remote Frame to be transmitted once, and message buffer becomes an RX message buffer for Data Frames	0100
0	1010 <sup>1</sup>	Data Frame to be transmitted only as a response to a Remote Frame	1010
0	1110	Data Frame to be transmitted only once, unconditionally, and then only as a response to Remote Frame	1010

- When a matching remote request frame is detected, the code for such a message buffer is changed to be 1110.

### 7.5.1.2 Fields for Extended Format Frames

**Table 7-4** describes the Message Buffer (MB) fields used only for Extended Identifier Format Frames.

**Table 7-4. Extended Format Frames**

Field	Description
ID[28:18]/[17:15]	Contains the 14 most significant bits of the extended identifier, located in the ID HIGH word of the message buffer.
Substitute Remote Request  (SRR)	Contains a fixed recessive bit, used only in Extended Format. Users should set the bit to 1 for Transmit Buffers. It will be stored as received on the CAN bus for Receive Buffers. This is a bit in the ID HIGH word of the Message Buffer (MB).  If the FlexCAN module transmits a value and receives a matching response, a successful bit transmission is indicated. However, if the FlexCAN module transmits this bit as 1 and receives it as a 0, an <i>arbitration loss</i> is indicated. And, if the FlexCAN module transmits this bit as 0 and receives it as 1, a bit error is indicated.
ID Extended (IDE)	If Extended Format Frame is used, this field should be set to 1. If 0, Standard Format Frame should be used. This is a bit in the ID HIGH word of the Message Buffer (MB).
ID[14:0]	Bits [14:0] of the Extended Identifier, located in the ID LOW word of the Message Buffer (MB).
Remote Transmission Request (RTR)	This bit is located in the Least Significant Bit (LSB) of the ID LOW word of the Message Buffer (MB); 0 Data Frame 1 Remote Frame.

### 7.5.1.3 Fields for Standard Format Frames

**Table 7-5** describes the Message Buffer (MB) fields used only for Standard Identifier Format Frames.

**Table 7-5. Standard Format Frames**

Field	Description
ID[28:18]	Contains bits [28:18] of the identifier, located in the ID HIGH word of the Message Buffer (MB). The four Least Significant Bits (LSBs) in this register (corresponding to the IDE bit and ID[17:15] for an Extended Identifier Message) must all be written as Logic 0s, ensuring proper operation of the FlexCAN module, illustrated in <b>Section 7-4</b> .
Remote Transmission Request  (RTR)	This bit is located in the ID HIGH word of the Message Buffer (MB): 0 Data Frame 1 Remote Frame  If the FlexCAN module transmits a value and receives a matching response, a successful bit transmission is indicated. However, if the FlexCAN module transmits this bit as 1 but receives it as 0, an <i>arbitration loss</i> is indicated. And if the FlexCAN module transmits this bit as 0 and receives it as 1, a bit error is indicated.
16-Bit Time Stamp	The 16-bit time stamp, located in the ID LOW word of the Message Buffer (MB), is not needed for standard format, and is used in a Standard Format Buffer to store the 16-bit value of the Free-Running Timer. The timer value is captured at the beginning of the identifier field of the frame on the CAN bus.

## 7.6 Functional Overview

The FlexCAN module is flexible, allowing each one of its 16 Message Buffers (MBs) to be assigned either as a Transmit Buffer or a Receive Buffer. Each MB is also assigned an interrupt flag bit indicating successful completion of either transmission or reception.

**Note:** For both processes, the first device action in preparing a MB should be to deactivate it by setting its code field to the proper value. This requirement is mandatory to assure proper operation.

### 7.6.1 Transmit Process

The device prepares, or changes an MB for transmission by executing the following steps:

- Writing the Control/Status word to hold the transmit MB inactive (CODE = 1000)
- Writing the ID\_HIGH and ID\_LOW words
- Writing the Data bytes
- Writing the Control/Status word (active CODE, LENGTH)

**Note:** The first and last steps are mandatory.

Beginning from the last step, this MB will participate in the Internal Arbitration Process, taking place every time the receiver, or at the interframe space, senses the CAN Bus is free with at least one MB ready for transmission. This Internal Arbitration Process is intended to select the next MB to be transmitted.

When this process is over with a *winning* MB for transmission, the frame is transferred to the Serial Message Buffer (SMB, see [Section 7.6.3.1](#)) for transmission (Move Out).

While transmitting, the FlexCAN module transmits up to eight data bytes, even if the LENGTH field value is larger. (When LENGTH > 8, DLC = 8 in the transmitted frame.)

At the end of the successful transmission

- The value of the Free-Running Timer (captured at the beginning of the Identifier Field on the CAN bus) is written into the TIME\_STAMP field in the MB.
- The CODE field in the Control/Status word of the MB is updated
- A Status Flag is set in the Interrupt Flag Register (FCIFLAG1, [Section 7.8.12](#))

#### 7.6.1.1 Transmit Interrupts

Each one of the MBs can be an interrupt source if its corresponding FCIMASK1 bit is set. No distinction is made between Receive and Transmit Interrupts for a particular MB. Refer to

[Section 7.8.11](#), [Section 7.8.12](#), and [Section 7.9](#) for additional information about the FCIMASK1 register and interrupts.

### 7.6.1.2 Transmit Polling

If using software polling for transmitting, the FCIFLAG1 register should be read to determine transmitter status.

**Warning:** Do not read the MB's Control/Status to determine Transmitter Status because this procedure causes the MB to lock.

Refer to [Section 7.8.12](#) for additional information on the FCIFLAG1 register.

## 7.6.2 Receive Process

The device prepares, or changes a MB for frame reception through the following steps:

- Writing the Control/Status word to Hold the Receive MB Inactive (CODE = 0000)
- Writing the ID\_HIGH and ID\_LOW words
- Writing the Control/Status word to mark the Receive MB as Active and Empty (CODE = 0100)

**Note:** The first and last steps are mandatory.

Beginning from the last step, this MB is an active Receive Buffer and will participate in the internal matching process, taking place every time the Receiver receives an error-free frame. In this process, all active Receive Buffers compare their ID value to the newly received one. If a match occurs, the frame is transferred (Move In) to the first (the lowest entry) matching MB. The value of the Free-Running Timer (captured at the beginning of the IDENTIFIER field on the CAN bus) is written into the TIME\_STAMP field in the MB. The ID field, Data field (8 bytes at most), and the LENGTH field are then stored, the CODE field is updated, and a BUF $n$ I flag is set in the Interrupt Flag register (FCIFLAG1).

The device should read a receive frame from its MB in the following way:

- Control/Status word (mandatory—activates internal lock for this buffer)
- ID (Optional—essential only if a mask was used)
- Data field word(s)
- Free-Running Timer (releases internal lock—optional)

The read of the Free-Running Timer is not mandatory. If not executed, the MB remains locked unless the device starts the read process for another MB.

**Note:** Only a single MB is locked at any time.

The only mandatory device read operation is of the Control/Status word, assuring data coherency. If the BUSY bit (the LSB in the CODE field) is set, the device should defer until this bit is cleared.

**Note:** The received IDENTIFIER field is always stored in the matching MB, thus the contents of the ID field in a MB may change if the match was due to a mask.

**Note:** If the number of data bytes is odd (as indicated in LENGTH), the last byte will be duplicated to fill the last 16-bit word. The entire data field received is written during Move In, so if the number of data bytes received is less than eight, unused bytes in the MB do not retain their previous values.

### 7.6.2.1 Receive Interrupts

Each one of the MBs can be an interrupt source if its corresponding FCIMASK1 bit is set. No distinction is made between Receive and Transmit Interrupts for a particular MB. Refer to [Section 7.8.11](#), [Section 7.8.12](#), and [Section 7.9](#) for additional information about the FCIMASK1 register and interrupts.

### 7.6.2.2 Receive Polling

If using software polling for receiving, the FCIFLAG1 register should be read to determine receiver status.

**Warning:** Do not read the MB's Control/Status to determine Receiver Status because this procedure causes the MB to lock.

Refer to [Section 7.8.12](#) for additional information about the FCIFLAG1 register.

### 7.6.2.3 Self-Received Frames

The FlexCAN module receives self-transmitted frames if a matching receive MB exists.

## 7.6.3 Message Buffer Handling

To maintain data coherency and proper FlexCAN operation, the device must obey the rules listed in [Section 7.6.1](#) and in [Section 7.6.2](#). Deactivation of an MB is a core action causing that MB to be excluded from FlexCAN Transmit or Receive processes. Any device write access to a Control/Status word of MB structure deactivates MB. This process excludes it from Receive/Transmit processes. Any form of device MB structure access within the FlexCAN module (other than those specified in [Section 7.6.1](#) and in [Section 7.6.2](#)) may cause the FlexCAN module to behave unpredictably.

The Match/Arbitration Processes are performed only during one period by the FlexCAN module. Once a winner, or match is determined there is no re-evaluation whatsoever, ensuring a Receive

Frame is not lost. Two or more Receive MBs holding a matching ID to a Received Frame do not assure reception in the FlexCAN module when deactivating the matching MB after FlexCAN scanned the second.

Suppose MB0 and MB1 are configured to receive a frame matching the same ID. The lowest number MB (MB0) has the priority to receive the message. If matching the ID is received it should Move In to MB0. But if MB0 is locked, it will remain in the Serial Message Buffer (SMB), discussed in [Section 7.6.3.1](#), and the Move is not guaranteed if MB0 is deactivated after first scanning MB1.

In the scanning process, the FlexCAN module will read the Control/Status word of each MB and search for an Active Receive Code. If the ID\_HIGH word shows the MB is configured to receive extended frames, the FlexCAN module will examine ID\_LOW, otherwise it will go on to the next MB's Control/Status word.

In this case FlexCAN will find a match after scanning MB0, but it will not quit after scanning because MB0 is locked and it will go on to MB1. It will also find MB1 active and match the ID, but it has a lower priority, hence the data is not stored. Thus, if purposely deactivating MB0 after scanning MB1, at the end of the scanning process, FlexCAN will not find an active MB into which data is moved.

### 7.6.3.1 Serial Message Buffers (SMBs)

To allow double buffering of messages, the FlexCAN module has two shadow buffers called Serial Message Buffers (SMBs). These two buffers are used by the FlexCAN module for buffering both received messages and messages to be transmitted. Only one SMB is active at a time, and its function depends upon the operation of the FlexCAN module at that time. Neither of these two buffers are accessible or visible at any time.

### 7.6.3.2 Transmit Message Buffer Deactivation

Any write access to the Control/Status word of a Transmit MB while selecting a MB for transmission will immediately deactivate that MB, thereby removing it from the transmission process.

- If a Transmit MB is deactivated while a message is being transferred from a Transmit Message Buffer to a SMB, the message is not transmitted.
- If a Transmit MB is deactivated after the message is transferred to the SMB, the message is transmitted, but an Interrupt is not generated and the TX CODE is not updated.
- A message may not be transmitted if an MB containing the lowest ID is deactivated while that message is undergoing the internal arbitration process in order to determine which message should be sent.



### 7.6.3.3 Receive Message Buffer Deactivation

Any write access to the Control/Status word of a Receive MB while selecting an MB for reception will immediately deactivate that MB, thereby removing it from the Reception process.

- If a Receive MB is deactivated while a message is being transferred into it, the transfer is halted and no interrupt is requested. If this occurs, that Receive MB may contain mixed data from two different frames.

**Warning:** *Data should never be written into a Receive MB.* If this is performed while a message is being transferred from an SMB, the Control/Status word reflects a full, or overrun condition, but no interrupt will be requested. *This action is absolutely forbidden!*

### 7.6.4 Lock/Release/Busy Mechanism and SMB Usage

This mechanism is implemented, assuring data coherency in both the Receive and Transmit processes. The mechanism includes Lock Status for an MB and two SMBs to Buffer Frame Transfers within the FlexCAN module.

- Device reading a Control/Status word of a MB triggers a Lock for that MB, i.e. a new Receive Frame matching this MB *cannot* be written into this MB.
- In order to release a Locked MB, the device should either lock another MB by reading its Control/Status word, or globally release any Locked MB by reading the Free-Running Timer.
- If, while a MB is locked, a Receive Frame with a matching ID is received, it can not be stored within that MB, remaining in the SMB. There is *no* indication for that situation.
- If, while a MB is locked, two or more Receive Frames with matching IDs are received, the last received is kept within the SMB while all preceding Receive Frames are lost. There is *no* indication for that situation.
- If a locked MB is released, and there exists a matching frame within the SMB, the frame is then transferred to the matching MB.
- If the device reads a Receive MB while the SMB is being transferred, the BUSY CODE bit is set in the Control/Status word. The device should wait until this bit is cleared before further reading from that MB, assuring data coherency. In this case the MB is *not* locked.
- If the device deactivates a Locked MB, its Lock status is cleared, but no data is transferred into that MB.

### 7.6.5 Remote Frames

The Remote Frame is a message frame transmitted to request a Data Frame. The FlexCAN module can be configured to transmit a Data Frame automatically in response to a Remote

Frame, or to transmit a Remote Frame and then wait for the responding Data Frame to be received.

When transmitting a Remote Frame:

- Initialize an MB as a Transmit MB with the Remote Transmission Request (RTR) bit set to 1. Once this Remote Frame is successfully transmitted, the Transmit MB automatically becomes a Receive MB with the same ID as the transmitted remote frame.

When a Remote Frame is received by the FlexCAN module:

- The Remote Frame ID is compared to the IDs of all Transmit MBs programmed with a CODE of 1010.
- If there is an exact matching ID, the Data Frame in that MB is transmitted.
- If the RTR bit in the matching Transmit MB is set, the FlexCAN module transmits a Remote Frame as a response.

A Received Remote Frame is not stored in a Receive MB. It is only used to trigger the automatic transmission of a frame in response. The Mask registers are not used in Remote Frame ID matching. All ID bits (except RTR) of the incoming Received Frame must match for the Remote Frame to trigger a response transmission.

In the case a Remote Request Frame is received and matched to a Transmit MB, this MB immediately enters the Internal Arbitration Process, but only as a normal transmit MB with no higher priority.

## 7.6.6 Overload Frames

Overload Frame transmissions are not initiated by the FlexCAN module unless certain conditions are detected on the CAN bus. These conditions include:

- Detection of a dominant bit in the first or second bit of intermission
- Detection of a dominant bit in the seventh (last) bit of the End-of-Frame (EOF) field in Receive Frames
- Detection of a dominant bit in the eighth (last) bit of the Error Frame Delimiter or Overload Frame Delimiter

## 7.6.7 Time Stamp

The value of the Free-Running, 16-bit Timer is sampled at the beginning of the Identifier field on the CAN bus.

- For a message being received the Time Stamp will be stored in the TIME\_STAMP field of the Receive MB at the moment the message is written into that buffer. For a message

being transmitted, the TIME\_STAMP field will be written into the Transmit MB once the transmission is successfully completed.

The Free-Running Timer can optionally be reset upon the reception of a frame into Message Buffer0 (MB0). This feature allows network time synchronization to be performed. See the TSYNC bit in the FCCTL0 Register, [Section 7.8.2](#).

### 7.6.8 Listen-Only Mode

In Listen-Only mode, the FlexCAN module is able to receive messages acknowledged by another CAN station. Whenever the module enters this mode the status of the Error Counters is frozen and the FlexCAN module operates in Error Passive mode. The Fault Confinement state bits in the Error and Status register indicate a Passive Error state irrespective of Error Counter values. Because the module does not influence the CAN bus in this mode, the device is capable of functioning like a monitor, or for automatic bit-rate detection. However, FlexCAN will only monitor valid transmissions not resulting in an error. This requires another CAN module be on the bus to provide the Acknowledge (ACK) bit and complete the transmission.

Once this mode is set, the FlexCAN module waits to be in either Intermission, Passive Error, Busoff, or Idle state. During this wait period, the FlexCAN module awaits all internal activity to complete before entering this mode other than that activity in the CAN bus interface.

### 7.6.9 Bit Timing

The FlexCAN module supports a variety of means to setup the bit timing parameters required by the CAN protocol. There are two 16-bit registers (FCCTL0 and FCCTL1) enabling the determination of the value of various fields of the bit timing parameters. Propagation Segment (PROPSEG), Phase Segment 1/2 (PSEG1 and PSEG2), and Resynchronization Jump Width (RJW) are programmed through fields in the FCCTL0 and FCCTL1 registers. Please see [Section 7.8.2](#) and [Section 7.8.3](#). Also, the FlexCAN module holds a Prescaler (PRES\_DIV) enabling determination of the ratio between the system clock and the SCLOCK, the actual Time Quanta Clock.

**Table 7-6. Examples of System Clock/CAN Bit-Rate/S-Clock**

System Clock Freq (Mhz)	Can Bit-Rate (Mhz)	Possible S-Clock Freq (Mhz)	Possible Number of Time Quanta/Bit	Pre-Scaler Programed Value + 1	Comments
60	1	20, 15, 12, 10	20, 15, 12, 10	3, 4, 5, 6	Min 8-Time Quanta Max 25-Time Quanta
56	1	14, 8	14, 8	4, 7	
54	1	18, 9	18, 9	3, 6	
50	1	25, 10	25, 10	2, 5	
60	0.125	3, 2.5, 2, 1.875, 1.5, 1.25, 1	24, 20, 16, 15, 12, 10, 8	20, 24, 30, 32, 40, 48, 60	

### 7.6.9.1 Configuring the FlexCAN Module Bit Timing

For details of CAN bit timing refer to the CAN Protocol Revision 2.0 Specification. The following considerations must be observed when programming bit timing functions:

- If the programmed PRES\_DIV value results in a single system clock per one time quantum, the PSEG2 field in FCCTL1 register *should not* be programmed to zero.
- If the programmed PRES\_DIV value results in a single system clock per one time quantum, the Information Processing Time (IPT) equals three time quanta, otherwise it equals two time quanta. If PSEG2 = 2, the FlexCAN module transmits one time quantum late relative to the scheduled sync segment.
- If the two conditions below are met, the FlexCAN module may not be able to prepare a MB for transmission in time to begin its own transmission and arbitrate against the CAN node which transmitted an early SOF.
  - When the prescaler and bit timing control fields are programmed to values resulting in fewer than 10-system clock periods per CAN bit time, and
  - CAN bus loading is 100 percent, anytime the rising edge of a Start-of-Frame (SOF) symbol transmitted by another node occurs during the third bit of the intermission between messages
- The FlexCAN module bit time must be programmed to be greater than, or equal to, nine system clocks, or correct operation is not guaranteed.

### 7.6.10 FlexCAN Initialization/Reset Sequence

The FlexCAN module may be reset in two ways:

1. Hard reset using system reset line
2. Assert the SOFT\_RST in the FCMCR

Following the negation of either reset, the FlexCAN module is not synchronized with the CAN bus, and HALT and FRZ1 bits in FCMCR are set. Its main control is disabled, and FREEZ\_ACK and NOT\_RDY bits in the FCMCR are set. The FlexCAN module CANTX pin is in recessive state and it does not initiate frame transmission, nor does it receive any frames from CAN bus. The MB contents are *not* changed following SOFT\_RESET or hard reset.

It is required for any configuration change/initialization the FlexCAN module either to be *frozen* by asserting the HALT bit in FCMCR or be reset. Please see [Section 7.7.1](#).

The following is a generic initialization sequence applicable for the FlexCAN module:

- Initialize all operation modes:
  - Bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW (FCCTL0 and FCCTL1 registers)
  - Determine the bit rate by programming the PRES\_DIV field (FCCTL1 register)
  - Determine Internal Arbitration mode (LBUF bit in FCCTL0 register)
- Initialize Message Buffers (MBs):
  - The Control/Status word of *all* MBs *must* be written either as active or inactive MB
  - Other entries in each MB should be initialized as required
- Initialize MASK registers for Acceptance Mask as required
- Initialize FlexCAN's Interrupt Handler
  - Set required BUF $n$ I bits in the FCIMASK1 register (for all MBs interrupts), in the FCCTL0 register for Busoff and Error Interrupts, and in FCMCR for Wake Interrupt
- Clear the HALT bit in the FCMCR
  - Starting with this event, the FlexCAN module attempts to synchronize with the CAN bus

## 7.7 Special Operating Modes

### 7.7.1 Debug Mode

Debug mode is qualified by FRZ1. Assuming FRZ1=1, debug mode is entered when HALT is set. Once this mode is set, the FlexCAN module waits to be in either Intermission, Passive Error, Busoff, or Idle states. When in one of these states, the FlexCAN module waits for all internal activity to complete other than for the CAN bus interface before the following occurs:

- FlexCAN module will stop transmitting/receiving frames
- Prescaler is stopped, thus halting all related activities
- The device is allowed to read *and* write into the Error Counters register

- FlexCAN module ignores its CAN\_RX input pin, driving its CAN\_TX pins as recessive
- FlexCAN loses synchronization with the CAN bus; the NOT\_RDY and FREEZ\_ACK bits in FCMCR are set

After asserting Debug mode configuration bits, wait for FREEZ\_ACK bit to be set in FCMCR before executing any other action to the FlexCAN module.

**Warning:** Failure to wait for FREEZ\_ACK bit to be set in may result in the FlexCAN module operating in an unpredictable manner.

Exiting Debug mode is achieved in one of the following ways:

- Clear HALT bit
- Clear FRZ1 bit

Once exited from Debug mode, the FlexCAN module tries to re-synchronize with the CAN bus by waiting for 11 consecutive recessive bits.

## 7.7.2 Stop Mode for Power Saving

**Note:** There are two different Stop modes discussed in this section. Stop mode described herein as Stop is referring to Internal Stop mode achieved by writing the STOP bit in the FCMCR. LPStop refers to Stop mode activated by the Stop instruction to the core affecting the whole chip.

Stop mode in the FlexCAN module is intended as a power-saving feature. When setting this mode, the FlexCAN module checks for the CAN bus to be either in Busoff or Idle, or it waits for the third bit of intermission, checking it to be recessive. The FlexCAN module waits for all internal activity to complete other than for the CAN bus interface before the following occurs:

- FlexCAN shuts down its clocks, stopping most of the internal circuits, thereby saving maximum power
- The IPBus Interface Logic continues operation, enabling the device to access the FCMCR
- The FlexCAN module ignores its CAN\_RX input pin driving its CAN\_TX pins as recessive
- FlexCAN loses synchronization with the CAN bus, setting the STOP\_ACK and NOT\_RDY bits in FCMCR

Exiting Stop mode is executed in one of the following ways:

- Reset the FlexCAN module either by hard reset or by asserting the SOFT\_RST bit in the FCMCR
- Clear the STOP bit in FCMCR

- If the SELF\_WAKE bit in FCMCR was set at the time the FlexCAN module entered Stop mode, upon detection of recessive to dominant transition on the CAN bus, the FlexCAN module resets the STOP bit in FCMCR, resuming its clocks

When in Stop mode, or in LPStop, a Recessive-to-Dominant transition on the CAN bus causes the WAKE\_INT bit in the Error and Status register to be set. This event can cause a device interrupt if the WAKE\_MASK bit in FCMCR is set.

### 7.7.2.1 Stop Mode Operation Notes

- Upon Stop/Self Wake mode, the FlexCAN module tries to receive the frame responsible for waking it, i.e. it assumes the dominant bit detected is a Start-of-Frame bit. It does *not* arbitrate for the CAN bus.
- Before asserting Stop mode, the device should disable all interrupts in the FlexCAN module. Failure to do so may cause an interruption while in Stop mode upon a non wake-up condition. If desired, the WAKE\_MASK bit should be set, enabling the Wake-Up Interrupt.
- If Stop is asserted while the FlexCAN module is in Busoff, the FlexCAN module moves to Stop. At this point it stops counting the synchronization sequence. It continues this count once Stop is cleared. Please see [Section 7.8.10](#).
- The correct flow to enter Stop with SELF\_WAKE:
  - Assert SELF\_WAKE concurrently with Stop
  - Wait for STOP\_ACK bit to be set
- The correct flow to clear STOP while in SELF\_WAKE:
  - Clear SELF\_WAKE concurrently with STOP
  - Wait for STOP\_ACK negation
- SELF\_WAKE should be set *only* when the STOP bit in the FCMCR is cleared and the FlexCAN module is ready, i.e. the NOT\_RDY bit in the FCMCR is cleared
- If STOP and SELF\_WAKE are set, and if a Recessive-to-Dominant edge immediately follows on the CAN bus, the STOP\_ACK bit in FCMCR may never be set, resetting the FCMCR STOP bit
- Avoid unwanted old frames being sent when the FlexCAN module is awakened (STOP with SELF\_WAKE) by disabling *all* transmit sources before Stop, Remote-Response included
- If Debug mode is active at the time the STOP bit is asserted, the FlexCAN module assumes Debug mode should be exited. Debug mode tries to synchronize to the CAN bus (11 consecutive recessive bits), searching for the correct conditions to Stop
- Trying to stop the FlexCAN module immediately after reset is permitted only after basic initialization is accomplished
- If Stop is activated with SELF\_WAKE, and the FlexCAN module operates with single system clock per time quanta, there are extreme cases FlexCAN's wake-up upon

Recessive-to-Dominant edge may not conform BOSCH CAN protocol in the sense the FlexCAN module synchronization is shifted one time quanta from the required situation. This shift lasts until the next Recessive-to-Dominant edge, re-synchronizing the FlexCAN module back to conform the protocol. The same holds for Auto Power Save mode upon wake-up by Recessive-to-Dominant edge. Please refer to [Section 7.7.3](#).

- If LPStop is executed during a transmission data can be lost, causing nonconformity with BOSCH CAN protocol. Exercise caution in ensuring the FlexCAN module is in an Idle state before putting the device into LPStop mode.

### 7.7.3 Auto Power Save Mode

This mode in the FlexCAN module is intended to enable normal operation while optimizing power savings. Setting the AUTO\_PWR\_SAVE bit in the FCMCR, the FlexCAN module looks for a set of conditions where clocks are not required to be running. If all these conditions are met, the FlexCAN module stops its clocks, thereby saving power. If, while FlexCAN clocks are stopped, any of the conditions mentioned below are no longer true, the FlexCAN module resumes its clocks. FlexCAN continues to monitor the conditions, stopping and resuming its clocks accordingly.

Conditions for automatic clocks shut-off and power saving are:

- No Receive or Transmit Frame in progress
- Receive/Transmit Frames do not move between SMB and MB, or there is no CANTX Frame pending for transmission in any MB
- FlexCAN module core access is unavailable
- the FlexCAN module is neither in:
  - Debug mode (FCMCR bit 8)
  - Stop mode (FCMCR bit 15)
  - Busoff

## 7.8 Register Definitions

**Table 7-7. FlexCAN Memory Map**

Device	Peripheral	Address
8100/8300	FC_BASE	\$00F800
	FC2_BASE	\$00FA00

The address of each FlexCAN register is the sum of a base and offset address. Please refer to the device's data sheet for the definition of the FlexCAN base address. All memory location base and offset addresses are given in hex FLexCAN Register Summary.



Table 7-8. FlexCAN Register Summary

Address Offset	Register Acronym	Register Name	Access Type	Chapter Location
Base + \$0	FCMCR	Module Configuration Register	Read/Write	<a href="#">Section 7.8.1</a>
		Reserved		
		Reserved		
Base + \$3	FCCTL0	Control 0 Register	Read/Write	<a href="#">Section 7.8.2</a>
Base + \$4	FCCTL1	Control 1 Register	Read/Write	<a href="#">Section 7.8.3</a>
Base + \$5	FCTMR	Free-Running Timer Register	Read/Write	<a href="#">Section 7.8.4</a>
Base + \$6	FCMAXMB	Maximum Message Buffer Register	Read/Write	<a href="#">Section 7.8.5</a>
		Reserved		
Base + \$8	FCRnGMASK_H	Receive Data Global Mask High Register	Read/Write	<a href="#">Section 7.8.7</a>
Base + \$9	FCRnGMASK_L	Receive Data Global Mask Low Register	Read/Write	
Base + \$A	FCRn14MASK_H	Receive Data Buffer 14 Mask High Register	Read/Write	<a href="#">Section 7.8.8</a>
Base + \$B	FCRn14MASK_L	Receive Data Buffer 14 Mask Low Register	Read/Write	
Base + \$C	FCRn15MASK_H	Receive Data Buffer 15 Mask High Register	Read/Write	<a href="#">Section 7.8.9</a>
Base + \$D	FCRn15MASK_L	Receive Data Buffer 15 Mask Low Register	Read/Write	
		Reserved		
		Reserved		
Base + \$10	FCSTATUS	Error and Status Register	Read/Write	<a href="#">Section 7.8.10</a>
Base + \$11	FCIMASK1	Interrupt Mask 1 Register	Read/Write	<a href="#">Section 7.8.11</a>
Base + \$12	FCIFLAG1	Interrupt Flag 1 Register	Read/Write	<a href="#">Section 7.8.12</a>
Base + \$13	FC_ERR_CNTRS	Transmission Data/Receive Data Error Counter Register	Read/Write	<a href="#">Section 7.8.13</a>
		Reserved		
Base + \$40	FCMB0_CTL	Message Buffer 0 Control/Status Register	Read/Write	<a href="#">Section 7.5</a>
Base + \$41	FCMB0_ID_H	Message Buffer 0 ID High Register	Read/Write	
Base + \$42	FCMB0_ID_L	Message Buffer 0 ID Low Register	Read/Write	
Base + \$43 - Base + \$46	FCMB0_DATA	Message Buffer 0 Data Registers	Read/Write	
		Reserved		
Base + \$48	FCMB1_CTL	Message Buffer 1 Control/Status Register	Read/Write	<a href="#">Section 7.5</a>
Base + \$49	FCMB1_ID_H	Message Buffer 1 ID High Register	Read/Write	
Base + \$4A	FCMB1_ID_L	Message Buffer 1 ID Low Register	Read/Write	
Base + \$4B - Base + \$4E	FCMB1_DATA	Message Buffer 1 Data Registers	Read/Write	
		Reserved		
Base + \$50	FCMB2_CTL	Message Buffer 2 Control/Status Register	Read/Write	<a href="#">Section 7.5</a>
Base + \$51	FCMB2_ID_H	Message Buffer 2 ID High Register	Read/Write	
Base + \$52	FCMB2_ID_L	Message Buffer 2 ID Low Register	Read/Write	
Base + \$53 - Base + \$56	FCMB2_DATA	Message Buffer 2 Data Registers	Read/Write	
		Reserved		

**Table 7-8. FlexCAN Register Summary (Continued)**

Address Offset	Register Acronym	Register Name	Access Type	Chapter Location
Base + \$58	FCMB3_CTL	Message Buffer 3 Control/Status Register	Read/Write	Section 7.5
Base + \$59	FCMB3_ID_H	Message Buffer 3 ID High Register	Read/Write	
Base + \$5A	FCMB3_ID_L	Message Buffer 3 ID Low Register	Read/Write	
Base + \$5B - Base + \$5E	FCMB3_DATA	Message Buffer 3 Data Registers	Read/Write	
		Reserved		
Base + \$60	FCMB4_CTL	Message Buffer 4 Control/Status	Read/Write	Section 7.5
Base + \$61	FCMB4_ID_H	Message Buffer 4 ID High Register	Read/Write	
Base + \$62	FCMB4_ID_L	Message Buffer 4 ID Low Register	Read/Write	
Base + \$63 - Base + \$66	FCMB4_DATA	Message Buffer 4 Data Registers	Read/Write	
		Reserved		
Base + \$68	FCMB5_CTL	Message Buffer 5 Control/Status	Read/Write	Section 7.5
Base + \$69	FCMB5_ID_H	Message Buffer 5 ID High Register	Read/Write	
Base + \$6A	FCMB5_ID_L	Message Buffer 5 ID Low Register	Read/Write	
Base + \$6B - Base + \$6E	FCMB5_DATA	Message Buffer 5 Data Registers	Read/Write	
		Reserved		
Base + \$70	FCMB6_CTL	Message Buffer 6 Control/Status	Read/Write	Section 7.5
Base + \$71	FCMB6_ID_H	Message Buffer 6 ID High Register	Read/Write	
Base + \$72	FCMB6_ID_L	Message Buffer 6 ID Low Register	Read/Write	
Base + \$73 - Base + \$76	FCMB6_DATA	Message Buffer 6 Data Registers	Read/Write	
		Reserved		
Base + \$78	FCMB7_CTL	Message Buffer 7 Control/Status	Read/Write	Section 7.5
Base + \$79	FCMB7_ID_H	Message Buffer 7 ID High Register	Read/Write	
Base + \$7A	FCMB7_ID_L	Message Buffer 7 ID Low Register	Read/Write	
Base + \$7B - -Base + \$7E	FCMB7_DATA	Message Buffer 7 Data Registers	Read/Write	
		Reserved		
Base + \$80	FCMB8_CTL	Message Buffer 8 Control/Status	Read/Write	Section 7.5
Base + \$81	FCMB8_ID_H	Message Buffer 8 ID High Register	Read/Write	
Base + \$82	FCMB8_ID_L	Message Buffer 8 ID Low Register	Read/Write	
Base + \$83 - Base + \$86	FCMB8_DATA	Message Buffer 8 Data Registers	Read/Write	
		Reserved		
Base + \$88	FCMB9_CTL	Message Buffer 9 Control/Status	Read/Write	Section 7.5
Base + \$89	FCMB9_ID_H	Message Buffer 9 ID High Register	Read/Write	
Base + \$8A	FCMB9_ID_L	Message Buffer 9 ID Low Register	Read/Write	
Base + \$8B - Base + \$8E	FCMB9_DATA	Message Buffer 9 Data Registers	Read/Write	
		Reserved		

**Table 7-8. FlexCAN Register Summary (Continued)**

Address Offset	Register Acronym	Register Name	Access Type	Chapter Location
Base + \$90	FCMB10_CTL	Message Buffer 10 Control/Status	Read/Write	Section 7.5
Base + \$91	FCMB10_ID_H	Message Buffer 10 ID High Register	Read/Write	
Base + \$92	FCMB10_ID_L	Message Buffer 10 ID Low Register	Read/Write	
Base + \$93 - Base + \$96	FCMB10_DATA	Message Buffer 10 Data Registers	Read/Write	
		Reserved		
Base + \$98	FCMB11_CTL	Message Buffer 11 Control/Status	Read/Write	Section 7.5
Base + \$99	FCMB11_ID_H	Message Buffer 11 ID High Register	Read/Write	
Base + \$9A	FCMB11_ID_L	Message Buffer 11 ID Low Register	Read/Write	
Base + \$9B - Base + \$9E	FCMB11_DATA	Message Buffer 11 Data Registers	Read/Write	
		Reserved		
Base + \$A0	FCMB12_CTL	Message Buffer 12 Control/Status	Read/Write	Section 7.5
Base + \$A1	FCMB12_ID_H	Message Buffer 12 ID High Register	Read/Write	
Base + \$A2	FCMB12_ID_L	Message Buffer 12 ID Low Register	Read/Write	
Base + \$A3 - Base + \$A6	FCMB12_DATA	Message Buffer 12 Data Registers	Read/Write	
		Reserved		
Base + \$A8	FCMB13_CTL	Message Buffer 13 Control/Status	Read/Write	Section 7.5
Base + \$A9	FCMB13_ID_H	Message Buffer 13 ID High Register	Read/Write	
Base + \$AA	FCMB13_ID_L	Message Buffer 13 ID Low Register	Read/Write	
Base + \$AB - Base + \$AE	FCMB13_DATA	Message Buffer 13 Data Registers	Read/Write	
		Reserved		
Base + \$B0	FCMB14_CTL	Message Buffer 14 Control/Status	Read/Write	Section 7.5
Base + \$B1	FCMB14_ID_H	Message Buffer 14 ID High Register	Read/Write	
Base + \$B2	FCMB14_ID_L	Message Buffer 14 ID Low Register	Read/Write	
Base + \$B3 - Base + \$B6	FCMB14_DATA	Message Buffer 14 Data Registers	Read/Write	
		Reserved		
Base + \$B8	FCMB15_CTL	Message Buffer 15 Control/Status	Read/Write	Section 7.5
Base + \$B9	FCMB15_ID_H	Message Buffer 15 ID High Register	Read/Write	
Base + \$BA	FCMB15_ID_L	Message Buffer 15 ID Low Register	Read/Write	
Base + \$BB - Base + \$BE	FCMB15_DATA	Message Buffer 15 Data Registers	Read/Write	

Bit fields of each of the 127 registers are illustrated in [Figure 7-6](#). Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$0	FCMCR	R	STOP	FRZ1	0	HALT	NOT_RDY	WAKE_MASK	SOFT_RST	FREEZ_ACK	0	SELF_WAKE	AUTO_PWR_SAVE	STOP_ACK	0	0	0	0
		W																
RESERVED																		
RESERVED																		
\$3	FCCTL0	R	BOFF_MASK	ERR_MASK	0	0	0	0	0	0		SAMP	LOOPB	TSYNC	LBUF	LOM	PROPSEG	
		W																
\$4	FCCTL1	R	PRES_DIV								RJW		PSEG1		PSEG2			
		W																
\$5	FCTMR	R	TMR															
		W																
\$6	FCMAXMB	R	0	0	0	0	0	0	0	0	0	0	0	0	MAXMB			
		W									TEST_EN							
RESERVED																		
\$8	FCRnGMASK_H	R	MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
		W																
\$9	FCRnGMASK_L	R	MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
		W																
\$A	FCRn14MASK_H	R	MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
		W																
\$B	FCRn14MASK_L	R	MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
		W																
\$C	FCRn15MASK_H	R	MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
		W																
\$D	FCRn15MASK_L	R	MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
		W																
RESERVED																		
RESERVED																		
\$10	FCSTATUS	R	BIT1_ERR	BIT0_ERR	ACK_ERR	CRC_ERR	FORM_ERR	STUFF_ERR	TX_WARN	RX_WARN	IDLE	TX/RX	FCS		0	BOFF_INT	ERR_INT	WAKE_INT
		W																
\$11	FCIMASK1	R	BUF_15M	BUF_14M	BUF_13M	BUF_12M	BUF_11M	BUF_10M	BUF_9M	BUF_8M	BUF_7M	BUF_6M	BUF_5M	BUF_4M	BUF_3M	BUF_2M	BUF_1M	BUF_0M
		W																
\$12	FCIFLAG1	R	BUF_15I	BUF_14I	BUF_13I	BUF_12I	BUF_11I	BUF_10I	BUF_9I	BUF_8I	BUF_7I	BUF_6I	BUF_5I	BUF_4I	BUF_3I	BUF_2I	BUF_1I	BUF_0I
		W																
\$13	FC_ERR_CNTRS	R	CANRX_ERR_CNTR								CANTX_ERR_CNTR							
		W																
RESERVED																		
RESERVED																		

R 0 Read as 0  
W Reserved

Figure 7-5. FlexCan Register Map Summary

## 7.8.1 Module Configuration Register (FCMCR)

Base + \$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Read</b>	STOP	FRZ1	0	HALT	NOT_RDY	WAKE_MASK	SOFT_RST	FREEZ_ACK	0	SELF_WAKE	AUTO_PWR_SAVE	STOP_ACK	0	0	0	0
<b>Write</b>																
<b>Reset</b>	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0

**Figure 7-6. Module Configuration Register (FCMCR)**

### 7.8.1.1 Stop (STOP)—Bit 15

This bit is a Low Power Sleep mode. It may be set by the device. The bit can be cleared either by the device or by FlexCAN only if the SELF\_WAKE bit in FCMCR is set.

- 0 = Enable FlexCAN clocks to run
- 1 = Shutdown FlexCAN clocks

Please see [Section 7.7.2](#).

### 7.8.1.2 FREEZE Enable (FRZ1)—Bit 14

The FRZ1 bit specifies the FlexCAN module response to asserting the HALT bit in the FCMCR. This bit is initialized to 1 during reset. Clearing this bit causes the FlexCAN to exit debug mode. For a detailed description of debug mode, please refer to [Section 7.7.1](#).

- 0 = Ignore HALT bit in FCMCR. FlexCAN operates normally.
- 1 = FlexCAN module can enter debug mode

When FREZ = 1, it enables the FlexCAN module to enter FRZ1/HALT mode. In order to enter this mode, FRZ1 bit should be set to 1, and FRZ1/HALT bit in FCMCR should be set. Clearing this bit field causes FlexCAN to exit from FRZ1/HALT mode. For detailed description of FRZ1/HALT mode, please refer to [Section 7.7.1](#).

### 7.8.1.3 Reserved—Bit 13

This bit is reserved or not implemented. It is initialized as 0 and should not be modified by writing. (In other implementations of FlexCAN this is bit FRZ0.)

#### 7.8.1.4 Halt FlexCAN SClock (HALT)—Bit 12

When HALT and FRZ1 are set the FlexCAN enters debug mode. This bit is initialized to 1 during reset. The HALT bit should be cleared after initializing the MBs and control registers. No receive or transmit operations occur until this bit is cleared.

When HALT is asserted the Error Counters register is write able. Please refer to Section 7.7.1.

- 0 = FlexCAN operates normally
- 1 = Enter Debug mode if FRZ1 = 1

#### 7.8.1.5 FlexCAN Not Ready (NOT\_RDY)—Bit 11

This *read-only* bit indicates FlexCAN is either in Stop or in Debug states.

This bit is cleared once the FlexCAN module has exited Debug mode either by synchronization to the bus (11 recessive bits), or by a self-wake mechanism.

#### 7.8.1.6 Wake-Up Interrupt Mask (WAKE\_MASK)—Bit 10

This bit enables the Wake-Up interrupt generation.

- 0 = Wake-Up Interrupt is disabled
- 1 = Wake-Up Interrupt is enabled

#### 7.8.1.7 Soft Reset (SOFT\_RST)—Bit 9

When SOFT\_RST is asserted FlexCAN resets its Internal State Machines (Sequencer, Error Counters, Error Flags, Timer) and the interface registers (FCMCR, FCIMASK1, FCIFLAG1, FCMAXMB).

Configuration bits controlling the interface with the CAN Bus (FCCTL0 and FCCTL1) the MBs, and the Receive Message Masks are not altered. This enables the device to use the SOFT\_RST as a debug feature during run-time of the system.

SOFT\_RST also affects the FCMCR, causing the STOP bit in the FCMCR to reset. FlexCAN then resumes its clocks after it is in Stop Low-Power mode. The SOFT\_RST bit asserts the HALT/FRZ1 bits, causing the FlexCAN module to enter Debug mode. This bit is self-clearing.

**Note:** After setting the SOFT\_RST bit, the next device access should *not* be to the FlexCAN module to allow full reset of internal FlexCAN circuitry.

#### 7.8.1.8 FlexCAN Disabled (FREEZ\_ACK)—Bit 8

This *read-only* bit provides status of when FlexCAN is in Debug mode, stopping its prescaler.

The value of this bit is:

- 0 = FlexCAN exited Debug mode and the prescaler is enabled
- 1 = FlexCAN entered Debug mode and the prescaler is disabled

The FREEZ\_ACK bit is set when the FlexCAN module enters Debug mode. The device can poll this bit to determine if the FlexCAN module entered Debug mode. This bit is cleared when FlexCAN exits Debug. It is also cleared when the prescaler is running. Please see [Section 7.8.1.5](#).

#### 7.8.1.9 Reserved—Bit 7

This bit is reserved or not implemented. It is read and written as 0.

#### 7.8.1.10 Self Wake-Up (SELF\_WAKE)—Bit 6

This bit enables the Self Wake-Up of FlexCAN after Stop, without device intervention. If this bit is set when entering Stop, the FlexCAN module will be looking for a dominant bit on the bus during Stop. If a transition from Recessive-to-Dominant is detected, the FlexCAN module immediately clears the STOP bit, resuming the clocks.

If a write to FCMCR with SELF\_WAKE set occurs at the same time a Recessive-to-Dominant edge appears on the CAN bus, the bit will not be set and the module clocks will not stop. Verify this bit was set by reading FCMCR.

**Note:** This bit should *not* be set if eventually the LPStop command is executed. For more details, please refer to [Section 7.7.2.1](#).

#### 7.8.1.11 Auto Power Save (AUTO\_PWR\_SAVE)—Bit 5

This bit enables FlexCAN to automatically shut-off its clocks, saving power when it has no process to execute. Those same clocks automatically resume when it has a task to execute, without any device intervention.

**Note:** The IPBus clocks are not stopped, enabling device access. The Auto Power Save action does not depend on the SELF\_WAKE (Bit 6) or WAKE\_MASK (Bit 10) bit values.

- 0 = Auto Power Save mode is not active; clocks are running normally
- 1 = Auto Power Save mode is active; clocks stop and resume as required

#### 7.8.1.12 FlexCAN Stopped (STOP\_ACK)—Bit 4

FlexCAN is in Stop mode with its main clocks stop.

This is a *read-only* bit. This bit has a value of one when the FlexCAN module enters Stop mode and its clocks are stopped, but a value of zero when Stop mode is cleared and the FlexCAN module's clocks are running again. Please see [Section 7.7.2](#).

When FlexCAN enters Stop mode and stops its clocks, it sets the STOP\_ACK bit. The device can poll this bit to know if FlexCAN entered Stop mode and stopped its clocks. If Stop mode bit is cleared, this bit is off once the FlexCAN module's clocks are running.

- 0 = FlexCAN exited Stop mode
- 1 = FlexCAN entered Stop mode

### 7.8.1.13 Reserved—Bits 3–0

This bit field is reserved or not implemented. The field is read as 0 and cannot be modified by writing.

## 7.8.2 Control Register 0 (FCCTL0)

These registers (FCCTOL0, FCCTL1, and FCTIMER) provide control means related to the CAN bus, such as bit rate, programmable sampling point within an CANRX bit and a global Free-Running Timer.

Base + \$3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	BOFF_MASK	ERR_MASK	0	0	0	0	0	0	SAMP	LOOPB	TSYNC	LBUF	LOM	PROPSEG		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-7. Control Register 0 (FCCTL0)

### 7.8.2.1 Busoff Mask (BOFF\_MASK)—Bit 15

This bit provides a mask for the Busoff Interrupt.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

### 7.8.2.2 Error Mask (ERR\_MASK)—Bit 14

This bit provides a mask for the Error Interrupt.

- 0 = Interrupt disabled
- 1 = Interrupt enabled

### 7.8.2.3 Reserved—Bits 13–8

This bit field is reserved or not implemented. These are read/write bits using 0s only. *Attempting to change the values of these bits can cause unintended functionality changes.*



**Warning:** Exercise care to only write 0s to these bits when accessing this register.

#### 7.8.2.4 Sampling Mode (SAMP)—Bit 7

The Sample bit determines whether the FlexCAN module will sample each received bit once or three times to determine its value.

- 0 = (Reset Value) One sample is used to determine the value of received bit.
- 1 = Three samples are used to determine the value of received bit, the normal one (sample point), and two preceding periods of SClk, using a majority rule.

#### 7.8.2.5 Loop Back Mode (LOOPB)—Bit 6

This bit can only be set when in Test mode, but only when the TEST\_EN bit in the FCMAXMB register is set. Please see [Section 7.8.5.2](#). If this bit is set, the internal Loop Back is enabled, preventing the module from seeing bus activity.

- 0 = Loop Back mode is disabled
- 1 = Loop Back mode is enabled

#### 7.8.2.6 Timer Synchronization Mode (TSYNC)—Bit 5

This bit enables a mechanism to reset or clear the Free-Running Timer each time a message is received in MB0. This feature provides means to synchronize multiple FlexCAN stations with a special SYNC message, i.e. Global Network Time. Buffer0 interrupt is also available.

- 0 = Timer Sync is disabled
- 1 = Timer Sync is enabled

**Note:** There is a possibility of four to five tick count skew between the different FlexCAN stations operating in this mode.

#### 7.8.2.7 Lowest Buffer Transmitted First (LBUF)—Bit 4

This bit defines the transmit-first scheme.

- 0 = Lowest ID is transmitted first
- 1 = Lowest number buffer is transmitted first

**Note:** LBUF = 0: If there are multiple MBs with the same ID, the Data Frame(s) is sent before the Remote Frame(s). Within Data and Remote Frames with the same ID, the lowest number buffer is transmitted first.

### 7.8.2.8 Listen-Only Mode (LOM)—Bit 3

This control bit configures FlexCAN to be in Listen-Only mode. In Listen-Only mode the FlexCAN module is able to receive messages without acknowledging or being active on the bus for diagnostics. For more details, please refer to [Section 7.6.8](#).

- 0 = Regular operation Listen-Only mode is off
- 1 = Enable Listen-Only mode

### 7.8.2.9 Propagation Segment (PROPSEG)—Bits 2–0

This bit field defines the length of the Propagation Segment in the bit time. The valid programmed values are zero through seven.

Propagation Segment Time = (PROPSEG + 1) × Time Quanta

1 Time Quantum = 1 SClock Period (please see [Section 7.8.3](#))

## 7.8.3 Control Register 1 (FCCTL1)

Base + \$4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PRES_DIV								RJW		PSEG1			PSEG2		
Write	PRES_DIV								RJW		PSEG1			PSEG2		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 7-8. Control Register 1 (FCCTL1)**

### 7.8.3.1 Prescaler Divide Factor (PRES\_DIV)—Bits 15–8

This bit field determines the ratio between the system clock frequency and the Serial Clock (Sclock); (1 Sclock = 1 Time Quantum).

The Sclock is equal to the system clock divided by the value of this register, plus one. Reset value of this register is zero, meaning the Sclock is the same as the system clock frequency.

The maximum value of this 8-bit field is \$FF, giving the minimum Sclock frequency = system clock/256. Please see [Section 7.6.9](#) for additional information.

### 7.8.3.2 Resynchronization Jump Width (RJW)—Bits 7–6

This bit field defines the maximum number of time quanta a bit may be changed by one resynchronization. Valid programmed values are zero through three.

Resync Jump Width = (RJW + 1) × Time Quanta

### 7.8.3.3 Phase Segment 1 (PSEG1)—Bits 5–3

This bit field defines the length of Phase Buffer Segment 1 in the bit time. The valid programmed values are zero through seven.

Phase Buffer Segment 1 = (PSEG1 + 1) × Time Quanta

### 7.8.3.4 Phase Segment 2 (PSEG2)—Bits 2–0

This bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmed values are zero through seven.

Phase Buffer Segment 2 = (PSEG2 + 1) × Time Quanta

## 7.8.4 Free-Running Timer (FCTMR)

Base + \$5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TMR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-9. Free-Running Timer Register (FCTMR)

### 7.8.4.1 Free-Running Timer (TMR)—Bits 15–0

This 16-bit field Free-Running counter is read/written by the device. The timer begins from zero after Reset, counting linearly to \$FFFF, then wraps around.

This timer is clocked by the FlexCAN module bit clock. During a message it increments by one for each bit received or transmitted. When there is no message on the bus, it counts at the nominal bit rate.

The timer value is captured at the beginning of the Identifier field of any frame on the CAN bus. This captured value is written into the TIME\_STAMP entry in an MB after a successful reception/transmission of a message.

## 7.8.5 Maximum Message Buffer Register (FCMAXMB)

Base + \$6	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	MAXMB			
Write									TEST_EN							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

**Figure 7-10. Maximum Message Buffer Register (FCMAXMB)**

**Warning:** Changing values of bits 15–8, 6–4 can result in unwanted functionality.

### 7.8.5.1 Reserved—Bits 15–8

This bit field is reserved or not implemented. Bits are initialized as 0 and should be written as 0 *only*.

### 7.8.5.2 Test Enable (TEST\_EN)—Bit 7

This *write-only* bit is set prior to setting the LOOPB bit in the FCCTL0 register. Please see [Section 7.8.2.5](#) for additional information about Loop Back mode.

- 0 = Test mode is disabled
- 1 = Test mode is enabled

### 7.8.5.3 Reserved—Bits 6–4

This bit field is reserved or not implemented. Bits are initialized as 0 and should be written as 0 *only*.

### 7.8.5.4 Maximum Message Buffer (MAXMB)—Bits 3–0

This bit field defines the maximum MB configuration of the module. The reset value of these bits is \$F for a 16-MB configuration.

Maximum MBs in use = MAXMB + 1

## 7.8.6 Receive Mask Registers

These registers are used as Acceptance Masks for received frame IDs. Two Global Masks, delineated as High and Low, are used for Receive Buffers 0 through 13. Four additional masks for buffers 14 and 15 are delineated as High and Low.

- 0 = Mask bit: The corresponding incoming ID bit is *don't care*

- 1 = Mask bit: The corresponding ID bit is checked against the incoming ID bit, determining if a match exists.

**Note:** 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 matching an MB through a mask may be transferred into the MB (upon release), but may no longer match.

**Table 7-9. Mask Examples for Normal/Extended Messages**

		Base ID ID28.....ID18	IDE	Extended ID ID17.....ID0	Match	Note
FlexCAN Configuration	MB2-ID	11111111000	0	—	—	—
	MB3-ID	11111111000	1	010101010101010101	—	—
	MB4-ID	00000011111	0	—	—	—
	MB5-ID	00000011101	1	010101010101010101	—	—
	MB14-ID	11111111000	1	010101010101010101	—	—
	CANRX_Global_Mask	11111111110	—	111111100000000001	—	—
	CANRX_14_Mask	01111111111	—	111111100000000000	—	—
Received Message ID	CANRXMsg in	11111111001	1	010101010101010101	3	(1)
	CANRXMsg in	11111111001	0	—	2	(2)
	CANRXMsg in	11111111001	1	010101010101010100	—	(3)
	CANRXMsg in	01111111000	0	—	—	(4)
	CANRXMsg in	01111111000	1	010101010101010101	14	(5)
	CANRXMsg in	10111111000	1	010101010101010101	—	(6)
	CANRXMsg in	01111111000	1	010101010101010101	14	(7)

Notes:

- (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
- (6) Mismatch for MB14 because of ID27
- (7) Match for MB14

## 7.8.7 Receive Global Mask (FCRXGMASK\_H and FCRXGMASK\_L)

Receive Global Mask registers are composed of four bytes. The mask bits are applied to all receive identifiers, excluding MBs14 and 15, having specific Receive Mask registers.

Base + \$8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read	MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

**Figure 7-11. Receive Global Mask High Register (FCRXGMASK\_H)**

Base + \$9	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

**Figure 7-12. Receive Global Mask Low Register (FCRXGMASK\_L)**

### 7.8.7.1 Mask Identification 28 to 18 (MID28 through MID18)—Bits 31–21

These bits are the same mask bits for the Standard and Extended formats.

### 7.8.7.2 Reserved—Bit 20

This bit of a received frame is never compared to the corresponding bit in the MB ID field.

**Note:** Remote frames (RTR = 1) are never received into MBs. RTR mask bits locations in the mask (bits 20 and zero) are always read as 0 regardless of any device write to these bits.

### 7.8.7.3 Reserved—Bit 19

The Identification Extended (IDE) bit of a Received Frame is always compared. Its location in the mask is always 1 regardless of any device write to this bit.

### 7.8.7.4 Mask Identification 17 to 1 (MID17 through MID0S)—Bits 18–1

These bits are used to mask comparison only in Extended format.

### 7.8.7.5 Reserved—Bit 0

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

## 7.8.8 Receive Buffer 14 Mask Registers (FCRX14MASK\_H / \_L)

The CANRX Buffer14 Mask register has the same structure as the CANRX Global Mask register. It is used to mask Buffer14.

Base + \$0A	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read	MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

Figure 7-13. Receive Buffer 14 Mask High Register (FCRX14MASK\_H)

Base + \$0B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Figure 7-14. Receive Buffer 14 Mask Low Register (FCRX14MASK\_L)

## 7.8.9 Receive Buffer 15 Mask Registers (FCRX15MASK\_H / \_L)

The CANRX Buffer15 Mask register has the same structure as the Receive Global Mask register. It is used to mask Buffer15.

Base + \$0C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read	MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

Figure 7-15. Receive Buffer 15 Mask High Register (FCRX15MASK\_H)

Base + \$0D	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Figure 7-16. Receive Buffer 15 Mask Low Register (FCRX15MASK\_L)

## 7.8.10 Error and Status Register (FCSTATUS)

This register reflects various error conditions; some general status of the device. It is the source of three interrupts to the core. The reported error conditions, bits 15 through 10, are those occurring since the last time the core read this register. Reading this register clears these bits to zero.

All bits in this register are *read-only*, except BOFF\_INT, ERR\_INT, and WAKE\_INT. These bits are interrupt sources and can be written by the core as zero. See [Section 7.9](#) for additional information. Bits nine through three are status bits.

Base + \$10	15	14	13	12	11	10	9	8
Read	BIT1_ERR	BIT0_ERR	ACK_ERR	CRC_ERR	FORM_ERR	STUFF_ERR	TX_WARN	RX_WARN
Write								
Reset	0	0	0	0	0	0	0	0

Base + \$10	7	6	5	4	3	2	1	0
Read	IDLE	TX/RX	FCS		0	BOFF_INT	ERR_INT	WAKE_INT
Write								
Reset	0	0	0	0	0	0	0	0

**Figure 7-17. Error and Status Register (FCSTATUS)**

### 7.8.10.1 Bit 1 Error (BIT1\_ERR)—Bit 15

- 0 = No errors seen
- 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 Acknowledge (ACK) slot, or in case of a node sending a passive edge flag detects dominant bits.

### 7.8.10.2 Bit 0 Error (BIT0\_ERR)—Bit 14

- 0 = No errors seen
- 1 = At least one bit sent as dominant is received as recessive

### 7.8.10.3 Acknowledge Error (ACK\_ERR)—Bit 13

- 0 = No occurrence



- 1 = An ACK error occurred since the last read of this register

#### 7.8.10.4 Cyclic Redundancy Code Error (CRC\_ERR)—Bit 12

- 0 = No occurrence
- 1 = A CRC error occurred since the last read for this register

#### 7.8.10.5 Form Error (FORM\_ERR)—Bit 11

- 0 = No occurrence
- 1 = A FORM error occurred since the last read for this register

#### 7.8.10.6 Stuff Bit Error (STUFF\_ERR)—Bit 10

- 0 = No occurrence
- 1 = A STUFF bit error occurred since the last read for this register

#### 7.8.10.7 Transmit Warning (TX\_WARN)—Bit 9

This status flag does not cause an interrupt.

- 0 = CANTX\_ERR\_Counter < 96
- 1 = CANTX\_ERR\_Counter ≥ 96

#### 7.8.10.8 Receive Warning (RX\_WARN)—Bit 8

This status flag does not cause an interrupt.

- 0 = CANRX\_ERR\_Counter < 96
- 1 = CANRX\_ERR\_Counter ≥ 96

#### 7.8.10.9 Idle (IDLE)—Bit 7

- 0 = The CAN bus is not idle. See TX/ $\overline{\text{RX}}$  bit
- 1 = The CAN bus is idle

#### 7.8.10.10 Transmission Receive (TX/ $\overline{\text{RX}}$ )—Bit 6

- 0 = This FlexCAN is receiving a message if IDLE = 0
- 1 = This FlexCAN is transmitting a message if IDLE = 0

### 7.8.10.11 Fault Confinement State (FCS)—Bits 5–4

This two-bit field describes the state of the device.

- 00 = Error Active
- 01 = Error Passive
- 1X = Busoff

If SOFT\_RST bit in FCMCR is asserted while the FlexCAN module is in Busoff state, the Error and Status register is then reset (including the FCS bits), but when exiting disabled state, the FCS bits return to reflect the Busoff state.

Update of the Fault Confinement State from Error Passive to Busoff occurs approximately one bit time after detection of the error that generates the Busoff state transition.

### 7.8.10.12 Reserved—Bit 3

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 7.8.10.13 Busoff Interrupt (BOFF\_INT)—Bit 2

This bit is set each time the FlexCAN module state changes to Busoff. If the FCCTL0 BOFF\_MASK bit is set, an interrupt is generated to the core. Please see [Section 7.8.2.1](#). This interrupt is *not* generated after the reset. This bit must be cleared to 0. To clear this bit, read the Error and Status register, then write the bit as 1. Writing 0 has no effect.

Assertion of the Busoff interrupt occurs approximately one bit time after detection of the bit error that generates the Busoff state transition.

### 7.8.10.14 Error Interrupt (ERR\_INT)—Bit 1

This bit is set anytime one of the error bits in this register is set. This is true even if an error bit is already set. If the FCCTL0 ERR\_MASK bit is set, an interrupt is generated to the core. Please refer to [Section 7.8.2.2](#). This bit must be cleared to 0. To clear this bit, read the Error and Status register, then write the bit as 1. Writing 0 has no effect.

### 7.8.10.15 Wake Interrupt (WAKE\_INT)—Bit 0

This bit is set when the FlexCAN module is in Stop mode, and a Recessive-to-Dominant transition is detected on the CAN bus. If the FCMCR WAKE\_MASK bit is set, an interrupt is generated to the core. Please see [Section 7.8.1](#). To clear this bit, read the Error and Status register, then write the bit as 1. Writing 0 has no effect.

## 7.8.11 Interrupt Mask Register 1 (FCIMASK1)

This register contains one interrupt mask bit per MB. It enables FlexCAN to determine which buffer will generate an interrupt after a successful transmission/reception when the corresponding FCIFLAG1 bit is set.

Base + \$11	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Read</b>	BUF15 M	BUF14 M	BUF13 M	BUF12 M	BUF11 M	BUF10 M	BUF9 M	BUF8 M	BUF7 M	BUF6 M	BUF5 M	BUF4 M	BUF3 M	BUF2 M	BUF1 M	BUF0 M
<b>Write</b>	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
<b>Reset</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-18. Interrupt Mask Register 1 (FCIMASK1)

### 7.8.11.1 Buffer Interrupt Masks (BUF15M – BUF0M)—Bits 15–0

- 0 = The corresponding buffer interrupt is disabled
- 1 = The corresponding buffer interrupt is enabled

**Note:** Setting or clearing a bit in the FCIMASK1 register can assert, or clear an interrupt request, respectively.

## 7.8.12 Interrupt Flag Register 1 (FCIFLAG1)

The Interrupt Flag register contains one interrupt flag bit per MB. Each successful transmission/reception sets the corresponding BUF $n$ I bit and, if the corresponding FCIMASK1 bit is set, will generate an interrupt.

Base + \$12	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Read</b>	BUF 15I	BUF 14I	BUF 13I	BUF 12I	BUF 11I	BUF 10I	BUF9I	BUF8I	BUF7I	BUF6I	BUF5I	BUF4I	BUF3I	BUF2I	BUF1I	BUF0I
<b>Write</b>																
<b>Reset</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7-19. Interrupt Flag Register 1 (FCIFLAG1)

### 7.8.12.1 Buffer Interrupt (BUF15I - BUF0I)—Bits 15–0

- 0 = The corresponding buffer did not successfully complete transmission or reception.
- 1 = The corresponding buffer successfully completed transmission or reception.

Interrupt flag must be cleared to zero by first reading the Interrupt Flag Register 1, then write 1 to this bit. Writing 0 has no effect. This register is initialized to all zeros on reset.

### 7.8.13 Error Counters (FC\_ERR\_CNTRS)

There are two error counters in the FlexCAN module:

1. Transmit error counter
2. Receive error counter

Base + \$13	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CANRX_ERR_CNTR								CANTX_ERR_CNTR							
Write	CANRX_ERR_CNTR								CANTX_ERR_CNTR							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 7-20. Error Counters (FC\_ERR\_CNTRS) Register**

Rules for increasing and decreasing these counters are described by the CAN protocol. These rules are fully implemented in the FlexCAN module.

Each counter is comprised of:

- 8-bit up/down counter
- Increment by eight (CANRX\_ERR\_CNTR also by one)
- Decrement by one
- Avoid decrement when equal to zero
- CANRX\_ERR\_CNTR preset to a value  $119 \leq x \leq 127$
- Value after reset = 0
- Detect values for error passive, busoff and error active transitions and to alert the core
- Cascade usage of CANRX\_ERR\_CNTR with an internal other counter to detect the 128 occurrences of 11 consecutive recessive bits to determine move from Busoff into Error Active

**Note:** Both counters are *read-only*, except for Freeze/Halt modes.

The FlexCAN module responds to any bus state as described in the protocol, for example Transmit Error Active or Error Passive Flag, delay its transmission start time (Error Passive) and avoid any influence on the bus when in Busoff state. The following are the basic rules for FlexCAN bus state transitions:

- If the value of CANTX\_ERR\_CNTR or CANRX\_ERR\_CNTR increases to be greater than, or equal to 128, the FCS field in the Error and Status register is updated to reflect it (set Error Passive state).
- If the FlexCAN module state is Error Passive and either CANTX\_ERR\_CNTR or CANRX\_ERR\_CNTR decreases to a value less than, or equal to 127 while the other already satisfies this condition, the FCS field in the Error and Status register is updated to reflect it (set Error Active state).
- If the value of the CANTX\_ERR\_CNTR increases to be greater than 255, the FCS field in the Error and Status register is updated to reflect it (set Busoff state) and an interrupt may be issued. The value of CANTX\_ERR\_CNTR is then reset to zero. Resetting of the CANTX\_ERR\_CNTR to zero occurs approximately one bit time after detection of the error that generates the Busoff state transition.
- If the FlexCAN module state is Busoff, CANTX\_ERR\_CNTR, together with an internal counter, are cascaded to count the 128 occurrences of 11 consecutive recessive bits on the bus. Hence, CANTX\_ERR\_CNTR is reset to zero and counts in a manner where the internal counter computes 11 such bits. It then wraps around while incrementing the CANTX\_ERR\_CNTR. When CANTX\_ERR\_CNTR reaches the value of 128, FCS 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, but does *not* affect the CANTX\_ERR\_CNTR.
- If, during system startup, only one node is operating, its CANTX\_ERR\_CNTR increases each message it is trying to transmit as a result of ACK\_ERR. A transition to bus state Error Passive should be executed as described, while this device never enters the Busoff state.
- If the CANRX\_ERR\_CNTR increases to a value greater than 127 it is not increased any more 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 enable resume to Error Active state.

## 7.9 Interrupts

The module can generate up to 19 interrupt sources, 16 interrupts due to MBs and three interrupts due to Busoff, Error and Wake-up. Each one of the MBs can be an interrupt source, if its corresponding IMASK bit is set.

There is no distinction between receive and transmit interrupts for a particular buffer with the assumption the buffer is initialized for either transmission or reception. Therefore, its interrupt routine can be fixed at compilation time. Each of the buffers is assigned a bit in the FCIFLAG1 register. The bit is set when the corresponding buffer completes a successful transmission or reception, and cleared when the device reads the Interrupt Flag register (FCIFLAG1) while the

associated bit is set. Once the associated bit is set, the bit writes it back as *one* with no new event of the same type occurring between the read and the write actions.

A combined interrupt for all 16-MBs is generated by combining all the interrupt sources from MBs. This interrupt gets generated when any of the 16-MB interrupt sources generates a interrupt. The device must read the FCIFLAG1 register to determine which MB caused the interrupt. See the device's data sheet for information on the location of these interrupts within the interrupt vector table.

The other three interrupt sources:

- Busoff
- Error
- Wake Up

Each act in the same way. They are located in the Error and Status register. The BOFF\_MASK and ERR\_MASK bits are located in the FCCTL0 register, while the WAKE\_INT MASK bit is located in the FCMCR.

## 7.10 Resets

FlexCAN module may be reset in two ways:

1. Hard reset using system reset line
2. Assert the SOFT\_RST bit in the FCMCR

Following the negation of either reset, the FlexCAN module is not synchronized with the CAN bus, the HALT and FRZ1 bits in FCMCR are set. Its main control is disabled and FREEZ\_ACK and NOT\_RDY bits in the FCMCR are set. The FlexCAN module CANTX pin is in recessive state and it does not initiate frame transmission nor receives any frames from CAN bus. The MBs contents are *not* changed following SOFT\_RST or hard reset.

It is required for any configuration change/initialization the FlexCAN module either be *frozen* by asserting the HALT bit in FCMCR or be reset. Please see [Section 7.7.1](#).

## 7.11 Interaction of FlexCAN Message Buffers and CodeWarrior Debugger

FlexCAN MBs are mapped into locations in data RAM and are visible in the CodeWarrior debugger via memory windows. However, note interaction of CodeWarrior and MBs can produce unpredictable results when the FlexCAN module is active and one or more CodeWarrior memory windows are open. The act by CodeWarrior to query the contents of these MBs can change the behavior of the FlexCAN module.

**Note:** *Do not open memory windows on top of MBs unless the FlexCAN module is disabled.*

## 7.12 Use of Message Buffers as Data RAM

If the FlexCAN module is not in use, the first seven words of every MB can be used as normal data RAM.

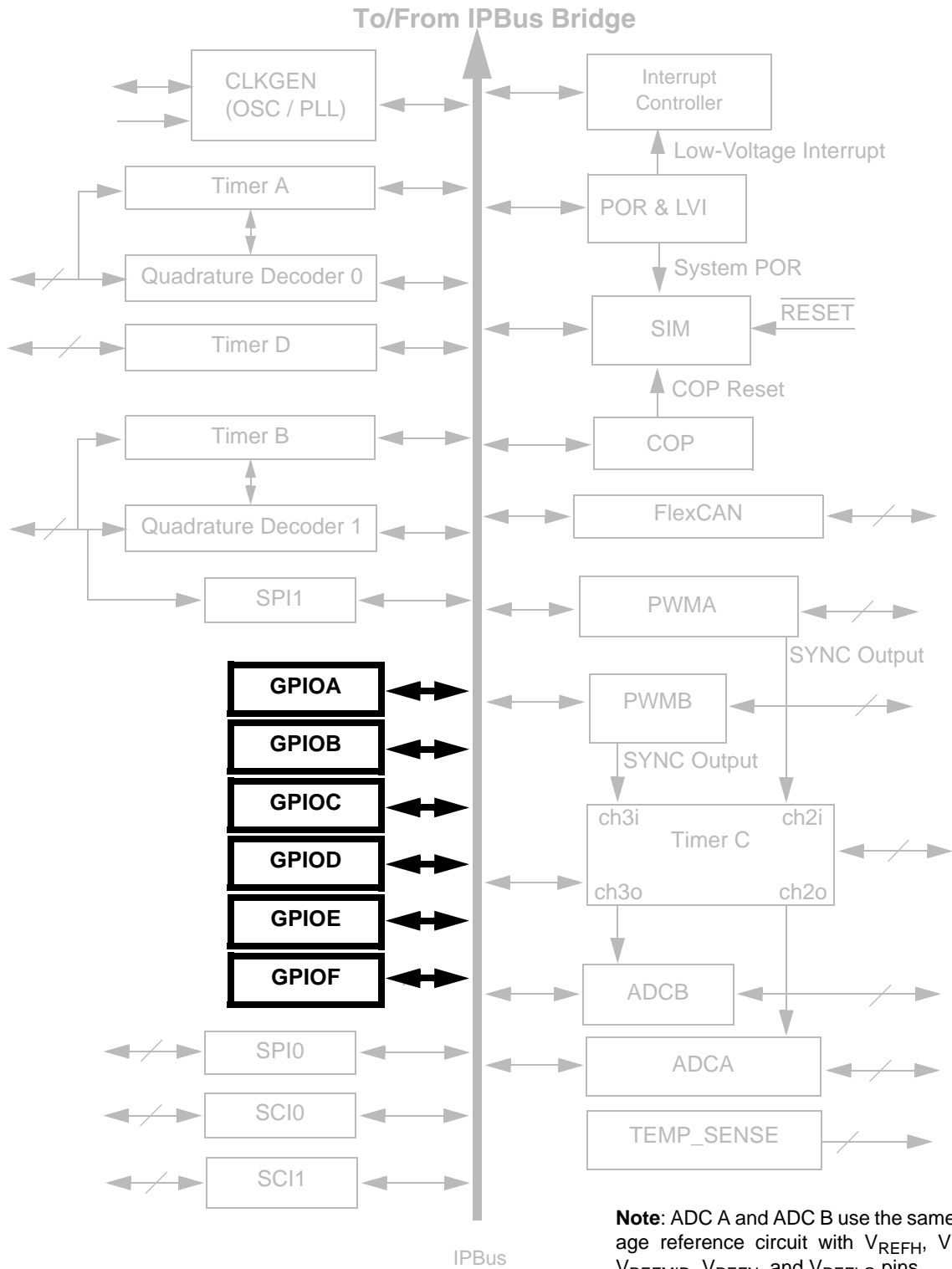
**Note:** The eighth word (offset \$7) is reserved and cannot be used.





# Chapter 8

## General Purpose Input/Output (GPIO)



## Document Revision History for Chapter 8, General Purpose Input/Output (GPIO)

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 3.0	Clarified wording in Table 8-3, second to last row, for Data Access result for DR value Corrected reset value in Figure 8-12
Rev 5.0	Corrected reversed Push/Pull, Section 8.6.10, page 12 to: 0 = Open Drain Mode; 1 = Push/Pull Mode Corrected Figures 8-2 (pg 8) and 8-10 (pg 12) to reflect IPR is 8 bits instead of 16 Converted chapter to Freescale design standards
Rev 6.0	Clarified Section 8.6.8 and 8.6.9. Corrected level sensitive to edge sensitive in section 8.8.
Rev 7.0	Updated PPMODE writeup on page 3 and 4. Clarification to GPIO configuration in Section 8.1.
Rev 8.0	Corrected document revision history comments and revision numbers.

## 8.1 Introduction

The General Purpose Input/Output (GPIO) module allows direct read or write access to pin values, or the ability to assign a pin to be used as an external interrupt. GPIO pins may be dedicated for GPIO use only. They can also be multiplexed with other peripherals on the chip as well. The *Device Data Sheet* specifies the assigned GPIO ports and its multiplexed pin package.

A GPIO pin may be configured in three ways:

1. As GPIO input with, or without, pull-up
2. As GPIO output with Push-Pull mode or open drain mode
3. As a peripheral pin when multiplexed with another module

If a GPIO pin is multiplexed with a peripheral, then it can also be configured to be used by the peripheral.

GPIOs are placed on the chip in groups of one to 16 bits, called ports and designated as A, B, C, etc. Please refer to the Data Sheet for the specific definition of each of the GPIO ports on the chip. For purposes of illustration, a port width of eight bits is assumed throughout this chapter.

## 8.2 Features

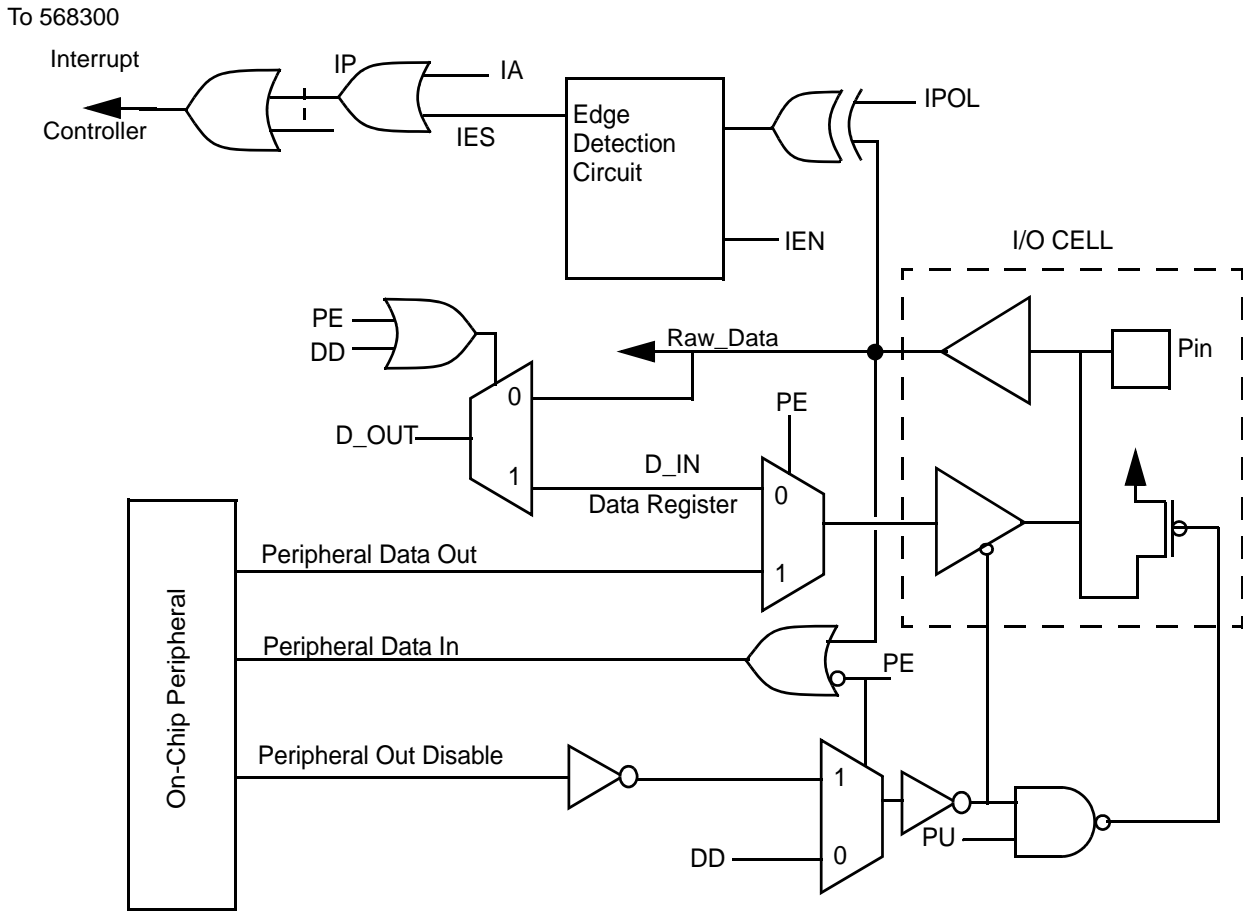
The GPIO module design includes these characteristics:

- Individual control for each pin to be in either Normal or GPIO mode
- Individual direction control for each pin in GPIO mode
- Individual pull-up enable control for each pin in either Normal or GPIO mode
- Optimized for use with a keypad interface with push-pull I/O
- Ability to monitor pin logic values even when GPIO are not enabled by using the `GPIOn_RAWDATA` register
- Interrupt assert capability

## 8.3 Block Diagram

**Figure 8-1** illustrates the logic associated with each GPIO register. Each GPIO pin can be configured as either an input (with or without pull-up), an output, or an interrupt. The pull-up is configured by writing to the Pull-Up Enable Register (PUR). The Data Direction Register (DDR) when the Peripheral Enable Register (PER) is set to zero. However, when PER is set to one, the pull-ups are controlled by the PER and the peripheral output is disabled. In any case, *if the GPIO pin is set to be an output, the pull-up is disabled, thus making the value of the PER irrelevant.* The GPIO features support putting the output driver into Open Drain mode by setting the PPMODE bit to zero. This setting will apply to all functions of the pin even when the pin is

configured for alternate functionality.



**Figure 8-1. Bit-Slice View of the GPIO Logic**

## 8.4 Operating Modes

The GPIO module contains two major modes of operation:

- Normal Mode—This can also be thought of as Peripheral Controlled mode. A peripheral module controls the output enable and any output/input data from/to the pin is passed to the peripheral. Pull-up enables are controlled by the PUR when a pin is configured as an input. When the pin is configured as an output the Push Pull characteristics of the output driver are controlled by the PPMODE register setting.
- GPIO Mode—In this mode, the GPIO module controls the output enable to the pin, supplying any data to be output. Also in GPIO mode, any input data can be read from a GPIO memory mapped register. Pull-up enables are controlled by the PUR when a pin is configured as an input.

## 8.5 Configurations

Each GPIO port is controlled by the registers listed in [Table 8-1](#).

**Note:**  $n$  in the following tables refers to the Port (A, B, C, etc.). For example,  $GPIO_n\_PUR$  for Port A would be  $GPIOA\_PUR$ .

**Table 8-1. GPIO Registers With Default Reset Values**

Register Acronym	Register Name	Default Reset State <sup>1, 2</sup>	Description
$GPIO_n\_PUR$	Port $n$ Pull-up Enable Register	\$00FF	Pull-ups are enabled
$GPIO_n\_DR$	Port $n$ Data Register	\$0000	DR is used for data interface between the pin and the IPBus
$GPIO_n\_DDR$	Port $n$ Data Direction Register	\$0000	All pins are configured as inputs
$GPIO_n\_PER$	Port $n$ Peripheral Enable Register	\$00FF	Peripheral controls the GPIO. The DDR does not determine the direction of the I/O.
$GPIO_n\_IAR$	Interrupt Assert Register	\$0000	No Interrupt
$GPIO_n\_IENR$	Interrupt Enable Register	\$0000	Interrupt is disabled
$GPIO_n\_IPOLR$	Interrupt Polarity Register	\$0000	When set to one, interrupts are active low. When set to zero, interrupts are active high.
$GPIO_n\_IPR$	Interrupt Pending Register	\$0000	No interrupt is registered. A one indicates an interrupt.
$GPIO_n\_IESR$	Interrupt Edge Sensitive Register	\$0000	A one indicates an edge has been detected
$GPIO_n\_PPMODE$	Push-Pull Output Mode Control Register	\$00FF	Push-Pull mode enabled out of reset. For modules without this control, the outputs are hardwired for push-pull operation.
$GPIO_n\_RAWDATA$	Provides an unlocked version of the data values currently present on each GPIO pin, even when not in the GPIO mode.	—	Values are not clocked and are subject to change at any time. Read several times to ensure stable values.

1. Assumes 8-bit GPIO. The reset value will be different if the port is not eight bits wide.

2. PER and PUR reset values may be different depending on the reset function of a specific part. See the chip data sheet for actual reset values.

[Table 8-3](#) provides the state of the pin and Pull-up resistor as a function of current peripheral output state and control register values.

**Table 8-2. GPIO Pull-Up Enable Functionality**

Peripheral Output Disable	PER	PUR	DDR	Pin State	Pin Pull-Up
x	0	0	0	Input	Disabled
x	0	0	1	Output	Disabled
x	0	1	0	Input	Enabled
x	0	1	1	Output	Disabled
0	1	x	x	Output	Disabled
1	1	0	x	Input	Disabled
1	1	1	x	Input	Enabled

**Note:** When the PER is set to zero, the PUR and DDR control the pin pull-up. When the PER is set to one, the pin pull-up is controlled by the PUR and peripheral output disable. The PUR value is only recognized when the pin is configured as an input.

**Table 8-3. GPIO Data Transfers Between I/O Pin and IPBus**

Peripheral Output Disable	PER	DDR	Pin State	Access	Data Access Result
x	0	0	Input	Write to DR	Data is written into DR by IPBus. No effect on the pin value.
x	0	1	Output	Write to DR	Data is written into DR by IPBus. DR value seen at pin.
x	0	0	Input	Read from DR	Pin state is read by the IPBus. No effect on pin value.
x	0	1	Output	Read from DR	DR value is read by the IPBus. DR value seen at pin.
1	1	x	Input	Write to DR	Data is written into DR by IPBus. No effect on pin value.
0	1	x	Output	Write to DR	Data is written into DR by IPBus. Peripheral pin output is seen at the pin.
1	1	x	Input	Read from DR	DR value is read by the IPBus. No effect on the pin or DR value.
0	1	x	Output	Read from DR	DR value is read by the IPBus. Peripheral pin output is seen at the pin.

## 8.6 Register Definitions

**Note:** Please refer to the Data Sheet for the base address of this peripheral.

Each GPIO register contains one to 16 readable/writable bits. For illustration purposes, an eight-bit definition is assumed for all registers in this chapter. Each register bit performs an

identical function for each of the GPIO pins controlled by a specific GPIO port. The only difference is with regard to initial operating conditions at reset. Some GPIO ports are on as default, while others are not. The same is true with pull-up resistors. Some may be enabled, others not.

**Note:** Please be certain to consider the specific chip's availability on GPIO ports. Not all GPIO ports are available on all chips.

A register address is the sum of a base address and an address offset. The base address is defined at the system level and the address offset is defined at the module level. The GPIO module has 11 registers.

**Table 8-4. GPIO Register Summary**

Register Address	Register Acronym	Register Name	Access Type	Chapter Location
Base + \$0	GPIO <sub>n</sub> _PUR	Pull-Up Enable Register	Read/Write	<a href="#">Section 8.6.1</a>
Base + \$1	GPIO <sub>n</sub> _DR	Data Register	Read/Write	<a href="#">Section 8.6.2</a>
Base + \$2	GPIO <sub>n</sub> _DDR	Data Direction Register	Read/Write	<a href="#">Section 8.6.3</a>
Base + \$3	GPIO <sub>n</sub> _PER	Peripheral Enable Register	Read/Write	<a href="#">Section 8.6.4</a>
Base + \$4	GPIO <sub>n</sub> _IAR	Interrupt Assert Register	Read/Write	<a href="#">Section 8.6.5</a>
Base + \$5	GPIO <sub>n</sub> _IENR	Interrupt Enable Register	Read/Write	<a href="#">Section 8.6.6</a>
Base + \$6	GPIO <sub>n</sub> _IPOLR	Interrupt Polarity Register	Read/Write	<a href="#">Section 8.6.7</a>
Base + \$7	GPIO <sub>n</sub> _IPR	Interrupt Pending Register	<i>Read-Only</i>	<a href="#">Section 8.6.8</a>
Base + \$8	GPIO <sub>n</sub> _IESR	Interrupt Edge Sensitive Register	Read/Write	<a href="#">Section 8.6.9</a>
Base + \$9	GPIO <sub>n</sub> _PPMODE	Push/Pull Mode Register	Read/Write	<a href="#">Section 8.6.10</a>
Base + \$A	GPIO <sub>n</sub> _RAWDATA	Raw Data Register	<i>Read-Only</i>	<a href="#">Section 8.6.11</a>

Note: Please see the chip specific Data Sheet for the base address.

**Note:** Reserved fields are read as zeros and they cannot be modified by writing

Addr. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$0	GPIO <sub>n</sub> _PUR	R	0	0	0	0	0	0	0	0	PU							
		W																
\$1	GPIO <sub>n</sub> _DR	R	0	0	0	0	0	0	0	0	D							
		W																
\$2	GPIO <sub>n</sub> _DDR	R	0	0	0	0	0	0	0	0	DD							
		W																
\$3	GPIO <sub>n</sub> _PER	R	0	0	0	0	0	0	0	0	PE							
		W																
\$4	GPIO <sub>n</sub> _IAR	R	0	0	0	0	0	0	0	0	IA							
		W																
\$5	GPIO <sub>n</sub> _IENR	R	0	0	0	0	0	0	0	0	IEN							
		W																
\$6	GPIO <sub>n</sub> _IPOLR	R	0	0	0	0	0	0	0	0	IPOL							
		W																
\$7	GPIO <sub>n</sub> _IPR	R	0	0	0	0	0	0	0	0	IP							
		W																
\$8	GPIO <sub>n</sub> _IESR	R	0	0	0	0	0	0	0	0	IES							
		W																
\$9	GPIO <sub>n</sub> _PPMODE	R	0	0	0	0	0	0	0	0	OEN							
		W																
\$A	GPIO <sub>n</sub> _RAWDATA	R	0	0	0	0	0	0	0	0	RAWDATA							
		W																

R	0	Read as 0
W		Reserved

**Note:** Assumes 8-bit GPIO. This information may be different depending on the specific part. Please see the device Data Sheet for actual value.

**Figure 8-2. GPIO Register Map Summary**

### 8.6.1 Pull-Up Enable Register (PUR)

This read/write register enables and disables the pull-up on each GPIO pin.

Base + \$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	PU							
Write																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

**Note:** Assumes 8-bit GPIO. This information may be different depending on the specific part. Please see the device Data Sheet for actual value.

**Figure 8-3. Pull-Up Enable Register (PUR)**

When PUR is set to one, DDR is set to zero (an input), and PER set to zero, the pull-up is enabled. However, if PER is set to one, the pull-up is controlled by the peripheral output disable and the PUR. If the pin is configured as an output, the PUR is not used. Please see [Table 8-2](#) for all possible combinations.

- 0 = Pull-Up is disabled



- 1 = Pull-Up is enabled (when DDR = 0 and PER = 0)

## 8.6.2 Data Register (DR)

This read/write register holds data coming either from the pin or the IPBus. That ability makes the DR data interface between the pin and the IPBus. Please see [Table 8-3](#) for possible data transfers between the pin and the IPBus.

Base + \$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	D							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Note:** Assumes 8-bit GPIO. This information maybe different depending on the specific part. Please see the device Data Sheet for actual value.

**Figure 8-4. Data Register (DR)**

## 8.6.3 Data Direction Register (DDR)

This read/write register configures the state of the pin as either an input or output when PE is set to zero. When DD is set to zero, the pin is an input. When DD is set to one, the pin is an output. Please see [Table 8-2](#) for additional information.

- 0 = Pin is an input
- 1 = Pin is an output

Base + \$2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	DD							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Note:** Assumes 8-bit GPIO. This information maybe different depending on the specific part. Please see the device Data Sheet for actual value.

**Figure 8-5. Data Direction Register (DDR)**

## 8.6.4 Peripheral Enable Register (PER)

This read/write register determines the configuration of the GPIO. When PE value is one, the GPIO module is configured for Normal mode. In this mode, a peripheral masters the GPIO pin. This master condition includes configuring the GPIO pin as a required input (with or without pull-up), or output depending on the status of the peripheral output disable. It also includes data

transfers from the pin to the peripheral. Please see [Table 8-2](#) and [Table 8-3](#) for additional information.

The GPIO module is configured for GPIO mode when the PE value is zero. In this mode, the GPIO module controls the output enable to the pin and the supplying of any data to be output. Also in GPIO mode, any input data can be read from a GPIO memory mapped register.

- 0 = Pin is for GPIO (GPIO mode)
- 1 = Pin is for peripheral (Normal mode)

Base + \$3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	PE							
Write																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

**Note:** Assumes 8-bit GPIO. This information maybe different depending on the specific part. Please see the device Data Sheet for actual value.

**Figure 8-6. Peripheral Enable Register (PER)**

### 8.6.5 Interrupt Assert Register (IAR)

This read/write register is used for software testing only. An interrupt is asserted when the IA is set to one. It is cleared by writing zeros into the Interrupt Assert register.

Base + \$4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	IA							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Note:** Assumes 8-bit GPIO. This information maybe different depending on the specific part. Please see the device Data Sheet for actual value.

**Figure 8-7. Interrupt Assert Register (IAR)**

### 8.6.6 Interrupt Enable Register (IENR)

This read/write register enables or disables the edge detection for any incoming interrupt from the pin. This register is set to one for interrupt detection. The interrupts are recorded in the `GPIOn_IPR`.

Base + \$5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Read	0	0	0	0	0	0	0	0	IEN								
Write																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Note:** Assumes 8-bit GPIO. This information maybe different depending on the specific part. Please see the device Data Sheet for actual value.

**Figure 8-8. Interrupt Enable Register (IENR)**

- 0 = Interrupt is disabled
- 1 = Interrupt is enabled

### 8.6.7 Interrupt Polarity Register (IPOLR)

This read/write register is used for polarity detection caused by any external interrupts. The interrupt at the pin is active low when this register is set to one (falling edge causes the interrupt). The interrupt seen at the pin is active high when this register is set to zero (rising edge causes the interrupt). This is true only when the IEN is set at one. There is no effect on the interrupt if the IEN is set to zero.

Base + \$6	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	IPOL							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Note:** Assumes 8-bit GPIO. This information maybe different depending on the specific part. Please see the device Data Sheet for actual value.

**Figure 8-9. Interrupt Polarity Register (IPOLR)**

- 0 = Interrupt is active high
- 1 = Interrupt is active low

### 8.6.8 Interrupt Pending Register (IPR)

This read/write register determines which pin has caused an interrupt. Writing zeros into the IAR register will clear the interrupt if it was caused by software (writing to the IAR). For external interrupts, the IP is cleared by writing ones into the IESR.

Base + \$7	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	IP							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Note:** Assumes 8-bit GPIO. This information may be different depending on the specific part. Please see the device Data Sheet for actual value.

**Figure 8-10. Interrupt Pending Register (IPR)**

### 8.6.9 Interrupt Edge Sensitive Register (IESR)

When an edge is detected by the edge detector circuit, and IEN is set to one, IES records the interrupt. Please see [Figure 8-14](#).

This read/write register clears the corresponding IPR bit field by writing one to a bit in the IES. Writing zero to an IES bit is ignored.

Base + \$8	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	IES							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Note:** Assumes 8-bit GPIO. This information may be different depending on the specific part. Please see the device Data Sheet for actual value.

**Figure 8-11. Interrupt Edge Sensitive Register (IESR)**

### 8.6.10 Push/Pull Output Mode Control Register (PPMODE)

This read/write register explicitly sets each output driver to either push/pull or Open Drain mode.

Base + \$9	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	OEN							
Write																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

**Note:** Assumes 8-bit GPIO. This information may be different depending on the specific part. Please see the device Data Sheet for actual value.

**Figure 8-12. Push/Pull Output Mode Control Register (PPMODE)**

- 0 = Open Drain mode

- 1 = Push/Pull mode

### 8.6.11 Raw Data Register (RAWDATA)

This *read-only* register allows the hybrid controller direct access to the logic values on each GPIO pin, even when pins are not in the GPIO mode. The value at reset is unknown.

Base + \$A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	RAW DATA							
Write																
Reset	0	0	0	0	0	0	0	0	U	U	U	U	U	U	U	U

**Note:** Assumes 8-bit GPIO. This information may be different depending on the specific part. Please see the device Data Sheet for actual value.

**Figure 8-13. Raw Data Register (RAWDATA)**

- 0 = Logic 0 present on GPIO pin
- 1 = Logic 1 present on GPIO pin

## 8.7 Clocks and Resets

The GPIO module has no critical reset issues. The module assumes reset states defined here and in each chip specification. Reset occurs whenever any source of system reset occurs (POR, external reset or COP).

## 8.8 Interrupts

The GPIO has two types of interrupts:

1. Software interrupt for testing purposes

The Interrupt Assert Register (IAR) is used to generate the software interrupt. It can be tested by writing ones to the IAR. The IPR will record the value of IAR. The IPR can be cleared by writing zeros into the IAR during the IAR testing.

**Note:** When testing the IAR, the IPOLR, IESR, and the IENR must be set to zero to guarantee the interrupt registered in the IPR is due to IAR only.

2. An edge sensitive hardware interrupt from the pin

When a GPIO is used as an interrupt, the IAR must be set to zero. Both the IPOLR and IENR must be set to one for an active low interrupt. When the signal at the pin goes low (interrupts are active low) and propagates through the edge detection mechanism, the value is seen at the IESR

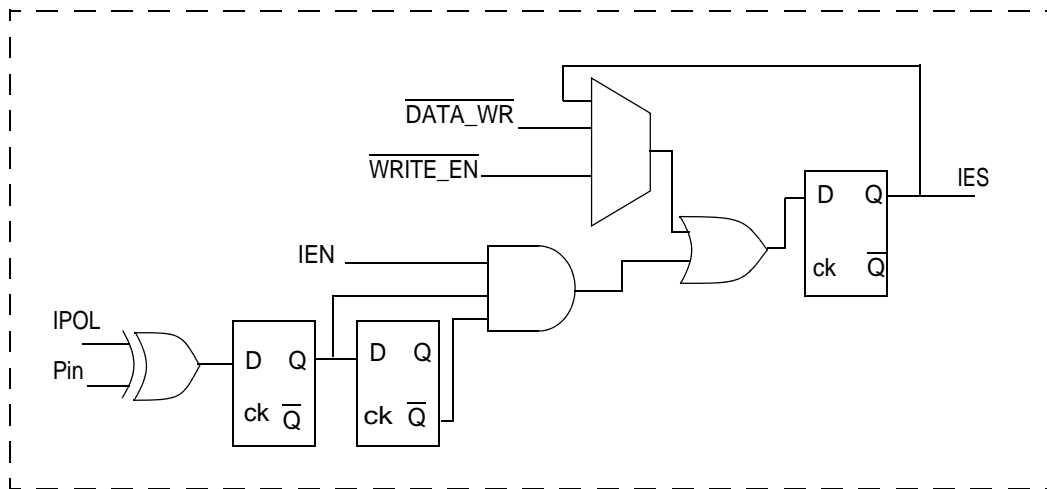
and recorded by the IPR. Please see [Table 8-5](#). The IPR is cleared by writing ones into the IESR. If IPOLR is set to zero, the interrupt at the pin is active high.

The interrupt signals in each port are ORed together, presenting only a single interrupt per port to the 16-bit controller core. The interrupt service routine must then check the contents of the IPR to determine which pin(s) caused the interrupt.

External interrupt sources do not need to remain asserted due to the edge sensitive nature of the detection mechanism.

**Table 8-5. GPIO Interrupt Assert Functionality**

IPOLR	Interrupt Asserted	Remark
0	High	If the IENR is set to one, as the pin goes to high an interrupt will be recorded by the IPR.
1	Low	If the IENR is set to one, as the pin goes to low an interrupt will be recorded by the IPR.



**Figure 8-14. Edge Detection Circuit**

---

# Chapter 9

## Joint Test Action Group Port (JTAG)

## Document Revision History for [Chapter 9, Joint Test Action Group Port \(JTAG\)](#)

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 4.0	Added reference to BSDL file in Section 9.9
Rev 5.0	Converted chapter to Freescale design standards



## 9.1 Introduction

Joint Test Action Group (JTAG) Boundary scan is an IEEE 1149.1 standard methodology enabling access to test features using a Test Access Port (TAP). JTAG Boundary scan consists of a TAP controller and boundary scan registers. This TAP, referred to as the Chip TAP in this document includes all IEEE 1149.1 required instructions plus several additional instructions and registers to accommodate Flash Mass Erase. Since this device also has a 56800E core containing its own TAP, or *Core TAP*, a TAP Linking Module (TLM) is included to manage the TAP access. Normal operation of this part will use the Chip TAP as the Master TAP controller, thereby disabling the 56800E TAP (Core TAP) controller. This chapter discusses the Master TAP only.

## 9.2 Features

Test Access Port (TAP) port characteristics include:

- Perform boundary scan operations to test circuit board electrical continuity
- Bypass the TAP for a given circuit board test by replacing the Boundary Scan Register (BSR) with a single-bit register
- Sample system pins during operation and transparently shift-out the results in the BSR
- Preload output pins prior to invoking the EXTEST instruction
- Disable the output drive to pins during circuit board testing
- Provide a means of accessing the EOnCE module controller and circuits to control a target system
- Query the IDCODE from any TAP in the system
- Force test data onto the peripheral outputs while replacing its BSR with a single bit register
- Enable/disable pull-up devices on peripheral boundary scan pins

### 9.3 Block Diagram

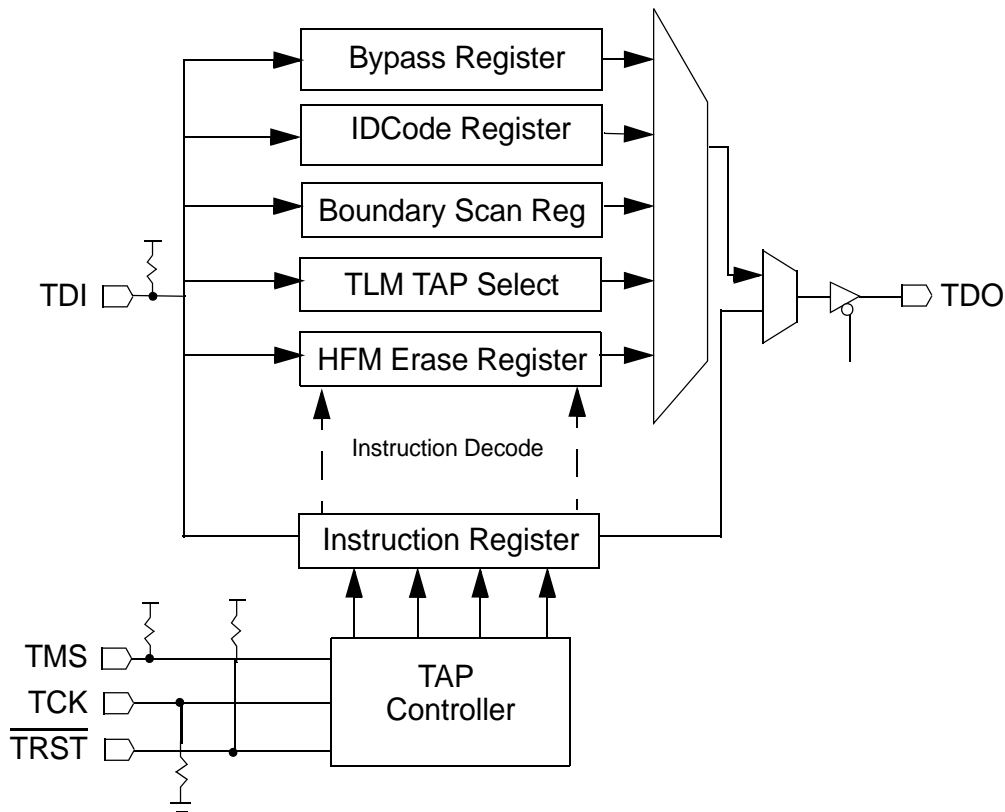


Figure 9-1. JTAG Block Diagram

### 9.4 Functional Description

The Master TAP consists of a synchronous finite 16-bit state machine, an eight-bit instruction register, a boundary scan register, a bypass register, and an identification code register. Two additional registers (KTR/DTR) have been added to support BIST and other test modes.

#### 9.4.1 Master TAP Instructions

The eight-bit Master TAP Instruction register is in support of all JTAG functions. It is described in [Table 9-1](#). This register includes all IEEE 1149.1 required instructions plus several additional instruction registers accommodating debug and BIST testing.

**Table 9-1. Master TAP Instructions Opcode**

Instruction	Target Register	Opcode
EXTEST	BOUNDARY	00000000
BYPASS	BYPASS	11111111
SAMPLE_PRELOAD	BOUNDARY	00000001
IDCODE	IDCODE	00000010
	Reserved	
	Reserved	
TLM_SEL	TLM	00000101
HIGHZ	BYPASS	00000110
CLAMP	BYPASS	00000111
Lock Out Recovery (Flash_Erase)	FLASH_ERASE	00001000

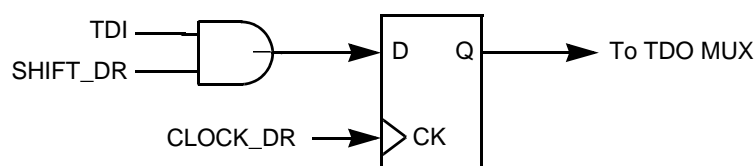
#### 9.4.1.1 External Test Instruction (EXTEST)

The External Test (EXTEST) instruction enables the BSR between TDI and TDO, including cells for all digital device signals and associated control signals. The EXTAL and RESET pins, and any codec pins associated with analog signals, are not included in the BSR path.

In EXTEST, the BSR is capable of scanning user-defined values onto output pins, capturing values presented to input signals and controlling the direction and value of bidirectional pins. EXTEST instruction asserts internal system reset for the 16-bit controller system logic during its run in order to force a predictable internal state while performing external boundary scan operations.

#### 9.4.1.2 Bypass Instruction (BYPASS)

The BYPASS instruction is a required JTAG instruction, selecting the TAP bypass register. This register is a single stage shift register providing a serial path between the TDI and the TDO pins illustrated in [Figure 9-2](#). This instruction enhances test efficiency by shortening the overall path between TDI and TDO when no test operation of a component is required.

**Figure 9-2. Bypass Register Diagram**

### 9.4.1.3 Sample and Preload Instructions (SAMPLE/PRELOAD)

The SAMPLE/PRELOAD instruction is a required JTAG instruction enabling the BSR between TDI and TDO. This instruction provides two major functions:

1. It provides a means to obtain a snapshot of system data and control signals (SAMPLE). The snapshot occurs on the rising edge of TCK in the Capture-DR controller state. The data can be observed by shifting it transparently through the BSR and out of the TDO pin.
2. It initializes the BSR output cells prior to selection of the EXTEST instruction.

### 9.4.1.4 IDCODE

The IDCODE instruction is an optional JTAG instruction enabling the chip identification register between TDI and TDO. This 32-bit register identifies the manufacturer, part and version numbers.

### 9.4.1.5 TLM\_SEL

The TLM\_SEL instruction is a user-defined JTAG instruction, disabling the Master TAP and enabling the TAP Linking Module, or TLM. The TLM then selects the 5683xx core TAP, or the Master TAP, as the enabled TAP.

### 9.4.1.6 High Z Instruction (HIGHZ)

The HIGHZ instruction is an optional JTAG instruction capable of tri-stating all boundary scan outputs. This instruction will hold the chip in reset forcing a predictable internal state while concurrently performing external boundary scan operations. HIGHZ also enables the Bypass register between TDI and TDO, required by the IEEE 1149.1 specification.

### 9.4.1.7 CLAMP

The CLAMP instruction is an optional JTAG instruction holding all boundary scan states loaded during an EXTEST or SAMPLE/PRELOAD instruction. This instruction holds the chip in reset, forcing a predictable internal state while performing external boundary scan operations. This instruction also enables the Bypass register between TDI and TDO, as required by the IEEE 1149.1 specification.

### 9.4.1.8 Lock Out Recovery (FLASH\_Erase)

Lock Out Recovery Completely erases the Flash memory to start again. This is used in the case when Flash is inaccessible for changes due to security locks. When this opcode is loaded, the next 16 bits to be loaded in the Flash Erase Register, [Table 9-2](#), will include the FLASH\_CLKDIV. After scanning in the 16 bits, the value of CLKDIV is sent to FLASH.

**Table 9-2. Flash Erase Register**

Bits	Test Mode Scan	Test Mode	Capture Data
0	FLASH_CLKDIV[0]	Not Connected	FLASH_CLKDIV[0]
1	FLASH_CLKDIV[1]	Not Connected	FLASH_CLKDIV[1]
2	FLASH_CLKDIV[2]	Not Connected	FLASH_CLKDIV[2]
3	FLASH_CLKDIV[3]	Not Connected	FLASH_CLKDIV[3]
4	FLASH_CLKDIV[4]	Not Connected	FLASH_CLKDIV[4]
5	FLASH_CLKDIV[5]	Not Connected	FLASH_CLKDIV[5]
6	FLASH_CLKDIV[6]	Not Connected	FLASH_CLKDIV[6]
7	Not Connected	Not Connected	0
8	Not Connected	Not Connected	0
9	Not Connected	Not Connected	0
10	Not Connected	Not Connected	0
11	Not Connected	Not Connected	0
12	Not Connected	Not Connected	0
13	Not Connected	Not Connected	0
14	Not Connected	Not Connected	0
15	Not Connected	Not Connected	0

The Flash responds to erase only once. It remains in an inoperable state until reset. For example, every time the Flash is erased in this manner, the part must be reset in order to use the Flash for Normal operation. The Flash Erase and FLASH\_CLKDIV signals retains the same value until the system is reset, turning off the Flash Erase signal and resetting the FLASH\_CLKDIV signals back to zeros. Allow Flash about 100ms to complete the erase before resetting the part.

## 9.5 TAP Controller

The TAP Controller is a synchronous 16-bit finite state machine illustrated in [Figure 9-3](#). TAP Controller responds to changes at the TMS and TCK pins. Transitions from one state to another occur on the rising edge of TCK. The value shown adjacent to each state transition represents the signal present on TMS at the time of a rising edge of TCK.

The TDO pin remains in the high impedance state except during the Shift-DR and Shift-IR TAP Controller states. In Shift-DR and Shift-IR controller states, TDO updates on the falling edge of TCK. TDI is sampled on the rising edge of TCK.

The TAP Controller executes the last instruction decoded until a new instruction is entered at the Update-IR state, or Test-Logic-Reset is entered.

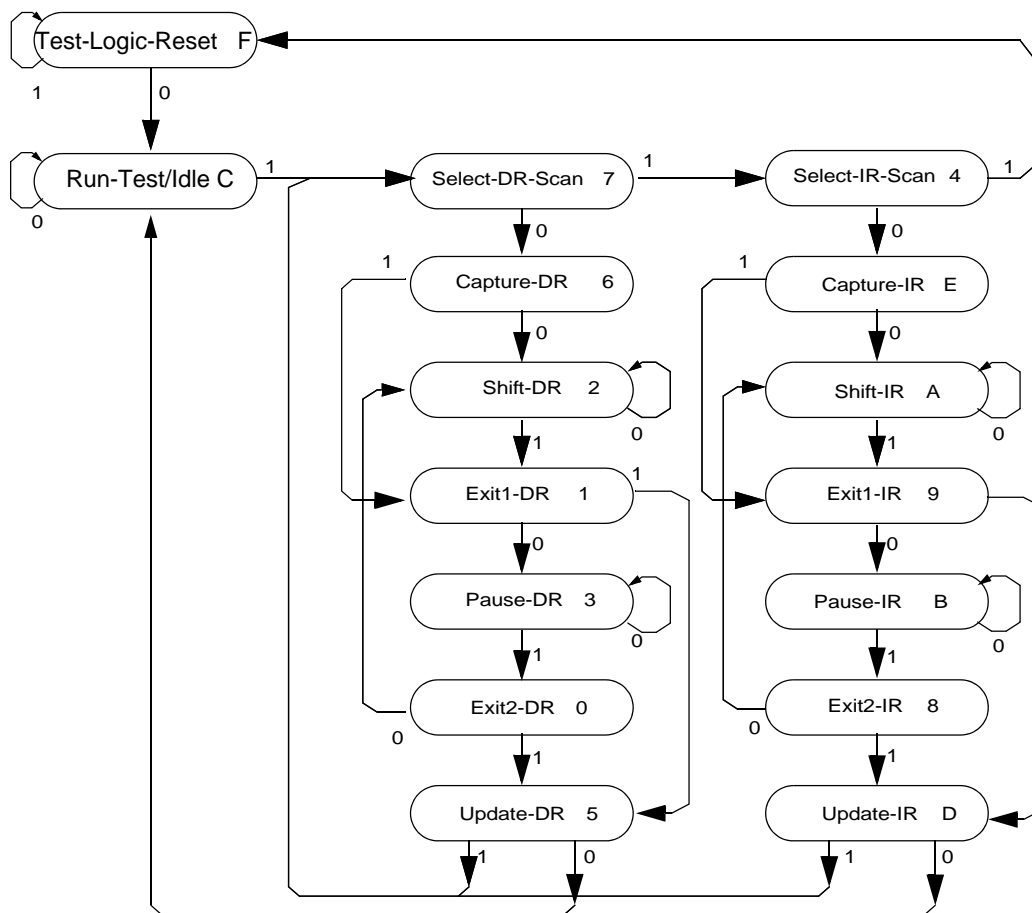


Figure 9-3. TAP Controller State Diagram

## 9.5.1 Operation

All state transitions of the TAP Controller occur based on the value of TMS at the time of a rising edge of TCK. Actions of the instructions occur on the falling edge of TCK in each controller state illustrated in [Figure 9-3](#).

### 9.5.1.1 Test Logic Reset (pstate = F)

During Test-Logic-Reset, all JTAG test logic is disabled so the chip can operate in Normal mode. This is achieved by initializing the Instruction Register (IR) with the IDCODE instruction. By holding TMS high for five rising edges of TCK, the device always remains in Test-Logic-Reset no matter what state the TAP Controller was in previously.

### 9.5.1.2 Run-Test-Idle (pstate = C)

Run-Test-Idle is a controller state between scan operations. When EOnCE is entered, the controller remains in Run-Test-Idle mode as long as TMS is held low. When TMS is high and a rising edge of TCK occurs, the controller moves to the Select-DR state.

### 9.5.1.3 Select Data Register (pstate = 7)

The Select-Data register state is a temporary state. In this state, all Test Data registers selected by the current instruction retains their previous states. If TMS is held low and a rising edge of TCK occurs when the controller is in this state, the controller moves into the Capture-DR state and a scan sequence for the selected test data register is initiated. If TMS is held high and a rising edge of TCK occurs, the controller moves to the Select-IR state.

### 9.5.1.4 Select Instruction Register (pstate = 4)

The Select-Instruction register state is a temporary state. In this state, all Test Data registers selected by the current instruction retain their previous states. If TMS is held low and a rising edge of TCK occurs when the controller is in this state, the controller moves into the Capture-IR state and a scan sequence for the instruction register is initiated. If TMS is held high and a rising edge of TCK occurs, the controller moves to the Test-Logic-Reset state.

### 9.5.1.5 Capture Data Register (pstate = 6)

In this controller state, data may be parallel loaded into test registers selected by the current instruction on the rising edge of TCK. If a Test Data register selected by the current instruction does not have a parallel input, the register retains its previous value.

### 9.5.1.6 Shift Data Register (pstate = 2)

In this controller state, the Test Data register is connected between TDI and TDO. This data is then shifted one stage towards its serial output on each rising edge of TCK. The TAP Controller remains in this state while TMS is held at low. When one is applied to TMS and a positive edge of TCK occurs, the controller will move to the Exit1-DR state.

### 9.5.1.7 Exit1 Data Register (pstate = 1)

This is a temporary controller state. If TMS is held high, and a rising edge is applied to TCK while in this state causes the controller to advance to the Update-DR state. This terminates the scanning process.

### 9.5.1.8 Pause Data Register (pstate = 3)

This controller state permits shifting of the Test Data register in the serial path between TDI and TDO to be temporarily halted. All Test Data registers selected by the current instruction retain

their previous state unchanged. The controller remains in this state while TMS is held low. When TMS goes high and a rising edge is applied to TCK, the controller advances to the Exit2-DR state.

#### **9.5.1.9 Exit2 Data Register (pstate = 0)**

This is a temporary controller state. If TMS is held high, and a rising edge is applied to TCK while it is in this state, the scanning process terminates and the TAP Controller advances to the Update-DR state. If TMS is held low and a rising edge of TCK occurs, the controller advances to the Shift-DR state.

#### **9.5.1.10 Update Data Register (pstate = 5)**

All Boundary Scan registers contain a two-stage data register. It isolates the shifting and capturing of data on the peripheral from what is applied to internal logic during Scan mode. This register is the second stage, or parallel output, and applies a stimulus to internal logic. Data is latched on the parallel output of these Test Data registers from the Shift register path on the falling edge of TCK in the Update-DR state. On a rising edge of TCK, the controller advances to the Select\_DR state if TMS is held high or the Run-Test-Idle state if TMS is held low.

#### **9.5.1.11 Capture Instruction Register (pstate = E)**

When the TAP Controller is in this state and a rising edge of TCK occurs, the controller advances to the Exit1-IR state if TMS is held at one, or the Shift-IR state if TMS is held at zero.

#### **9.5.1.12 Shift Instruction Register (pstate = A)**

In this controller state, the Shift register contained in the instruction register is connected between TDI and TDO and shifts data one stage toward its serial output on each rising edge of TCK. When the TAP Controller is in this state and a rising edge of TCK occurs, the controller advances to the Exit1-IR state if TMS is held at one or remains in the Shift-IR state if TMS is held at zero.

#### **9.5.1.13 Exit1 Instruction Register (pstate = 9)**

This is a temporary controller state. If TMS is held high, and a rising edge is applied to TCK while in this state causes the controller to advance to the Update-IR state. This terminates the scanning process. If TMS is held low and a rising edge of TCK occurs the controller advances to the Pause-IR state.

#### **9.5.1.14 Pause Instruction Register (pstate = B)**

This controller state allows shifting of the instruction register in the serial path between TDI and TDO to be temporarily halted. All Test Data registers selected by the current instruction retain their previous state unchanged. The controller remains in this state while TMS is held low. When TMS goes high and a rising edge is applied to TCK, the controller advances to the Exit2-IR state.



### 9.5.1.15 Exit2 Instruction Register (pstate = 8)

This is a temporary controller state. If TMS is held high, and a rising edge is applied to TCK while in this state, the scanning process terminates and the TAP Controller advances to the Update-IR state. If TMS is held low and a rising edge of TCK occurs, the controller advances to the Shift-IR state.

### 9.5.1.16 Update Instruction Register (pstate = D)

During this state, instruction shifted into the Instruction register is latched from the Shift register path on the falling edge of TCK and into the instruction latch. It becomes the current instruction. On a rising edge of TCK, the controller advances to the Selector state if TMS is held high, or the Run-Test-Idle state if TMS is held low.

## 9.6 Memory Map

JTAG has no memory mapped registers.

## 9.7 Pin Description

The signal summaries for the JTAG are located in [Table 9-3](#).

**Table 9-3. JTAG Pin Description**

Pin Name	Pin Description
TCK	<b>Test Clock Input</b> —This input pin provides the clock to synchronize the test logic and shift serial data to and from all TAP Controllers and the TLM. If the EOnCE module is not being accessed using the Master or 56800E core TAP Controllers, the maximum TCK frequency is 1/4 the maximum frequency for the 56800E core. When accessing the EOnCE module through the 56800E core TAP Controller, the maximum frequency for TCK is 1/8 the maximum frequency for the 56800E core. The TCK pin has a pull down non-disabled resistor.
TDI	<b>Test Data Input</b> —This input pin provides a serial input data stream to the TAP and the TLM. It is sampled on the rising edge of TCK. TDI has an on-chip pull-up resistor which can be disabled through SIM_PUDR register in the SIM module.
TMS	<b>Test Mode Select Input</b> —This input pin is used to sequence the TAP Controller's TLM state machine. It is sampled on the rising edge of TCK. TMS has an on-chip pull-up resistor which can be disabled through SIM_PUDR register in the SIM module.
$\overline{\text{TRST}}$	<b>Test Reset</b> —This input pin provides an asynchronous reset signal to the TLM and all TAP Controllers. If the JTAG is not going to be used, prevent signal interference by holding it low during operation.
TDO	<b>Test Data Output</b> —This tri-state output pin provides a serial output data stream from the Master TAP, or 56800E core TAP Controller. It is driven in the Shift-IR and Shift-DR controller states of the TAP Controller state machines. Output data changes on the falling edge of TCK.

## 9.8 JTAG Port Architecture

The TAP Controller is a simple state machine used to sequence the JTAG port through its varied operations:

- Serially shift in or out a JTAG port command
- Update and decode the JTAG port Instruction Register (IR)
- Serially input or output a data value
- Update a JTAG port or EOnCE module register

**Note:** The JTAG port supervises the shifting of data into and out of the EOnCE module through TDI and TDO pins respectively. In this case, the shifting is guided by the same controller used when shifting JTAG information.

A block diagram of the JTAG port is provided in [Figure 9-1](#). The JTAG port has four read/write registers:

1. Instruction Register (JTAGIR)
2. Chip Identification (CID) register
3. Bypass Register (JTAGBR)
4. Boundary Scan Register (BSR)

Access to the EOnCE registers is described in the *56800E Data Sheet*.

## 9.9 JTAG Boundary Scan Register

This register is enabled via the JTAG Master TAP by issuing the EXTEST or SAMPLE\_PRELOAD instructions, enabling the Boundary Scan registers between TDI and TDO. Boundary scan cell number one is connected to TDO making it the first data bit shifted into TDI and the first bit shifted out of TDO when loading and unloading the boundary scan chain. The exact order of the scan chain is unique to each chip and can be found in the product's BSDL file.

## 9.10 Clocks

### 9.10.1 TCK

This is the sole clock used by the Master TAP module. If the EOnCE module is not being accessed using the Master or 56800E core TAP controllers, the maximum TCK frequency is one-quarter the maximum frequency for the 56800E core. When accessing the EOnCE module

through the 56800E core TAP controller, the maximum frequency for TCK is one eighth the maximum frequency for the 56800E core.

**Table 9-4. Clock Summary**

Clock	Priority	Source	Characteristics
TCK	1	External	This user provided clock shifts data and control the state machine.

## 9.11 Interrupts

This module has no interrupt capabilities.

## 9.12 Resets

### 9.12.1 $\overline{\text{TRST}}$ Reset

This is the external reset signal provided by the user to reset the TAP Controller.

**Table 9-5. Reset Summary**

Reset	Priority	Source
$\overline{\text{TRST}}$	1	External



---

# Chapter 10

## Power Supervisor (PS)

## Document Revision History for **Chapter 10, Power Supervisor (PS)**

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 3.0	Figure 10-1 over stated, simplified in this revision for clarity.
Rev 5.0	Converted chapter to Freescale design standards

## 10.1 Introduction

This chapter details the on-chip Power Supervisor (PS) module. Its function ensures those chips are operated only within required voltage ranges while assisting in a orderly shutdown of the chip in the event the power supply is interrupted, or there is a brownout, or a voltage sag.

## 10.2 Features

The Power Supervisor (PS) monitors on-chip voltages (digital 3.3V and 2.5V rails), providing the following qualities:

- Power-On Reset (POR) is generated until  $V_{DD}$  core exceeds 1.8V
- Low Voltage Interrupt (LVI) is generated when the 2.5V rail drops below 2.2V
- Low Voltage Interrupt (LVI) is generated when the 3.3V rail drops below 2.7V

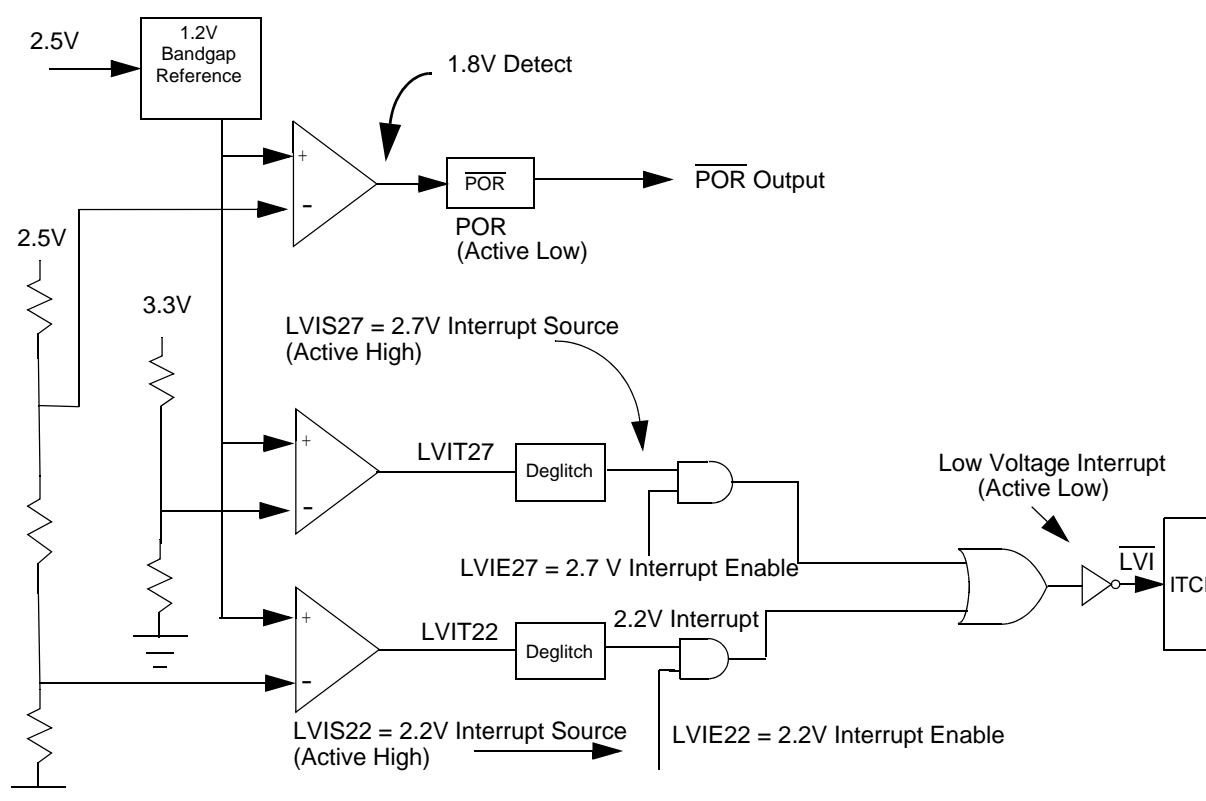
Working under the assumption power supply voltages move relatively slowly with respect to the system clocks, low voltage interrupts should provide ample warning of impending problems providing orderly shutdown of the chip be accomplished.

## 10.3 Block Diagram

The basic Power-On Reset (POR) and low voltage detect module is illustrated in [Figure 10-1](#). The POR circuit is designed to assert the internal reset from  $V_{DD} = 0V$  to  $V_{DD} = 1.8V$ .  $\overline{POR}$  switching high indicates there is enough voltage for on-chip logic to operate at the oscillator frequency.

**Note:** High speed operation is not guaranteed until both the 2.7V and 2.2V interrupt sources are inactive.

The deglitch blocks are essentially strings of four flops in series. The outputs of all four must indicate an active interrupt before the output switches to the active state. This is intended to prevent the LVI circuitry from responding to momentary glitches brought about as a result of normal operation. Once set, the sticky status bits LVIS27S, LVIS22S, and LVI must be explicitly reset by writing one.



**Figure 10-1. Power Supervisor Block Diagram**

## 10.4 Functional Description

The Power-On Reset (POR) is always enabled under normal circumstances and the low voltage interrupts are disabled out of reset. As the device is powered up, the POR will be released. High speed operation via the PLL should be deferred until both LVI sources show supply voltages are above required minimums. The LVIs should then be enabled and the PLL used to increase the system clock frequency.

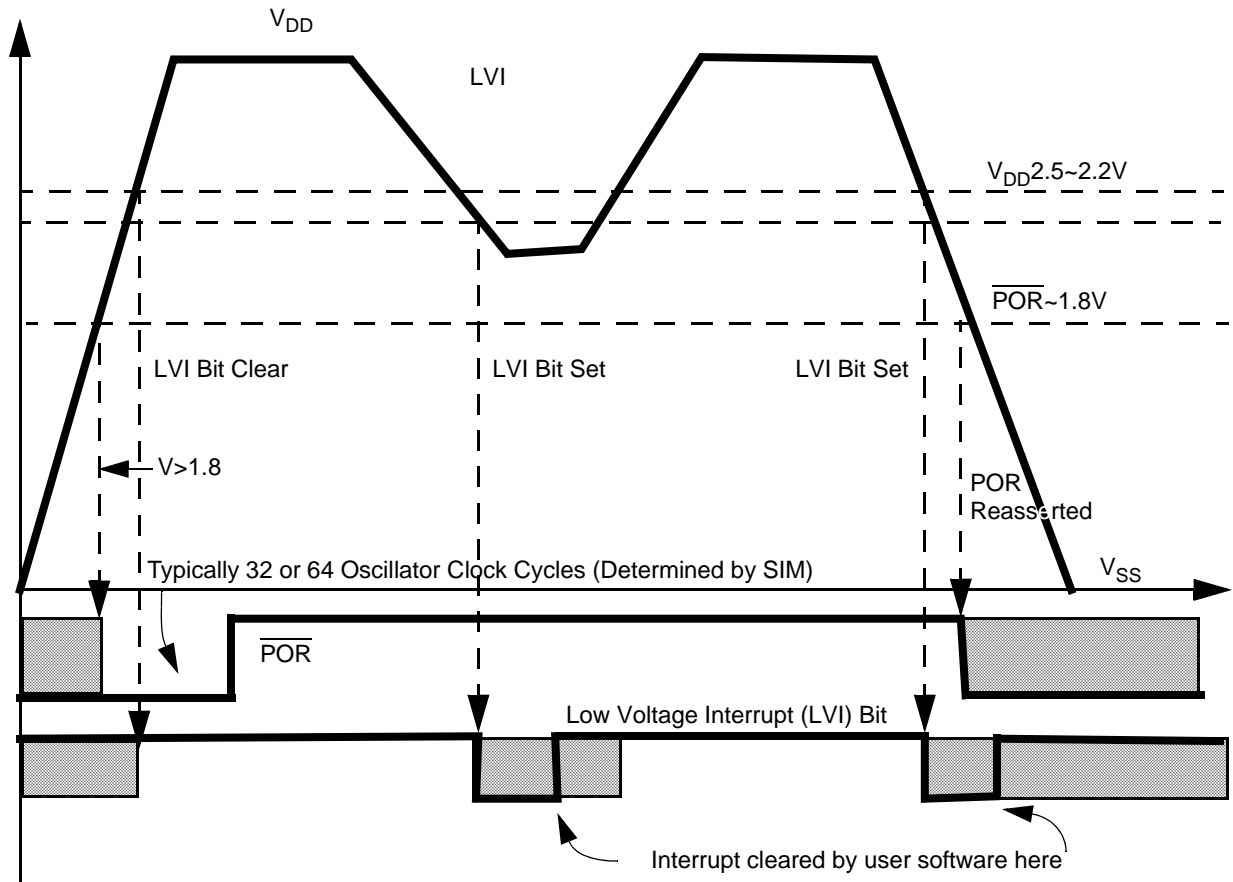
Only when both LVIs are clear should the PLL be used to run the system clock at full speed. In typical application code interrupts are enabled after turning on the PLL. The time between PLL lock and interrupt enable represents a window of vulnerability to LVI events. As long as this window is small it can be tolerated. Otherwise, move the enabling of interrupts to another part of the start-up code.

For most initializations, waiting for the PLL to lock represents the largest component in the start-up time line. Hence, it is prudent to check the LVI status bits again just after PLL lock is achieved.



**Figure 10-2** illustrates operation of the  $\overline{\text{POR}}$  versus low voltage detect circuits. The LVI should be explicitly cleared, and disabled by the interrupt service routine responsible for shutting down the part when a low voltage is detected.

Low voltage interrupts are masked upon  $\overline{\text{POR}}$ . They must be explicitly enabled and disabled thereafter. Low voltage status bits in the Power Supervisor Status Register (LVISR) are always active, independent of whether low voltage interrupts are enabled. Please refer to [Section 10.5.2](#). LVIs include roughly 50-100mV of hysteresis.



Hashed levels show outputs from analog circuitry.

**Figure 10-2.  $\overline{\text{POR}}$  Vs. Low-Voltage Interrupts**

## 10.5 Register Definitions

**Table 10-1. PS Memory Map**

Device	Peripheral	Address
8100/8300	LVI_BASE	\$00F

A register address is the sum of a base address and an address offset. The base address is defined at the system level and the address offset is defined at the module level. The Low Power Voltage Supervisor module has two registers.

**Table 10-2. Power Supervisor Register Summary**

Register Address	Register Acronym	Register Name	Access Type	Chapter Location
Base + \$0	LVICONTROL	Control Register	Read/Write	<a href="#">Section 10.5.1</a>
Base + \$1	LVISTATUS	Status Register	Read/Write	<a href="#">Section 10.5.2</a>

Bit fields of each of the two LVI registers are illustrated in [Figure 10-3](#) and [Figure 10-4](#). Details of each follow.

### 10.5.1 Power Supervisor Control Register (LVICTLR)

Base + \$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LVIE27	LVIE22
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 10-3. Power Supervisor Control Register (LVICTLR)**

#### 10.5.1.1 Reserved—Bits 15–2

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 10.5.1.2 2.7V Low Voltage Interrupt Enable (LVIE27)—Bit 1

- 0 = Interrupt is disabled
- 1 = Interrupt is enabled

### 10.5.1.3 2.2V Low Voltage Interrupt Enable (LVIE22)—Bit 0

- 0 = Interrupt is disabled
- 1 = Interrupt is enabled

## 10.5.2 Power Supervisor Status Register (LVISR)

Base + \$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	LVI	LVIS27S	LVIS22S	LVIS27	LVIS22
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-4. Power Supervisor Status Register (LVISR)

### 10.5.2.1 Reserved—Bits 15–5

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 10.5.2.2 Low Voltage Interrupt (LVI)—Bit 4

LVI is a sticky status bit derived from the combination of LVIS27 and LVIE27, *or* LVIS22 and LVIE22. Write 1 to this bit to clear it. It may be set again immediately if voltages are still low. Writing 0 has no effect.

- 0 = Interrupt not asserted
- 1 = Interrupt asserted

### 10.5.2.3 Sticky 2.7V Low Voltage Interrupt Source (LVIS27S)—Bit 3

When this bit is set it must be cleared explicitly by writing 1 to it. Writing 0 has no effect.

- 0 = 2.7V interrupt threshold has not been passed
- 1 = 3.3V supply has dropped below the 2.7V interrupt threshold

This bit is also set whenever the LVIT27 bit is one long enough to pass through the glitch filter. Bit LVIS27S may be set again in the very next clock cycle after being cleared if the condition causing the supply(ies) to drop still persists.

### 10.5.2.4 Sticky 2.2V Low Voltage Interrupt Source (LVIS22S)—Bit 2

When this bit is set, it must be cleared explicitly by writing 1 to it. Writing 0 has no effect.

- 0 = 2.2V interrupt threshold has not been passed
- 1 = 2.5V supply dropped below the 2.2V interrupt threshold

This bit is also set whenever the LVIT22 bit is one long enough to pass through the glitch filter. Bit LVIS22S may be set again in the very next clock cycle after being cleared if the condition causing the supply(ies) to drop still persists.

#### 10.5.2.5 Non-Sticky 2.7V Low Voltage Interrupt Source (LVIS27)—Bit 1

This bit may reset itself if the supply voltage raises above the comparator threshold point. This bit cannot be modified.

- 0 = 2.7V interrupt threshold has not been passed
- 1 = 3.3V supply has dropped below the 2.7V interrupt threshold

This bit is set whenever the LVIT27 bit, illustrated in [Figure 10-1](#), is one long enough to pass through the glitch filter.

#### 10.5.2.6 Non-Sticky 2.2V Low Voltage Interrupt Source (LVIS22)—Bit 0

This *read-only* bit will reset itself if the supply voltage raises above the comparator threshold point. This bit cannot be modified.

- 0 = 2.2V interrupt threshold has not been passed
- 1 = 2.5V supply is below the 2.2V interrupt threshold

This bit is set whenever the LVIT22 bit, illustrated in [Figure 10-1](#), is one long enough to pass through the glitch filter.

## 10.6 Suggestions for LVI Interrupt Service Routines

The presence of one or more LVI bits high in the Power Supervisor Status Register (LVISR) indicates normal operation of the chip is no longer possible. If the 2.7V interrupt has fired (LVIS27S or LVIS27 are one) then operation of the chip I/O ring is questionable. (Logic may operate correctly but high output levels will not meet requirements.) If the 2.2V Interrupt has fired, the core's operation at high speed is questionable.

Prudence demands the ISR for either LVI events basically do the following:

- Run chip at low speed
- Disable COP
- Switch OFF the PLL
- Set COP to wake up chip in one second
- Enable COP for operation in Stop mode
- Shutdown peripherals in an orderly manner

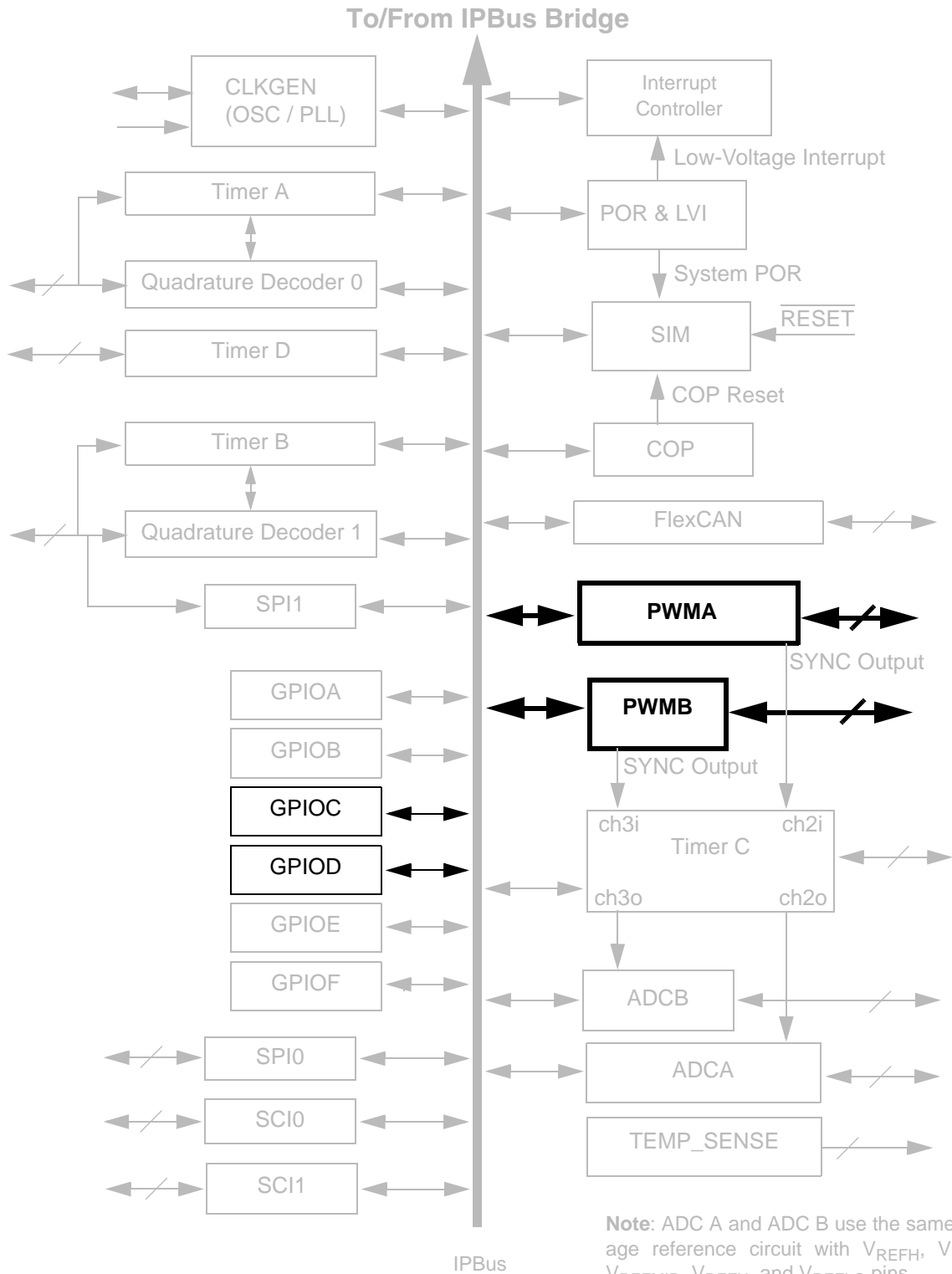
- Put chip in Stop mode

With the COP timer enabled in the Stop mode, the chip will timeout and wake-up the chip if the POR circuitry is not active (supply voltage > 1.8V). Then LVI status bits can be checked again before switching to the PLL. If the LVI bits are still active after being initially cleared, the chip should again go into a *safe mode* through the execution of the LVISR.



# Chapter 11

## Pulse Width Modulator (PWM)



## Document Revision History for Chapter 11, Pulse Width Modulator (PWM)

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 3.0	No changes
Rev 4.0	Corrected values in Figures 11-12 and 11-13, clarified text in section 11.10.13 and added Figure 11-19, grammar edits throughout chapter.
Rev 5.0	Converted chapter to Freescale design standards
Rev 7.0	Added reference to Figure 11-1 to clarify introduction
Rev 8.0	Minor formatting edits.



## 11.1 Introduction

This chapter describes the Pulse Width Modulator (PWM) module. The PWM can be configured as three complementary pairs, six independent PWM signals, or their combinations, such as one complementary or four independent. Both Edge- and Center-Aligned synchronous pulse width control, from zero to 100 percent modulation, are supported.

A 15-bit common PWM counter is applied to all six channels. PWM resolution is one clock period for Edge-Aligned operation and two clock periods for Center-Aligned operation. The clock period is dependent on the IPBus frequency and a programmable prescaler as shown in [Figure 11-1](#).

When generating complementary PWM signals, the module features automatic deadtime insertion to PWM output pairs. Each PWM output can be controlled by PWM generator or software manually.

## 11.2 Features

- Six PWM signals
  - All independent
  - Complementary pairs
  - Mix independent and complementary
- Features of complementary channel operation
  - Deadtime insertion
  - Separate top and bottom pulse width correction via current status inputs or software
  - Asymmetric PWM output within Center Align operation
  - Separate top and bottom polarity control
- Edge- or Center-Aligned PWM signals
- 15-bits of resolution
- Half-cycle reload capability
- Integral reload rates from 1 to 16
- Individual software controlled PWM output
- Programmable fault protection
- Polarity control
- 10/12 mA current source/sink capability on PWM pins
- Write protected registers

## 11.3 Block Diagram

The PWM block diagram is illustrated in [Figure 11-1](#).

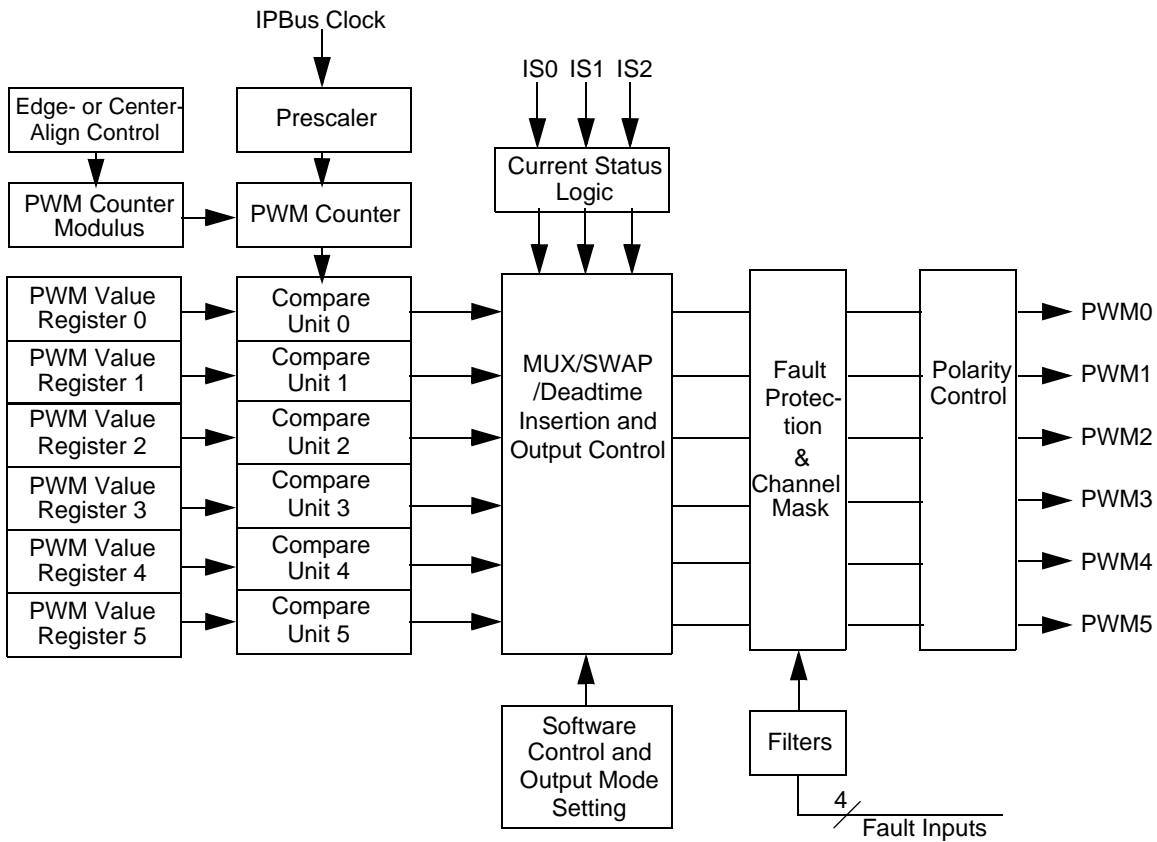


Figure 11-1. PWM Block Diagram

## 11.4 Functional Description

### 11.4.1 Prescaler

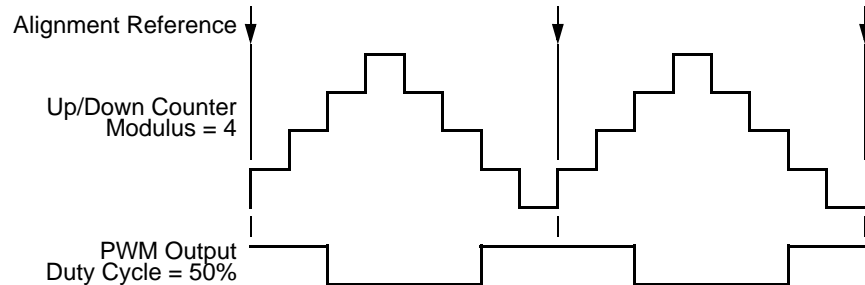
To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the IPBus clock frequency by one, two, four, or eight. The prescaler bits, PRSC0 and PRSC1 in the PWM Control (PMCTL) register, select the prescaler divisor. This prescaler is buffered and will not be used by the PWM generator until the LDOK bit is set and a new PWM reload cycle begins.

### 11.4.2 PWM Generator

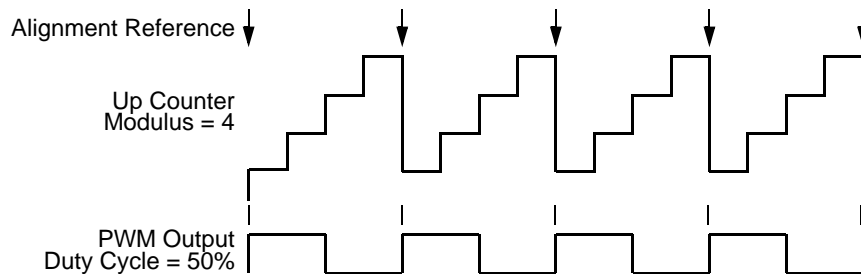
The PWM generator contains a 15-bit up/down PWM counter producing output signals with software-selectables.

### 11.4.2.1 Alignment

The Edge-Align (EDGE) bit in the PWM Configure (PMCFG) register selects either Center-Aligned or Edge-Aligned PWM generator outputs.



**Figure 11-2. Center-Aligned PWM Output**



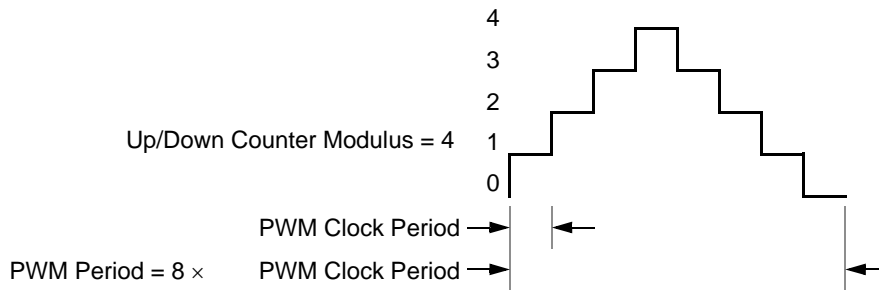
**Figure 11-3. Edge-Aligned PWM Output**

**Note:** Because of the equals-comparator architecture of this PWM, the modulus equals zero case is considered illegal. However, the deadtime constraints and fault conditions will still be guaranteed.

### 11.4.2.2 Period

The PWM period is determined by the value written to the PWMCM register. The PWM counter is an up/down counter in a Center-Aligned operation. In this mode the PWM highest output resolution is two IPBus clock cycles. The modulus is one-half of the PWM output period in PWM clock cycles.

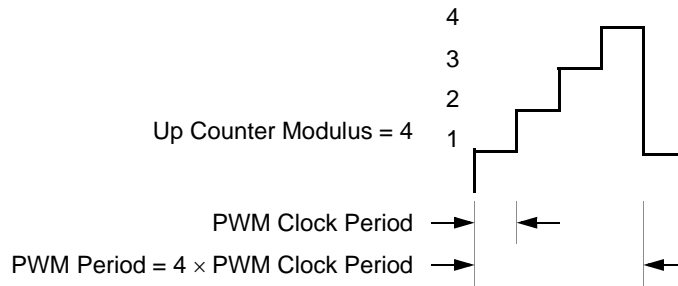
$$\text{PWM period} = (\text{PWM modulus}) \times (\text{PWM clock period}) \times 2$$



**Figure 11-4. Center-Aligned PWM Period**

The PWM counter is an up-counter during an Edge-Aligned operation. In this mode, the PWM highest output resolution is one IPBus clock cycle. The modulus is the period of the PWM output in PWM clock cycles.

$$\text{PWM period} = \text{PWM modulus} \times \text{PWM clock period}$$



**Figure 11-5. Edge-Aligned PWM Period**

### 11.4.2.3 Pulse Width Duty Cycle

The signed 16-bit number written to the PWM value registers is the pulse width in PWM clock periods of the PWM prescaler output.

$$\text{Duty Cycle} = \frac{\text{PWM value}}{\text{Modulus}} \times 100$$

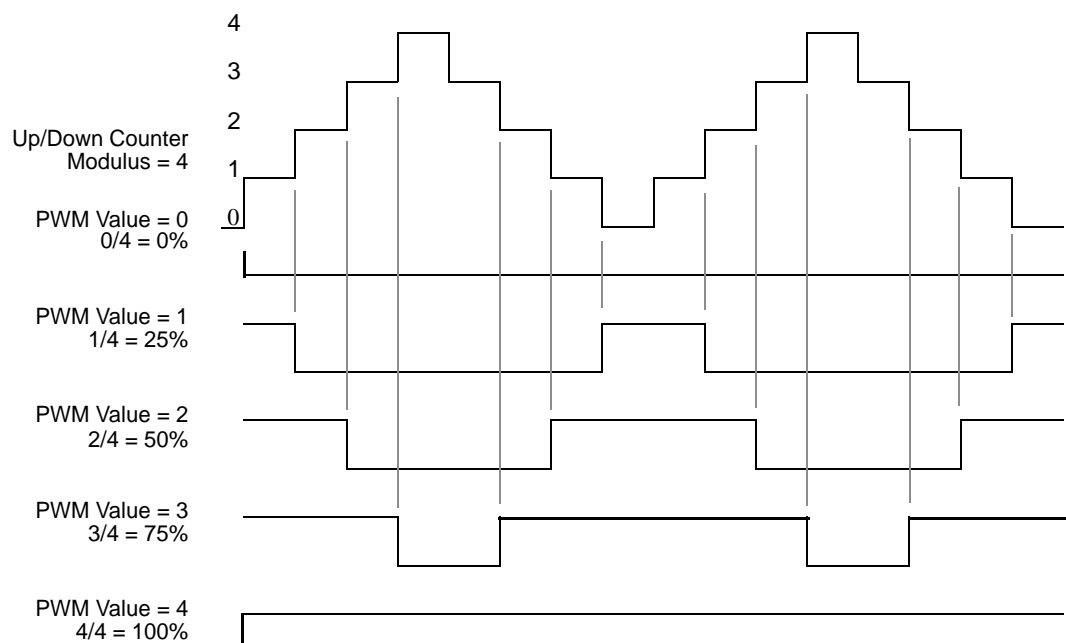
**Note:** A PWM value less than, or equal to, zero deactivates the PWM output for the entire PWM period. A PWM value greater than or equal to the modulus activates the PWM output for the entire PWM period.

**Table 11-1. PWM Value and Underflow Conditions**

PWMVAL $n$	Condition	PWM Value Used
\$0000–\$7FFF	Normal	Value in Registers
\$8000–\$FFFF	Underflow	\$0000

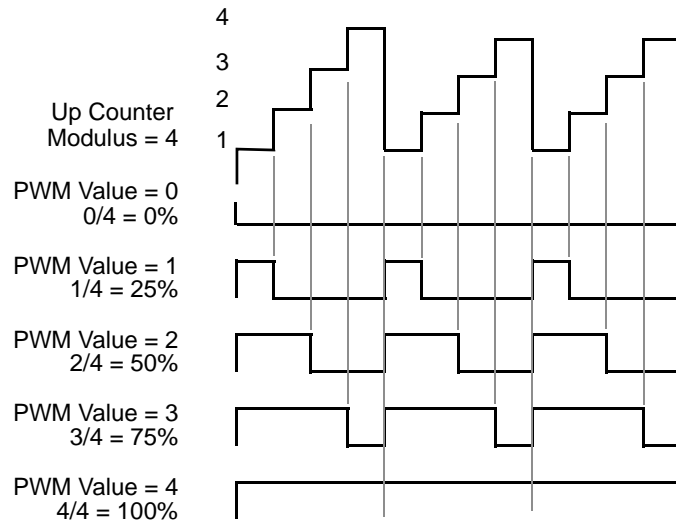
A Center-Aligned operation is illustrated in [Figure 11-6](#). The pulse width is twice the value written to the PWM Value register with Center-Aligned output in PWM clock cycles.

$$\text{Pulse width} = (\text{PWM value}) \times (\text{PWM clock period}) \times 2$$

**Figure 11-6. Center-Aligned PWM Pulse Width**

An Edge-Aligned operation is illustrated in [Figure 11-7](#). The pulse width is the value written to the PWM Value register with Edge-Aligned output in PWM clock cycles.

$$\text{Pulse width} = (\text{PWM value}) \times (\text{PWM clock period})$$

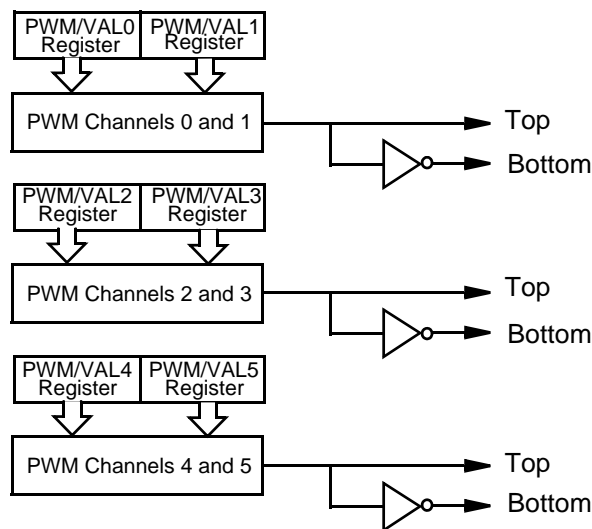


**Figure 11-7. Edge-Aligned PWM Pulse Width**

### 11.4.3 Independent or Complementary Channel Operation

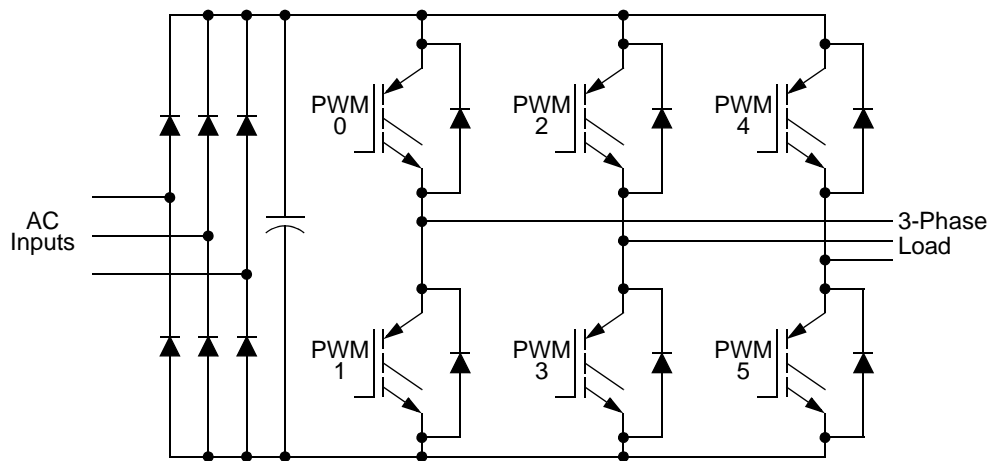
In the PWM Configure register, writing Logic 1 to the Independent or complement pair operation (INDEP $nn$ ) bit configures a pair of the PWM outputs as two independent PWM channels. Each PWM output has its own PWM value register operating independently of the other channels in independent channel operation.

Writing Logic 0 to the INDEP $nn$  bit configures the PWM output as a pair of complementary channels. The PWM pins are paired in complementary channel operation, illustrated in [Figure 11-8](#).



**Figure 11-8. Complementary Channel Pairs**

The complementary channel operation drives top and bottom transistors in an inverter circuit, such as the one in [Figure 11-9](#).



**Figure 11-9. Typical 3-Phase Inverter**

In complementary channel operation, there are three additional features:

1. Deadtime insertion
2. Separate top and bottom pulse width correction for distortions caused by deadtime inserted and reactive load characteristics
3. Separate top and bottom output polarity control

#### 11.4.4 Deadtime Generators

While in the Complementary mode, each PWM pair can be used to drive top/bottom transistors, illustrated in [Figure 11-10](#). Ideally, the PWM pairs are an inversion of each other. When the top PWM channel is active, the bottom PWM channel is inactive and vice versa.

To avoid short circuiting between top and bottom transistor, there must be no overlap of conducting intervals between top and bottom transistor. But the transistor's characteristics make its switching-off time longer than switching-on time. To avoid the conducting overlap of top and bottom transistors, deadtime needs to be inserted in the switching period.

Deadtime generators automatically insert software-selectable activation delays into each pair of PWM outputs. The Pulse Module Deadtime (PMDEADTM) register specifies the number of PWM clock cycles to use for deadtime delay. Every time the PWM generator output changes state, deadtime is inserted. Deadtime forces both PWM outputs in the pair to the inactive state. A method of correcting this inserted deadtime, adding to or subtracting from the PWM value used, is discussed subsequently.

Figure 11-11, Figure 11-12, and Figure 11-13 illustrate deadtime insertion in different operation conditions.

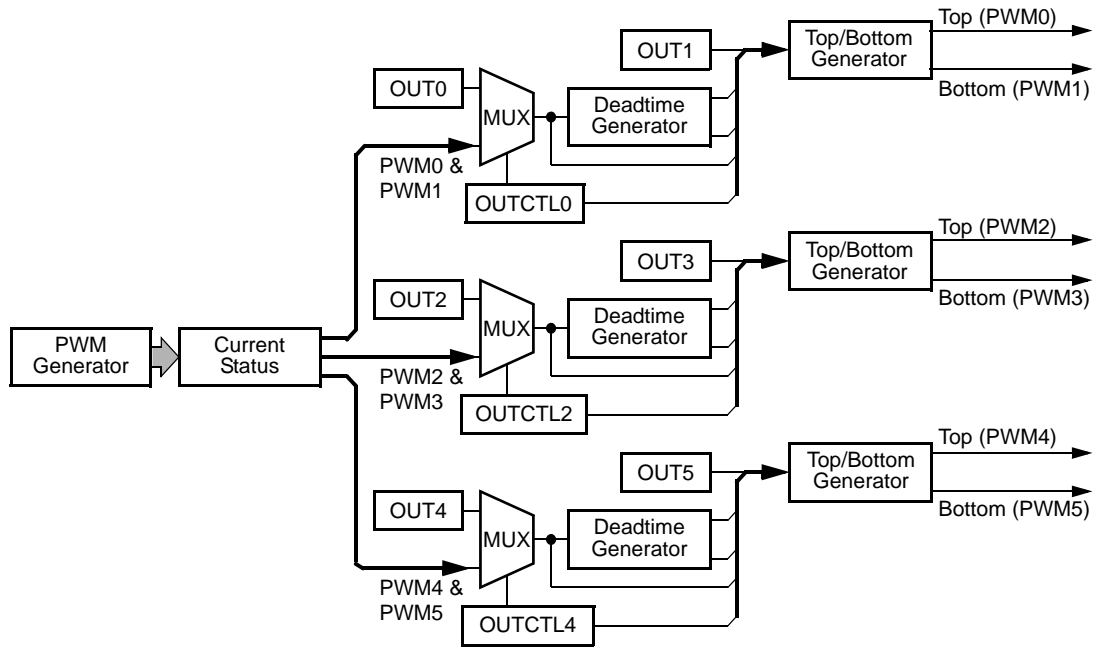


Figure 11-10. Deadtime Generators

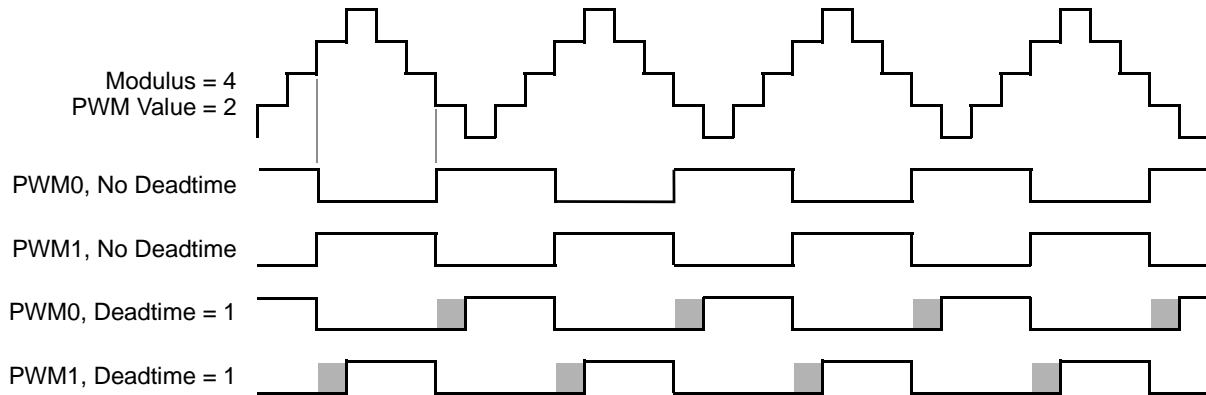
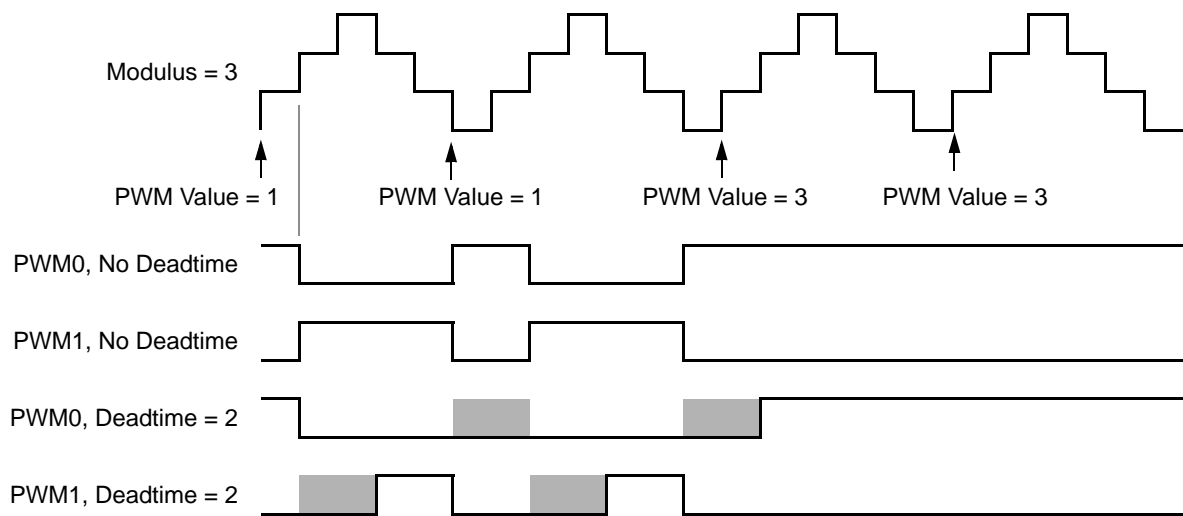
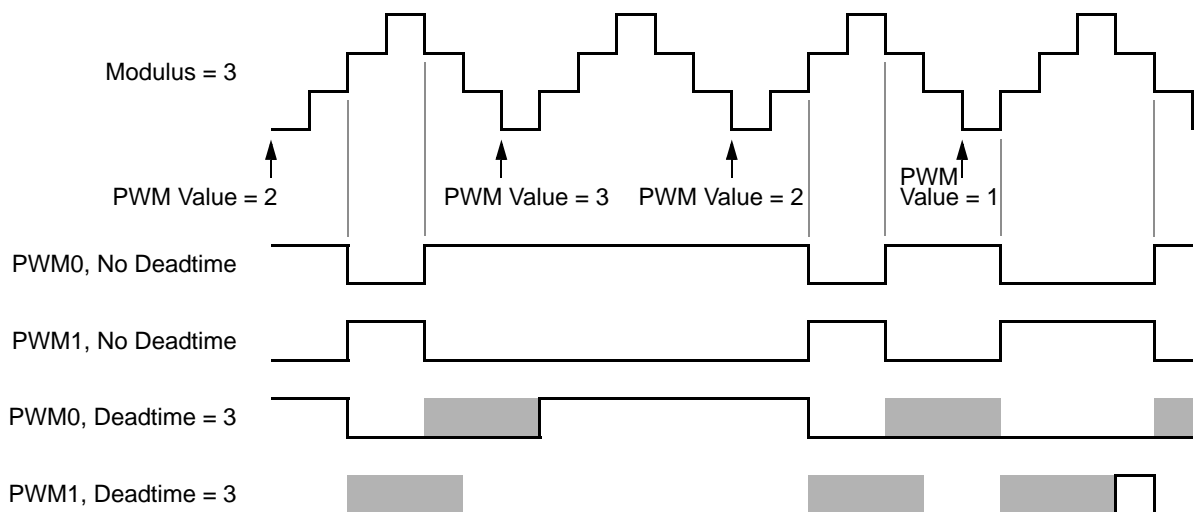


Figure 11-11. Deadtime Insertion, Center Alignment





**Figure 11-12. Deadtime at Duty Cycle Boundaries**



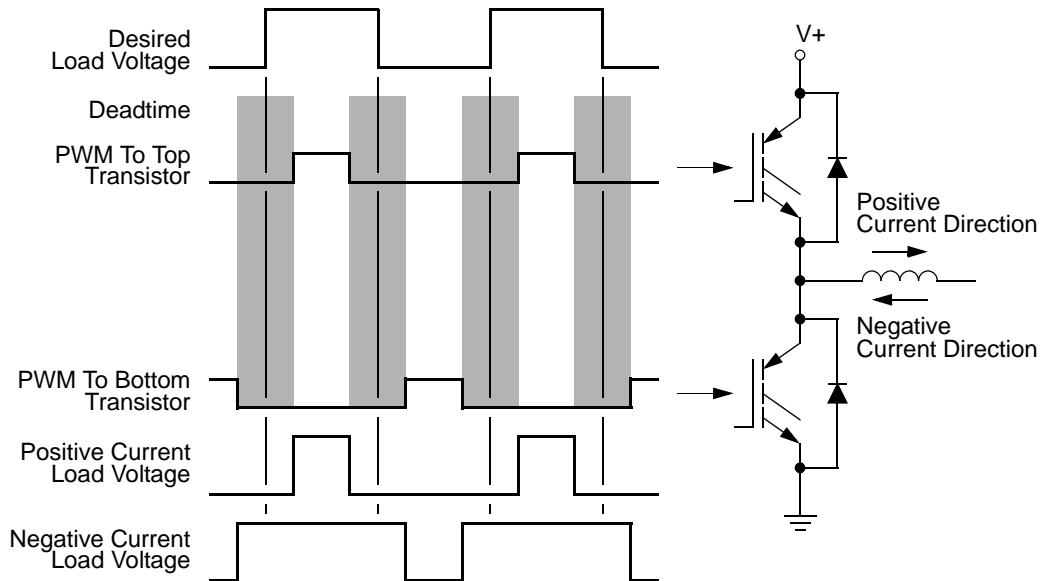
**Figure 11-13. Deadtime and Small Pulse Widths**

**Note:** The waveform at the output pin is delayed by two IPBus clock cycles for deadtime insertion.

#### 11.4.4.1 Top/Bottom Deadtime Correction

In the Complementary mode, either the top or the bottom transistor controls the output voltage. However, deadtime has to be inserted to avoid overlap of conducting interval between the top and

bottom transitions. Both transistors in Complementary mode are off during deadtime inserted, allowing the output voltage to be determined by the current status (direction) of load and introduce distortion in the output voltage. Please refer to [Figure 11-14](#). The distortion typically manifests itself as poor output waveforms with visible glitches and harmonics.



**Figure 11-14. Deadtime Distortion**

Load inductance distorts output voltage by keeping current flowing through the anti-body diode of transistor during deadtime. This deadtime current flow creates an output voltage varying with current direction. With a positive current flow, the output voltage during deadtime is equal to the bottom supply voltage, putting the top transistor in control. With a negative current flow, the output voltage during deadtime is equal to the top supply voltage, putting the bottom transistor in control. This results in the original pulse widths shortened by deadtime insertion, the averaged output will be less than desired value. However, when deadtime is inserted, it creates a distortion in load current waveform. This distortion is aggravated by dissimilar turn-on and turn-off delays of each of the transistors. By giving the PWM module information regarding which transistor is controlling at a given time, distortion can be corrected.

For a typical circuit in complementary channel operation, only one of the transistors will be effective in controlling the output voltage at any given time. This depends on the direction of the load current for that pair. Please see [Figure 11-14](#). To correct distortion one of two different factors must be added to the desired PWM value, depending on whether the top or bottom transistor is controlling the output voltage. Therefore, the software is responsible for calculating both compensated PWM values prior to placing them in an odd/even numbered PWM register pair. Either the odd or the even PWM Value (PWMVAL $n$ ) registers control the pulse width at

any given time. For a given PWM pair, whether the odd or even PWMVAL register is active depends on either:

- The state of the current status pin ( $IS_n$ ) for that driver
- The state of the odd/even correction bit ( $IPOL_n$ ) for that driver
- The direction of the PWM counter if ICC bits in the PM ICCR register are set (check data sheet to determine if this register is implemented)

To correct deadtime distortion, software can decrease or increase the value in the appropriate PWMVAL register.

- In Edge-Aligned operation, decreasing or increasing the PWM value by a correction value equal to the deadtime typically compensates for deadtime distortion.
- In Center-Aligned operation, decreasing or increasing the PWM value by a correction value equal to one-half the deadtime typically compensates for deadtime distortion.

In the complementary channel operation, the ISENS[1:0] bits in the PMCTL register select one of three correction methods:

- Manual deadtime correction
- Automatic deadtime correction during deadtime sampling the current status pins ( $IS_n$ )
- Automatic current status deadtime correction when the PWM counter value equals the value in the PWM counter modulus registers samples the current status pins ( $IS_n$ )

**Table 11-2. Correction Method Selection**

ISENS[1:0]	Correction Method	Comments
0n	Manual correction or no correction	—
10	Current status sampling on pins IS0, IS1, and IS2 during deadtime.	The polarity of the $IS_n$ pin is latched when both the top and bottom PWMs are off. At the 0% and 100% duty cycle boundaries, there is no deadtime, so no new current value is sensed.
11	Current status sampling on pins IS0, IS1, and IS2. At the half cycle in Center-Aligned operation At the end of the cycle in Edge-Aligned operation	Current is sensed even with 0%, or 100% duty cycle.

**Note:** Assumes the user will provide current status sensing circuitry causing the voltage at the corresponding input pin to be low for positive current and high for negative current. Additionally, it assumes the top PWMs are PWM 0, 2, and 4 while the bottom PWMs are PWM 1, 3, and 5.

### 11.4.4.2 Manual Deadtime Correction

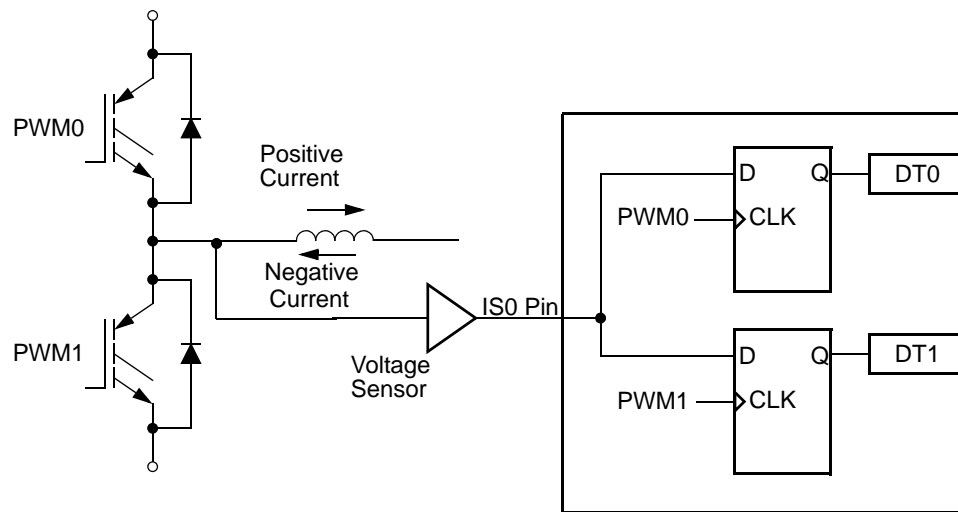
The IPOL0–IPOL2 bits in PWM Control (PMCTL) register select either the odd or the even PWM value registers to use in the next PWM cycle in Complementary mode when ISENS[1:0] = 0n.

**Table 11-3. Top/Bottom Manual Correction**

Bit	Logic State	Output Control
IPOL0	0	PWMVAL0 Controls PWM0/PWM1 Pair
	1	PWMVAL1 Controls PWM0/PWM1 Pair
IPOL1	0	PWMVAL2 Controls PWM2/PWM3 Pair
	1	PWMVAL3 Controls PWM2/PWM3 Pair
IPOL2	0	PWMVAL4 Controls PWM4/PWM5 Pair
	1	PWMVAL5 Controls PWM4/PWM5 Pair

**Note:** IPOL $n$  bits are buffered allowing only one PWM register to be used per PWM cycle. If an IPOL $n$  bit changes during a PWM period, the new value does not take effect until the next PWM period. IPOL $n$  bits take effect at the end of each PWM cycle regardless of the state of the LOAD OKAY (LDOK) bit.

To detect the current status, the voltage on each IS $n$  pin is sampled at the end of each deadtime. The value is stored in the DT $n$  bits in the fault acknowledge register. The DT $n$  bits are a timing marker especially indicating when to toggle between PWM value registers. Software can then set the IPOL $n$  bit to toggle PWMVAL registers according to DT $n$  values.

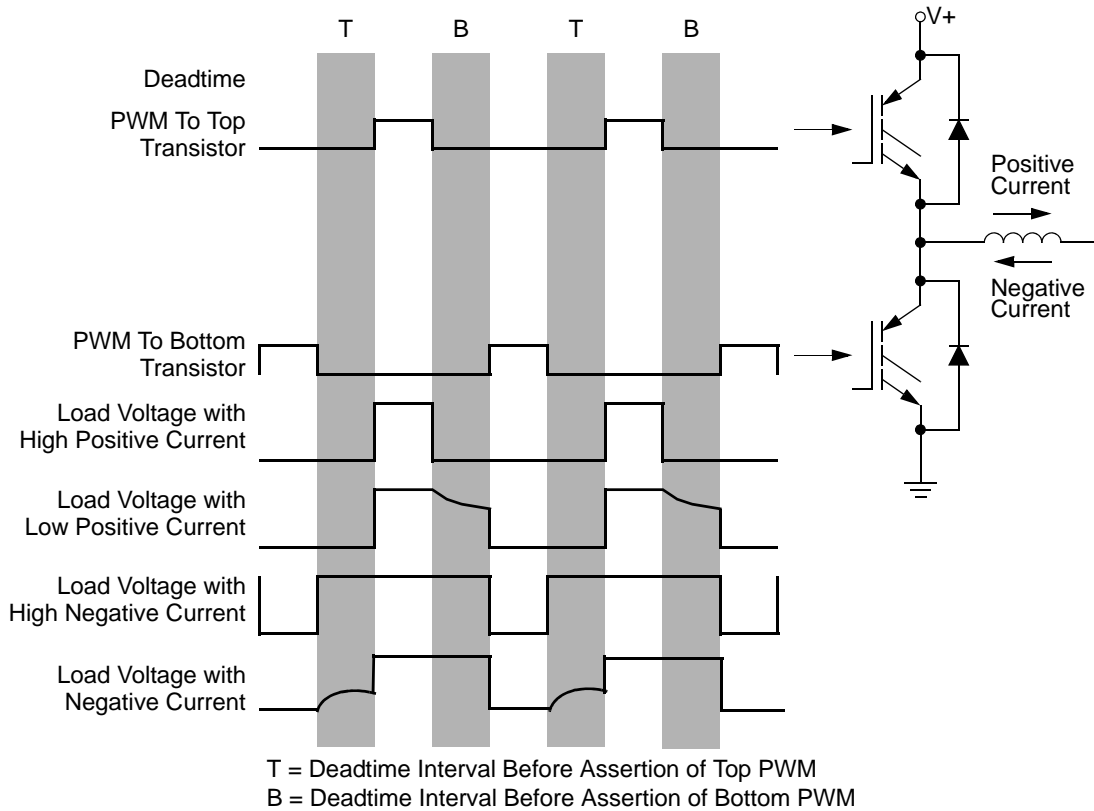


**Figure 11-15. Current Status Sense Scheme for Deadtime Correction**

Both D flip-flops latch *low* ( $DT0 = 0$  and  $DT1 = 0$ ) during deadtime periods if the current is large and flowing out of the complementary circuit. Please refer to [Figure 11-15](#).

Both D flip-flops latch the *high* ( $DT0 = 1$ ,  $DT1 = 1$ ) during deadtime periods if current is large and flowing into the complementary circuit. Please see [Figure 11-15](#).

However, under low-current, the output voltage of the complementary circuit during deadtime is somewhere between the high and low levels. The current cannot free-wheel throughout the opposition anti-body diode, regardless of polarity, giving additional distortion when the current crosses zero. Sampled results will be  $DT0 = 0$  and  $DT1 = 1$ . Thus, the best time to change one PWM value register to another is just before the current zero crossing. Please refer to [Figure 11-16](#).



**Figure 11-16. Output Voltage Waveforms**

### 11.4.5 Automatic Deadtime Correction

A Current Status pin,  $IS_n$ , for a PWM pair, selects either the odd or the even PWM value registers to use in the next PWM cycle. The selection is based on user-provided current sense circuitry driving the  $IS_n$  pin high for negative current and low for positive current.

**Table 11-4. Top/Bottom Automatic Correction**

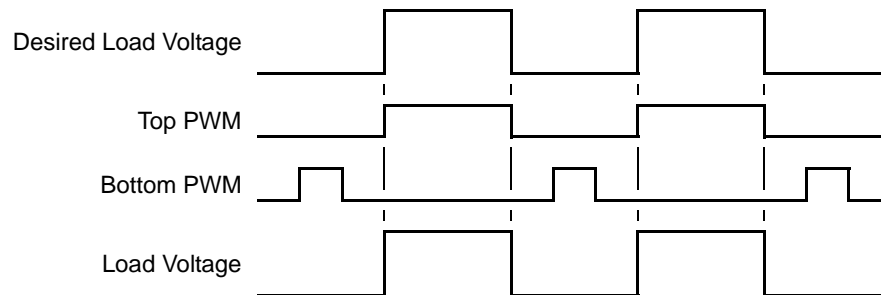
Pin	Logic State	Output Control
$\overline{IS0}$	0	PWMVAL0 Controls PWM0/PWM1 Pair
	1	PWMVAL1 Controls PWM0/PWM1 Pair
$\overline{IS1}$	0	PWMVAL2 Controls PWM2/PWM3 Pair
	1	PWMVAL3 Controls PWM2/PWM3 Pair
$\overline{IS2}$	0	PWMVAL4 Controls PWM4/PWM5 Pair
	1	PWMVAL5 Controls PWM4/PWM5 Pair

The first automatic deadtime correction mode, where  $ISENS[1:0] = 10$ , is accomplished by sampling the current status pin ( $IS_n$ ) during deadtime. No samples can be taken at the 100 percent and zero percent duty cycle boundaries because deadtime does not exist.

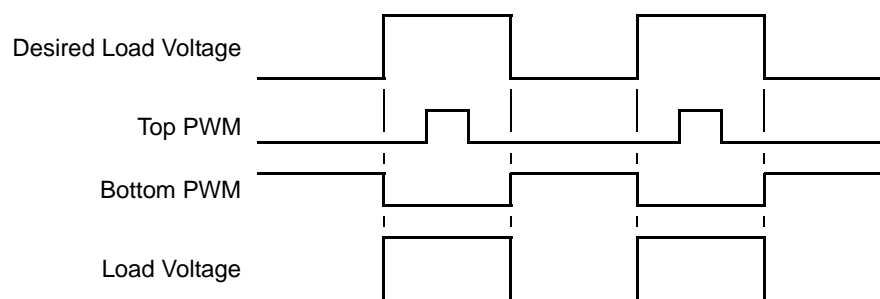
The second automatic deadtime correction mode, where  $ISENS[1:0] = 11$ , is accomplished by sampling the current status pin ( $IS_n$ ) at the half cycle during Center-Aligned operation and at the end of the cycle during Edge-Aligned operation.

**Note:** Values latched on the  $IS_n$  pins are buffered so only one PWM register is used per PWM cycle. If a current status changes during a PWM period, the new value does not take effect until the next PWM period.

When initially enabled by setting the PWMEN bit, no current status has previously been sampled. Even PWM value registers initially control the three PWM pairs when configured for current status correction.



**Figure 11-17. Correction with Positive Current**



**Figure 11-18. Correction with Negative Current**

### 11.4.6 Asymmetric PWM Output

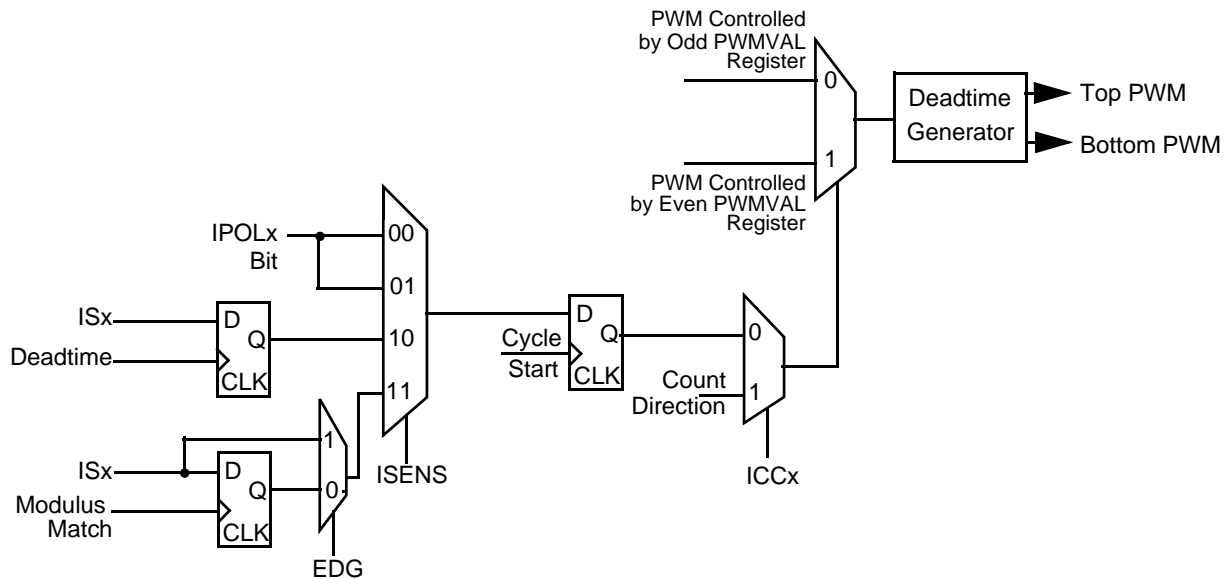
In Complementary mode with Center-Align operation, the PWM duty cycle is able to change alternatively at every half cycle. The count direction of the PWM counter selects either the odd or

the even PWM value registers to use in the PWM cycle. For counting up, select even PWM value registers to use in the PWM cycle. For counting down, select odd PWM value registers to use in the PWM cycle.

**Table 11-5. Top/Bottom Corrections Selected by ICC<sub>n</sub> Bits**

Bit	Logic State	Output Control
ICC0	0	ISENS[1:0] Controls PWM0/PWM1 Pair
	1	PWM Count Direction Controls PWM0/PWM1 Pair
ICC1	0	ISENS[1:0] Controls PWM2/PWM3 Pair
	1	PWM Count Direction Controls PWM2/PWM3 Pair
ICC2	0	ISENS[1:0] Controls PWM4/PWM5 Pair
	1	PWM Count Direction Controls PWM4/PWM5 Pair

**Note:** If an ICC<sub>n</sub> bit in PMICCR register changes during a PWM period, the new value does not take effect until the next PWM period so ICC<sub>n</sub> bits take effect at the end of each PWM cycle regardless of the state of the LOAD OKAY (LDOK) bit.



**Figure 11-19. Correction Logic**

### 11.4.7 Output Polarity

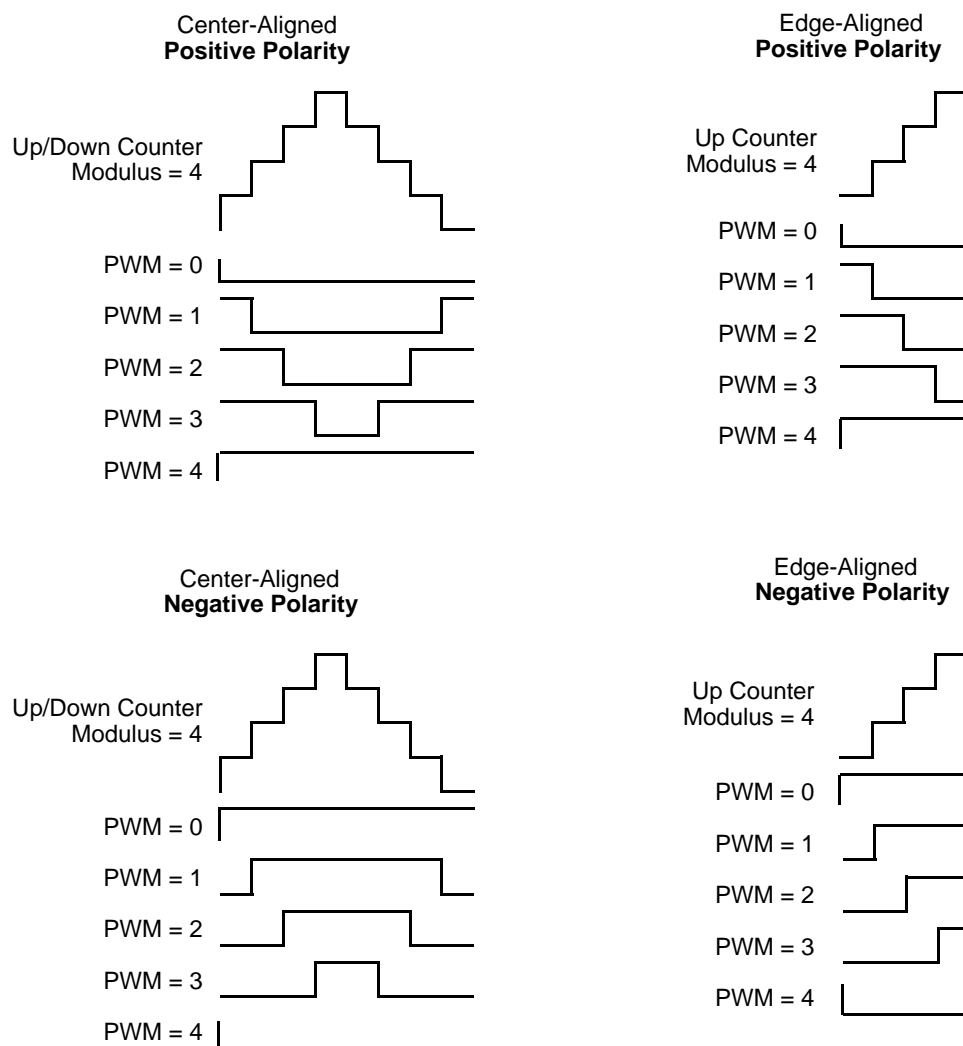
*Positive* polarity means when the PWM is active its output is high. Conversely, *negative* polarity means when the PWM is active its output is low.



Output polarity of the PWMs is determined by two options:

1. TOPNEG controls the polarity of PWM0, PWM2 and PWM4 outputs, which typically drive the top transistors of the pair. When TOPNEG is set these outputs are active-low.
2. BOTNEG controls the polarity of PWM1, PWM3 and PWM5 outputs, which typically drive the bottom transistors of the pair. When BOTNEG is set these outputs are active-low.

Both TOPNEG and BOTNEG bits are in the Configure register (PMCFG). Please see [Figure 11-20](#).



**Figure 11-20. PWM Polarity**

## 11.5 Software Output Control

Setting Output Control Enable (OUTCTRL $n$ ) bit, the PWM outputs is driven by software rather than the PWM generator.

In an Independent mode, with OUTCTRL $n$  =1, the output bit OUT $n$ , controls the PWM $n$  channel. Setting and clearing the OUT $n$  bit activates and deactivates the corresponding PWM channel.

The OUTCTRL $n$  and OUT $n$  bits are in the PWM Output Control (PMOUT) register.

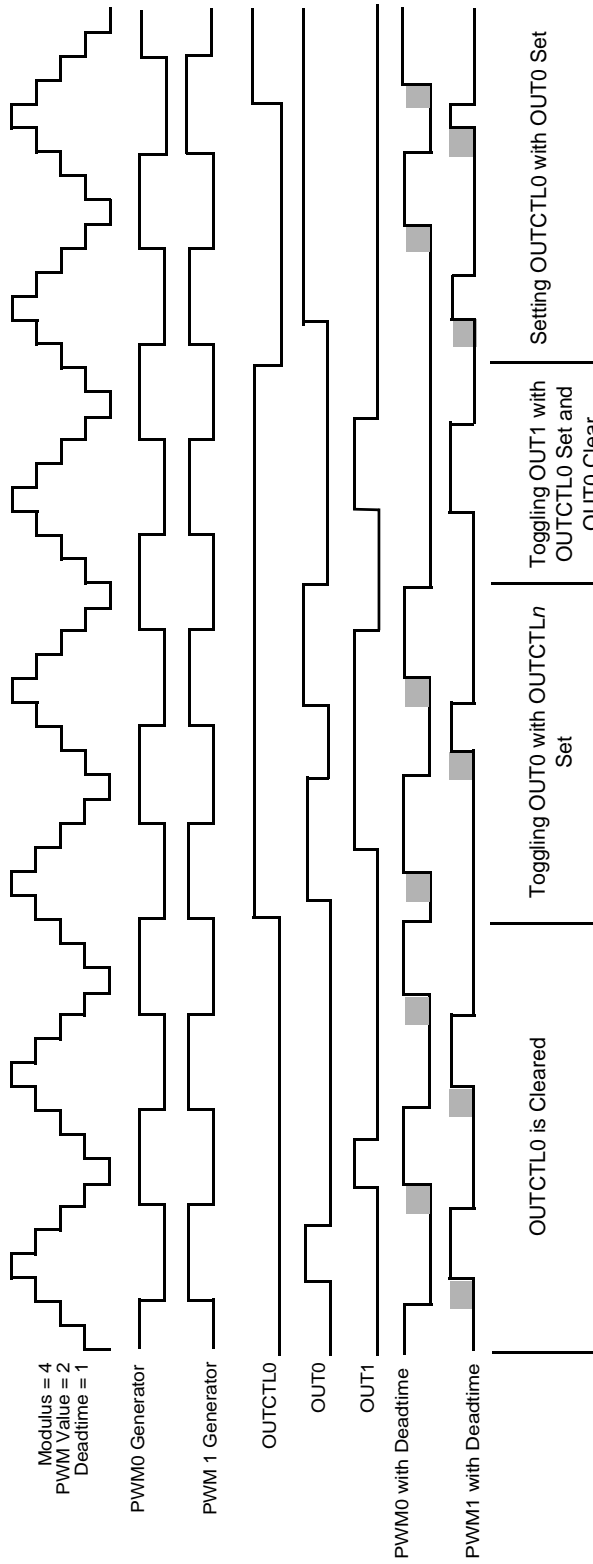
During software output control, TOPNEG and BOTNEG still control output polarity.

In complementary channel operation odd and even OUTCTRL $n$  must be switched concurrently for proper operation, the even-numbered OUT $n$  bits replace the PWM generator outputs. Complementary channel pairs cannot be active simultaneously, and the deadtime generators continue to insert deadtime whenever an even OUT $n$  bit toggles. Deadtime is not inserted when the odd OUT $n$  bit toggles. The even OUT $n$  bit controls complementary channel pairs when the odd OUT $n$  bit is set. However, the even OUT $n$  bit still controls complementary channel pairs with odd PWM $n$  deactivated if the odd OUT $n$  bit is cleared. In other words, setting the odd OUT $n$  bit makes its corresponding PWM $n$  the complement of its even pair, while clearing the odd OUT $n$  bit deactivates the odd PWM $n$ . Please refer to [Figure 11-21](#).

Setting the OUTCTRL $n$  bits do not disable the PWM generators and current status sensing circuitry. They continue to run, but no longer control the output pins. When the OUTCTRL $n$  bits are cleared, the outputs of the PWM generator takes control of PWM outputs at the beginning of the next PWM cycle. Please refer to [Figure 11-21](#).

Software can drive the PWM outputs even when PWM Enable (PWMEN) bit is set to zero.

**Note:** Avoid an unexpected deadtime insertion by clearing the OUT $n$  bits before setting and after clearing the OUTCTRL $n$  bits.



**Figure 11-21. Software Output Control in Complementary Mode**

## 11.6 PWM Generator Loading

### 11.6.1 Load Enable

The Load Okay (LDOK) bit enables loading the PWM generator with:

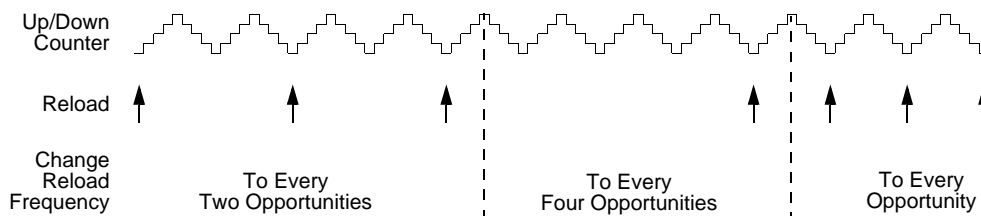
- A prescaler divisor from the PRSC1 and PRSC0 bits in PWM Control register
- A PWM period from the PWM Counter Modulus registers
- A PWM pulse width from the all PWM Value registers

LDOK ensures reloading of these PWM parameters simultaneously. Setting LDOK allows the prescale bits, PWMCM and PWMVAL<sub>n</sub> registers to be loaded into a set of buffers. The loaded buffers are used by the PWM generator at the beginning of the next PWM reload cycle. Set LDOK by reading it, then writing a Logic 1 to it. After loading, LDOK is automatically cleared.

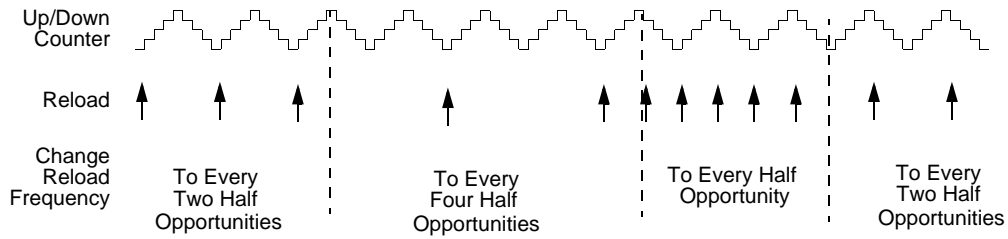
### 11.6.2 Load Frequency

The LDFQ3, LDFQ2, LDFQ1, and LDFQ0 bits in the PMCTL register select an integral loading frequency of one to 16-PWM reload opportunities. The LDFQ bits take effect at every PWM reload opportunity, regardless the state of the LDOK bit. The *half* bit in the PMCTL register controls half cycle reloads for Center-Aligned PWMs. If the *half* bit is set, a reload opportunity occurs at both beginning of PWM cycle or PWM half cycle. If the half bit is not set, a reload opportunity occurs only at the beginning of the cycle. Reload opportunities can only occur at the beginning of a PWM cycle in Edge-Aligned mode.

**Note:** Loading a new modulus on a half cycle will force the count to the new modulus value *minus one* on the next clock cycle. Half cycle reloads only changes reload rate in Center-Aligned mode. Enabling or disabling half cycle reloads in Edge-Aligned mode will have no effect on the reload rate.



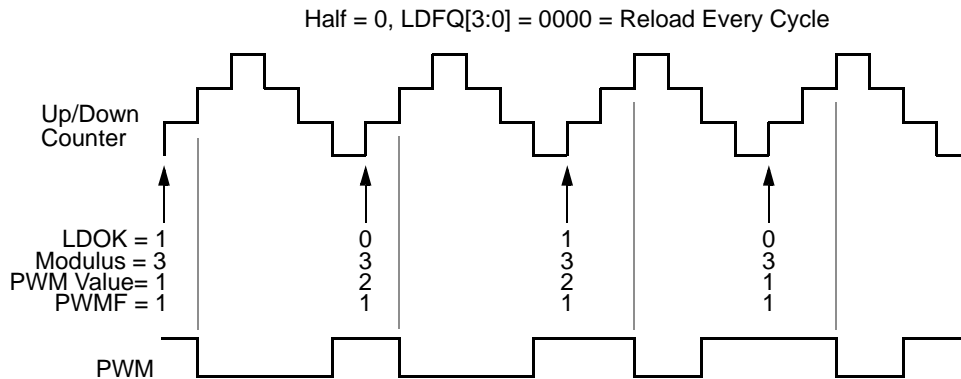
**Figure 11-22. Full Cycle Reload Frequency Change**



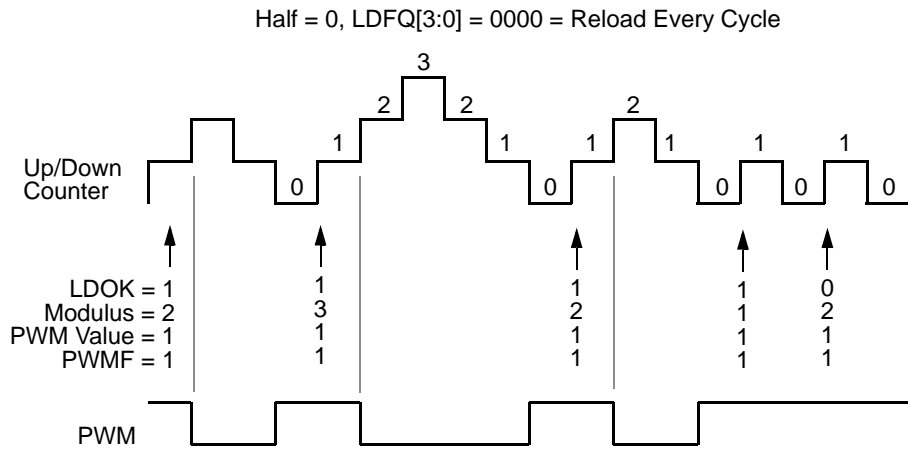
**Figure 11-23. Half Cycle Reload Frequency Change**

### 11.6.3 Reload Flag

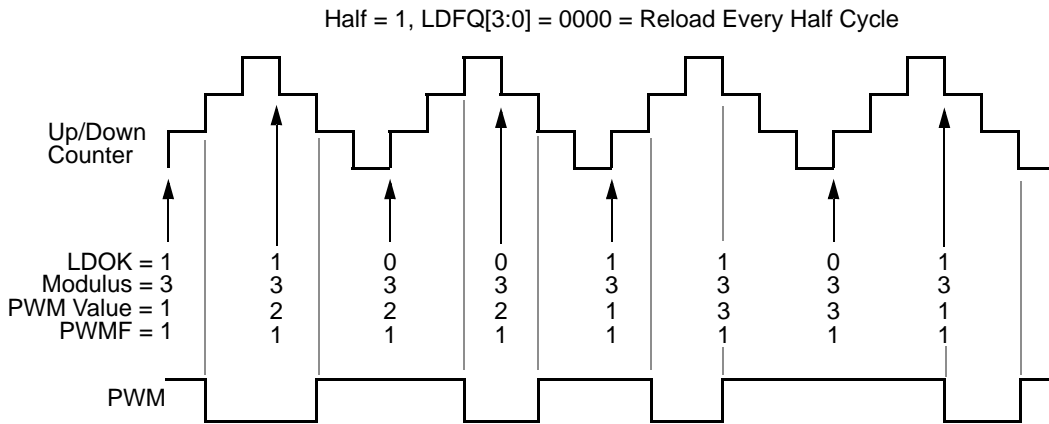
At every reload opportunity the PWM Reload Flag (PWMF) bit in the PMCTL register is set. Setting PWMF happens even if an actual reload is prevented by the LDOK bit. If the PWM Reload Interrupt Enable (PWMRIE) bit is set, the PWMF generates core interrupt requests allowing software to calculate new PWM parameters in real time. When PWMRIE is not set, reloads still occur at the selected reload rate without generating interrupt requests. Clear PWMF by reading it then write a Logic 0 to it.



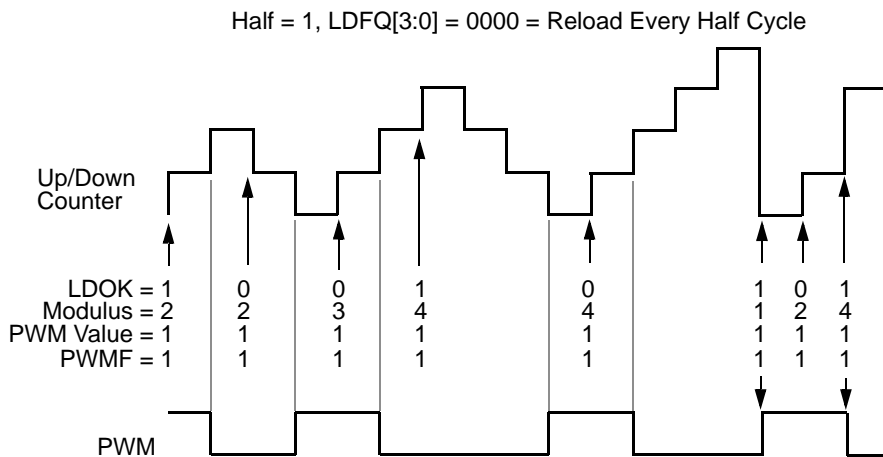
**Figure 11-24. Full Cycle Center-Aligned PWM Value Loading**



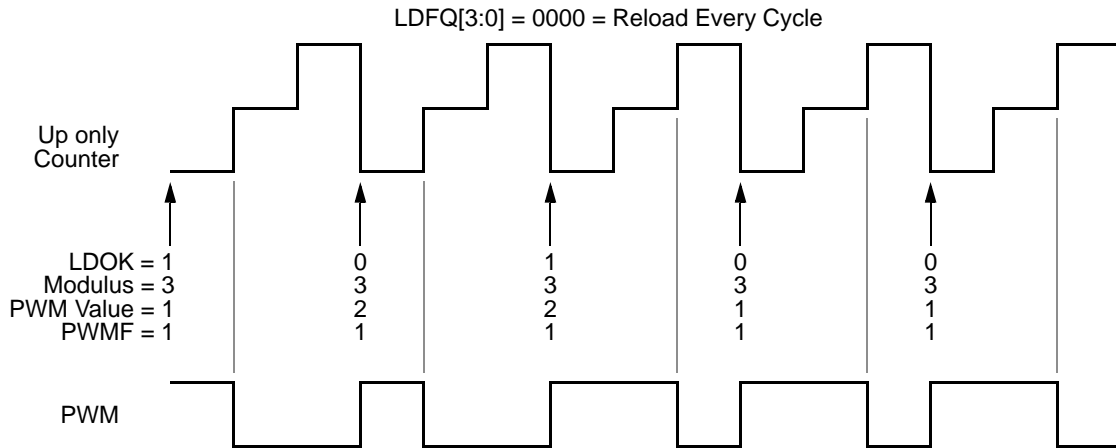
**Figure 11-25. Full Cycle Center-Aligned Modulus Loading**



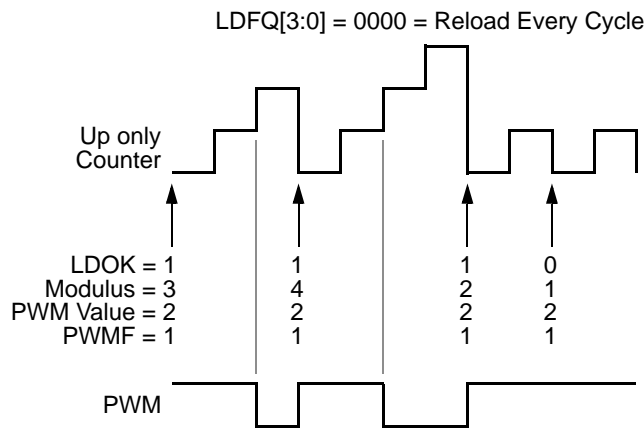
**Figure 11-26. Half Cycle Center-Aligned PWM Value Loading**



**Figure 11-27. Half Cycle Center-Aligned Modulus Loading**



**Figure 11-28. Edge-Aligned PWM Value Loading**



**Figure 11-29. Edge-Aligned Modulus Loading**

### 11.6.4 Synchronization Output

The PWM outputs a synchronization pulse connected as an input to the synchronization module, Timer C. A high-true pulse occurs for each reload of the PWM regardless of the state of the LDOK bit. When half cycle reloads are enabled, HALF = 1 in the PMCTL register, the pulse can occur on the half cycle.

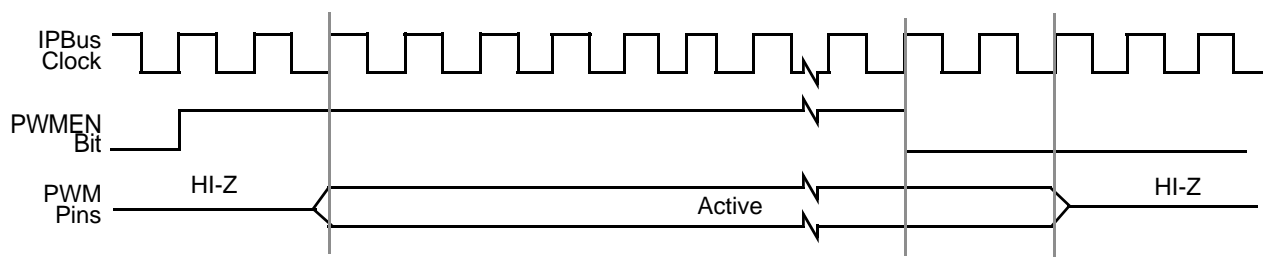
### 11.6.5 Initialization

Initialize all registers and set the LOAD OKAY (LDOK) bit before setting the ENABLE (PWMEN) bit. With LDOK set, when the PWMEN bit is first set, a reload will immediately occur, thereby setting the PWMF bit. The PWMF bit generates an interrupt request if the

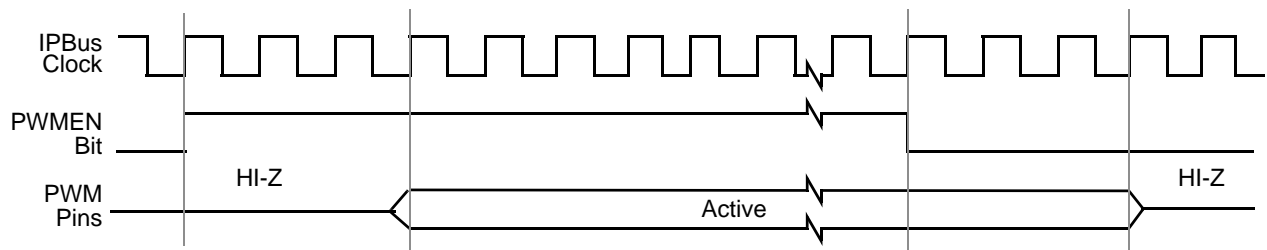
PWMRIE bit is set. In complementary channel operation with current status correction selected, even numbered PWM Value registers control the outputs for the first PWM cycle.

**Note:** Even if LDOK is not set, setting PWMEN also sets the PWMF. To prevent a core interrupt request, clear the PWMRIE bit before setting PWMEN.

Setting PWMEN for the first time after reset without first setting LDOK loads a prescaler divisor of one, a PWM value of \$0000, and an unknown modulus. If the LDOK bit is not set after the PWMEN bit is cleared, then set (without a RESET) the value last loaded will be used in the PWM generated. If deadtime register is changed after PWMEN or OUTCTLn bit are set, an improper deadtime insertion could occur.



**Figure 11-30. PWMEN and PWM Pins in Independent Operation (OUTCTL0–5 = 0)**



**Figure 11-31. PWMEN & PWM Pins in Complement Operation (OUTCTL0,2,4 = 0)**

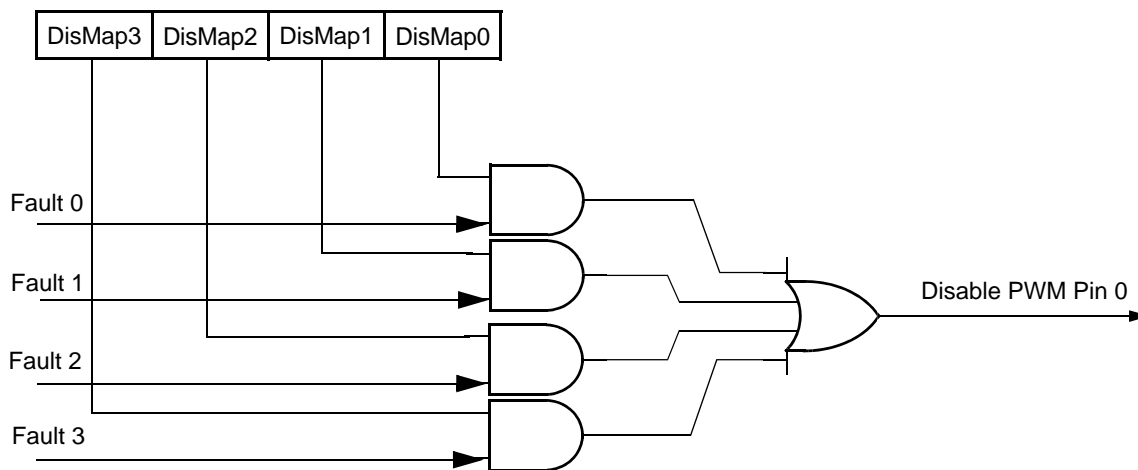
When the PWMEN bit is cleared:

- The PWMn pins will be in their inactive status unless OUTCTLn = 1
- The PWM counter is cleared and does not count
- The PWM generator forces its outputs to zero
- The PWMF and pending interrupt requests are not cleared
- All fault circuitry remains active
- Software output control remains active if OUTCTLn = 1
- Deadtime insertion continues during software output control



## 11.7 Fault Protection

Fault protection can disable any combination of PWM pins. Faults are generated by a Logic 1 on any of the FAULT pins. Each FAULT pin can be mapped arbitrarily to any of the PWM pins. When fault protection hardware disables PWM pins, the PWM generator continues to run, only the output pins are deactivated. The fault decoder disables PWM pins selected by the fault logic and the disable mapping register. Please see [Figure 11-32](#). Each bank of four bits in the disable mapping register control the mapping for a single PWM pin. Please refer to [Table 11-6](#). The fault protection is enabled even when the PWM is not enabled; therefore, if a fault is latched in, it must be cleared prior to enabling the PWM to prevent an unexpected interrupt. Please see [Section 11.10.3.4](#).



**Figure 11-32. Fault Decoder for PWM 0**

**Table 11-6. Fault Mapping**

PWM Pin	Controlling Register Bits
PWM0	DISMAP3–DISMAP0
PWM1	DISMAP7–DISMAP4
PWM2	DISMAP11–DISMAP8
PWM3	DISMAP15–DISMAP12
PWM4	DISMAP19–DISMAP16
PWM5	DISMAP23–DISMAP20

**Note:** For parts with less than four fault pins, the same controls apply. The unavailable DISMAP field bits should be set to zero. For example, if Fault3 is not available as an input, set DISMAP3 = 0.

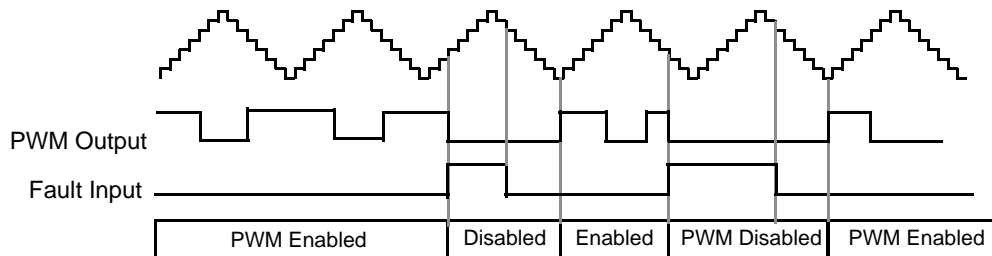
### 11.7.1 Fault Pin Filter

Each fault pin has a filter to test for fault conditions. A fault input transition to a high state is not declared until the input is sampled high on two consecutive IPBus clocks. Only then FFLAG $n$  and FPIN $n$  are set. The FPIN $n$  bit will remain set until the Fault input is detected low on two consecutive IPBus clocks. Clear FFLAG $n$  by writing a Logic 1 to the corresponding fault acknowledge bit, FTACK $n$ . If the FIEN $n$ , FAULT $n$  pin interrupt enable bit is set, the FFLAG $n$  flag generates an interrupt request. The interrupt request latch remains set until one of the following actions occur:

- Software clears the FFLAG $n$  flag by writing a Logic 1 to the FTACK $n$  bit
- Software clears the FIEN $n$  bit by writing a Logic 0 to it
- A reset occurs

### 11.7.2 Automatic Fault Clearing

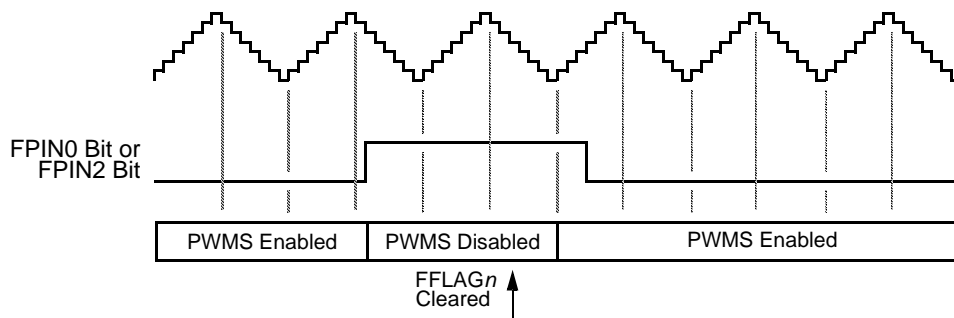
In Automatic mode, when FMODE $n$  is set, disabled PWM pins are enabled when the FAULT $n$  pin returns to Logic 0 and a new PWM half cycle begins. Please refer to [Figure 11-33](#). Clearing the FFLAG $n$  flag does not affect disabled PWM pins when FMODE $n$  is set.



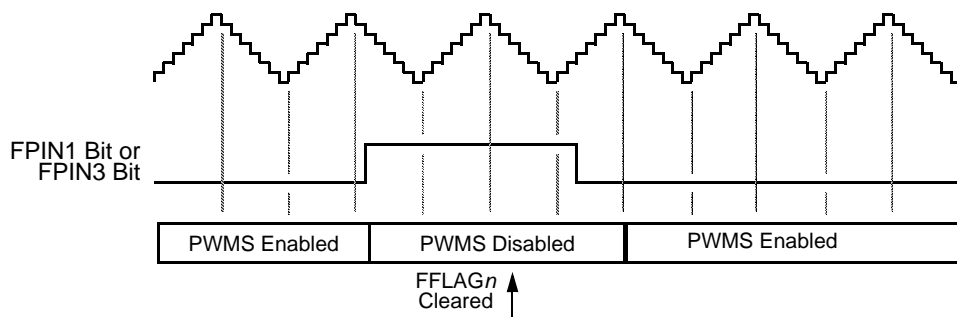
**Figure 11-33. Automatic Fault Clearing**

### 11.7.3 Manual Fault Clearing

In Manual mode, the fault pins are grouped in pairs, each pair sharing common functionality. A fault condition on Fault pins 0 and 2 can be cleared by software clearing the corresponding FFLAG bit, allowing the PWM(s) to enable at the next PWM half cycle regardless of the logic level at the Fault pin. [Figure 11-34](#). A fault condition on Fault pins 1 and 3 can only be cleared by software clearing corresponding FFLAG $n$  bit, allowing the PWM(s) to enable if a Logic Low at the Fault pin is detected at the start of the next PWM half cycle boundary. Please see [Figure 11-35](#).



**Figure 11-34. Manual Fault Clearing (Example 1)**



**Figure 11-35. Manual Fault Clearing (Example 2)**

**Note:** PWM half cycle boundaries occur at both the PWM cycle start and when the counter equals the modulus, so in Edge-Aligned operation full cycles and half cycles are equal.

**Note:** Fault protection also applies during software output control when the  $OUTCTLn$  bits are set. Fault clearing still occurs at half PWM cycle boundaries while the PWM generator is engaged where  $PWMEN$  equals one. But the  $OUTn$  bits can also control the PWM pins while the PWM generator is off where  $PWMEN$  equals zero. Thus, fault clearing occurs at IPBus cycles while the PWM generator is off and at the start of PWM cycles when the generator is engaged.

## 11.8 Operating Modes

Exercise care when using this module in certain chip operating modes. Some applications require regular software updates for proper operation. Failure to use caution could result in damaging the circuit. Because of this, PWM outputs are placed in their inactive states in Stop mode, and optionally under Wait and EOnCE modes. PWM outputs will be reactivated (assuming they were active to begin with) when these modes are exited.

**Table 11-7. Modes When PWM Operation is Restricted**

Mode	Description
STOP	PWM Outputs are Disabled
WAIT	PWM Outputs are Disabled as a Function of the PMCFG WAIT_EN Bit
EOnCE	PWM Outputs are Disabled as a Function of the PMCFG DBG_EN Bit

## 11.9 Pin Descriptions

The Pulse Width Modulator (PWM) is capable of having the following external pins.

### 11.9.1 PWM0–PWM5 Pins—(PWM0–5)

PWM0–PWM5 are output pins of the six PWM channels.

### 11.9.2 FAULT0–FAULT3 Pins—(FAULT0–3)

FAULT0–FAULT3 are input pins for disabling selected PWM outputs.

### 11.9.3 IS2 Pins—(IS0–2)

IS0–IS2 are current status pins for Top/Bottom pulse width correction in complementary channel operation while deadtime is asserted.

## 11.10 Register Definitions

**Table 11-8. PWM Memory Map**

Devices	Peripheral	Address
8100/8300	PWMA_BASE	\$00F140
	PWMB_BASE	\$00F160

The address of a register is the sum of a base address and an address offset. The base address is defined at the core level and the address offset is defined at the module level. PWMA uses PWMA\_BASE plus the given offset while PWMB uses PWMB\_BASE plus the given offset in the table below.

**Table 11-9. PWM Register Summary**

Address + Offset	Address Acronym	Register Name	Access Type	Chapter Location
Base + \$0	PMCTL	Control Register	Read/Write	<a href="#">Section 11.10.1</a>
Base + \$1	PMFCTL	Fault Control Register	Read/Write	<a href="#">Section 11.10.2</a>
Base + \$2	PMFSA	Fault Status & Acknowledge Register	Read/Write	<a href="#">Section 11.10.3</a>
Base + \$3	PMOUT	Output Control Register	Read/Write	<a href="#">Section 11.10.4</a>
Base + \$4	PMCNT	Counter Register	Read-Only	<a href="#">Section 11.10.5</a>
Base + \$5	PWMCM	Counter Modulo Register	Read/Write	<a href="#">Section 11.10.6</a>
Base + \$6 to \$B	PWMVAL0-5	Value Register 0-5	Read/Write	<a href="#">Section 11.10.7</a>
Base + \$C	PMDEADTM	Deadtime Register	Read/Write	<a href="#">Section 11.10.8</a>
Base + \$D	PMDISMAP1	Disable Mapping Register 1	Read/Write	<a href="#">Section 11.10.9</a>
Base + \$E	PMDISMAP2	Disable Mapping Register 2	Read/Write	
Base + \$F	PMCFG	Configure Register	Read/Write	<a href="#">Section 11.10.10</a>
Base + \$10	PMCCR	Channel Control Register	Read/Write	<a href="#">Section 11.10.11</a>
Base + \$11	PMPORT	Port Register	Read/Write	<a href="#">Section 11.10.12</a>
Base + \$12	PMICCR	Internal Correction Control Register	Read/Write	<a href="#">Section 11.10.13</a>

Bit fields of each of the 19 registers are illustrated in [Figure 11-36](#). Details of each follow.

Addr. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$0	PMCTL	R	LDFQ				HALF	IPOL2	IPOL1	IPOL0	PRSC		PWMRIE	PWMF	ISENS		LDOK	PWMEN
\$1	PMFCTL	R	0	0	0	0	0	0	0	0	FIE3	FMODE3	FIE2	FMODE2	FIE1	FMODE1	FIE0	FMODE0
\$2	PMFSA	R	FPIN3	FFLAG3	FPIN2	FFLAG2	FPIN1	FFLAG1	FPIN0	FFLAG0	0	0	DT5	DT4	DT3	DT2	DT1	DT0
\$3	PMOUT	R	PAD_EN	0	OUT CTL5	OUT CTL4	OUT CTL3	OUT CTL2	OUT CTL1	OUT CTL0	0	0	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
\$4	PMCNT	R	0	CNT														
\$5	PWMCM	R	0	CM														
\$6-\$B	PWMVAL0-5	R	VAL															
\$C	PMDEADTM	R	0	0	0	0	PWMDT											
\$D	PMDISMAP1	R	DISMAP[15:0]															
\$E	PMDISMAP2	R	0	0	0	0	0	0	0	0	DISMAP[23:16]							
\$F	PMCFG	R	0	DBG_EN	WAIT_EN	EDG	0	TOP NEG 45	TOP NEG 23	TOP NEG 01	0	BOT NEG 45	BOT NEG 23	BOT NEG 01	INDEP 45	INDEP 23	INDEP 01	WP
\$10	PMCCR	R	ENHA	nBX	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	0	0	VLMODE		0	SWP 45	SWP 23	SWP 01
\$11	PMPORT	R	0	0	0	0	0	0	0	0	0	IS2	IS1	IS0	FAULT 3	FAULT 2	FAULT 1	FAULT 0
\$12	PMICCR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	ICC2	ICC1	ICC0

R 0 Read as 0  
W Reserved

Figure 11-36. PWM Register Map Summary

### 11.10.1 PWM Control Register (PMCTL)

Base + \$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LDFQ				HALF	IPOL2	IPOL1	IPOL0	PRSC		PWMRIE	PWMF	ISENS		LDOK	PWMEN
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-37. PWM Control Register (PMCTL)

### 11.10.1.1 Load Frequency (LDFQ)—Bits 15–12

These buffered read/write bits select the PWM load frequency according to [Table 11-10](#). Reset clears the LDFQ bits, selecting loading every PWM opportunity. The occurrence of a PWM opportunity is determined by the half bit.

**Note:** The LDFQ $n$  bits take effect when the current load cycle is complete, regardless of the state of the Load Okay (LDOK) bit. Reading the LDFQ $n$  bits reads the buffered values and not necessarily the values currently in effect.

**Table 11-10. PWM Reload Frequency**

LDFQ	PWM Reload Frequency	LDFQ	PWM Reload Frequency
0000	Every PWM Opportunity	1000	Every 9 PWM Opportunities
0001	Every 2 PWM Opportunities	1001	Every 10 PWM Opportunities
0010	Every 3 PWM Opportunities	1010	Every 11 PWM Opportunities
0011	Every 4 PWM Opportunities	1011	Every 12 PWM Opportunities
0100	Every 5 PWM Opportunities	1100	Every 13 PWM Opportunities
0101	Every 6 PWM Opportunities	1101	Every 14 PWM Opportunities
0110	Every 7 PWM Opportunities	1110	Every 15 PWM Opportunities
0111	Every 8 PWM Opportunities	1111	Every 16 PWM Opportunities

### 11.10.1.2 Half Cycle Reload (HALF)—Bit 11

This read/write bit enables half cycle reloads in Center-Aligned PWM mode. This bit has no effect on Edge-Aligned PWMs.

- 0 = Half cycle reloads disabled
- 1 = Half cycle reloads enabled

### 11.10.1.3 Current Polarity 2 (IPOL2)—Bit 10

This buffered read/write bit selects the PWM value register for the PWM4 and PWM5 pins in top/bottom manual deadtime correction.

- 0 = PWM value register four in next PWM cycle
- 1 = PWM value register five in next PWM cycle

**Note:** The IPOL $n$  bits take effect at the beginning of the next load cycle regardless of the state of the LDOK bit. Select top/bottom software correction by writing 0 $n$  to the current status bits, ISENS, in the PWM Control register. Reading the IPOL $n$  bits reads the buffered values and not necessarily the values currently in effect.

#### 11.10.1.4 Current Polarity 1 (IPOL1)—Bit 9

This buffered read/write bit selects the PWM value register for the PWM2 and PWM3 pins in top/bottom manual deadtime correction.

- 0 = PWM value register two in next PWM cycle
- 1 = PWM value register three in next PWM cycle

#### 11.10.1.5 Current Polarity 0 (IPOL0)—Bit 8

This buffered read/write bit selects the PWM value register for the PWM0 and PWM1 pins in top/bottom manual deadtime correction.

- 0 = PWM value register zero in next PWM cycle
- 1 = PWM value register one in next PWM cycle

#### 11.10.1.6 Prescaler (PRSC)—Bits 7–6

These buffered read/write bits select the PWM clock frequency illustrated in [Table 11-11](#).

**Table 11-11. PWM Prescaler**

PRSC	PWM Clock Frequency
00	$f_{IPBus}$
01	$f_{IPBus}/2$
10	$f_{IPBus}/4$
11	$f_{IPBus}/8$

**Note:** Reading the  $PRSC_n$  bits reads the buffered values, and not necessarily the values currently in effect. The  $PRSC_n$  bits take effect at the beginning of the next PWM cycle and only when the LDOK bit is set.

#### 11.10.1.7 PWM Reload Interrupt Enable (PWMRIE)—Bit 5

This read/write bit enables the PWMF flag to generate interrupt requests.

- 0 = PWMF interrupt request disabled
- 1 = PWMF interrupt request enabled

#### 11.10.1.8 PWM Reload Flag (PWMF)—Bit 4

This read/write flag is set at the beginning of every reload cycle regardless of the state of the LDOK bit. Clear PWMF by reading it, then write a Logic 0 to it. If another reload occurs before the clearing sequence is complete, writing Logic 0 to PWMF has no effect.



- 0 = No new reload cycle since last PWMF clearing
- 1 = New reload cycle since last PWMF clearing

**Note:** Clearing PWMF satisfies pending PWMF interrupt request.

### 11.10.1.9 Current Status (ISENS)—Bits 3–2

These read/write bits select the top/bottom deadtime correction scheme, illustrated in [Table 11-2](#). Reset clears the ISENS $n$  bits. This selects manual correction or no correction, or automatic deadtime correction.

**Note:** The ISENS $n$  bits are not buffered. Changing the current status sensing method can affect the present PWM cycle.

### 11.10.1.10 Load Okay (LDOK)—Bit 1

This read/write bit loads the prescaler bits of PMCTL and the entire PWMCM register and PWMVAL registers into a set of buffers. The buffered prescaler divisor, PWM counter modulus value, and PWM pulse width take effect at the next PWM reload. Set LDOK by reading it, then write a Logic 1 to it. LDOK is automatically cleared after the new values are loaded, or can be manually cleared before a reload by writing a Logic 0 to it.

- 0 = No action is taken
- 1 = Load prescaler, PWM modulus and PWM values into buffers

### 11.10.1.11 PWM Enable (PWMEN)—Bit 0

This read/write bit enables the PWM generator. When PWMEN equals zero, the PWM outputs are in their inactive states unless OUTCTL $n$  equals one.

- 0 = PWM generator and PWM outputs disabled unless OUTCTRL = 1
- 1 = PWM generator and PWM outputs enabled

**Note:** PWM pin outputs are in tri-state if Output Pad Enable (PAD\_EN) bit in PWM Output Control (PMOUT) is cleared.

## 11.10.2 PWM Fault Control Register (PMFCTL)

Base+ \$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	FIE3	FMODE3	FIE2	FMODE2	FIE1	FMODE1	FIE0	FMODE0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-38. PWM Fault Control Register (PMFCTL)

### 11.10.2.1 Reserved—Bits 15–8

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 11.10.2.2 FAULT $n$ Pin Interrupt Enable (FIE $n$ )—Bits 7, 5, 3, 1

These read/write bits enable the interrupt request generated by the FAULT $n$  pin.

- 0 = FAULT $n$  interrupt request disabled
- 1 = FAULT $n$  interrupt request enabled

**Note:** The fault protection circuit is independent of the FIE $n$  bits and is always active. If a fault is detected, the PWM pins are disabled according to the PWM disable mapping register.

### 11.10.2.3 FAULT $n$ Pin Clearing Mode (FMODE $n$ )—Bits 6, 4, 2, 0

These read/write bits select automatic or manual clearing of FAULT $n$  pin faults.

- 0 = Manual fault clearing of FAULT $n$  pin faults
- 1 = Automatic fault clearing of FAULT $n$  pin faults

## 11.10.3 PWM Fault Status and Acknowledge Register (PMFSA)

Base + \$2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FPIN3	FFLAG3	FPIN2	FFLAG2	FPIN1	FFLAG1	FPIN0	FFLAG0	0	0	DT5	DT4	DT3	DT2	DT1	DT0
Write										FTACK3		FTACK2		FTACK1		FTACK0
Reset	U	0	U	0	U	0	U	0	0	0	0	0	0	0	0	0

Figure 11-39. PWM Fault Status and Acknowledge Register (PMFSA)

### 11.10.3.1 FAULT $n$ Pin (FPIN $n$ )—Bits 15, 13, 11, 9

These *read-only* bits reflect the current state of the filtered FAULT $n$  bit. A reset has no effect on FPIN $n$ .

- 0 = Logic 0 on the FAULT $n$  bit
- 1 = Logic 1 on the FAULT $n$  bit

### 11.10.3.2 FAULT $n$ Pin Flag (FFLAG $n$ )—Bits 14, 12, 10, 8

These *read-only* flags are set within two IPBus cycles after a rising edge on the FAULT $n$  pin. Clear FFLAG $n$  by writing a Logic 1 to the FTACK $n$  bits in the PMFSA register. A reset clears FFLAG $n$ .

- 0 = No fault on the FAULT $n$  bit
- 1 = Fault on the FAULT $n$  bit

### 11.10.3.3 Reserved—Bit 7

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 11.10.3.4 FAULT $n$ Pin Acknowledge (FTACK $n$ )—Bits 6, 4, 2, 0

Writing a Logic 1 to FTACK $n$  clears FFLAG $n$ . Writing a Logic 0 has no effect. Reading these bits reads the appropriate DT $n$  bit. Please see [Section 11.10.3.5](#). Reset clears FTACK $n$ . The fault protection is enabled even when the PWM is not enabled; therefore if a fault is latched, it must be cleared prior to enabling the PWM to prevent an unexpected interrupt.

### 11.10.3.5 Deadtime $n$ (DT $n$ )—Bits 5–0

These are *read-only* bits. The DT $n$  bits are grouped in pairs, DT0 and DT1, DT2 and DT3, DT4 and DT5. Each pair reflects the corresponding IS $n$  pin value as sampled during deadtime period. A reset clears these bits.

## 11.10.4 PWM Output Control Register (PMOUT)

This is a read/write register.

Base + \$3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PAD_EN	0	OUT	OUT	OUT	OUT	OUT	OUT	0	0	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
Write			CTL5	CTL4	CTL3	CTL2	CTL1	CTL0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 11-40. PWM Output Control Register (PMOUT)**

### 11.10.4.1 Output Pad Enable (PAD\_EN)—Bit 15

The PWM output pads can be enabled or disabled by setting the PAD\_EN bit. The power-up default has the pads disabled. This bit does not affect the functionality of the PWM, so the PWM module can be energized with the output pads disabled. This enable is to power-up with a safe default value for the PWM drivers.

- 0 = Output pads disabled (tri-stated)
- 1 = Output pads enabled (not tri-stated)

### 11.10.4.2 Reserved—Bit 14

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 11.10.4.3 Output Control Enables (OUTCTRL5–0)—Bits 13–8

These read/write bits enable software control of their corresponding PWM pin. When OUTCTRL $n$  is set, the OUT $n$  bit activates and deactivates the PWM $n$  output. A reset clears the OUTCTRL bits. When operating the PWM in Complementary mode, these bits must be switched in pairs for proper operation. Please see [Section 11.5](#) for details.

- 0 = Software control disabled
- 1 = Software control enabled

### 11.10.4.4 Reserved—Bits 7–6

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 11.10.4.5 Output Control (OUT5–0)—Bits 5–0

When the corresponding OUTCTRL bit is set, these read/write bits control the PWM pins, illustrated in [Table 11-12](#).

**Table 11-12. Software Output Control**

Out $n$ Bit	Complementary Channel Operation	Independent Channel Operation
OUT0	1—PWM0 is Active 0—PWM0 is Inactive	1—PWM0 is Active 0—PWM0 is Inactive
OUT1	1—PWM1 is Complement of PWM 0 0—PWM1 is Inactive	1—PWM1 is Active 0—PWM1 is Inactive
OUT2	1—PWM2 is Active 0—PWM2 is Inactive	1—PWM2 is Active 0—PWM2 is Inactive
OUT3	1—PWM3 is Complement of PWM 2 0—PWM3 is Inactive	1—PWM3 is Active 0—PWM3 is Inactive
OUT4	1—PWM4 is Active 0—PWM4 is Inactive	1—PWM4 is Active 0—PWM4 is Inactive
OUT5	1—PWM5 is Complement of PWM 4 0—PWM5 is Inactive	1—PWM5 is Active 0—PWM5 is Inactive

### 11.10.5 PWM Counter Register (PMCNT)

Base + \$4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	CNT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-41. PWM Counter Register (PMCNT)

#### 11.10.5.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 11.10.5.2 Counter (CNT)—Bits 14–0

These *read-only* bits display the state of the 15-bit PWM counter.

### 11.10.6 PWM Counter Modulo Register (PWMCM)

Base + \$5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	CM														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-42. PWM Counter Modulo Register (PWMCM)

#### 11.10.6.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 11.10.6.2 Counter Modulo (CM)—Bits 14–0

The 15-bit unsigned value written to this buffered read/write register defines the number of PWM clocks to use in determining the PWM period. Do not write a modulus value of zeros into these bits.

**Note:** The PWM counter modulo register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins. Reading PWMCM reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

### 11.10.7 PWM Value Registers (PWMVAL0–5)

The 16-bit signed value in these buffered, read/write registers is the PWM pulse width in PWM clock periods for each of the output channels.

Base + \$6	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	VAL															
Write	VAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-43. PWM Value Register (PWMVAL0-5)

#### 11.10.7.1 Value (VAL)—Bits 15–0

The PWM Value registers are buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins. Reading PWMVAL $n$  reads the value in a buffer and not necessarily the value the PWM generator is currently using.

A PWM value less than, or equal to zero, deactivates the PWM output for the entire PWM period. A PWM value greater than, or equal to the modulus, activates the PWM output for the entire PWM period. Please see [Table 11-1](#).

**Note:** The terms activate and deactivate refer to the high and low logic states of the PWM outputs.

### 11.10.8 PWM Deadtime Register (PMDEADTM)

Base + \$C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	PWMDT											
Write					PWMDT											
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 11-44. PWM Deadtime Register (PMDEADTM)

#### 11.10.8.1 Reserved—Bits 15–12

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 11.10.8.2 Deadtime (PWMDT)—Bits 11–0

The 12-bit value written to this write-protectable register is the number of PWM clock cycles in complementary channel operation. A reset sets the PWM Deadtime register to a default value of 0x0FFF, selecting a deadtime of 4096-PWM clock cycles minus one IPBus clock cycle. This register is write protected after the Write Protect (WP) bit in the PWM configuration register is set. Please refer to [Section 11.10.10](#).

**Note:** Deadtime is affected by changes to the prescaler value. The deadtime duration is determined as follows:  $DT = P \times PWMDT - 1$  IPBus clocks, where DT is deadtime, P is the prescaler value, PWMDT is the programmed value of deadtime. For example: if the prescaler is programmed for a divide-by-two and PWMDT is set to five, then  $P = 2$  and the deadtime value is equal to:

$$DT = 2 \times 5 - 1 = 9 \text{ IPBus clock cycles.}$$

### 11.10.9 PWM Disable Mapping Registers (PMDISMAP1-2)

These *write-protectable* registers determine which PWM pins are disabled by the fault protection inputs, provided in [Table 11-6](#). Reset sets all of the bits used in the PWM disable mapping registers. These registers are write-protected after the WP bit in the PWM Configure register is set. Reserved bits 15-8 in the PMDISMAP2 register cannot be modified. The bits are read as 0.

Base + \$D	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DISMAP[15:0]															
Write	DISMAP[15:0]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 11-45. PWM Disable Mapping Register 1 (PMDISMAP1)

Base + \$E	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	DISMAP [23:16]							
Write									DISMAP [23:16]							
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 11-46. PWM Disable Mapping Register 2 (PMDISMAP2)

### 11.10.10 PWM Configure Register (PMCFG)

This *write-protectable* register contains the configuration bits determining PWM modes of operation detailed below. This register cannot be modified after the WP bit is set.

Base + \$F	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Read</b>	0				0				0							
<b>Write</b>		DBG_EN	WAIT_EN	EDG		TOPNEG45	TOPNEG23	TOPNEG01		BOTNEG45	BOTNEG23	BOTNEG01	INDEP45	INDEP23	INDEP01	WP
<b>Reset</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 11-47. PWM Configure Register (PMCFG)**

#### 11.10.10.1 Reserved—Bit 15

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 11.10.10.2 Debug Enable (DBG\_EN)—Bit 14

When this *write-protectable* bit is set to one, the PWM will continue to run while the chip is in EOnCE Debug mode. If the device enters the EOnCE mode and this bit is zero, the PWM outputs will be switched to their inactive state until the EOnCE mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers.

**Note:** PWM parameter updates will not occur in the EOnCE Debug mode.

#### 11.10.10.3 Wait Enable (WAIT\_EN)—Bit 13

When this *write-protectable* bit is set to one, the PWM will continue to run while the chip is in the Wait mode. In this mode, the peripheral clock continues to run; however, the core clock does not. If the device enters the Wait mode and this bit is zero, the PWM outputs will be switched to their inactive state until the Wait mode is exited. At the point of exit the PWM pins will resume operation as programmed in the PWM registers.

**Note:** PWM parameter updates will not occur in Wait mode.

#### 11.10.10.4 Edge-Aligned or Center-Aligned PWMs (EDG)—Bit 12

This *write-protectable* bit determines whether all PWM channels will use Edge-Aligned or Center-Aligned wave forms.

- 0 = Center-Aligned PWMs
- 1 = Edge-Aligned PWMs



### 11.10.10.5 Reserved—Bit 11

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 11.10.10.6 Top-Side PWM Polarity (TOPNEG)—Bits 10–8

This *write-protectable* bit determines the polarity for the top-side PWMs.

- 0 = Positive top-side polarity
- 1 = Negative top-side polarity

### 11.10.10.7 Bottom-Side PWM Polarity (BOTNEG)—Bits 6–4

This *write-protectable* bit determines the polarity for the bottom-side PWMs.

- 0 = Positive bottom-side polarity
- 1 = Negative bottom-side polarity

### 11.10.10.8 Independent or Complement Pair Operation (INDEP)—Bits 3–1

This *write-protectable* bit determines if the PWM channels will be independent PWMs or complementary PWM pairs.

- 0 = Complementary PWM pair
- 1 = Independent PWMs

**Note:** Each pair of PWM channels can be configured: channel zero to one, channel two to three, and channel four to five.

### 11.10.10.9 Write Protect (WP)—Bit 0

This bit enables write-protection for all write-protectable registers. While clear, *WP allows write-protected registers and bits to be written*. When set, WP prevents writes to write-protectable registers and bits. Once set, WP can be cleared only by a reset.

*Write-protectable* registers and bits include: PMDISMAP1–2, PMDEADTM, PMCFG, and the ENHA bit in the Channel Control Register (PMCCR). The VLMODE, SWP45, SWP23, and SWP01 bits in the PMCCR are protected when the Enable Hardware Acceleration (ENHA) bit is set to zero in the PMCCR. ENHA is in turn, protected by setting the WP bit in the PMCFG.

- 0 = Write-protectable registers may be written
- 1 = Write-protectable registers are write protected

**Note:** The write to PMCFG setting the WP bit is the last write accepted to the register until reset.

### 11.10.11 PWM Channel Control Register (PMCCR)

Base + \$10	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Read</b>	ENHA	nBX	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	0	0	VLMODE		0	SWP45	SWP23	SWP01
<b>Write</b>																
<b>Reset</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 11-48. PWM Channel Control Register (PMCCR)**

This *write-protectable* register contains the configuration bits determining PWM modes of operation detailed below. The ENHA bit cannot be modified after the WP bit in the PMCFG register is set. In turn, ENHA provides protection for the nBX, VLMODE, SWP45, SWP23 and SWP01 bits. Mask bits 0-5 are not write-protectable.

#### 11.10.11.1 Enable Hardware Acceleration (ENHA)—Bit 15

This bit enables writing to the nBX, VLMODE, SWP45, SWP23, and SWP01 bits. The bit is *write-protectable* by WP bit in the PMCFG register.

- 0 = Disable writing to nBX, VLMODE, SWP45, SWP23, and SWP01 bits
- 1 = Enable writing to nBX, VLMODE, SWP45, SWP23, and SWP12 bits

#### 11.10.11.2 56F80x Compatibility (nBX)—Bit 14

This bit is used to enable/disable improved SWAP and MASK operations. If it is cleared, SWAP/MASK operates identical to the 56F80x version of this module. See 56F80x User's Manual for details. If is set, SWAP and MASK operations are described in this manual which are not compatible features with 56F80 and are not supported.

- 0 = SWAP and MASK provide 56F80x compatible operation
- 1 = SWAP<sub>n</sub> and MASK<sub>n</sub> provide new functionality described in this manual

This bit is write-protected when ENHA is zero.

**Note:** Unless expecting SWAP/MASK operate identical to 56F80x version of this module, nBX = 1 mode is highly recommended.

#### 11.10.11.3 Mask (MSK5–0)—Bits 13–8

These six bits determine the mask for each of the PWM logical channels.

- 0 = Unmasked
- 1 = Masked, channel deactivated

#### 11.10.11.4 Reserved—Bits 7–6

These bits are reserved or not implemented. They are read as 0 and cannot be modified by writing.

#### 11.10.11.5 Value Register Load Mode (VLMODE)—Bits 5–4

These two bits determine the way the PWM value registers are being loaded. These bits are write-protected when ENHA is zero.

- 00 = Each PWM value register is accessed independently
- 01 = Writing to PWM value register zero also writes to PWM value registers one to five
- 10 = Writing to PWM value register zero also writes to PWM value registers one to three
- 11 = Reserved

#### 11.10.11.6 Reserved—Bit 3

This bit is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 11.10.11.7 Swap45 (SWP45)—Bit 2

This bit is write-protected when ENHA is zero.

- 0 = No swap
- 1 = Channel four and channel five are swapped

#### 11.10.11.8 Swap23 (SWP23)—Bit 1

This bit is write-protected when ENHA is zero.

- 0 = No swap
- 1 = Channel two and channel three are swapped

#### 11.10.11.9 Swap01 (SWP01)—Bit 0

This bit is write-protected when ENHA is zero.

- 0 = No swap
- 1 = Channels zero and one are swapped

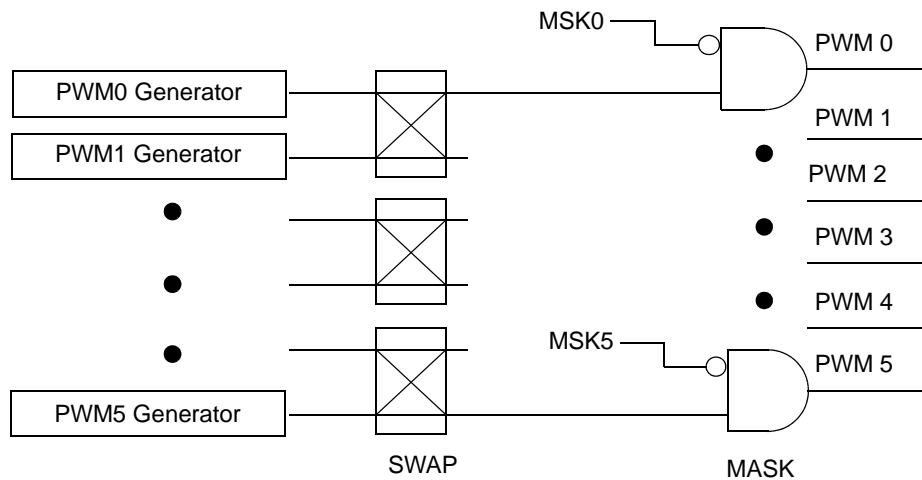


Figure 11-49. Channel Swapping

### 11.10.12 PWM Port Register (PMPORT)

This register contains values of the three current status inputs, bits six, five, and four. Additionally, it contains the four fault inputs, bits three, two, one, and zero. This register may be read while the PWM is active.

Base + \$11	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	IS2	IS1	IS0	FAULT3	FAULT2	FAULT1	FAULT0
Write																
Reset	0	0	0	0	0	0	0	0	0	U	U	U	U	U	U	U

Figure 11-50. PWM Port Register (PMPORT)

#### 11.10.12.1 Reserved—Bits 15–7

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 11.10.12.2 Port (PORT)—Bits 6–0

These are *read-only* bits; therefore, any writes to them are not affected.

### 11.10.13 PWM Internal Correction Control Register (PMICCR)

These bits only apply in Center-Aligned operation during Complementary mode. Setting these bits overrides the ISENS[1:0] settings. These control bits determine whether values latched on

the  $IS_n$  pins control which PWM value register is used, or PWM count direction controls which PWM value register is used.

Base + \$12	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	ICC2	ICC1	ICC0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 11-51. PWM Internal Correction Control Register (PMICCR)**

### 11.10.13.1 Reserved—Bits 15–3

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

### 11.10.13.2 Internal Current Control 2 (ICC2)—Bit 2

This bit controls PWM4/PWM5 pair.

- 0 = PWM value register to use is determined by the  $ISENS[0:1]$  setting. The current direction sensed is captured in the DT4 and DT5 bits of the PMFSA register.
- 1 = Use PWMVAL4 register when the PWM counter is counting up. Use PWMVAL5 register when counting down.

### 11.10.13.3 Internal Current Control 1 (ICC1)—Bit 1

This bit controls PWM2/PWM3 pair.

- 0 = PWM value register to use is determined by the  $ISENS[0:1]$  setting. The current direction sensed is captured in the DT2 and DT3 bits of the PMFSA register.
- 1 = Use PWMVAL2 register when the PWM counter is counting up. Use PWMVAL3 register when counting down.

### 11.10.13.4 Internal Current Control 0 (ICC0)—Bit 0

This bit controls PWM0/PWM1 pair.

- 0 = PWM value register to use is determined by the  $ISENS[0:1]$  setting. The current direction sensed is captured in the DT0 and DT1 bits of the PMFSA register.
- 1 = Use PWMVAL0 register when the PWM counter is counting up. Use PWMVAL1 register when counting down.

## 11.11 Clocks

The system IPBus clock is the only clock required by this module. Along with the PWM prescaler, it determines the amount of time allocated for a single PWM bit value.

## 11.12 Interrupts

Five PWM sources can generate two interrupt requests to the core:

- Reload Flag (PWMF)—PWMF is set at the beginning of every reload cycle. The reload interrupt enable bit, PWMRIE, enables PWMF to generate core interrupt requests. PWMF and PWMRIE are in PWM Control (PMCTL) register.
- Fault Flags (FFLAG0–FFLAG3)—The FFLAG $n$  bit is set when a Logic 1 occurs on the FAULT $n$  pin. The fault pin interrupt enable bits, FIE0–FIE3, enable the FFLAG $n$  flags to generate core interrupt requests. FFLAG0–FFLAG3 are in the Fault Status (PMFSA) register. FIE0–FIE3 are in the Fault Control (PMFCTL) register.

## 11.13 Resets

All PWM registers are reset to their default values upon any system reset.

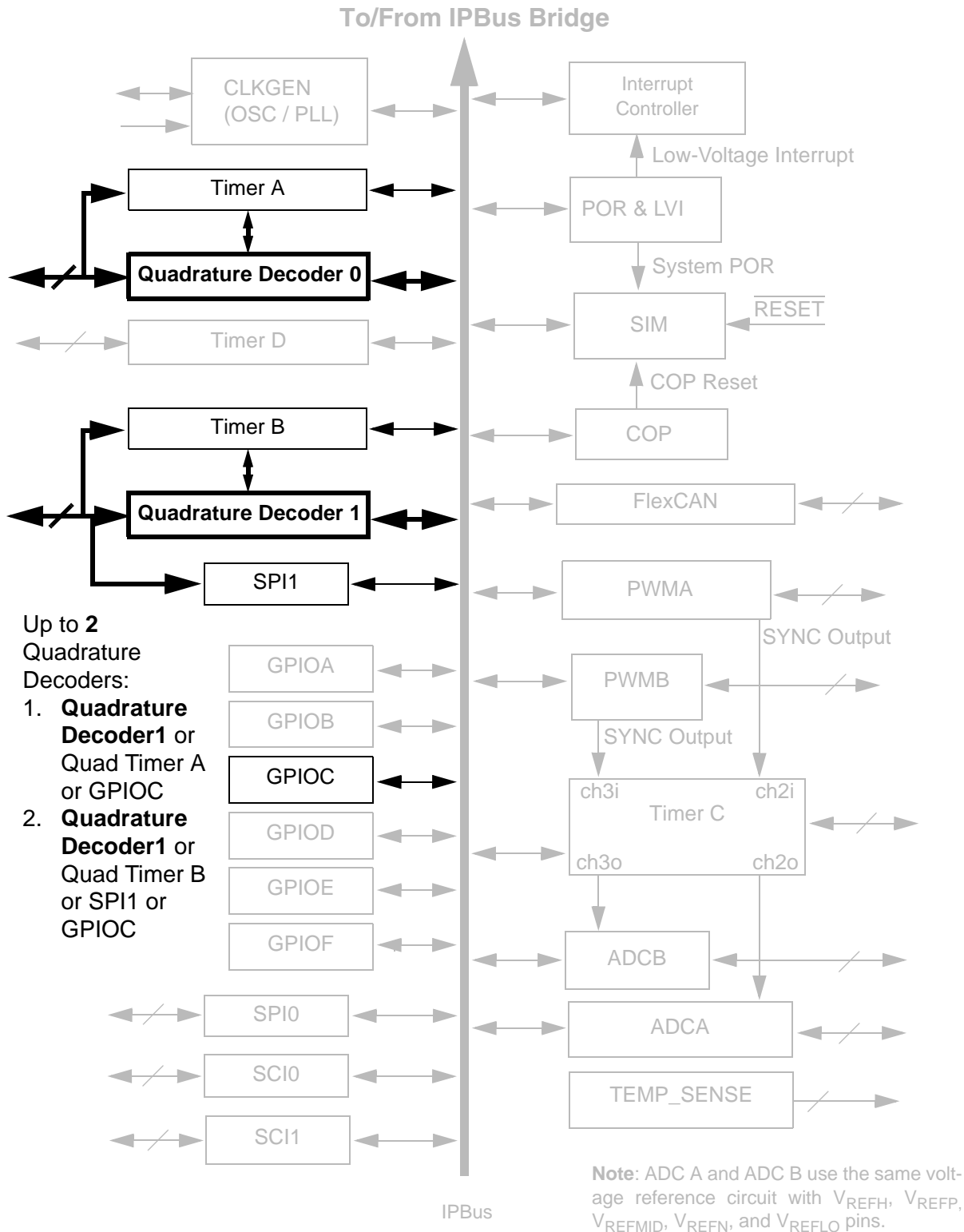






# Chapter 12

## Quadrature Decoder (DEC)



Quadrature Decoder (DEC), Rev. 10

## Document Revision History for **Chapter 12, Quadrature Decoder (DEC)**

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 5.0	Converted chapter to Freescale design standards
Rev 6.0	Clarification edits pertaining to timing inputs throughout chapter where pertinent as well as corrected wrong referenced table for timing inputs

## 12.1 Introduction

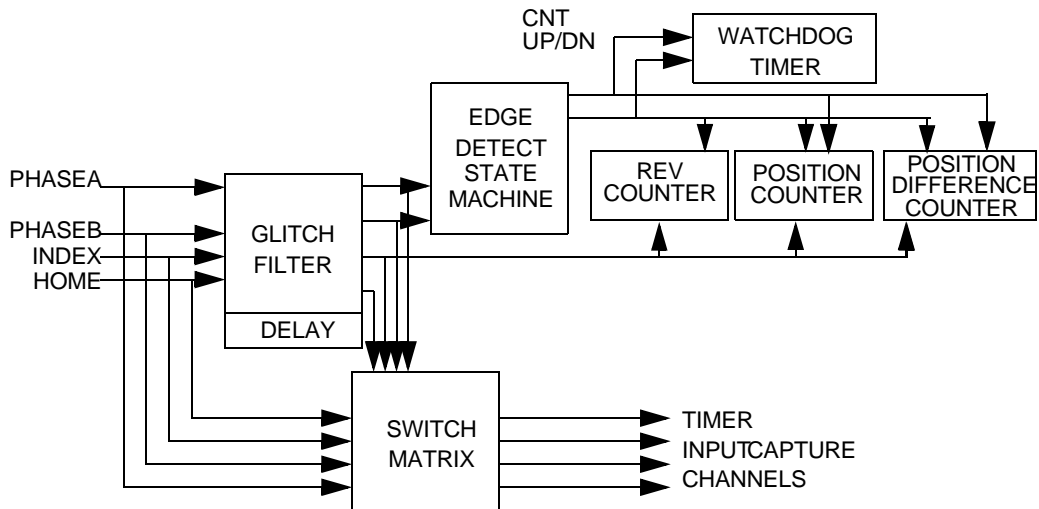
A quadrature decoder circuit decodes the quadrature encoder signals, normally two 90 degrees out-of-phase pulses, into count and direction information. The maximum counting resolution is  $4 \times$  input signal. The quadrature decoder samples both incoming pulses. Based on the previous pulse information of the two signals and the present state, it outputs a count signal and a direction signal to internal position counts.

## 12.2 Features

- Includes logic to decode quadrature signals
- Configurable digital filter for inputs to remove glitches and ensure only true transitions are recorded
- 32-bit position counter register
- 16-bit position difference register
- Maximum count frequency equals the IPBus clock rate
- Position counter can be initialized by software or external events
- Preloadable 16-bit revolution counter
- Inputs can be connected to a general purpose timer, aiding low speed velocity measurements
- A watchdog timer to detect a non-rotating shaft condition
- Can be optionally used as a single phase pulse accumulator

## 12.3 Block Diagram

Figure 12-1 illustrates the Quadrature Decoder module block diagram.



**Figure 12-1. Quadrature Decoder Block Diagram**

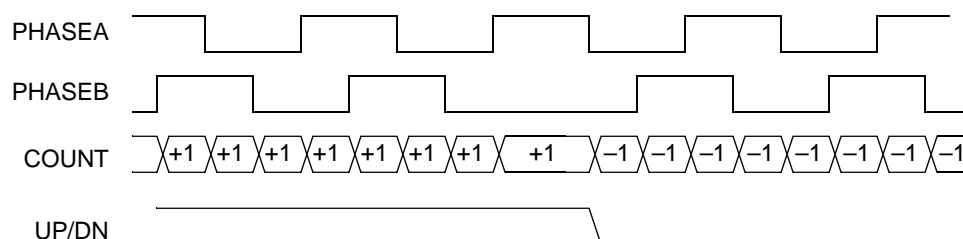
The Quadrature Decoder module has four input signals. They are:

1. PHASEA
2. PHASEB
3. INDEX
4. HOME

The Quadrature Decoder also includes a circuitry called the switch matrix. The switch matrix provides a mean to share input pins with an associated timer module.

## 12.4 Functional Description

A timing diagram illustrating the basic operation of a quadrature incremental position Quadrature Decoder is illustrated in [Figure 12-2](#).



**Figure 12-2. Quadrature Decoder Signals**

### 12.4.1 Positive Vs. Negative Direction

A typical quadrature encoder has three outputs:

1. PHASEA signal
2. PHASEB signal
3. INDEX pulse, not shown

If PHASEA leads PHASEB, assuming motion is in the positive direction. However, if PHASEA lags PHASEB, the motion is in the negative direction. Transitions on these phases can be integrated to yield position, or differentiated to yield velocity. The Quadrature Decoder is designed to perform these functions in hardware.

### 12.4.2 Position Counter

The 32-bit position counter calculates up or down on every count pulse, generated by the difference of PHASEA and PHASEB. The direction of the count is determined by direction signal. This counter acts as an integrator, whose count value is proportional to position. Position counters may be initialized to a predetermined value by one of three different methods:

1. Software-triggered event
2. INDEX signal transition
3. HOME signal transition

The INDEX and HOME signals can be programmed to interrupt the processor. Whenever the position counter is read, either Upper Position (UPOS) Counter Register or Lower Position Counter Register (LPOS), snapshots of the position counter, the position difference counter, and

the revolution counters are placed into their Hold registers and position difference counter is cleared.

### 12.4.3 Position Difference Counter

The 16-bit Position Difference Counter contains the position difference value occurring between each read of the Position register. This register counts up or down on every count pulse generated by the difference of PHASEA and PHASEB. The direction of count is determined by the direction signal. The counter acts as a differentiator whose count value is proportional to the change in position since the last time the position counter was read. When the position difference counter is read, snapshots of position counter, the position difference counter, and the revolution counter are placed into their Hold register and position difference register is cleared.

### 12.4.4 Revolution Counter

The 16-bit up/down Revolution Counter is used to count, or integrate revolutions. This is achieved by counting INDEX pulses. The direction of the count is determined by the direction signal. If the direction of the count is different on the rising and falling edges of the INDEX pulse, it indicates the quadrature encoder changed direction on the INDEX pulse when the Revolution counter is placed into their Holding registers.

### 12.4.5 Holding and Initializing Registers

Hold registers are associated with three counters:

1. Position
2. Position difference
3. Revolution

When any of the counter registers are read, the contents of each counter register is written to the corresponding hold register. Taking a snapshot of the counters' values provides a consistent view of a system position and a velocity to be attained.

The Counter and Hold registers are read/write capable. However, if writing values into any Hold register before reading any counter, Hold register contents is overwritten with the values in the counters after any counter is read.

The position counter is 32 bits wide. It can be initialized with two 16-bit registers, an upper and a lower initialization register, are provided. The Upper Initialization Register (UIR) and Lower Initialization Register (LIR) should be modified with the desired value to initialize the counter.

The Position counter can be loaded by writing 1 to the SWIP bit in the Decoder Control Register (DECCR). Alternatively, either the XIP or the HIP bits in the DECCR can enable the Position counter to be initialized in response to a HOME or INDEX signal transition.

### 12.4.6 Prescaler for Slow or Fast Speed Measurement

For a fast moving shaft encoder, the speed can be computed by calculating the change in the position counter per unit time, or by reading the Position Difference Counter (POSD) register and calculating speed. For applications with slow shaft movement speeds and low line count quadrature encoders, high resolution velocity measurements can be made with the timer module by measuring the time period between quadrature phases. The timer module uses a 16-bit free running counter operating from a prescaled version of the IPBus clock. The prescaler divides the IPBus clock by values ranging from one to 128. The counter period is calculated by following:

$$\text{Counter Period} = \frac{\text{Counter Value} \times \text{Prescaler}}{\text{IPBus Frequency}}$$

### 12.4.7 Pulse Accumulator Functionality

The module can be programmed, integrating only selected transitions of the PHASEA signal. In this manner the position counter is used as a pulse accumulator. The count direction is up. The pulse accumulator can also optionally be initialized by the INDEX input. Please see [Section 12.7.1.7](#).

### 12.4.8 Glitch Filter

Because the logic of the Quadrature Decoder must sense transitions, the inputs are first run through a glitch filter. This filter has a digital delay line sampling four time points on the signal and verifying a majority of the samples are at a new state before outputting this new state to the internal logic. The sample rate of this delay line is programmable to adapt to a variety of signal bandwidths. Please refer to [Section 12.7.2](#).

### 12.4.9 Edge Detect State Machine

The Edge Detect State Machine looks for changes in the four possible states of the filtered PHASEA and PHASEB inputs, calculating the direction of motion. This information is formatted as Count\_Up and Count\_Down signals. These signals are routed into up to three up/down counters:

1. Position counter
2. Revolution counter
3. Position difference counter

### 12.4.10 Watchdog Timer

A Watchdog Timer is included. This timer ensures the algorithm is indicating motion in the shaft. Two successive counts indicate proper operation, resetting the timer. The timeout value is programmable. When a timeout occurs, an interrupt request can be generated.

## 12.5 Operating Modes

**Table 12-1. Switch Matrix for Inputs to the Timer**

	PHASEA	PHASEA Filtered	PHASEB	PHASEB Filtered	INDEX	INDEX Filtered	HOME	HOME Filtered
<b>Mode 0</b>	Timer 0	—	Timer 1	—	Timer 2	—	Timer 3	—
<b>Mode 1</b>	—	Timer 0	—	Timer 1	—	Timer 2	—	Timer 3
<b>Mode 2</b>	—	Timer 0, 1	—	Timer 2, 3	—		—	—
<b>Mode 3</b>	Reserved							

The PHASEA and PHASEB inputs are routed through a switch matrix to a general purpose timer module. Possible connections of the switch matrix are provided in [Table 12-1](#). In mode zero, the timer module can use all four available inputs as normal timer input capture channels. This does not preclude use of the Quadrature Decoder, but normally it would not be used in this mode. Modes one and zero are similar, but mode one takes advantage of the digital filter incorporated in the Quadrature Decoder. Mode two is most likely to be used in conjunction with operation of the Quadrature Decoder.

Both the positive and negative edges of PHASEA and PHASEB can be captured in mode two. Full speed range measurement is able to achieve under mode two. The MODE bit field is described in [Section 12.7.1.15](#).

## 12.6 Pin Descriptions

The four external signals, each muxed to the four pins of a Quad Timer module, are discussed in the this sections.



### 12.6.1 Phase A Input (PHASEA)

The PHASEA input can be connected to one of the phases from a two-phase shaft quadrature encoder output. It is used by the Quadrature Decoder module in conjunction with the PHASEB input to indicate a decoder increment has passed, and to calculate its direction. PHASEA is the leading phase for a shaft rotating in the positive direction. PHASEA is the trailing phase for a shaft rotating in the negative direction. It can also be used as the single input when the Quadrature Decoder module is used as a single phase pulse accumulator. Please refer to [Section 12.7.1.15](#).

The PHASEA input is also an input capture channel for one of the timer modules. The exact connection to the timer module is specified in [Table 12-1](#) based on mode setting.

### 12.6.2 Phase B Input (PHASEB)

The PHASEB input is used as one of the phases from a two-phase shaft quadrature encoder output. It is used by the Quadrature Decoder module in conjunction with the PHASEA input indicating a decoder increment has passed, and to calculate its direction. PHASEB is the trailing phase for a shaft rotating in the positive direction. PHASEB is the leading phase for a shaft rotating in the negative direction.

The PHASEB input is also an input capture channel for one of the Timer modules. The exact connection to the Timer module is specified in [Table 12-1](#) based on mode setting.

### 12.6.3 Index Input (INDEX)

Normally connected to the index pulse output of a quadrature encoder, this pulse can be used to trigger the initialization of the position counters (UPOS and LPOS) and the pulse accumulator of the Quadrature Decoder module. It also causes a change of state on the revolution counter. The direction of this change, increment or decrement, is calculated from the PHASEA and PHASEB inputs.

The INDEX input is also an input capture channel for one of the Timer modules. The exact connection to the Timer module is specified in [Table 12-1](#) based on mode setting.

**Note:** If the motor shaft is stationary and positioned near, or at the index point, slight variations of the shaft position can cause multiple counts of the revolution counter. To avoid false indications, the software should check the position registers to verify the count has changed.

## 12.6.4 Home Switch Input (HOME)

The HOME input can be used by the Quadrature Decoder and the Timer module. This input can be used to trigger the initialization of the position counters (UPOS and LPOS). Often this signal is connected to a sensor signaling the motor or machine notifying it has reached a defined home position. This general purpose signal is also connected to the Timer module as specified in [Table 12-1](#).

## 12.7 Register Definitions

**Table 12-2. DEC Memory Map**

Device	Peripheral	Address
8100/8300	DEC0_BASE	\$00F180
	DEC1_BASE	\$00F190

The address of a register is the sum of a base address and an address offset. Please refer to the chip's data sheet for the definition of the base address. All memory locations base and offsets are provided in hex.

Each of the following set of registers is used for each Quadrature Decoder module. For example, if there are two quadrature decoders on the chip, there are two decoder control registers: DEC0\_DECCR at data memory location DEC0\_BASE+\$0 and DEC1\_DECCR at data memory location DEC1\_BASE+\$0.

**Table 12-3. DEC Register Summary**

Address Offset	Address Acronym	Register Name	Chapter Location
Base + \$0	DECCR	Decoder Control Register	<a href="#">Section 12.7.1</a>
Base + \$1	FIR	Filter Interval Register	<a href="#">Section 12.7.2</a>
Base + \$2	WTR	Watchdog Timer Register	<a href="#">Section 12.7.3</a>
Base + \$3	POSD	Position Difference Counter Register	<a href="#">Section 12.7.4</a>
Base + \$4	POSDH	Position Difference Counter Hold Register	<a href="#">Section 12.7.5</a>
Base + \$5	REV	Revolution Counter Register	<a href="#">Section 12.7.6</a>
Base + \$6	REXH	Revolution Hold Register	<a href="#">Section 12.7.7</a>
Base + \$7	UPOS	Upper Position Counter Register	<a href="#">Section 12.7.8</a>
Base + \$8	LPOS	Lower Position Counter Register	<a href="#">Section 12.7.9</a>
Base + \$9	UPOSH	Upper Position Hold Register	<a href="#">Section 12.7.10</a>
Base + \$A	LPOSH	Lower Position Hold Register	<a href="#">Section 12.7.11</a>
Base + \$B	UIR	Upper Initialization Register	<a href="#">Section 12.7.12</a>
Base + \$C	LIR	Lower Initialization Register	<a href="#">Section 12.7.13</a>
Base + \$D	IMR	Input Monitor Register	<a href="#">Section 12.7.14</a>

Bit fields of each of the 14 registers are illustrated in [Figure 12-3](#). Details of each follow.



- 0 = No interrupt
- 1 = HOME signal transition interrupt request

#### **12.7.1.2 HOME Interrupt Enable (HIE)—Bit 14**

This read/write bit enables HOME signal interrupts.

- 0 = Disable HOME interrupts
- 1 = Enable HOME interrupts

#### **12.7.1.3 Enable HOME to Initialize Position Counters UPOS and LPOS (HIP)—Bit 13**

This read/write bit allows the position counter to be initialized by the HOME signal.

- 0 = No action
- 1 = HOME signal initializes the position counter

#### **12.7.1.4 Use Negative Edge of HOME Input (HNE)—Bit 12**

This read/write bit determines the edge of the HOME input used to initialize the position counter.

- 0 = Use positive transition edge of HOME pulse
- 1 = Use negative transition edge of HOME pulse

#### **12.7.1.5 Software Triggered Initialization of Position Counters UPOS and LPOS (SWIP)—Bit 11**

Writing one to this bit will transfer the UIR and LIR contents to UPOS and LPOS respectively. This bit is always read as 0.

- 0 = No action
- 1 = Initialize position counter

#### **12.7.1.6 Enable Reverse Direction Counting (REV)—Bit 10**

This read/write bit reverses the interpretation of the quadrature signal, changing the direction of count.

- 0 = Count normally
- 1 = Count in the reverse direction

#### **12.7.1.7 Enable Signal Phase Count Mode (PH1)—Bit 9**

This read/write bit bypasses the Quadrature Decoder logic.

- 0 = Use standard Quadrature Decoder where PHASEA and PHASEB represents a two phase quadrature signal.
- 1 = Bypass the Quadrature Decoder, implement pulse accumulator, a positive transition of the PHASEA input generates a count signal. The PHASEB input and the REV bit controls the counter direction.
  - IF REV = 0, PHASEB = 0, then count up
  - IF REV = 0, PHASEB = 1, then count down
  - IF REV = 1, PHASEB = 0, then count down
  - IF REV = 1, PHASEB = 1, then count up

#### 12.7.1.8 INDEX Pulse Interrupt Request (XIRQ)—Bit 8

This bit is set when an INDEX interrupt occurs. It remains set until cleared by software. To clear, write 1 to this bit.

- 0 = No interrupt has occurred
- 1 = INDEX pulse interrupt has occurred

#### 12.7.1.9 INDEX Pulse Interrupt Enable (XIE)—Bit 7

This read/write bit enables INDEX interrupts.

- 0 = INDEX pulse interrupt is disabled
- 1 = INDEX pulse interrupt is enabled

#### 12.7.1.10 Index Triggered Initialization of Position Counters UPOS and LPOS (XIP)—Bit 6

This read/write bit enables the position counter to be initialized by the INDEX pulse.

- 0 = No action
- 1 = INDEX pulse initializes the position counter

#### 12.7.1.11 Use Negative Edge of INDEX Pulse (XNE)—Bit 5

This read/write bit determines the edge of the INDEX pulse used to initialize the position counter.

- 0 = Use positive transition edge of INDEX pulse
- 1 = Use negative transition edge of INDEX pulse

### 12.7.1.12 Watchdog Timeout Interrupt Request (DIRQ)—Bit 4

This bit is set when a watchdog timeout interrupt occurs. It remains set until it is cleared by software. To clear, write 1 to this bit.

- 0 = No interrupt has occurred
- 1 = Watchdog timeout interrupt has occurred

### 12.7.1.13 Watchdog Timeout Interrupt Enable (DIE)—Bit 3

This read/write bit enables watchdog timeout interrupts.

- 0 = Watchdog timer interrupt is disabled
- 1 = Watchdog timer interrupt is enabled

### 12.7.1.14 Watchdog Enable (WDE)—Bit 2

This bit operates the watchdog timer monitoring the PHASEA and PHASEB inputs for shaft movement.

- 0 = Watchdog timer is disabled
- 1 = Watchdog timer is enabled

### 12.7.1.15 Switch Matrix Mode (MODE)—Bits 1–0

These read/write bits select the Switch Matrix mode, connecting inputs to the Timer module. The modes are provided in [Table 12-1](#).

- 00 = Mode 0: Input captures connected to the four input pins as indicated in the first row of [Table 12-1](#).
- 01 = Mode 1: Input captures connected to the filtered versions of the four input pins
- 10 = Mode 2: PHASEA input connected to both channels zero and one of the timer to allow capture of both rising and falling edges; PHASEB input connected to both channels two and three of the timer to allow capture of both rising and falling edges
- 11 = Mode 3: Reserved

## 12.7.2 Filter Interval Register (FIR)

This read/write register sets the sample rate of the digital glitch filter. When the count reaches the specified value in FIR, the counter is reset and the filter takes a new sample of the raw PHASEA, PHASEB, INDEX, and HOME input signals. If the filter interval value is zero, the digital filter for the PHASEA, PHASEB, INDEX, and HOME inputs is bypassed.

Bypassing the digital filter enables the position/position difference counters to operate with count rates up to the IPBus frequency.

Base + \$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	DELAY							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 12-5. Filter Interval Register (FIR)**

The value of DELAY plus one represents the filter interval period in increments of the IPBus clock period.

The filter interval register works in this manner: Sample the inputs, PHASEA, PHASEB, INDEX, and HOME, monitoring their outputs in the Input Monitor Register (IMR). Use the following equations to determine how many IPBus clock cycles it needs before the outputs are available for Internal counters. Digital delay logic samples four times points on inputs signal and verifies a majority of samples, and then outputs them to Internal counters.

Where  $f$  is the number loaded in the FIR

- DELAY (IPBus clock cycles) =  $(f + 1) \times 4 + 1$  (to read the filtered output)
- DELAY (IPBus clock cycles) =  $(f + 1) \times 4 + 2$  (to monitor the output in the IMR)

One additional IPBus clock cycle is required to read the filtered output and two additional IPBus clock cycles to monitor the filtered output in the IMR. A one is added to  $f$  to account for the interval timer in the filter starting its counting from zero. The sample rate is set when it reaches the number  $f$ .

Two examples are provided:

1. When  $f = 0$ , the filter is bypassed and  $s$  is zero because there is no sampling. Therefore, DELAY = 1 or 2 clock cycles according to the equations above.
2. When  $f = 5$ , the DELAY =  $(5+1) \times 4 + (1 \text{ or } 2) = 25 \text{ or } 26$  clock cycles.

### 12.7.3 Watchdog Timer Register (WTR)

This read/write register stores the timeout count for the Quadrature Decoder module Watchdog Timer. This timer is separate from the Watchdog Timer in the COP module.



Base + \$2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COUNT															
Write	COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 12-6. Watchdog Timer Register (WTR)**

COUNT is a binary representation of the number of IPBus clock cycles plus one.

### 12.7.4 Position Difference Counter Register (POSD)

This read/write register contains the position change in value occurring between each read of the position register. The value of the Position Difference Counter (POSD) register can calculate velocity.

The 16-bit POSD computes up or down on every count pulse. This counter acts as a differentiator whose count value is proportional to the change in position since the last time the position counter was read. When read, the position difference counter's contents are copied into the POSDH register, and POSD is cleared.

Base + \$3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	POSD															
Write	POSD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 12-7. Position Difference Count Register (POSD)**

### 12.7.5 Position Difference Counter Hold Register (POSDH)

This read/write register contains a snapshot of the value of the POSD register, described in [Section 12.4.5](#). When the Position register is read, the Position Difference counter's contents are copied into the POSDH register and the Position Difference counter is cleared. The value of the POSDH can be used to calculate velocity.

Base + \$4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	POSDH															
Write	POSDH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 12-8. Position Difference Counter Hold Register (POSDH)**

## 12.7.6 Revolution Counter Register (REV)

This read/write register contains the current value of the revolution counter.

Base + \$5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	REV															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-9. Revolution Counter Register (REV)

## 12.7.7 Revolution Hold Register (RE VH)

This read/write register contains a snapshot of the value of the REV register, described in [Section 12.4.5](#).

Base + \$6	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RE VH															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-10. Revolution Hold Register (RE VH)

## 12.7.8 Upper Position Counter Register (UPOS)

This read/write register contains the upper and most significant half of the position counter. This is the binary count from the position counter.

Base + \$7	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	POS[31:16]															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-11. Upper Position Counter Register (UPOS)

## 12.7.9 Lower Position Counter Register (LPOS)

This read/write register contains the lower and least significant half of the current value of the position counter. It is the binary count from the position counter.

Base + \$8	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	POS[15:0]															
Write	POS[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 12-12. Lower Position Counter Register (LPOS)**

### 12.7.10 Upper Position Hold Register (UPOSH)

This read/write register contains a snapshot of the UPOS register with a description discussed in [Section 12.4.5](#).

Base + \$9	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	UPOSH[31:16]															
Write	UPOSH[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 12-13. Upper Position Counter Register (UPOSH)**

### 12.7.11 Lower Position Hold Register (LPOSH)

This read/write register contains a snapshot of the LPOS register with a description discussed in [Section 12.4.5](#).

Base + \$A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LPOSH[15:0]															
Write	LPOSH[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 12-14. Lower Position Hold Register (LPOSH)**

### 12.7.12 Upper Initialization Register (UIR)

This read/write register contains the value to initialize the upper half of UPOS.

Base + \$B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	INITIALIZATION VALUE[31:16]															
Write	INITIALIZATION VALUE[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-15. Upper Initialization Register (UIR)

### 12.7.13 Lower Initialization Register (LIR)

This read/write register contains the value to be used to initialize the lower half of LPOS.

Base + \$C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	INITIALIZATION VALUE[15:0]															
Write	INITIALIZATION VALUE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-16. Lower Initialization Register (LIR)

### 12.7.14 Input Monitor Register (IMR)

This *read-only* register contains the values of the raw and filtered PHASEA, PHASEB, INDEX, and HOME input signals. The reset value depends on the values of the raw and filtered values of the PHASEA, PHASEB, INDEX and HOME. If these input pins are connected to a pull-up, bits 0–7 of the IMR will all be ones. However, if these input pins are connected to a pull down device, bits 0–7 will all be zeros. If no pull-up or pull down is connected to these input pins, the reset value of the eight lower bits of the IMR will all be unknown.

Base + \$D	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	FPHA	FPHB	FIND	FHOM	PHA	PHB	INDEX	HOME
Write																
Reset	0	0	0	0	0	0	0	0	U	U	U	U	U	U	U	U

Figure 12-17. Input Monitor Register (IMR)

#### 12.7.14.1 Reserved—Bits 15–8

These bits are reserved or not implemented. They are read as 0 and cannot be modified by writing.

**12.7.14.2 Filtered PHASEA (FPHA)—Bit 7**

This is the filtered version of PHASEA input.

**12.7.14.3 Filtered PHASEB (FPHB)—Bit 6**

This is the filtered version of PHASEB input.

**12.7.14.4 Filtered Index (FIND)—Bit 5**

This is the filtered version of INDEX input.

**12.7.14.5 Filtered Home (FHOM)—Bit 4**

This is the filtered version of HOME input.

**12.7.14.6 Phase A Input (PHA)—Bit 3**

This is the raw PHASEA input.

**12.7.14.7 Phase B Input (PHB)—Bit 2**

This is the raw PHASEB input.

**12.7.14.8 Index Input (INDEX)—Bit 1**

This is the raw INDEX input.

**12.7.14.9 Home Switch Input (HOME)—Bit 0**

This is the raw HOME input.

**12.8 Interrupts****Table 12-4. Interrupt Summary**

Interrupt	Description	Section
HIRQ	HOME Signal Transition and Watchdog Timeout Interrupt Request	<a href="#">Section 12.7.1.1</a>
XIRQ	INDEX Signal Transition Interrupt Request	<a href="#">Section 12.7.1.12</a>

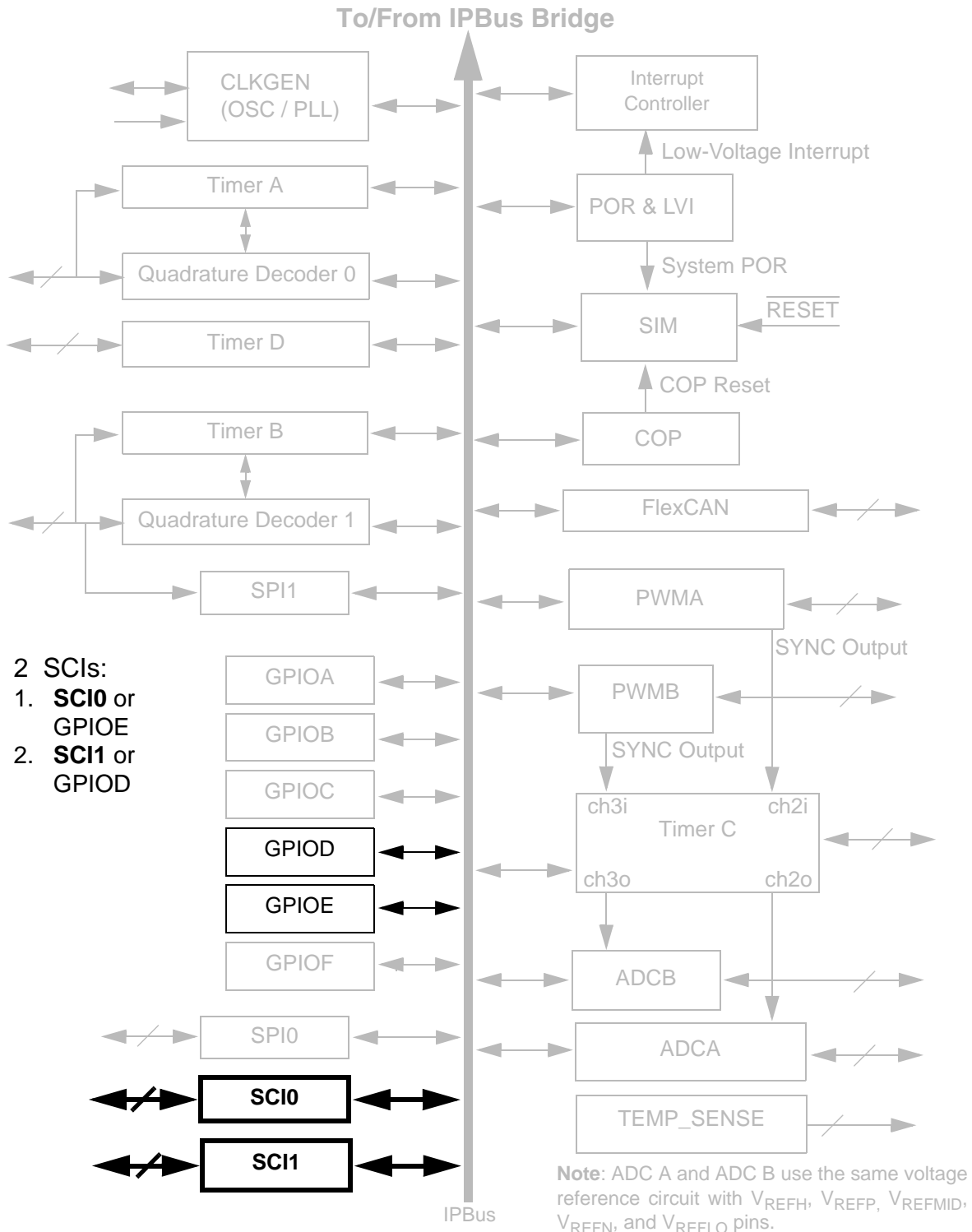
**Note:** Watchdog and HOME interrupts are combined into one interrupt. Software must read DECCR to determine which occurred.

See the chip's data sheet for information or the location of these interrupts within the interrupt vector table.



# Chapter 13

## Serial Communications Interface (SCI)



## Document Revision History for Chapter 13, Serial Communications Interface (SCI)

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 3.0	Corrected Figure 13-11 and Program Sheets reset value of bit 6 from 0 to 1
Rev 5.0	Converted chapter to Freescale design standards Added Table 13-4 <i>Example Baud Rates (Module Clock = 40MHz)</i> on page 7



## 13.1 Introduction

This chapter describes the Serial Communications Interface (SCI) module. The module allows asynchronous serial communications with peripheral devices and other controllers.

## 13.2 Features

- Full-duplex or Single-Wire Operation
- Standard mark/space Non-Return-to-Zero (NRZ) format
- 13-bit baud rate selection
- Programmable 8- or 9-bit data format
- Separately enabled transmitter and receiver
- Separate receiver and transmitter interrupt requests
- Programmable polarity for transmitter and receiver
- Two receiver wake-up methods:
  - Idle line
  - Address mark
- Interrupt-driven operation with seven flags:
  - Transmitter empty
  - Transmitter idle
  - Receiver full
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

## 13.3 Block Diagram

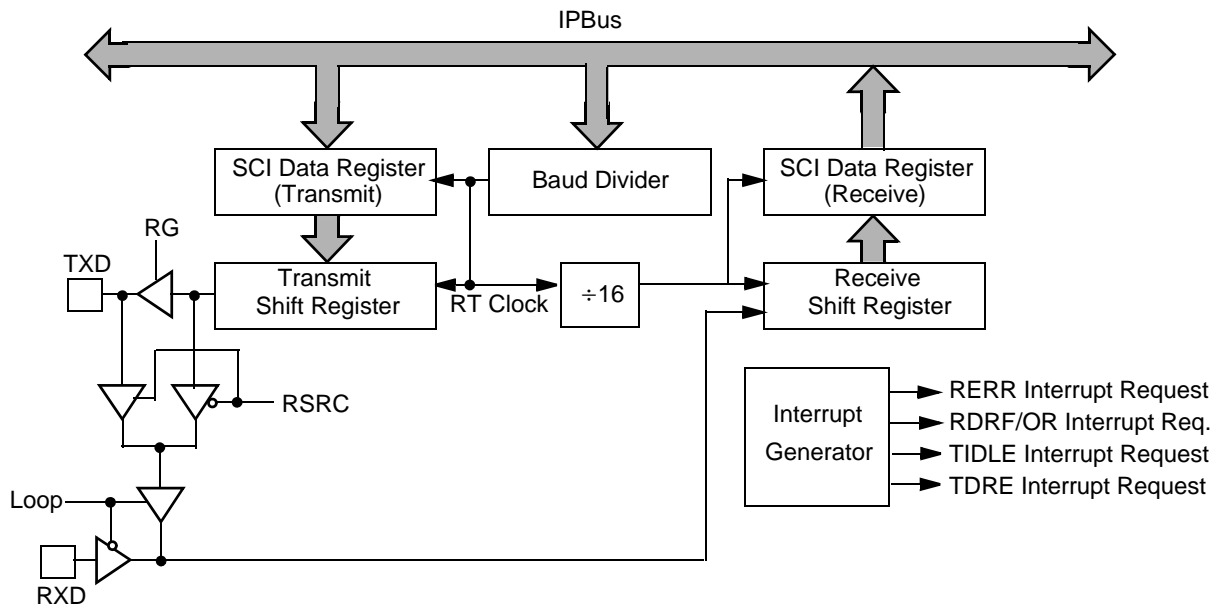


Figure 13-1. SCI Block Diagram

## 13.4 Functional Description

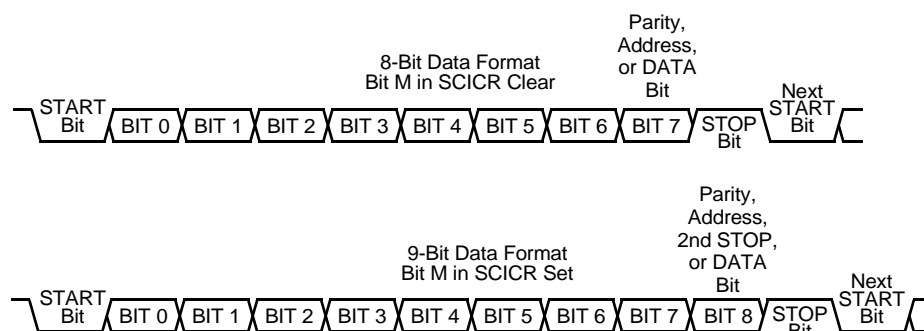
**Figure 13-1** explains the SCI module structure. The SCI allows full duplex, asynchronous, Non-Return-to-Zero (NRZ) serial communication between the controller and remote devices, including other controllers. The SCI transmitter and receiver operate independently although they use the same baud rate generator. The controller monitors the status of the SCI, writes the data to be transmitted, and processes received data.

The SCI has one input signal and one output signal. Data transmits on the TXD pin and receives on the RXD pin. SCI Data Register (SCIDR) holds received bytes and bytes to be transmitted, actually consists of two physically different registers. To send software writes a byte to SCIDR; to receive, software reads a byte from SCIDR. However, if software has not read the first byte by the time the second byte is received, the second byte will be lost and Overrun (OR) flag will be set.

When initializing the SCI, be certain to set the proper peripheral enable bits in the General Purpose Input/Output (GPIO) registers as well as any pull-up enables if the SCI pins are multiplexed with GPIO pins.

## 13.4.1 Data Frame Format

The SCI uses the standard NRZ mark/space data frame format illustrated in [Figure 13-2](#).



**Figure 13-2. SCI Data Frame Formats**

Each data character is contained in a frame including a START bit, eight or nine data bits, and a STOP bit. Clearing the Mode (M) bit in the SCI Control Register (SCICR) configures the SCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits. Formats are provided in [Figure 13-1](#).

**Table 13-1. Example 8-Bit Data Frame Formats**

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
1	8	0	0	1
1	7	0	1	1
1	7	1 <sup>1</sup>	0	1

1. The address bit identifies the frame as an address character. Please see [Section 13.4.4.6, Receiver Wake-Up](#)

Setting the M bit configures the SCI for 9-bit data characters. A frame with nine data bits has a total of 11 bits. Formats are provided in [Table 13-2](#).

**Table 13-2. Example 9-Bit Data Frame Formats**

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
1	9	0	0	1
1	8	0	0	2 <sup>2</sup>
1	8	0	1	1
1	8	1 <sup>1</sup>	0	1

1. The address bit identifies the frame as an address character. Please see [Section 13.4.4.6, Receiver Wake-Up](#)

2. The user must implement the second stop bit by setting the MSB of the data bits when transmitting and by masking the MSB when receiving

## 13.4.2 Baud Rate Generation

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and the transmitter. A value of 1 to 8191 written to the SBR bit in the SCI Baud Rate (SCIBR) register determines the module clock divisor. A value of zero disables the baud rate generator. The clock generated by SCIBR is called RT clock; a frequency of 16 times the baud rate. The RT clock is synchronized with the IPBus clock, driving the receiver. The RT 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 two sources of error:

1. The Integer division of the module clock may not give the exact target frequency.
2. Synchronization with the bus clock can cause phase shift.

**Table 13-3** lists examples of achieving target baud rates with a module clock frequency of 60MHz.

**Table 13-3. Example Baud Rates (Module Clock = 60MHz)**

SBR Bits	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
98	612,245	38,265	38,400	-0.35
195	307,692	19,231	19,200	0.16
391	153,453	9,591	9,600	-0.10
781	76,825	4,802	4,800	0.03
1563	38,388	2,399	2,400	-0.03
3125	19,200	1,200	1,200	0.00
6250	9,600	600	600	0.00

**Note:** Maximum baud rate is IPBus clock rate divided by 16. System overhead may preclude processing the data at this speed.

**Table 13-4** lists examples of achieving target baud rates with a module clock frequency of 40MHz.

**Table 13-4. Example Baud Rates (Module Clock = 40MHz)**

SBR Bits	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (%)
65	615,384.6	38,461.5	38,400	0.16
130	307,692.3	19,230.8	19,200	0.16
260	153,846.1	9,615.4	9,600	0.16
521	76,775.4	4,798.5	4,800	0.03
1042	38,387.7	2,399.2	2,400	0.03
2083	19,203.1	1,200.2	1,200	0.02
4167	9,599.2	600.0	600	0.01

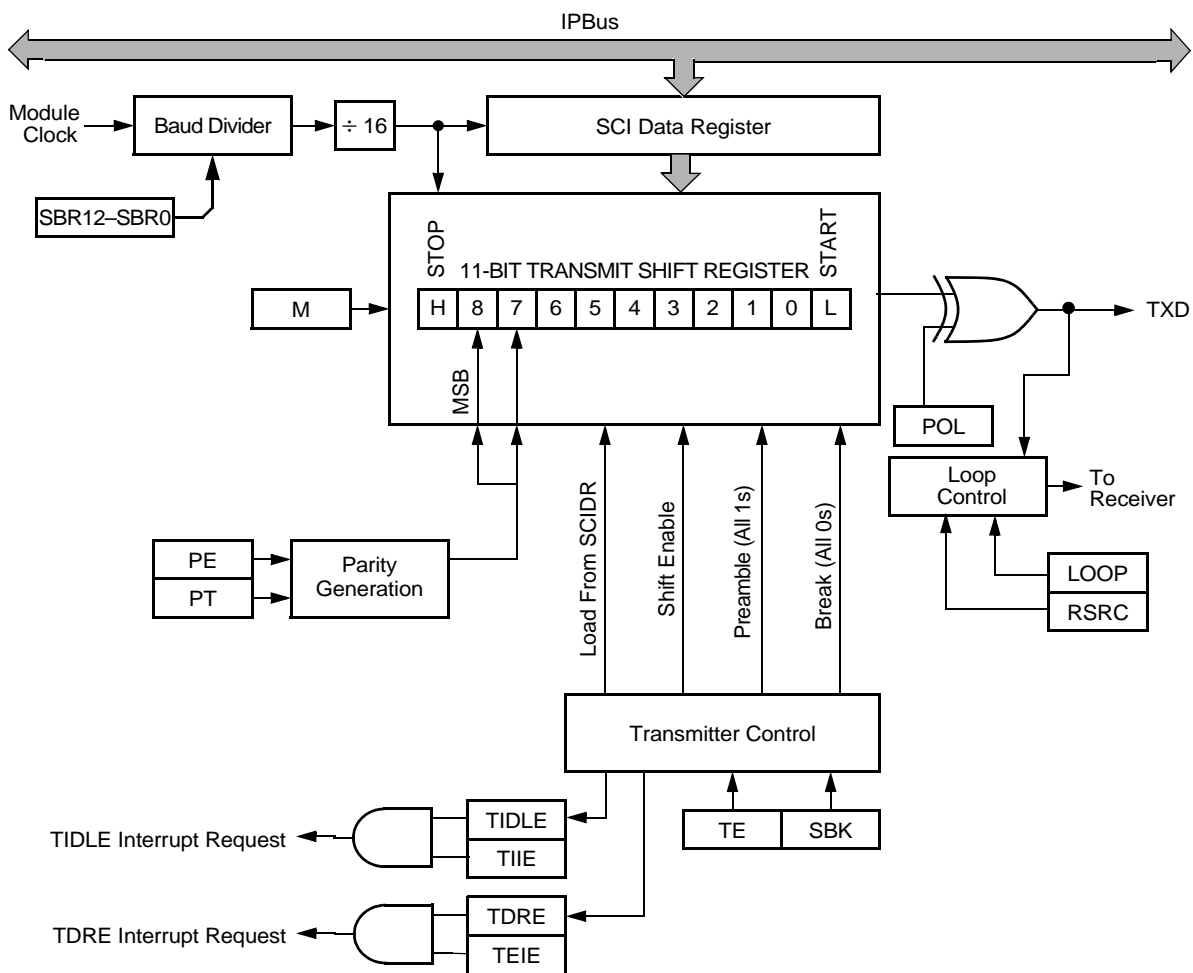
**Note:** Maximum baud rate is IPBus clock rate divided by 16. System overhead may preclude processing the data at this speed.

### 13.4.3 Transmitter

**Figure 13-3** illustrates the transmitter functions block diagram with detailed discussion of the transmitter in the following sections.

#### 13.4.3.1 Character Length

The SCI transmitter accommodates either 8- or 9-bit data characters, determined by the state of the M bit in the SCICR.



**Figure 13-3. SCI Transmitter Block Diagram**

### 13.4.3.2 Character Transmission

During an SCI transmission, the Transmit Shift register moves a frame out to the TXD pin. The SCI Data Register (SCIDR) is the buffer between the internal data bus and the Transmit Shift registers.

Modifying the Transmitter Enable (TE) bit from zero to one automatically loads the Transmit Shift register with a preamble containing all Logic 1s with no START, STOP, or PARITY bit. After the preamble shifts out, control logic automatically transfers the data from the SCIDR into the Transmit Shift register. A Logic 0 START bit automatically goes into the Least Significant Bit (LSB) position of the Transmit Shift register. A Logic 1 STOP bit goes into the Most Significant Bit (MSB) position of the frame.

Hardware supports odd or even parity. When parity is enabled, the MSB of the data character is replaced by the PARITY bit.

The Transmit Data Register Empty (TDRE) flag in the SCISR becomes set when the SCIDR transfers a character to the Transmit Shift register. The TDRE flag indicates the SCIDR can accept new transmit data. If the TEIE bit in the SCICR is also set, the TDRE flag generates a transmitter empty interrupt request.

When the SCI Transmit Shift register is not transmitting a frame and  $TE = 1$ , the TXD pin goes to the idle condition, Logic 1. If software clears TE while a transmission is in progress, the frame in the SCI Transmit Shift register continues to shift-out the transmitter, then relinquishes control of the port I/O pin upon completion of the current transmission. This action causes the TXD pin to go into a HighZ state even if there is data pending in the SCIDR. 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 initiate a SCI transmission:

1. Enable the transmitter by writing a Logic 1 to the TE bit in the SCICR.
2. Wait for the TDRE flag to be set
3. Clear the TDRE flag by first reading the SCISR, then write to the SCIDR.
4. Repeat Steps 2 and 3 for each subsequent transmission.

To separate messages with preambles having minimum idle line time, use this sequence between messages:

1. Write the last character of the first message to SCIDR.
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, then set the TE bit.
4. Write the first character of the second message to SCIDR.

### 13.4.3.3 Break Characters

Writing a Logic 1 to the Send Break (SBK) bit in the SCICR loads the Transmit Shift register with a break character. A break character contains all Logic 0s without START, STOP, or PARITY bits. Break character length depends on the Mode (M) bit in the SCICR. 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 Transmit Shift register finishes transmitting the last break character subsequently transmitting at least one Logic 1. The automatic Logic 1 at the end of the last break character guarantees the recognition of the START bit of the next frame.

The SCI 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 SCI registers:

- Sets the Framing Error (FE) flag
- Sets the Receive Data Register Full (RDRF) flag
- Clears the SCI Data Register (SCIDR)
- May set the Overrun (OR) flag, Noise Flag (NF), Parity Error (PE) flag, or the Receiver Active Flag (RAF). Please see SCISR in [Section 13.6.3](#).

### 13.4.3.4 Preambles

A preamble contains all Logic 1s with no START, STOP, or PARITY bit. A preamble length depends on the M bit in the SCICR. The preamble is a synchronizing mechanism initiating the first transmission begun after modifying the TE bit from zero to one.

To queue a preamble between two transmissions:

1. After writing the last character of the first transmission to the Transmit register, wait for TDRE flag to be set.
2. When the TDRE flag becomes set, clear and set the TE bit. This will queue a preamble after the last character of the first transmission.
3. After setting the TE bit, immediately write the first character of the second transmission to the SCIDR.

## 13.4.4 Receiver

[Figure 13-4](#) illustrates the block diagram of the SCI receiver function.

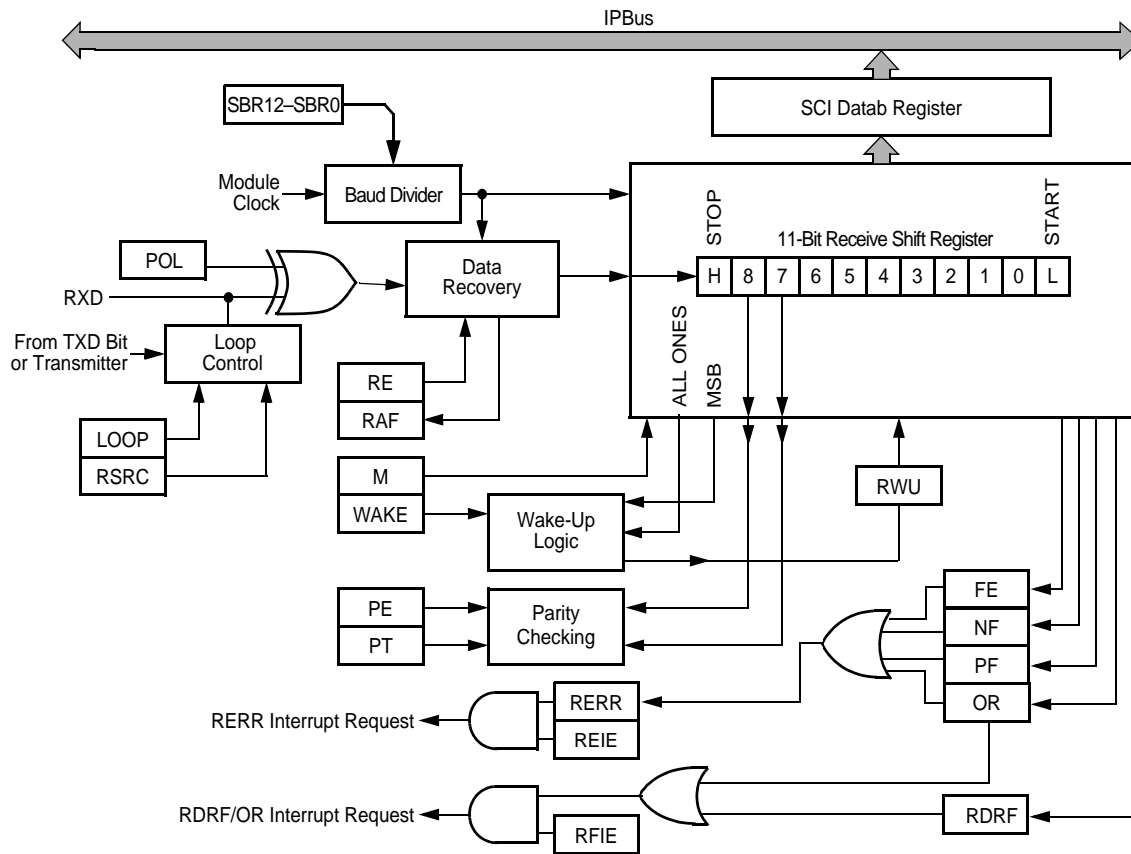


Figure 13-4. SCI Receiver Block Diagram

### 13.4.4.1 Character Length

The SCI receiver can accommodate either 8- or 9-bit data characters determined by the state of the M bit in the SCICR.

### 13.4.4.2 Character Reception

During an SCI reception, the Receive Shift register alters a frame in from the RXD pin. The data is read from the SCIDR.

After a complete frame shifts into the Receive Shift register, the data portion of the frame transfers to the SCIDR. The Receive Data Register Full (RDRF) flag in the SCISR is set, indicating the received character can be read. If the Receive Full Interrupt Enable (RFIE) bit in the SCICR is also set, the RDRF flag generates an RDRF interrupt request.

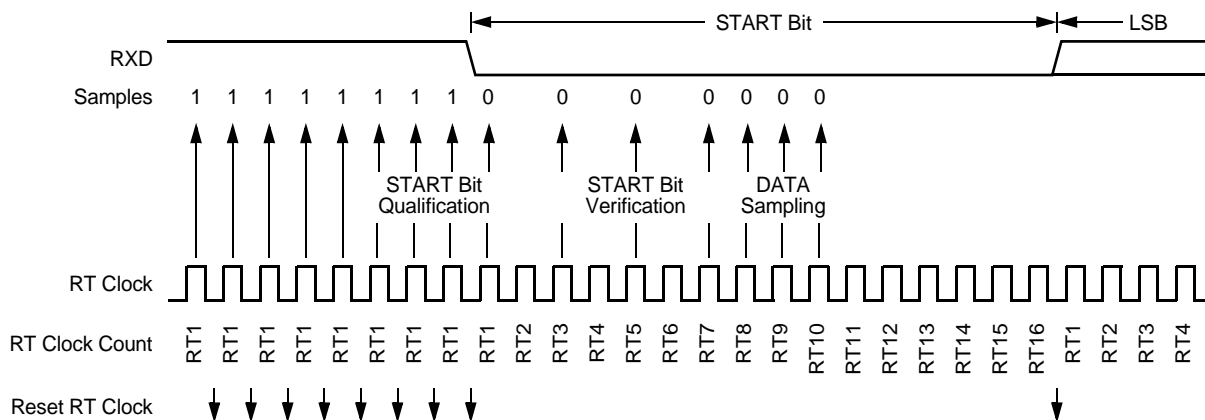


### 13.4.4.3 Data Sampling

The receiver samples the RXD pin at the Rate Tolerance (RT) clock rate. To adjust the baud rate mismatch, the RT clock illustrated in [Figure 13-5](#), is resynchronized:

- After every START bit
- After the receiver detects a data bit change from Logic 1 to 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 13-5. Receiver Data Sampling**

To verify the START bit and detect noise, data recovery logic takes samples at RT3, RT5, and RT7. [Table 13-5](#) summarizes the results of the START bit verification samples and the Noise Flag (NF). A majority vote of the three samples is used as the value of the bit.

**Table 13-5. 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
110	No	0
111	No	0

If two of three samples are 0s (not all 0s), Noise Flag (NF) is set. If START bit verification is not successful, the RT clock is reset and a new search for a START bit begins.

To determine the value of a DATA bit and to detect noise, data recovery logic takes samples at RT8, RT9, and RT10. A majority vote of the three samples is used as the value of the bit. **Table 13-6** summarizes the results of the DATA bit samples. If all three samples are not the same, Noise Flag (NF) is set.

**Table 13-6. 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 NF is set and the receiver assumes the bit is a START bit (Logic 0).

To verify a STOP bit and to detect noise, data recovery logic takes samples at RT8, RT9, and RT10. A majority vote of the three samples is used as the value of the bit. **Table 13-7** summarizes the results of the STOP bit samples. If all three samples are not the same, Noise Flag (NF) is set. If STOP bit detecting fails, Framing Error Flag (FE) is set.

**Table 13-7. 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

### 13.4.4.4 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 (FE) flag in SCISR. Reception of a break character also sets the FE flag because a break character has no STOP bit. The FE flag is set concurrently with the RDRF flag. The FE flag inhibits further data reception until it is cleared. The FE flag is cleared by reading the SCISR, thereafter write any value to the SCISR.

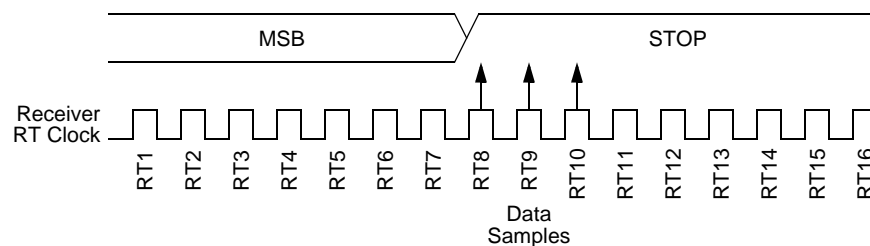
### 13.4.4.5 Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause either Noise Error, Framing Error, or both.

As the receiver samples an incoming frame, it resynchronizes the RT clock on any valid falling edge within the frame. Resynchronization within frames may correct misalignments between transmitter bit times and receiver bit times. The worst case is the data is all 0s or 1s without resynchronization occurring during the frame transmit.

#### 13.4.4.5.1 Slow Data Tolerance

**Figure 13-6** illustrates how much a slow received frame can be misaligned without causing a noise or framing error. The slow STOP bit begins at RT8 instead of RT1, but it arrives in time for the STOP bit data samples at RT8, RT9, and RT10.



**Figure 13-6. Slow Data**

For an 8-bit (all 0s) data character, data sampling of the STOP bit takes the receiver:

$$9\text{-bit} \times 16 \text{ RT cycles} + 10 \text{ RT cycles} = 154 \text{ RT cycles}$$

With the misaligned character, illustrated in **Figure 13-6**, the receiver counts 154 RT cycles at the point when the count of the transmitting device is:

$$9\text{-bit} \times 16 \text{ RT cycles} + 3 \text{ RT cycles} = 147 \text{ RT cycles}$$

The maximum percentage difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

For a 9-bit (all 0s) data character, data sampling of the STOP bit takes the receiver:

$$\left| \frac{154-147}{154} \right| \times 100 = 4.54\%$$

$$10\text{-bit} \times 16 \text{ RT cycles} + 10 \text{ RT cycles} = 170 \text{ RT cycles}$$

With the misaligned character illustrated in [Figure 13-6](#), the receiver counts 170 RT cycles at the point when the count of the transmitting device is:

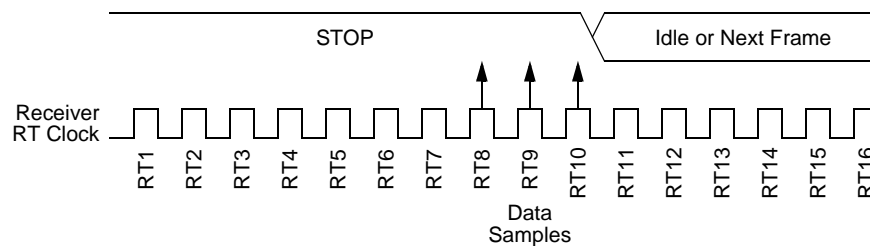
$$10\text{-bit} \times 16 \text{ RT cycles} + 3 \text{ RT cycles} = 163 \text{ RT cycles}$$

The maximum percentage difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$\left| \frac{170-163}{170} \right| \times 100 = 4.12\%$$

#### 13.4.4.5.2 Fast Data Tolerance

[Figure 13-7](#) illustrates how much a fast received frame can be misaligned without causing a noise error or a framing error. The fast STOP bit ends at RT10 instead of RT16 but it is still sampled at RT8, RT9, and RT10.



**Figure 13-7. Fast Data**

For an 8-bit (all 0s or 1s) data character, data sampling of the STOP bit takes the receiver:

$$9\text{-bit} \times 16 \text{ RT cycles} + 10 \text{ RT cycles} = 154 \text{ RT cycles}$$

With the misaligned character, illustrated in [Figure 13-7](#), the receiver counts 154 RT cycles at the point when the count of the transmitting device is:

$$10\text{-bit} \times 16 \text{ RT cycles} = 160 \text{ RT cycles}$$

The maximum percentage difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

For a 9-bit (all 0s or 1s) data character, data sampling of the STOP bit takes the receiver:

$$\left| \frac{154-160}{154} \right| \times 100 = 3.90\%$$

$$10\text{-bit} \times 16 \text{ RT cycles} + 10 \text{ RT cycles} = 170 \text{ RT cycles}$$

With the misaligned character, illustrated in [Figure 13-7](#), the receiver counts 170 RT cycles at the point when the count of the transmitting device is:

$$11\text{-bit} \times 16 \text{ RT cycles} = 176 \text{ RT cycles}$$

The maximum percentage difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

$$\left| \frac{170-176}{170} \right| \times 100 = 3.53\%$$

#### 13.4.4.6 Receiver Wake-Up

In order for the SCI 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 (RWU) bit in the SCICR puts the receiver into a standby state while receiver interrupts are disabled.

The transmitting device can address messages to selected receivers by including addressing information in the initial frame or frames of each message.

The WAKE bit in the SCICR determines how the SCI is brought out of the standby state to process an incoming message. The WAKE bit enables either Idle Input Line Wake-Up or Address Mark Wake-Up:

- Idle Input Line Wake-Up (WAKE = 0)—In this wake-up method, an idle condition on the RXD pin clears the RWU bit, waking up the SCI. Idle Input Line Wake-Up requires messages be separated by at least one preamble, and no message contains preambles. The initial frame or frames of every message contains addressing information. All receivers evaluate the addressing information. Receivers of the message then process the following frames. Any receiver a message does not address can set its RWU bit, returning to the standby state. The RWU bit remains set and the receiver remains on standby until another preamble appears on the RXD pin.

The receiver-waking preamble does not set the Receiver Idle (RIDLE) bit or the Receive Data Full Register (RDRF) flag.

With the WAKE bit clear, setting the RWU bit after the RXD pin has been idle can cause the receiver to wake-up immediately.

- Address Mark Wake-Up (WAKE = 1)—In this wake-up method, a Logic 1 in the MSB position of a frame clears the RWU bit, awakening the SCI. The Logic 1 in the MSB position marks a frame as an address frame containing addressing information. The address frame also sets the RDRF bit in the SCISR. All receivers evaluate the addressing information. Receivers of the message then process the following frames. Any receiver a message does not address can set its RWU bit, returning to the standby state. The RWU bit remains set and the receiver remains on standby until another address frame appears on the RXD pin. Address Mark Wake-Up allows messages to contain preambles but it requires the MSB to be reserved for use in address frames.

## 13.5 Special Operating Modes

**Table 13-8** summarizes how to configure for Normal Operation, Loop Back, or Single-Wire Operation as described in [Section 13.5.1](#) and [Section 13.5.2](#).

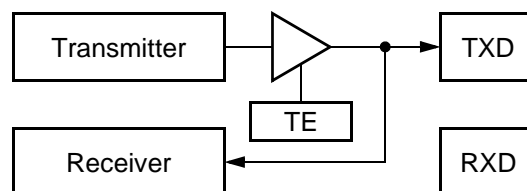
**Table 13-8. Loop Functions**

LOOP	RSRC	Function
0	X	Normal Operation
1	0	Loop Mode with Internal TXD Fed Back to RXD
1	1	Single-Wire Mode with TXD Output Fed Back to RXD

### 13.5.1 Single-Wire Operation

Normally, the SCI uses two pins for transmitting and receiving. In the Single-Wire Operation, the RXD pin is disconnected from the SCI and is available for other peripherals, illustrated in [Figure 13-8](#). The SCI uses the TXD pin for both receiving and transmitting.

Setting the TE bit in the SCICR enables the transmitter, configuring TXD as the output for transmitted data. Clearing the TE bit disables the transmitter, configuring TXD as the input for received data.



**Figure 13-8. Single-Wire Operation (LOOP = 1, RSRC = 1)**

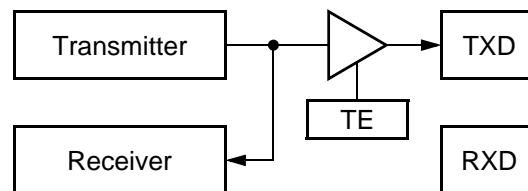
Enable Single-Wire Operation by setting the LOOP bit and the Receiver Source (RSRC) bit in the SCICR. Setting the LOOP bit disables the path from the RXD pin to the receiver. Setting the

RSRC bit connects the receiver input to the output of the TXD pin driver. To enable Receiver, Receiver Enable (RE) bit in the SCICR must be set.

### 13.5.2 Loop Operation

In Loop Operation, the transmitter output goes to the receiver input. The RXD pin is disconnected from the SCI and is available as a GPIO pin. Please see [Figure 13-9](#).

Setting the TE bit in the SCICR enables the transmitter, connecting the transmitter output to the TXD pin. Clearing the TE bit disables the transmitter, disconnecting the transmitter output from the TXD pin.



**Figure 13-9. Loop Operation (LOOP = 1, RSRC = 0)**

Enable Loop Operation by setting the LOOP bit and clearing the RSRC bit in the SCICR. Setting the LOOP bit disables the path from the RXD pin to the receiver. Clearing the RSRC bit connects the transmitter output to the receiver input. To enable Loop Operation both the Transmitter Enable (TE) and Receiver Enable (RE) bits in the SCICR must be set.

### 13.5.3 Low-Power Options

#### 13.5.3.1 Run Mode

Clearing the Transmitter Enable (TE) or Receiver Enable (RE) bits in the SCICR reduces power consumption in Run mode. SCI registers are still accessible when TE or RE bits are cleared, but clocks to the SCI module are disabled.

#### 13.5.3.2 Wait Mode

SCI operation in Wait mode depends on the state of the SWAI bit in the SCICR.

- If SWAI is clear, SCI operates normally when CPU is in Wait mode.
- If SWAI is set, SCI clock generation ceases and the SCI module enters a power conservation state when the CPU is in Wait mode. Setting SWAI does not affect the state of the RE bit or the TE bit.

When SWAI is set, any transmission or reception in progress stops at Wait mode entry. The transmission or reception resumes when either an internal or external interrupt brings the processor out of Wait mode.

When SWAI is set the SCI module cannot generate interrupt requests during Wait mode.

Any enabled SCI interrupt request can bring the processor out of Wait mode as long as SWAI is clear.

### 13.5.3.3 Stop Mode

The SCI is inactive in STOP mode for reduced power consumption. The Stop instruction does not affect the register states. SCI operation resumes after an external interrupt brings the processor out of Stop mode.

## 13.6 Register Definitions

**Table 13-9. SCI Memory Map**

Device	Peripheral	Address
8100/8300	SCI0_BASE	\$00F280
	SCI1_BASE	\$00F290

There are four accessible registers on SCI described in [Table 13-10](#) and summarized in [Table 13-11](#). A register address is the sum of a base address and an address offset. The base address is defined at the system level and the address offset is defined at the module level. The SCI has four registers.

**Table 13-10. SCI Register Summary**

Address Offset	Register Acronym	Register Description	Access Type	Chapter Location
Base + \$0	SCIBR	Baud Rate Register	Read/Write	<a href="#">Section 13.6.1</a>
Base + \$1	SCICR	Control Register	Read/Write	<a href="#">Section 13.6.2</a>
		Reserved		
Base + \$3	SCISR	Status Register	<i>Read-Only</i>	<a href="#">Section 13.6.3</a>
Base + \$4	SCIDR	Data Register	Read/Write	<a href="#">Section 13.6.4</a>

Bit fields of each of the four registers are illustrated in [Figure 13-10](#). Details of each follow.



Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$0	SCIBR	R	0	0	0	SBR												
		W																
\$1	SCICR	R	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIE	RFIE	REIE	TE	RE	RWU	SBK
		W																
RESERVED																		
\$3	SCISR	R	TDRE	TIDLE	RDRF	RIDLE	OR	NF	FE	PF	0	0	0	0	0	0	0	RAF
		W																
\$4	SCIDR	R	0	0	0	0	0	0	0	RECEIVE DATA								
		W								TRANSMIT DATA								

R	0	Read as 0
W		Reserved

Figure 13-10. SCI Register Map Summary

### 13.6.1 SCI Baud Rate Register (SCIBR)

This register can be read at any time. Bits 12 through 0 can be written at any time, but bits 15 through 13 are reserved.

Base + \$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	SBR												
Write																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Figure 13-11. SCI Baud Rate Register (SCIBR)

The count in this register determines the baud rate of the SCI. The formula for calculating baud rate is:

$$\text{SCI Baud Rate} = \frac{\text{IPBus Clock}}{16 \times \text{SBR}}$$

See [Section 13.4.2](#) for more details and examples.

**Note:** The baud rate generator is disabled until the TE or the RE bits are set for the first time after reset. The baud rate generator is disabled when SBR = 0.

#### 13.6.1.1 Reserved—Bits 15–13

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 13.6.1.2 SCI Baud Rate (SBR)—Bits 12–0

Contents of the Baud Rate register has a value of 1 to 8191.

## 13.6.2 SCI Control Register (SCICR)

The SCI Control Register (SCICR) can be read and written at anytime.

Base + \$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIIE	RFIE	REIE	TE	RE	RWU	SBK
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 13-12. SCI Control Register (SCICR)**

### 13.6.2.1 Loop Select Bit (LOOP)—Bit 15

This bit enables loop operation. Please see [Section 13.5.2](#). The loop operation disconnects the RXD pin from the SCI and the transmitter output goes into the receiver input.

- 0 = Normal operation enabled
- 1 = Loop operation enabled

### 13.6.2.2 Stop in Wait Mode (SWAI)—Bit 14

The SWAI bit disables the SCI in Wait mode. Please see [Section 13.5.3.2](#).

- 0 = SCI enabled in Wait mode
- 1 = SCI disabled in Wait mode

### 13.6.2.3 Receiver Source (RSRC)— Bit 13

When LOOP = 1, the RSRC bit determines the internal feedback path for the receiver. See [Section 13.5.1](#) and [Section 13.5.2](#) for more details.

- 0 = Receiver input internally connected to transmitter output
- 1 = Receiver input connected to TXD pin

### 13.6.2.4 Data Format Mode (M)—Bit 12

This bit determines whether data characters are eight or nine bits long.

- 0 = One START bit, eight data bits, one STOP bit
- 1 = One START bit, nine data bits, one STOP bit

### 13.6.2.5 Wake-Up Condition (WAKE)—Bit 11

This bit determines which condition wakes up the SCI.

- 0 = Idle Line Wake-Up
- 1 = Address Mark Wake-Up (a Logic 1 in the MSB position of a receive data character)

**Note:** Address Mark Wake-Up is not a valid option when the parity function is enabled (PE = 1) since the ADDRESS bit and the PARITY bit both occupy the MSB.

### 13.6.2.6 Polarity (POL)—Bit 10

This bit determines whether to invert the data as it goes from the transmitter to the TXD pin and from the RXD pin to the receiver. All bits, START, DATA, and STOP, are inverted as they leave the Transmit Shift register and before they enter the Receive Shift register.

- 0 = Doesn't invert transmit and receive data bits (Normal mode)
- 1 = Invert transmit and receive data bits (Inverted mode)

**Note:** It is recommended the POL bit be toggled only when both TE and RE = 0.

### 13.6.2.7 Parity Enable (PE)—Bit 9

This bit enables the parity function. When enabled, the parity function replaces the MSB of the data character with a parity bit.

- 0 = Parity function disabled
- 1 = Parity function enabled

**Note:** Address Mark Wake-Up (WAKE = 1) is not a valid option when the parity function is enabled because the ADDRESS bit and the PARITY bit both occupy the MSB.

### 13.6.2.8 Parity Type (PT)—Bit 8

This bit determines whether the SCI generates and checks for even or odd parity of the data bits. With *even parity*, an *even* number of *ones clears* the PARITY bit while an *odd* number of *ones, sets* the PARITY bit. However, with *odd parity*, an *odd* number of *ones clears* the PARITY bit while an *even* number of *ones, sets* the PARITY bit.

- 0 = Even parity
- 1 = Odd parity

### 13.6.2.9 Transmitter Empty Interrupt Enable (TEIE)—Bit 7

This bit enables the TDRE flag to generate interrupt requests.

- 0 = TDRE interrupt requests disabled
- 1 = TDRE interrupt requests enabled

### 13.6.2.10 Transmitter Idle Interrupt Enable (TIIE)—Bit 6

This bit enables the TIDLE flag to generate interrupt requests.

- 0 = TIDLE interrupt requests disabled
- 1 = TIDLE interrupt requests enabled

### 13.6.2.11 Receiver Full Interrupt Enable (RFIE)—Bit 5

This bit enables the RDRF flag or the OR flag to generate interrupt requests.

- 0 = RDRF and OR interrupt requests disabled
- 1 = RDRF and OR interrupt requests enabled

### 13.6.2.12 Receive Error Interrupt Enable (REIE)—Bit 4

This bit enables the Receive Error (RE) flags (NF, PF, FE, and OR) to generate interrupt requests. The status bits can be checked during the error interrupt process.

- 0 = Error interrupt requests disabled
- 1 = Error interrupt requests enabled

### 13.6.2.13 Transmitter Enable (TE)—Bit 3

This bit enables the SCI transmitter and configures the TXD pin as the SCI transmitter output. The TE bit can be used to queue an idle preamble.

- 0 = Transmitter disabled
- 1 = Transmitter enabled

### 13.6.2.14 Receiver Enable (RE)—Bit 2

This bit enables the SCI Receiver.

- 0 = Receiver disabled
- 1 = Receiver enabled

### 13.6.2.15 Receiver Wake-Up (RWU)—Bit 1

This bit enables the wake-up function, inhibiting further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU. Please refer to [Section 13.4.4.6](#) for a description of Receive Wake-Up.

- 0 = Normal operation
- 1 = Standby state

### 13.6.2.16 Send Break (SBK)—Bit 0

Setting SBK sends one break character (all Logic 0s, include START, DATA, STOP). As long as SBK is set, transmitter sends uninterrupted break characters.

- 0 = No break characters
- 1 = Transmit break characters

## 13.6.3 SCI Status Register (SCISR)

This register can be read at anytime; however, it cannot be modified by writing. Writes clear flags.

Base + \$3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TDRE	TIDLE	RDRF	RIDLE	OR	NF	FE	PF	0	0	0	0	0	0	0	RAF
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-13. SCI Status Register (SCISR)

### 13.6.3.1 Transmit Data Register Empty Flag (TDRE)—Bit 15

This bit is set when the Transmit Shift register receives a character from the SCIDR. Clear TDRE by reading SCISR, then write to the SCIDR.

- 0 = No character transferred to Transmit Shift register
- 1 = Character transferred to Transmit Shift register, TDRE

### 13.6.3.2 Transmitter Idle Flag (TIDLE)—Bit 14

This bit is set when the TDRE flag is set and no data, preamble, or break character is being transmitted. When TIDLE is set, the TXD pin becomes idle (Logic 1). Clear TIDLE by reading the SCISR, then write to the SCIDR.

- 0 = Transmission in progress

- 1 = No transmission in progress

### 13.6.3.3 Receive Data Register Full Flag (RDRF)—Bit 13

This bit is set when the data in the Receive Shift register transfers to the SCIDR. Clear RDRF by reading the SCISR, then read the SCIDR.

- 0 = Data not available in SCIDR
- 1 = Received data available in SCIDR

### 13.6.3.4 Receiver Idle Line Flag (RIDLE)—Bit 12

This bit is set when ten consecutive Logic 1s (if M = 0) or eleven consecutive Logic 1s (if M = 1) appear on the receiver input. Once the RIDLE flag is cleared by the receiver detecting a Logic 0, a valid frame must again set the RDRF flag before an idle condition can set the RIDLE flag.

- 0 = Receiver input is either active now or has never become active since the RIDLE flag was last cleared by reset
- 1 = Receiver input has become idle (after receiving a valid frame)

**Note:** When the Receiver Wake-Up (RWU) bit is set, an idle line condition does not set the RIDLE flag.

### 13.6.3.5 Overrun Flag (OR)—Bit 11

This bit is set when software fails to read the SCIDR before the Receive Shift register receives the next frame. The data in the Shift register is lost, but the data already in the SCIDR is not affected. Clear OR by reading the SCISR, then write the SCISR with any value.

- 0 = No overrun
- 1 = Overrun

### 13.6.3.6 Noise Flag (NF)—Bit 10

This bit is set when the SCI detects noise on the receiver input. The NF bit is set during the same cycle as the RDRF flag, but it is not set in the case of an overrun. Clear NF by reading the SCISR, then write the SCISR with any value.

- 0 = No noise
- 1 = Noise

### 13.6.3.7 Framing Error Flag (FE)—Bit 9

This bit is set when a Logic 0 is accepted as the STOP bit. The FE bit is set during the same cycle as the RDRF flag but it is not set in the case of an overrun. FE inhibits further data reception until it is cleared. Clear FE by reading the SCISR, then write the SCISR with any value.

- 0 = No framing error
- 1 = Framing error

### 13.6.3.8 Parity Error Flag (PF)—Bit 8

This bit is set when the Parity Enable (PE) bit is set and the parity of the received data does not match its parity bit. Clear PF by reading the SCISR, then write the SCISR with any value.

- 0 = No parity error
- 1 = Parity error

### 13.6.3.9 Reserved—Bits 7–1

These bits are reserved or not implemented. They are read as 0 and cannot be modified by writing.

### 13.6.3.10 Receiver Active Flag (RAF)—Bit 0

This bit 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 false start bits (usually from noise or baud rate mismatch) or when the receiver detects a preamble.

- 0 = No reception in progress
- 1 = Reception in progress

## 13.6.4 SCI Data Register (SCIDR)

The SCIDR can be read and modified at any time. Reading accesses the SCI Receive Data register. Writing to the register accesses the SCI Transmit Data register.

Base + \$4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	RECEIVE DATA								
Write								TRANSMIT DATA								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-14. SCI Data Register (SCIDR)

### 13.6.4.1 Reserved—Bits 15–9

These bits are reserved or not implemented. They are read as 0 and cannot be modified by writing.

### 13.6.4.2 Receive/Transmit Data—Bits 8–0

Writing to these bits loads the transmit data. Reading these bits accesses the receive data.

**Note:** When configured for 8-bit data, bits 7 to 0 contain the data received.

## 13.7 Clocks

All timing is derived from the IPBus clock, operating at the system clock rate. Please see [Section 13.4.2](#) for a description of how the data rate is determined.

## 13.8 Resets

Any system reset completely resets the SCI.

## 13.9 Interrupts

**Table 13-11. SCI Interrupt Sources**

Interrupt Source	Flag	Local Enable	Description
Transmitter	TDRE	TEIE	Transmit Empty
	TIDLE	TIIE	Transmit Idle
Receiver	RDRF	RFIE	Receive Full
	OR		
	FE	REIE	Receive Error
	PE		
	NF		
	OR		

### 13.9.1 Transmitter Empty Interrupt

This interrupt is enabled by setting the TEIE bit of the SCICR. When this interrupt is enabled, an interrupt is generated when data is transferred from the SCIDR to the Transmit Shift register.

### 13.9.2 Transmitter Idle Interrupt

This interrupt is enabled by setting the TIIE bit of the SCICR. This interrupt indicates the TIDLE flag is set and the transmitter is no longer sending data, preamble, or break characters. The interrupt service routine should initiate a preamble, a break, write a data character to the SCIDR or disable the transmitter.

### 13.9.3 Receiver Full Interrupt

This interrupt is enabled by setting the RFIE bit of the SCICR. This interrupt indicates either receive data is available in the SCIDR or a data overrun occurred. The interrupt service routine should read SCISR to determine which of RDRF flag, OR flag, or both were set.



### 13.9.4 Receive Error Interrupt

This interrupt is enabled by setting the REIE bit of the SCICR. This interrupt indicates any of the listed errors was detected by the receiver:

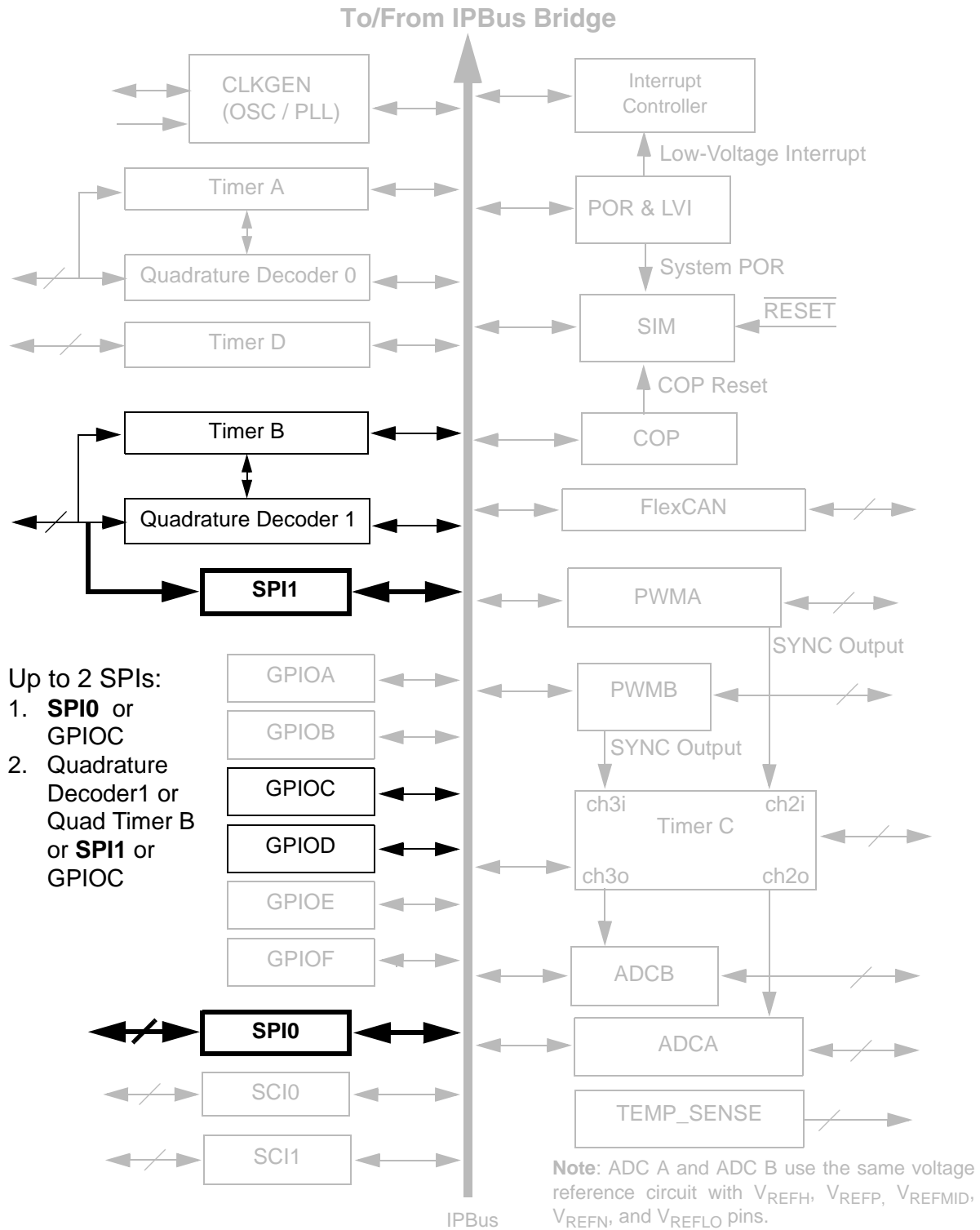
1. Noise Flag (NF) set
2. Parity Error Flag (PF) set
3. Framing Error (FE) flag set
4. Overrun (OR) flag set

The interrupt service routine should read the SCISR to determine which of the error flags was set. The error flag is cleared by writing anything to the SCISR. Then the appropriate action should be taken by the software to handle the error condition.



# Chapter 14

## Serial Peripheral Interface (SPI)



## Document Revision History for [Chapter 14, Serial Peripheral Interface \(SPI\)](#)

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 5.0	Converted chapter to Freescale design standards Added note to SPR Example Section 14.9.1.1.1, page 22 re: 8100 device 40MHz
Rev 6.0	Clarification made to Clock Polarity Bit 7 description (Section 14.9.1.7)

## 14.1 Introduction

This chapter describes the Serial Peripheral Interface (SPI) module. The module allows full-duplex, synchronous, serial communication between the 16-bit controller and peripheral devices, including other 16-bit controllers.

## 14.2 Features

Characteristics of the SPI module include:

- Full-duplex operation
- Master and Slave modes
- Double-buffered operation with separate transmit and receive registers
- Programmable length transmissions (two to 16 bits)
- Programmable transmit and receive shift order (MSB first or last bit transmitted)
- Eight Master mode frequencies (maximum = module clock  $\div$  2)
- Maximum Slave mode frequency = module clock  $\div$  2
- Clock ground for reduced Radio Frequency (RF) interference
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts
  - SPI Receiver Full (SPRF)
  - SPI Transmitter Empty (SPTE)
- Mode Fault Error flag interrupt capability
- Wired OR mode functionality enabling connection to multiple SPIs

**Note:** Throughout this chapter, there are references to the SPI module clock. The SPI module clock is the IPBus clock.

### 14.3 Block Diagram

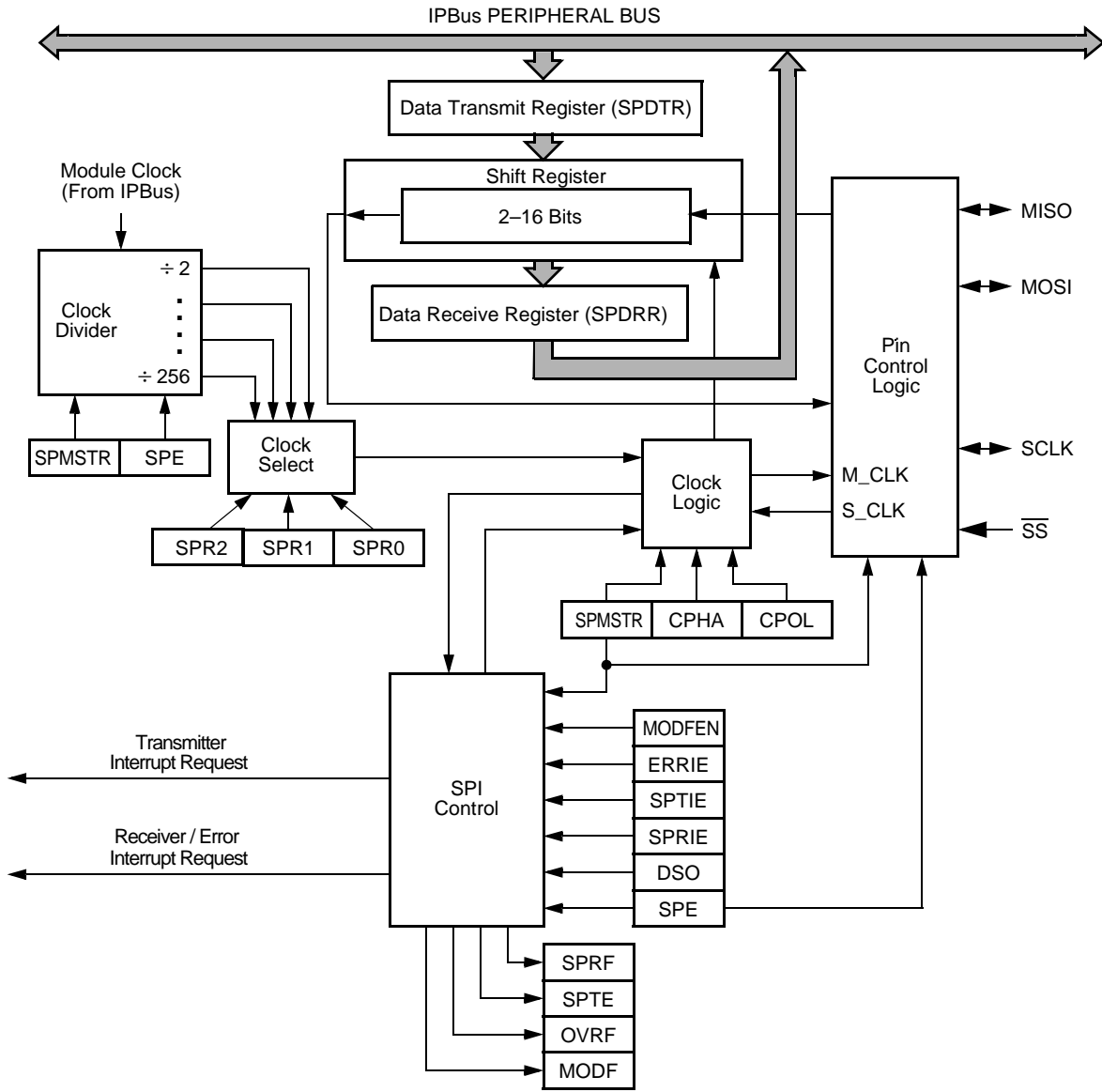


Figure 14-1. SPI Block Diagram

### 14.4 Operating Modes

The SPI has two operating modes:

- Master
- Slave

An operating mode is selected by the SPMSTR bit in the SPI Status and Control Register (SPSCR) as follows:

- SPMSTR = 0 Slave mode
- SPMSTR = 1 Master mode

**Note:** The SPMSTR bit should be configured before enabling the SPI (setting the SPE bit in the SPSCR). The master SPI should be enabled before enabling any slave SPI. All slave SPIs should be disabled before disabling the master SPI.

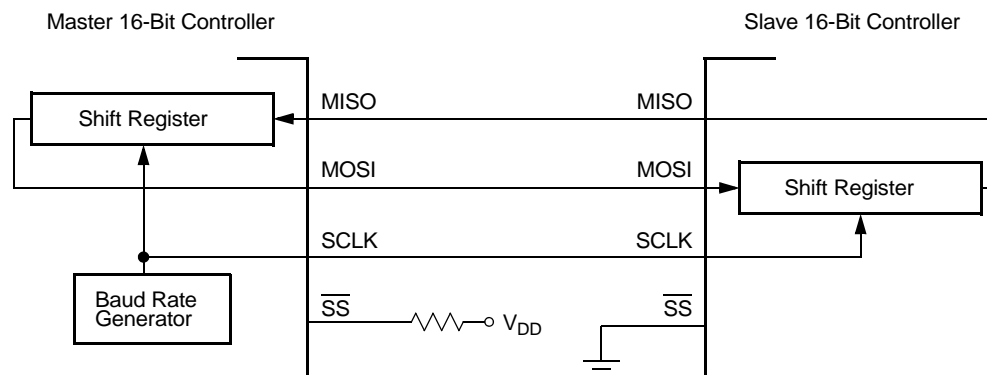
### 14.4.1 Master Mode

The SPI operates in Master mode when the SPI master bit, SPMSTR, is set. Only a master SPI module can initiate transmissions. With the SPI enabled, software begins the transmission from the master SPI module by writing to the SPI Data Transmit Register (SPDTR). If the Shift register is empty, the data immediately transfers to the Shift register, setting the SPI Transmitter Empty (SPTE) bit. The data begins shifting out on the Master Out/Slave In (MOSI) pin under the control of the SPI Serial Clock (SCLK).

The SPR[2:0] bits in the SPI Status and Control Register (SPSCR) control the Baud Rate Generator, determining the speed of the Shift register. The Baud Rate Generator of the master also controls the Shift register of the slave peripheral via the SCLK pin.

As the data shifts out on the MOSI pin of the master, external data shifts in from the slave on the Master In/Slave Out (MISO) pin. The transmission ends when the SPI Receiver Full (SPRF) bit in the SPSCR becomes set. At the same time the SPRF becomes set, the data from the slave transfers to the SPI Data Receive Register (SPDRR). In a normal operation, SPRF signals the end of a transmission. Software clears the SPRF by reading the SPSCR with SPRF set and then reading the SPDRR. Writing to the SPDTR clears the SPTE bit.

**Figure 14-2** is an example configuration for a Full-Duplex Master-Slave Configuration. Having the Slave Select ( $\overline{SS}$ ) bit of the master 16-bit controller held high is only necessary if MODFEN = 1. Tying the Slave 16-bit controller  $\overline{SS}$  bit to ground should only be executed if CPHA = 1.



**Figure 14-2. Full Duplex Master/Slave Connections**

## 14.4.2 Slave Mode

The SPI operates in Slave mode when the SPMSTR bit is cleared. While in Slave mode, the SCLK pin acts as the input for the serial clock from the master 16-bit controller. Before a data transmission occurs, the  $\overline{SS}$  pin of the slave SPI must be at Logic 0.  $\overline{SS}$  must remain low until the transmission is complete or a Mode Fault Error occurs.

**Note:** The SPI must be enabled (SPE = 1) for slave transmissions to be received.

**Note:** Data in the transmitter Shift register will be unaffected by SCLK transitions in the event the SPI is operating as a slave but is deselected.

In a slave SPI module, data enters the Shift register under the control of the Serial Clock, (SCLK), from the master SPI module. After a full length data transmission enters the Shift register of a slave SPI, it transfers to the SPI Data Receive Register (SPDDR) and the SPI Receiver Full (SPRF) bit in the SPSCR is set. If the SPI Receive Interrupt Enable (SPRIE) bit in the SPSCR is set, a Receive Interrupt is also generated. To prevent an overflow condition, slave software must read the SPDDR before another full length data transmission enters the Shift register.

The maximum frequency of the SCLK for the SPI configured as a slave is the module clock  $\div 2$ , or half the module clock. Frequency of the SCLK for the SPI configured as a slave does not have to correspond to any particular SPI baud rate. The baud rate only controls the speed of the SCLK generated by the SPI configured as a master. Therefore, the frequency of the SCLK for a SPI configured as a slave can be any frequency less than or equal to half of the module clock.

When the master SPI starts a transmission, the data in the slave Shift register begins shifting out on the MISO pin. The slave can load its Shift register with new data for the next transmission by writing to its SPDTR. The slave must write to its SPDTR at least one bus cycle before the master starts the next transmission. Otherwise, the data already in the slave Shift register shifts out on the MISO pin. Data written to the Slave Shift register during a transmission remains in a buffer until the end of the transmission.

When the CPHA bit is set, the first edge of SCLK starts a transmission. When CPHA is cleared, the falling edge of  $\overline{SS}$  starts a transmission.

**Note:** SCLK must be in the proper idle state (depends on setting of CPOL bit) before the slave is enabled to prevent SCLK from appearing as a clock edge.

## 14.4.3 Wired OR Mode

Wired-OR functionality is provided to permit the connection of multiple SPIs. [Figure 14-3](#) illustrates a single master controlling multiple slave SPIs. When the WOM bit is set, the outputs switch from conventional complementary CMOS output to open drain outputs. This lets the



internal pull-up resistor bring the line high, and whichever SPI drives the line pulls it low as needed.

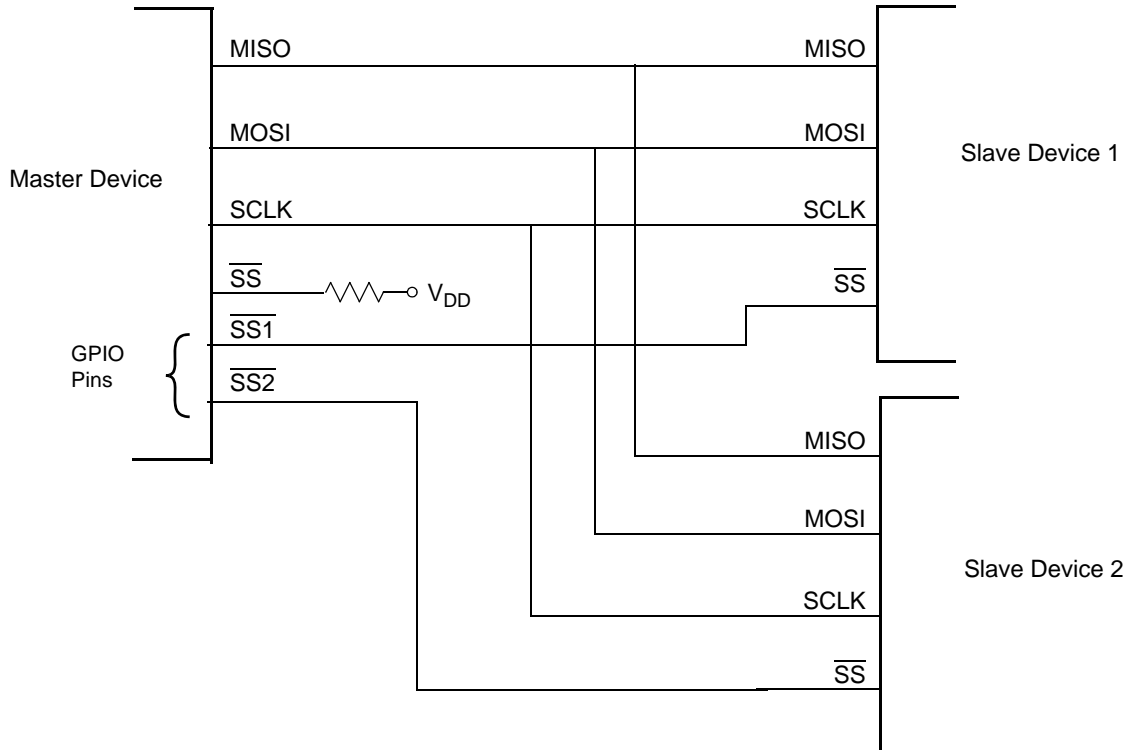


Figure 14-3. Master with Two Slaves

## 14.5 Pin Descriptions

There are four external SPI pins. Each is summarized in [Table 14-1](#).

Table 14-1. External I/O Signals

Signal Name	Description	Direction
MISO	Master-In Slave-Out Pad Pin	Bi-Directional
MOSI	Master-Out Slave-In Pad Pin	Bi-Directional
SCLK	Serial Clock Pad Pin	Bi-Directional
$\overline{SS}$	Slave Select Pad Pin (Active Low)	Input

### 14.5.1 Master In/Slave Out (MISO)

MISO is one of the two SPI module pins dedicated to transmit serial data. In full duplex operation, the MISO pin of the master SPI module is connected to the MISO pin of the slave SPI module. The master SPI simultaneously receives data on its MISO pin and transmits data from its

MOSI pin. The slave SPI simultaneously transmits data on its MISO pin and receives data from its MOSI pin.

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when the SPMSTR bit, discussed in [Section 14.9.1](#), is Logic 0 and its  $\overline{SS}$  pin is at Logic 0. To support a multiple slave system, a Logic 1 on the  $\overline{SS}$  pin puts the MISO pin in a High Impedance state.

### 14.5.2 Master Out/Slave In (MOSI)

MOSI is the other SPI module pin dedicated to transmit serial data. In full duplex operation, the MOSI pin of the master SPI module is connected to the MOSI pin of the slave SPI module. The master SPI simultaneously transmits data from its MOSI pin and receives data on its MISO pin. The slave SPI simultaneously receives data from its MOSI pin and transmits data on its MISO pin.

### 14.5.3 Serial Clock (SCLK)

The serial clock synchronizes data transmission between master and slave devices. In a master 16-bit controller, the SCLK pin is the clock output. In a slave 16-bit controller, the SCLK pin is the clock input. In full duplex operation, the master and slave 16-bit controller exchange data in the same number of clock cycles as the number of bits of transmitted data.

### 14.5.4 Slave Select ( $\overline{SS}$ )

The  $\overline{SS}$  pin has various functions depending on the current state of the SPI. For a SPI configured as a slave, the  $\overline{SS}$  is used to select a slave. When the Clock Phase (CPHA) bit in the SPSCR is cleared, the  $\overline{SS}$  is used to define the start of a transmission, so it must be toggled high and low between each full length data transmitted for the CPHA = 0 format. However, it can remain low between transmissions for the CPHA = 1 format as illustrated in [Figure 14-5](#).

When a SPI is configured as a slave, the  $\overline{SS}$  pin is always configured as an input. The MODFEN bit can prevent the state of the  $\overline{SS}$  from creating a MODF error.

**Note:** A Logic 1 voltage on the  $\overline{SS}$  pin of a slave SPI puts the MISO pin in a high impedance state. The slave SPI ignores all incoming SCLK clocks, even if it was already in the middle of a transmission. A Mode Fault occurs if the  $\overline{SS}$  pin changes state during a transmission.

When a SPI is configured as a master, the  $\overline{SS}$  input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SCLK. For the state of the  $\overline{SS}$  pin to set the MODF flag, the MODFEN bit in the SCLK register must be set.

**Table 14-2. SPI I/O Configuration**

SPE	SPMSTR	MODFEN	SPI Configuration	State of $\overline{SS}$ Logic
0	x	x	Not Enabled	$\overline{SS}$ ignored by SPI
1	0	x	Slave	Input-only to SPI
1	1	0	Master without MODF	$\overline{SS}$ ignored by SPI
1	1	1	Master with MODF	Input-only to SPI

x = Don't care

## 14.6 Transmission Formats

During a SPI transmission, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock synchronizes shifting and sampling on the two serial data lines. A slave select line allows selection of an individual slave SPI device; slave devices not selected do not interfere with SPI bus activities. On a master SPI device, the slave select line can optionally be used to indicate multiple-master bus contention.

### 14.6.1 Data Transmission Length

The SPI can support data lengths from two to 16 bits. This can be configured in the Data Size Register (SPDSR). When the data length is less than 16 bits, the Receive Data register will pad the upper bits with zeros.

**Note:** Data can be lost if the data length is not the same for both master and slave devices.

### 14.6.2 Data Shift Ordering

The SPI can be configured to transmit or receive the MSB of the desired data first or last. This is controlled by the Data Shift Order (DSO) bit in the SPSCR. Regardless which bit is transmitted or received first, the data shall always be written to the SPDTR and read from the Receive Data Register (SPDRR) with the LSB in bit zero and the MSB in the correct position, depending on the data transmission size.

### 14.6.3 Clock Phase and Polarity Controls

Software can select any of four combinations of Serial Clock (SCLK) phase and polarity using two bits in the SPSCR. The Clock Polarity is specified by the (CPOL) control bit. In turn, it selects an active high or low clock and has no significant effect on the transmission format.

The Clock Phase (CPHA) control bit selects one of two fundamentally different transmission formats. The clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between

transmissions to allow a master device to communicate with peripheral slaves having different requirements.

**Note:** Before writing to the CPOL bit or the CPHA bit, disable the SPI by clearing the SPI Enable (SPE) bit.

#### 14.6.4 Transmission Format When CPHA = 0

**Figure 14-4** exhibits a SPI transmission with CPHA as Logic 0.

**Note:** The figure should not be used as a replacement for data sheet parametric information.

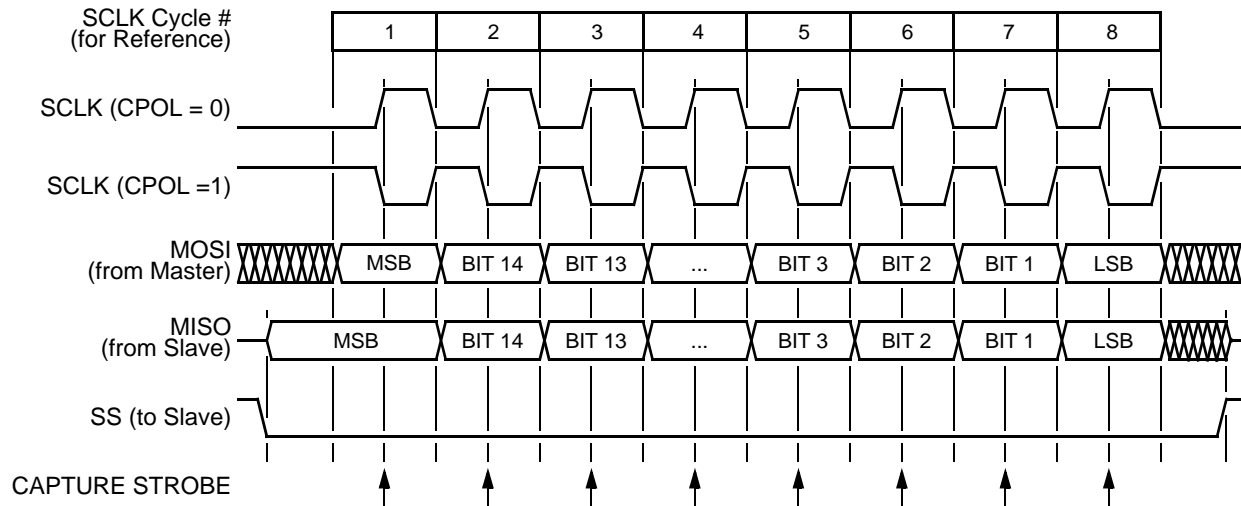
Two waveforms for the SCLK are shown:

1. CPOL = 0
2. CPOL = 1

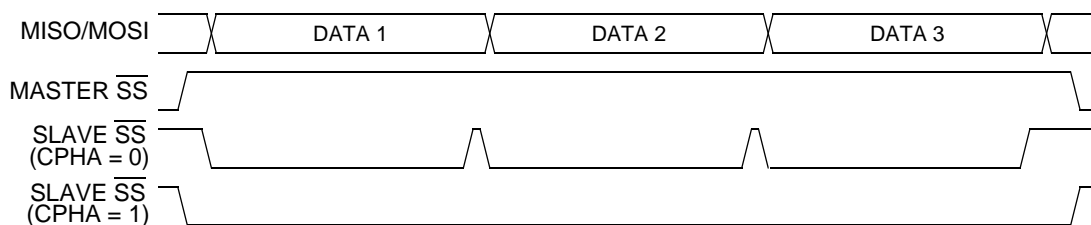
The diagram may be interpreted as a master or slave timing diagram since the Serial Clock (SCLK), Master In/Slave Out (MISO), and Master Out/Slave In (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master.

The  $\overline{SS}$  line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ( $\overline{SS}$ ) is at Logic 0, because only the selected slave drives to the master. The  $\overline{SS}$  pin of the master is not shown, but it is assumed to be inactive. The  $\overline{SS}$  pin of the master must be high or a Mode Fault Error will occur. When CPHA = 0, the first SCLK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SCLK edge and a falling edge on the  $\overline{SS}$  pin is used to start the slave data transmission. The slave's  $\overline{SS}$  pin must be toggled back to high and then low again between each full length data transmitted as depicted in **Figure 14-5**.

**Note:** **Figure 14-4** assumes 16-bit data lengths and the MSB shifted out first.



**Figure 14-4. Transmission Format (CPHA = 0)**



**Figure 14-5. CPHA/SS Timing**

When  $CPHA = 0$  for a slave, the falling edge of  $\overline{SS}$  indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the first bit of its data. Once the transmission begins, no new data is allowed into the Shift register from the SPDTR. Therefore, the SPI Data register of the slave must be loaded with transmit data before the falling edge of  $\overline{SS}$ . Any data written after the falling edge is stored in the SPDTR and transferred to the Shift register after the current transmission.

### 14.6.5 Transmission Format When CPHA = 1

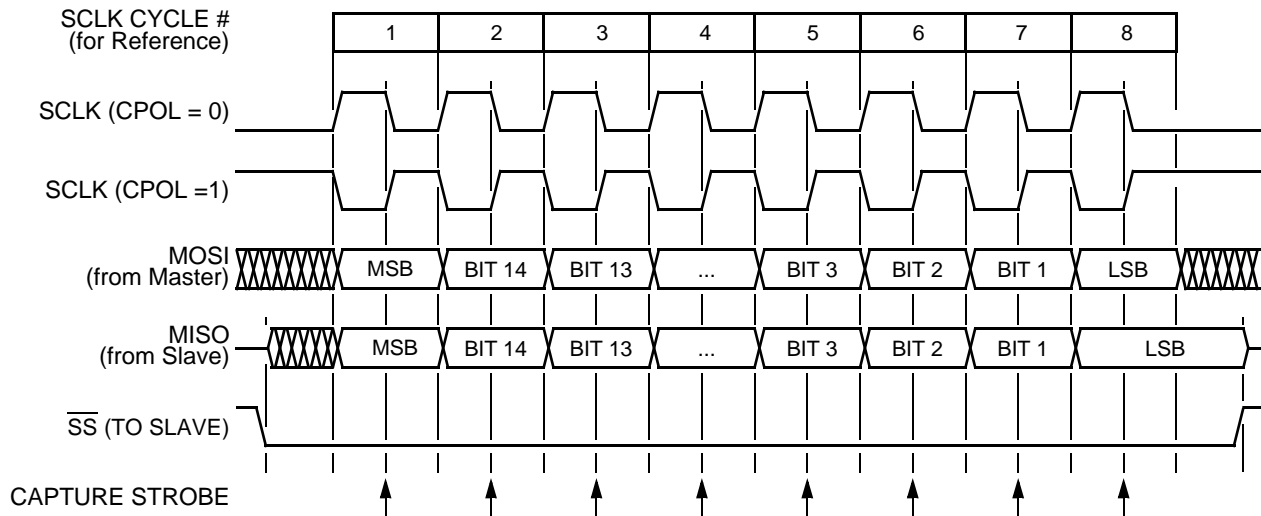
A SPI transmission is shown in [Figure 14-6](#) where CPHA is Logic 1.

**Note:** The figure should not be used as a replacement for data sheet parametric information.

Two waveforms are shown for SCLK: 1 for  $CPOL = 0$  and another for  $CPOL = 1$ . The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCLK), Master In/Slave Out (MISO), and Master Out/Slave In (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The  $\overline{SS}$  line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ( $\overline{SS}$ ) is at Logic 0, so only the selected slave drives

to the master. The  $\overline{SS}$  pin of the master is not shown but is assumed to be inactive. The  $\overline{SS}$  pin of the master must be high or a Mode Fault Error occurs. When  $CPHA = 1$ , the master begins driving its MOSI pin on the first SCLK edge. Therefore, the slave uses the first SCLK edge as a start transmission signal. The  $\overline{SS}$  pin can remain low between transmissions. This format may be preferable in systems having only one master and slave driving the MISO data line.

**Note:** [Figure 14-6](#) assumes 16-bit data lengths and the MSB shifted out first.



**Figure 14-6. Transmission Format (CPHA = 1)**

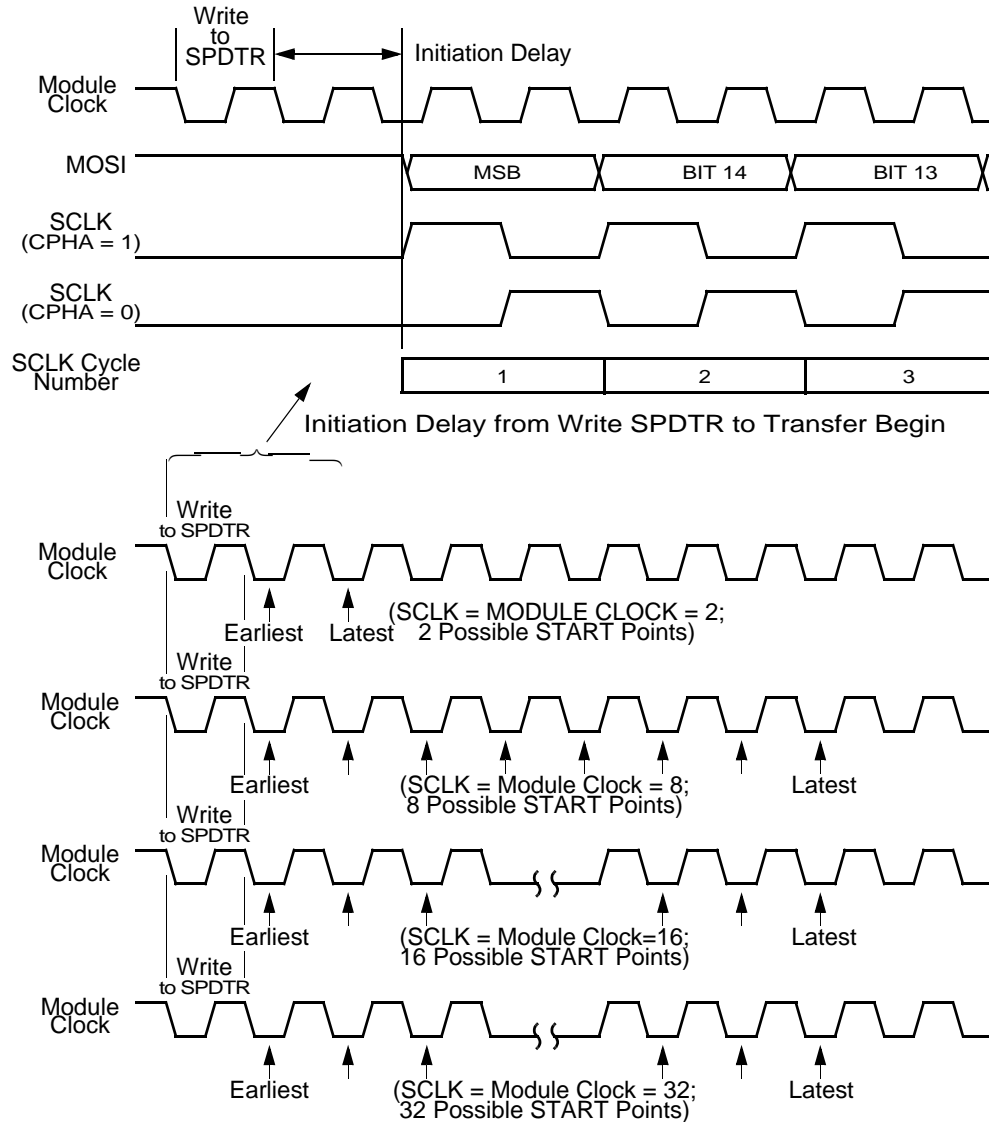
When  $CPHA = 1$  for a slave, the first edge of the SCLK indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the first bit of its data. Once the transmission begins, no new data is allowed into the Shift register from the SPDTR. Therefore, the SPI Data register of the slave must be loaded with transmit data before the first edge of SCLK. Any data written after the first edge is stored in the SPDTR and transferred to the Shift register after the current transmission.

### 14.6.6 Transmission Initiation Latency

When the SPI is configured as a master ( $SPMSTR = 1$ ), writing to the SPDTR starts a transmission.  $CPHA$  has no effect on the delay to the start of the transmission, but it does affect the initial state of the SCLK signal. When  $CPHA = 0$ , the SCLK signal remains inactive for the first half of the first SCLK cycle. When  $CPHA = 1$ , the first SCLK cycle begins with an edge on the SCLK line from its inactive to its active level. The SPI clock rate, selected by  $SPR[2:0]$ , affects the delay from the write to SPDTR and the start of the SPI transmission. The internal SPI clock in the master is a free-running derivative of the internal clock. To conserve power, it is enabled only when both the SPE and SPMSTR bits are set. Since the SPI clock is free-running, it is uncertain where the write to the SPDTR occurs relative to the slower SCLK. This uncertainty causes the variation in the initiation delay, demonstrated in [Figure 14-7](#). This delay is no longer

than a single SPI bit time. That is, the maximum delay is two bus cycles for DIV2, four bus cycles for DIV4, eight bus cycles for DIV8, and so on up to a maximum of 256 cycles for DIV256.

**Note:** **Figure 14-7** assumes 16-bit data lengths and the MSB shifted out first.



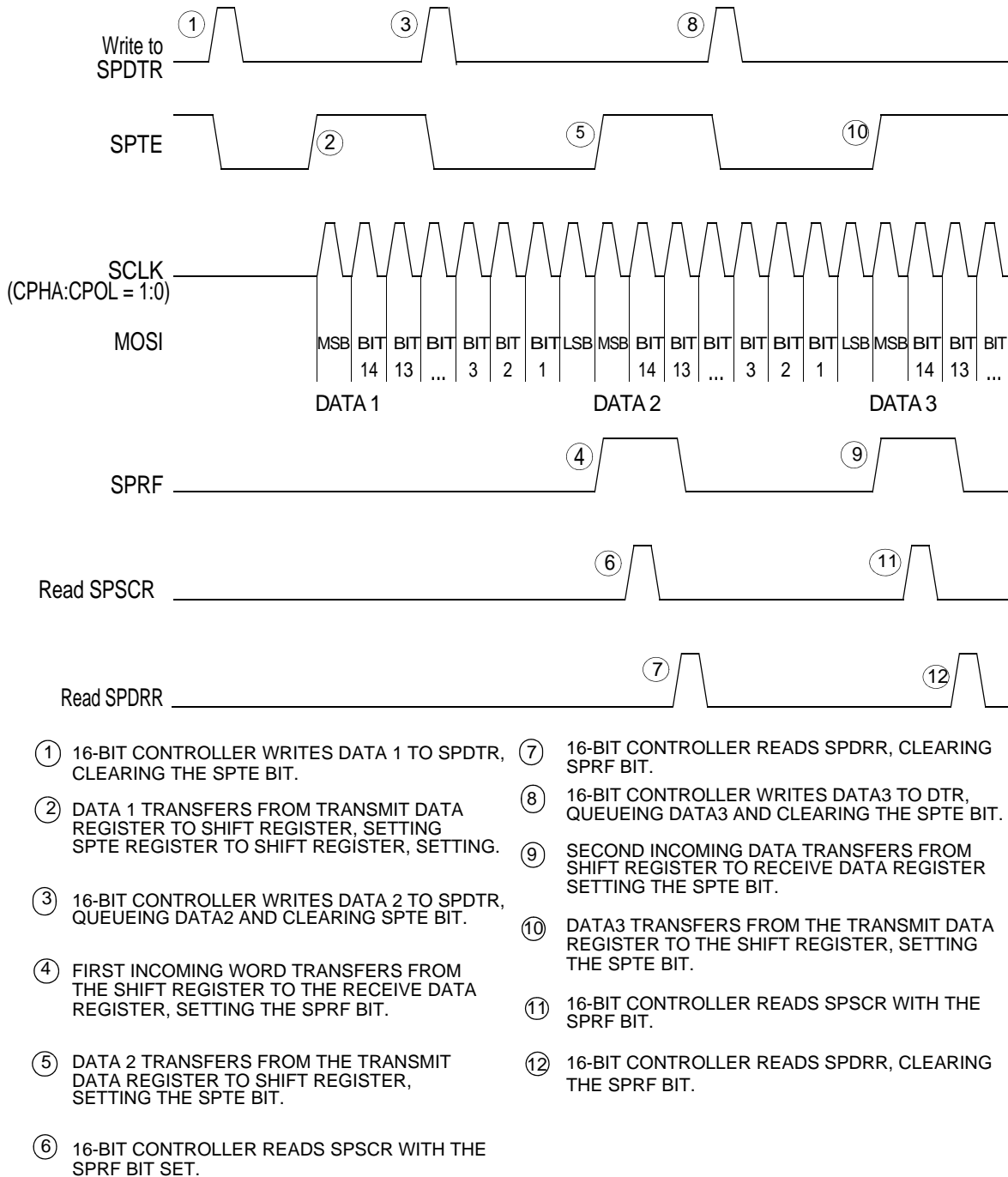
**Figure 14-7. Transmission Start Delay (Master)**

## 14.7 Transmission Data

The double-buffered SPDTR allows data to be queued and transmitted. For a SPI configured as a master, the queued data is transmitted immediately after the previous transmission has completed. The SPTE flag indicates when the transmit data buffer is ready to accept new data.

Write to the SPDTR only when the SPTE bit is high. **Figure 14-8** illustrates the timing associated with doing back-to-back transmissions with the SPI (SCLK has CPHA: CPOL = 1:0).

**Note:** **Figure 14-8** assumes 16-bit data lengths and the MSB shifted out first.



**Figure 14-8. SPRF/SPTE Interrupt Timing**



The transmit data buffer permits back-to-back transmissions without the slave precisely timing its writes between transmissions as is necessary in a system with a single data buffer. Also, if no new data is written to the data buffer, the last value contained in the Shift register is the next data to be transmitted.

An idle master or idle slave without loaded data in its transmit buffer, sets the SPTE again no more than two bus cycles after the transmit buffer empties into the Shift register. This allows a queue to send up to a 32-bit value. For an already active slave, the load of the Shift register cannot occur until the transmission is completed. This implies a back-to-back write to the SPDTR is not possible. The SPTE indicates when the next write can occur.

## 14.8 Error Conditions

The following flags signal SPI error conditions:

- **Overflow (OVRF)** — Failing to read the SPI Data register before the next full length data enters the Shift register sets the OVRF bit. The new data will not transfer to the Receive Data register, and the unread data can still be read. OVRF is in the SPSCR.
- **Mode Fault Error (MODF)** — The MODF bit indicates the voltage on the Slave Select pin ( $\overline{SS}$ ) is inconsistent with SPI mode. MODF is in the SPSCR.

### 14.8.1 Overflow Error

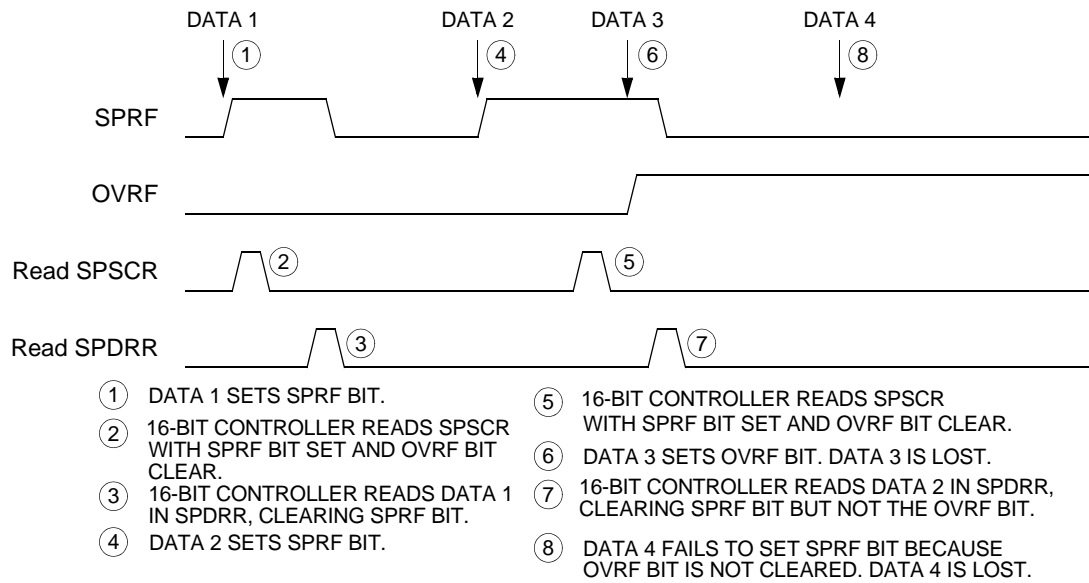
The Overflow Flag (OVRF) becomes set if the last bit of a current transmission is received and the Receive Data register still has unread data from a previous transmission. If an overflow occurs, all data received after the overflow, and before the OVRF bit is cleared, does not transfer to the Receive Data Register (SPRDR). It does not set the SPI Receiver Full (SPRF) bit. The unread data transferred to the Receive Data register before the overflow occurred can still be read. Therefore, an overflow error always indicates the loss of data. Clear the overflow flag by reading the SPSCR, then read the SPRDR.

OVRF generates a receiver/error interrupt request if the error interrupt enable bit (ERRIE) is also set. It is not possible to enable MODF or OVRF individually to generate a receiver/error interrupt request. However, leaving MODFEN low prevents MODF from being set.

If the SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. **Figure 14-9** explains how it is possible to miss an overflow. The first element of the same figure illustrates how it is possible to read the SPSCR and SPDRR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF bit can be set between the time SPSCR and SPDRR are read.

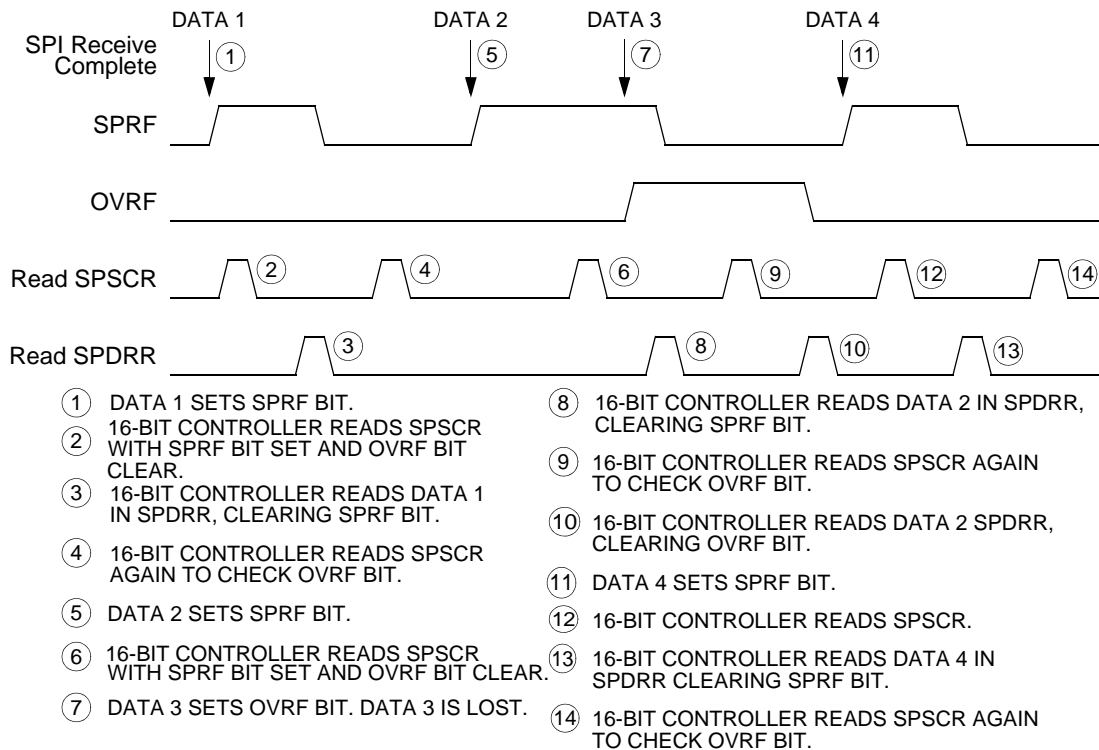
In this case, an overflow can easily be missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it is not obvious data is being lost as more transmissions are

completed. To prevent this loss, either enable the OVRF interrupt or take another read of the SPSCR following the read of the SPDRR. This ensures the OVRF was not set before the SPRF was cleared and future transmissions can set the SPRF bit.



**Figure 14-9. Missed Read of Overflow Condition**

**Figure 14-10** illustrates the described process. Generally, to avoid a second SPSCR read, enable the OVRF by setting the ERRIE bit.



**Figure 14-10. Clearing SPRF When OVRF Interrupt Is Not Enabled**

## 14.8.2 Mode Fault Error

Setting the SPMSTR bit selects Master mode, configuring the SCLK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects Slave mode, configuring the SCLK and MOSI pins as inputs and the MISO pin as an output. The Mode Fault (MODF) bit becomes set any time the state of the  $\overline{SS}$  pin is inconsistent with the mode selected by SPMSTR. To prevent SPI pin contention and damage to the 16-bit controller, a Mode Fault Error occurs if:

- The  $\overline{SS}$  pin of a slave SPI goes high during a transmission.
- The  $\overline{SS}$  pin of a master SPI goes low at any time.

To set the MODF flag, Mode Fault Error Enable (MODFEN) bit must be set. Clearing the MODFEN bit does not clear the MODF flag but it does prevent the MODF from being set again after the MODF is cleared.

MODF generates a receiver/error interrupt request if the Error Interrupt Enable (ERRIE) bit is also set. It is not possible to enable MODF or OVRF individually to generate a receiver/error interrupt request. However, leaving MODFEN low prevents MODF from being set.

### 14.8.2.1 Master SPI Mode Fault

In a master SPI with Mode Fault Enable (MODFEN) bit set, Mode Fault (MODF) flag is set if  $\overline{SS}$  goes to Logic 0. A Mode Fault in a Master SPI causes the following events to occur:

- If  $ERRIE = 1$ , the SPI generates a SPI receiver/error interrupt request.
- The SPE bit is cleared
- The SPTE bit is set
- The SPI state counter is cleared

In a master SPI, the MODF flag will not be cleared until the  $\overline{SS}$  pin is at a Logic 1 or the SPI is configured as a slave.

**Note:** When  $CPHA = 0$ , a MODF occurs if a slave is selected ( $\overline{SS}$  is at Logic 0) and later unselected ( $\overline{SS}$  is at Logic 1) even if no SCLK is sent to that slave. This happens because  $\overline{SS}$  at Logic 0 indicates the start of the transmission (MISO driven out with the value of MSB) for  $CPHA = 0$ . When  $CPHA = 1$ , a slave can be selected and then later unselected with no transmission occurring. Therefore, MODF does not occur since a transmission was never begun.

### 14.8.2.2 Slave SPI Mode Fault

In a slave SPI ( $SPMSTR = 0$ ), the  $MODF$  bit generates a SPI Receiver/Error Interrupt request if the  $ERRIE$  bit is set. The  $MODF$  bit does not clear the  $SPE$  bit or reset the SPI in any way. Software can abort the SPI transmission by clearing the  $SPE$  bit of the slave.

**Note:** A Logic 1 voltage on the  $\overline{SS}$  pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SCLK clocks, even if it was already in the middle of a transmission.

When configured as a slave ( $SPMSTR = 0$ ), the  $MODF$  flag is set if the  $\overline{SS}$  goes high during a transmission. When  $CPHA = 0$ , a transmission begins when  $\overline{SS}$  goes low and ends once the incoming SCLK goes back to its idle level, following the shift of the last data bit. When  $CPHA = 1$ , the transmission begins when the SCLK leaves its idle level and  $\overline{SS}$  is already low. The transmission continues until the SCLK returns to its idle level following the shift of the last data bit.

To clear the  $MODF$  flag, read the  $SPSCR$  with the  $MODF$  bit set and then write to the  $SPSCR$ . This entire clearing mechanism must occur with no  $MODF$  condition existing or else the flag is not cleared.

In a slave SPI, if the  $MODF$  flag is not cleared by writing a one to the  $MODF$  bit, the condition causing the Mode Fault still exists. In this case, the interrupt caused by the  $MODF$  flag can be cleared by disabling the  $EERIE$  or  $MODFEN$  bits (if set) or by disabling the SPI. Disabling the SPI using the  $SPE$  bit will cause a partial reset of the SPI and may cause the loss of a message currently being received or transmitted.

## 14.9 Register Definitions

**Table 14-3. SPI Memory Map**

Device	Peripheral	Address
8100/8300	SPI0_BASE	\$00F2A0
	SPI1_BASE	\$00F2B0

**Table 14-4** lists the SPI registers in ascending address, including their acronyms and address of each register. The read/write registers should be accessed only with word accesses. Accesses other than word lengths result in undefined results. **Table 14-6** portrays a map summary of the registers.

**Table 14-4. SPI Register Summary**

Address Offset	Register Acronym	Register Name	Access Type	Chapter Location
Base + \$0	SPSCR	Status and Control Register	Read/Write	<a href="#">Section 14.9.1</a>
Base + \$1	SPDSR	Data Size and Control Register	Read/Write	<a href="#">Section 14.9.2</a>
Base + \$2	SPDRR	Data Receive Register	Read-Only	<a href="#">Section 14.9.3</a>
Base + \$3	SPDTR	Data Transmit Register	Write-Only	<a href="#">Section 14.9.4</a>

Bit fields of each of the four registers are illustrated in [Table 14-11](#). Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
\$0	SPSCR	R	SPR			DSO	ERRIE	MOD FEN	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE	SPRF	OVRF	MODF	SPTE
		W																
\$1	SPDSR	R	WOM	0	0	0	0	0	0	0	0	0	0	0	DS3	DS2	DS1	DS0
		W																
\$2	SPDRR	R	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
		W																
\$3	SPDTR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0

R	0	Read as 0
W		Reserved

**Figure 14-11. SPI Register Map Summary**

### 14.9.1 SPI Status and Control Register (SPSCR)

The SPSCR:

- Selects master SPI baud rate
- Determines data shift order
- Enables SPI module interrupt requests
- Configures SPI module as master or slave
- Selects serial clock polarity and phase
- Indicates the SPI status

Base + \$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	SPR			DSO	ERRIE	MODFEN	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE	SPRF	OVRF	MODF	SPTIE
Write	SPR			DSO	ERRIE	MODFEN	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE				
Reset	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0	1

**Figure 14-12. SPI Status and Control Register (SPSCR)**

**Note:** Using BFCLR or BFSET instructions on the SPSCR can cause unintended side effects on the status bits. This is due to these instructions being a read/write modify instruction.

#### 14.9.1.1 SPI Baud Rate Select (SPR)—Bits 15–13

While in the Master mode, these read/write bits select one of eight baud rates depicted in [Table 14-5](#). SPR[2:0] have no effect in Slave mode. Use the formula below to calculate the SPI baud rate.

$$\text{Baud Rate} = \frac{\text{Module Clock}}{\text{Baud Rate Divisor}}$$

**Table 14-5. SPI Master Baud Rate Selection**

SPR[2:0]	Baud Rate Divisor (BD)
000	2
001	4
010	8
011	16
100	32
101	64
110	128
111	256

**Note:** The maximum data transmission rate for the SPI is typically limited by the bandwidth of the I/O drivers on the chip. Limits for the 5683x are normal at 20MHz and 650kHz for Wired OR. These apply to both Master and Slave modes. The BD field needs to be set to keep the module within these ranges.

### 14.9.1.1.1 SPR Example

If you have a SPI device with a maximum baud rate of 1MHz and the SPI module clock is 60MHz, then the baud rate divisor must be greater than or equal to 60 (60MHz/1MHz). According to [Table 14-5](#), the smallest baud rate divisor valid for this example would be 64. Thus, SPR[2:0] would be 5. The actual baud rate for this example would be 937.5kHz (60MHz/64).

**Note:** For 8100 devices, with 40MHz clock, the baud rate divisor must be greater than, or equal to, 40 (40MHz/1MHz). A divisor of 64 from Table 14-5 is still used, yielding an actual baud rate of 625kHz.

### 14.9.1.2 Data Shift Order (DSO)—Bit 12

This read/write bit determines whether the MSB or LSB bit is transmitted or received first. Both Master and Slave SPI modules must transmit and receive the same length packets. Regardless how this bit is set, when reading from the SPDRR or writing to the SPDTR, the LSB will always be at bit location zero. If the data length is less than 16 bits, the data will be zero padded on the upper bits.

- 0 = MSB transmitted first
- 1 = LSB transmitted first

### 14.9.1.3 Error Interrupt Enable (ERRIE)—Bit 11

This read/write bit enables the MODF, if MODFEN is also set, and OVRF bits to generate interrupt requests.

- 0 = MODF and OVRF cannot generate interrupt requests
- 1 = MODF and OVRF can generate interrupt requests

### 14.9.1.4 Mode Fault Enable (MODFEN)—Bit 10

This read/write bit, when set to one, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN does not clear the MODF flag.

If the MODFEN bit is low, the level of the  $\overline{SS}$  pin does not affect the operation of an enabled SPI configured as a master. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation.

### 14.9.1.5 SPI Receiver Interrupt Enable (SPRIE)—Bit 9

This read/write bit enables interrupt requests generated by the SPRF bit. The SPRF bit is set when data transfers from the Shift register to the Receive Data register.

- 0 = SPRF interrupt requests disabled
- 1 = SPRF interrupt requests enabled

#### 14.9.1.6 SPI Master (SPMSTR)—Bit 8

This read/write bit selects Master or Slave modes operation.

- 0 = Slave mode
- 1 = Master mode

#### 14.9.1.7 Clock Polarity (CPOL)—Bit 7

This read/write bit determines the logic state of the SCLK pin between transmissions. To transmit data between SPI modules, the SPI modules must have identical CPOL values. Please see [Figure 14-4](#) and [Figure 14-6](#).

#### 14.9.1.8 Clock Phase (CPHA)—Bit 6

This read/write bit controls the timing relationship between the serial clock and SPI data. Please see [Figure 14-4](#) and [Figure 14-6](#). To transmit data between SPI modules, the SPI modules must have identical CPHA values. When CPHA = 0, the  $\overline{SS}$  pin of the Slave SPI module must be set to Logic 1 between data transmissions, illustrated in [Figure 14-5](#).

#### 14.9.1.9 SPI Enable (SPE)—Bit 5

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. When setting/clearing this bit, *no* other bits in the SPSCR should be changed. Failure to following this statement may result in spurious clocks.

- 0 = SPI module disabled
- 1 = SPI module enabled

#### 14.9.1.10 SPI Transmit Interrupt Enable (SPTIE)—Bit 4

This read/write bit enables interrupt requests generated by the SPTE bit. SPTE is set when data transfers from the SPDTR to the Shift register.

- 0 = SPTE interrupt requests disabled
- 1 = SPTE interrupt requests enabled

#### 14.9.1.11 SPI Receiver Full (SPRF)—Bit 3

This *read-only* flag is set each time data transfers from the Shift register to the SPDRR. SPRF generates an interrupt request if the SPRIE bit in the SPI Control register is set. This bit is cleared by reading the SPDRR.

- 0 = Data Receive Register (SPDRR) not full
- 1 = Data Receive Register (SPDRR) full



#### 14.9.1.12 Overflow (OVRF)—Bit 2

This *read-only* flag is set if software does not read the data in the SPDRR before the next full data enters the Shift register. In an overflow condition, the data already in the SPDRR is unaffected, and the data shifted in last is lost. Clear the OVRF bit by reading the SPSCR and then reading the SPDRR. OVRF generates a Receiver/Error interrupt if the EERIE bit is set. See [Section 14.8.1](#) for more details.

- 0 = No overflow
- 1 = Overflow

#### 14.9.1.13 Mode Fault (MODF)—Bit 1

This *read-only* flag is set in a slave SPI if the  $\overline{SS}$  pin goes high during a transmission with the MODFEN bit set. In a master SPI, the MODF flag is set if the  $\overline{SS}$  pin goes low at any time with the MODFEN bit set. Clear the MODF bit by writing a one to the MODF bit when it is set. MODF generates a Receive/Error interrupt if the EERIE bit is set. See [Section 14.8.2](#) for more details

- 0 =  $\overline{SS}$  pin at appropriate Logic level
- 1 =  $\overline{SS}$  pin at inappropriate Logic level

#### 14.9.1.14 SPI Transmitter Empty (SPTE)—Bit 0

This *read-only* flag is set each time the SPDTR transfers data into the Shift register. SPTE generates an interrupt request if the SPTIE bit in the SPI Control register is set. This bit is cleared by writing to the SPDTR.

- 0 = Data Transmit Register (SPDTR) not empty
- 1 = Data Transmit Register (SPDTR) empty

**Note:** Do not write to the SPI Data register unless the SPTE bit is high.

### 14.9.2 SPI Data Size and Control Register (SPDSR)

This read/write register determines the data length for each transmission. The master and slave must transfer the same size data on each transmission. A new value will only take effect at the time the SPI is enabled (SPE bit in SPSCR set from zero to one). In order to have a new value take effect, disable then re-enable the SPI with the new value in the register. The DSR:

- Enables Wired OR mode on MISO and MOSI
- Configures the size of the transmission

Base + \$1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	WOM	0	0	0	0	0	0	0	0	0	0	0	DS3	DS2	DS1	DS0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

**Figure 14-13. SPI Data Size and Control Register (SPDSR)**

#### 14.9.2.1 Wired OR Mode (WOM)—Bit 15

This control bit is used to select the nature of the SPI pins. When the WOM bit is set, the SPI pins are configured as open-drain drivers with the pull-ups disabled. When the WOM bit is cleared, the SPI pins are configured as push-pull drivers.

- 0 = Wired OR mode disabled
- 1 = Wired OR mode enabled

#### 14.9.2.2 Reserved—Bits 14-4

These bits are reserved or not implemented. They are read as 0 and cannot be modified by writing.

#### 14.9.2.3 Data Size (DS)—Bits 3-0

Please see [Table 14-6](#) for detailed transmission data.

**Table 14-6. Data Size**

DS[3:0]	Size of Transmission
\$0	Not Allowed
\$1	2 Bits
\$2	3 Bits
\$3	4 Bits
\$4	5 Bits
\$5	6 Bits
\$6	7 Bits
\$7	8 Bits
\$8	9 Bits
\$9	10 Bits
\$A	11 Bits
\$B	12 Bits
\$C	13 Bits
\$D	14 Bits
\$E	15 Bits
\$F	16 Bits

### 14.9.3 SPI Data Receive Register (SPDRR)

This *read-only* register will show the last data word received after a complete transmission. The SPRF bit is set when new data is transferred to this register.

Base + \$2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-14. SPI Data Receive Register (SPDRR)

### 14.9.4 SPI Data Transmit Register (SPDTR)

This *write-only* register holds data to be transmitted. When the SPTE bit is set, new data should be written to this register. If new data is not written while in the Master mode, a new transaction will not be initiated until this register is written. When in Slave mode, the old data will be re-transmitted. All data should be written with the LSB at bit zero.

Base + \$3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Write	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-15. SPI Data Transmit Register (SPDTR)

## 14.10 Resets

Any system reset completely resets the SPI. Partial resets occur whenever the SPI enable bit (SPE) is low. Whenever SPE is low, the following will occur:

- SPTE flag is set
- Any Slave mode transmission currently in progress is aborted
- Any Master mode transmission currently in progress is continued to completion
- SPI state counter is cleared, making it ready for a new complete transmission
- All the SPI port Logic is disabled

The following items are reset only by a system reset:

- The SPDTR and SPDRR registers

- All control bits in the SPSCR (MODFEN, ERRIE, SPR[2:0])
- The status flags SPRF, OVRF, and MODF

By not resetting the control bits when SPE is low, it is possible to clear SPE between transmissions without having to set all control bits again when SPE is set back high for the next transmission.

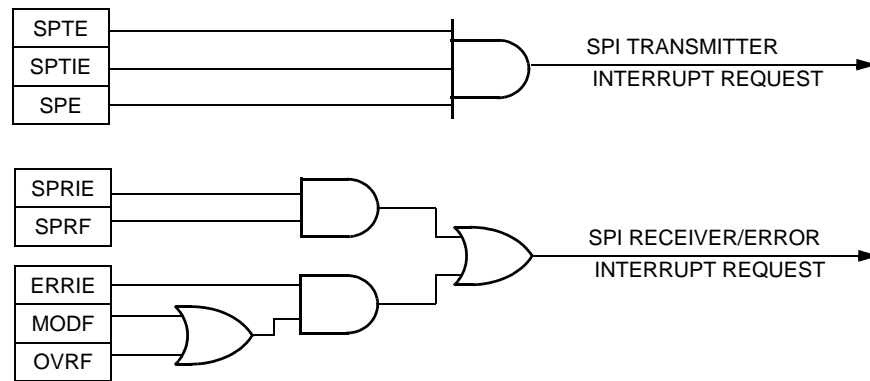
By not resetting the SPRF, OVRF, and MODF flags, it is possible to service the interrupts after the SPI has been disabled. Disable SPI by writing zero to the SPE bit. SPI can also be disabled by a Mode Fault occurring in a SPI configured as a master.

## 14.11 Interrupts

Four SPI status flags can be enabled to generate interrupt requests.

**Table 14-7. SPI Interrupts**

Flag	Interrupt Enabled By	Description
SPTIE (Transmitter Empty)	SPI Transmitter Interrupt Request (SPTIE = 1, SPE = 1)	The SPI transmitter interrupt enable bit (SPTIE) enables the SPTIE flag to generate transmitter interrupt requests provided that the SPI is enabled (SPE = 1). The SPTIE bit becomes set every time data transfers from the SPDTR to the Shift register. The clearing mechanism for the SPTIE flag is a write to the SPDTR.
SPRF (Receiver Full)	SPI Receiver Interrupt Request (SPRIE = 1)	The SPI Receiver Interrupt Enable bit (SPRIE) enables the SPRF bit to generate receiver interrupt requests regardless of the state of the SPE bit. The SPRF is set every time data transfers from the Shift register to the SPDRR. The clearing mechanism for the SPRF flag is to read the SPDRR.
OVRF (Overflow)	SPI Receiver/Error Interrupt Request (ERRIE = 1)	The error interrupt enable bit (ERRIE) enables both the MODF and OVRF bits to generate a receiver/error interrupt request.
MODF (Mode Fault)	SPI Receiver/Error Interrupt Request (ERRIE = 1, MODFEN = 1)	The Mode Fault enable bit (MODFEN) can prevent the MODF flag from being set so that only the OVRF bit is enabled by the ERRIE bit to generate receiver/error 16-bit controller interrupt requests.

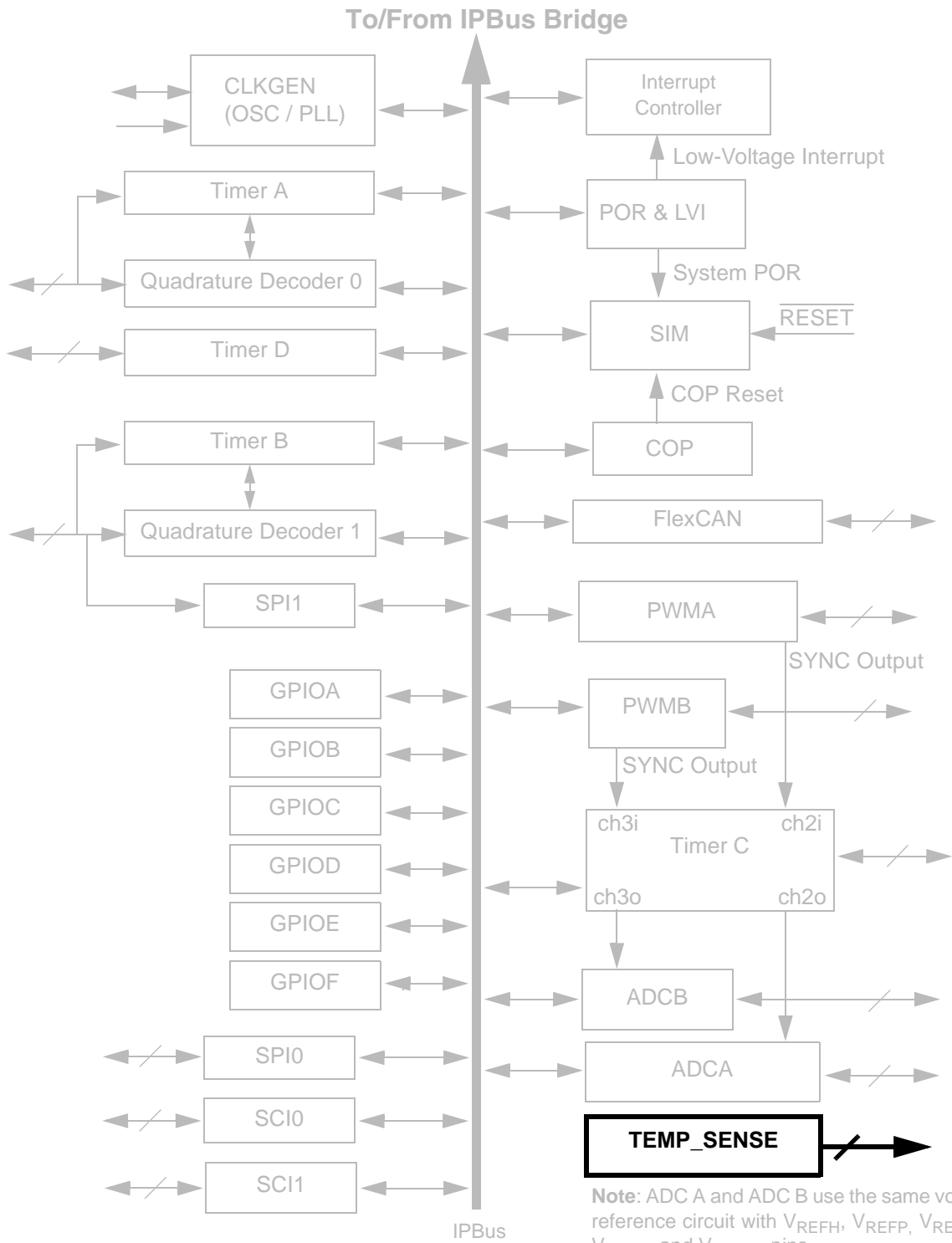


**Figure 14-16. SPI Interrupt Request Generation**



# Chapter 15

## Temperature Sensor System (TSENSOR)



Temperature Sensor System (TSENSOR), Rev. 10

## Document Revision History for **Chapter 15, Temperature Sensor System (TSENSOR)**

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 5.0	<p>Deleted the first sentence of the fourth bullet, page 3, leaving the remainder of the bullet in tact</p> <p>Section 15.4, page 4 Added "Conceptually" to the beginning of the first sentence</p> <p>Deleted "our", third line of Section 15.4, replaced with "the"</p> <p>Deleted "blocks" in third line of Section 15.4, replacing with "module"</p> <p>Deleted "Automotive" replacing with "all" in first line under Figure 15-2</p> <p>Deleted last two sentences in text under Figure 15-2 "Commercial temp...."</p> <p>Added sentence at the end of Section 15.6, page 6 to read: "Configurations are part and package dependent."</p> <p>Corrected reference to registers FMOPT1 to FMOPT2 on page 6.</p> <p>Deleted Clocks Section 15.8, page 7</p> <p>Converted chapter to Freescale design standards</p>
Rev 8.0	Minor formatting edits.



## 15.1 Introduction

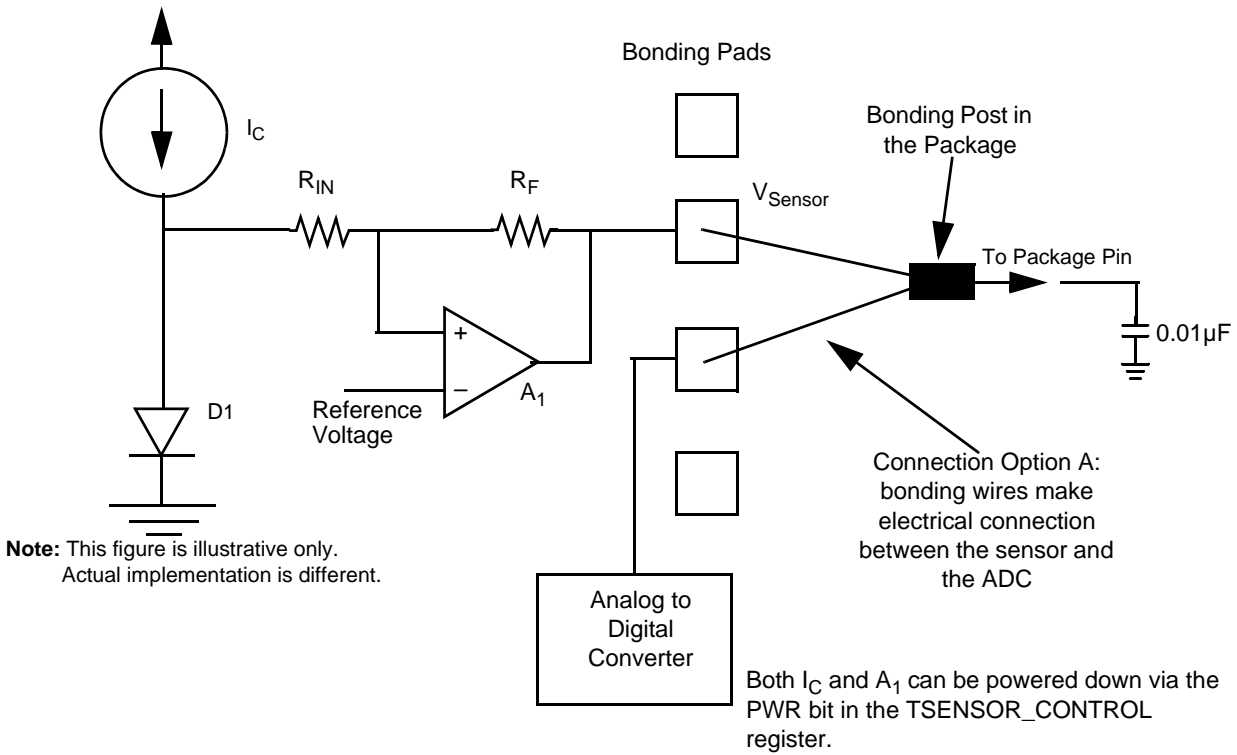
This chapter describes the Temperature Sensor module intended to operate in conjunction with an on-chip Analog-to-Digital Converter (ADC). The module is composed of a custom analog block and a simple IPBus interface. Temperature readings may be taken through the ADC. The ADC includes the ability to continuously monitor an input and issue an interrupt when a limit is exceeded. This provides the ability to easily implement an over temperature alert.

**Note:** Application Note AN1980/D *Using the 56F83xx Temperature Sensor* illustrates how to significantly increase accuracy based on identifying limitations and proper system design.

## 15.2 Features

- Operating range: -40 to 150°C Junction Temperature ( $T_J$ )(inclusive, 150°C is a legal value)
- Operation will continue in a linear fashion above 150°C, although the device is only guaranteed to this point
- Supply Voltage: 3.3V  $\pm$  .3V
- Connection to ADC input is achieved in several ways and is part and package dependent. The connection methods are:
  - Internal connection of the sensor output to an ADC input in the package
  - Bringing the sensor voltage out to an output pin permitting optional external connection to an ADC input
- Monotonic with temperature
- Resolution is better than 1°C per bit over a 10-bit range from 0 to 3.6V

## 15.3 Block Diagram



**Figure 15-1. Temperature Sensor Block Diagram**

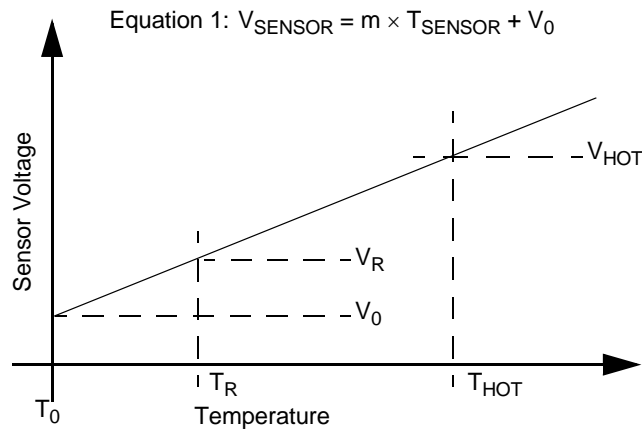
## 15.4 Functional Description

Conceptually, this module runs a constant current source through one or more PN junctions to generate a temperature dependent voltage. This voltage is then routed via an inverting amplifier to the input of one of the standard on-chip analog-to-digital module for conversion to a digital value which can be used to calculate temperature.

Assuming:

$T_{\text{SENSOR}}$	= Sensor Temperature
$T_{\text{R}}$	= Room temperature
$T_{\text{HOT}}$	= Ambient temperature used for hot temperature testing
$V_0$	= Y-intercept of the sensor voltage versus temperature curve
$V_{\text{SENSOR}}$	= Sensor voltage
$V_{\text{R}}$	= Sensor voltage at room temperature
$V_{\text{HOT}}$	= Sensor voltage at $T_{\text{Hot}}$

**Figure 15-2** (not to scale) illustrates the classic voltage versus temperature characteristics for a PN junction buffered by an inverting amplifier illustrated in **Figure 15-1**. The parameter  $m$  is the slope of the line,  $V_0$  represents the Y-intercept of the  $V_{\text{SENSOR}}$  vs. temperature curve.



**Figure 15-2. Temperature Sensor Output Characteristics**

Both  $V_0$  and  $m$  may vary from device to device. All devices will have the value of  $V_R$  (as converted by the ADC) and  $V_{HOT}$  stored in Non-Volatile Memory (NVM). This allows system software to compensate for variances in those two parameters.

Solving for  $T_{\text{SENSOR}}$  from Equation 1:

$$\text{Equation 2} \quad T_{\text{SENSOR}} = (V_{\text{SENSOR}} - V_0)/m$$

and substituting:

$$\text{Equation 3} \quad V_0 = V_R - m \times T_R$$

$$\text{Equation 4} \quad T_{\text{SENSOR}} = (V_{\text{SENSOR}} - V_R)/m + T_R$$

and finally, by inspection, the result is:

$$\text{Equation 5} \quad m = (V_{HOT} - V_R) / (T_{HOT} - T_R)$$

Therefore, equations 4 and 5 can be used to convert  $V_{\text{SENSOR}}$  as measured by the on-chip ADC into temperature.

## 15.5 Operating Modes

The sensor can be powered down when it is not in use. This is its default state. Thus, if it is not internally connected to the ADC, it can easily be ignored. The block diagram in [Figure 15-1](#) illustrates an internal ADC connection option. Other packaging options have the V sensor pad bonded directly out to a Tsensor pin and not shared with an ADC input.

## 15.6 Pin Descriptions

The Temperature Sensor module can optionally have a dedicated pin enabling its connection externally to an available ADC input or not at all. Other configurations have the temperature sensor output physically connected to a specific ADC input. Configurations are part and package dependent.

## 15.7 Register Definition

**Table 15-1. TSENSOR Memory Map**

Device	Peripheral	Address
8100/8300	TSENSOR_BASE	\$00F270

This module has only one peripheral register controlling the decision to switch the current source  $I_C$  into the circuit. Additionally, ADC measurements for room and hot temperatures are stored in internal NVM. These *read-only* values are available in the FMOPT2 and FMOPT0 registers, respectively. They can be accessed by application software to adjust real-time temperature sensor ADC readings to account for individual device variances. The bit of the TS register is illustrated in [Table 15-2](#). Details follow.

**Table 15-2. TSENSOR Register Summary**

Address Offset	Acronym	Name	Access Type	Location
Base + \$0	TSENSOR_CTRL	Control Register	Read/Write	<a href="#">Section 15.7.1</a>

### 15.7.1 Temperature Sensor Control Register (TSENSOR\_CTRL)

Base + \$0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PWR
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 15-3. Temperature Sensor Control Register (TSENSOR\_CTRL)**

### 15.7.1.1 Reserved—Bits 15–1

This bit field is reserved or not implemented. Bits are read as 0 and cannot be modified by writing.

### 15.7.1.2 Power-On (PWR)—Bit 0

This bit is used to power current source  $I_C$  ON or OFF.

- 0 = Off (default)
- 1 = On

## 15.8 Interrupts

The temperature sensor module relies on the associated Analog-to-Digital Converter for all reading and interrupt support. There are no interrupts directly from this module.

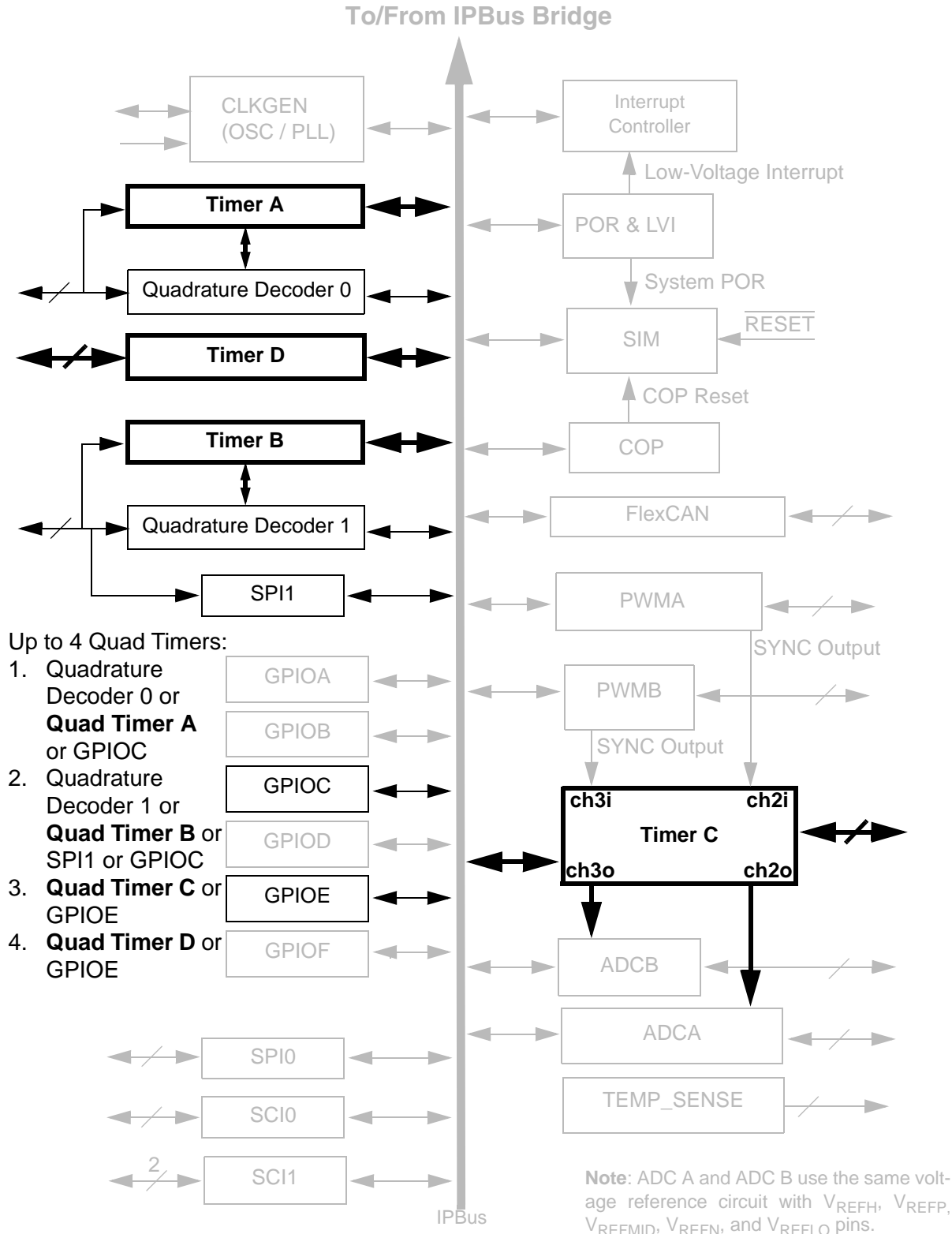
## 15.9 Resets

The sensor is turned off as a result of any system reset.



# Chapter 16

## Quad Timer (TMR)



Quad Timer (TMR), Rev. 10

## Document Revision History for Chapter 16, Quad Timer (TMR)

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 3.0	Added: "TCF asserts every time there is a compare event..." to Section 16.6.8.1
Rev 5.0	Added code examples in the Operating Modes section 16-1 through 16-12 Converted chapter to Freescale design standards Delineated differences in 8300 and 8100 TMR base clock rates, Section 6.5. page 7
Rev 6.0	Clarification verbiage added to Section 16.6.7.2
Rev 7.0	Clarification to Sections 16.6.7.4 and 16.6.8.7
Rev 8.0	Corrected description of change in document revision history for Rev 7.0.
Rev. 9	In Section 16.6.7.4 (TMRCTRL[CM] bit description), replaced the text " <i>...the timer is stopped by changing the timer's Count Mode to Stop Mode (CM=0)</i> " with " <i>...the timer module changes its Count mode to Stop Mode (CM=0)</i> ".



## 16.1 Introduction

The Quad Timer (TMR) module contains four identical counter/timer groups. Each 16-bit counter/timer group contains a:

- Prescaler
- Counter
- Load register
- Hold register
- Capture register
- Two compare registers
- One status and control register
- One control register

All of the registers except the prescaler are read/write capable.

**Note:** This document uses the terms *Timer* and *Counter* interchangeably because the counter/timers may perform either or both tasks.

The Load register provides the initialization value to the counter when the counter's terminal value has been reached. The Hold register captures the counter's value when other registers are being read. This feature supports the reading of cascaded counters. The Capture register enables an external signal to take a snapshot of the counter's current value.

TMRCMP1 and TMRCMP2 registers provide values to which the counter is compared. If a match occurs, the OFLAG signal can be set, cleared, or toggled. At match time, an interrupt is generated if enabled, and the new compare value is loaded into TMRCMP1 or TMRCMP2 registers from TMRCMPLD1 and TMRCMPLD2 when enabled.

The Prescaler provides different time bases useful for clocking the counter/timer. The Counter provides the ability to count internal or external events. Input pins can be shared within a Timer module (set of four timer/counters).

## 16.2 Features

- Four, 16-bit counters/timers
- Capable of counting up and down
- Counters will cascade
- Count modulo can be programmed
- Maximum count rate equals peripheral clock/2 for external clocks

- Maximum count rate equals peripheral clock for internal clocks
- Will count once or repeatedly
- Counters can be preloaded
- Counters can share available input pins
- Separate prescaler for each counter
- Each counter has capture and compare capability

### 16.3 Block Diagram

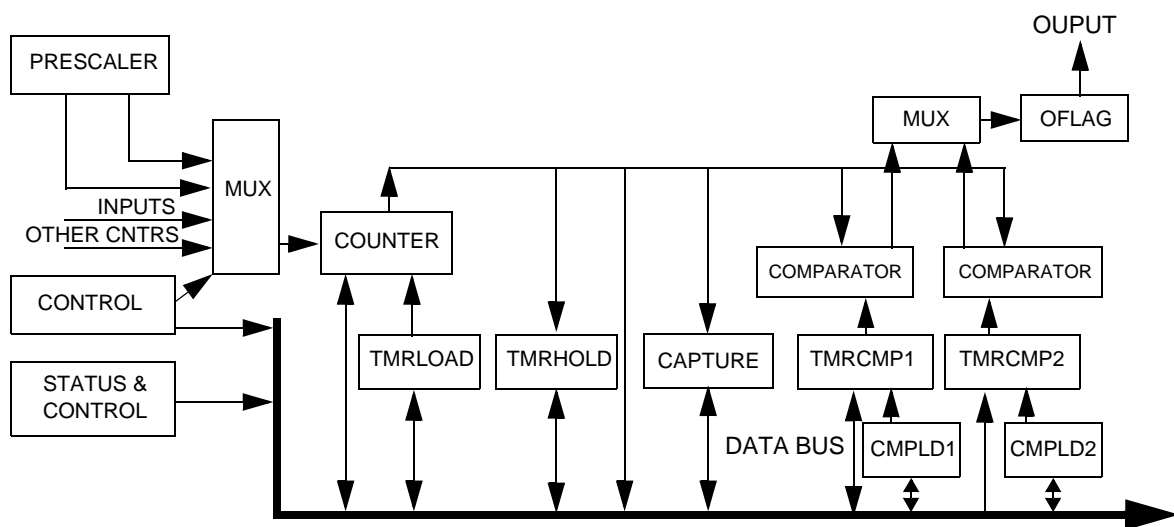


Figure 16-1. TMR Module Block Diagram

### 16.4 Functional Description

The counter/timer has two basic modes of operation:

1. Count internal or external events
2. Count an internal clock source while an external input signal is asserted, or an external event has occurred, thereby timing the width of the external input signal, or the time between external events.

The counter can count the rising, falling, or both edges of the selected input pin. The counter can decode and count quadrature encoded input signals. The counter can also count up and down using dual inputs in a count with direction format. The counter's terminal count value (modulo) is programmable. The value loaded into the counter after reaching its terminal count is

programmable. The counter can count repeatedly, or it can stop after completing one count cycle. The counter can be programmed to calculate a programmed value before immediately reinitializing, or it can count through the compare value until the count rolls over to zero.

The external inputs to each counter/timer can be shared among each of the four counter/timers within the module. The external inputs can be used to:

- Generate count commands
- Generate timer commands
- Trigger current counter value to be captured
- Generate interrupt requests

The polarity of the external inputs can be selected. The primary output of each timer/counter is the output signal, OFLAG. The OFLAG output signal can be set, cleared, or toggled when the counter reaches the programmed value. The OFLAG output signal may be output to an external pin shared with an external input signal.

The OFLAG output signal enables each counter to generate square waves, PWM, or pulse stream outputs. The polarity of the OFLAG output signal is selectable.

Any counter/timer can be assigned as a master. A master's compare signal can be broadcast to the other counter/timers within the module. The other counters can be configured to reinitialize their counters and/or force their OFLAG output signals to predetermined values when a master's counter/timer compare event occurs.

### 16.4.1 Compare Registers Usage

The dual Compare registers (TMRCMP1 and TMRCMP2) provide a bidirectional modulo count capability.

- TMRCMP1 register = used when the counter is *counting up*
- TMRCMP2 register = used when the counter is *counting down*

Alternating the Compare mode is the only exception.

The TMRCMP1 register should be set to the desired maximum count value, or \$FFFF, indicating the maximum unsigned value prior to roll-over. The TMRCMP2 register should be set to the minimum count value, or \$0000, indicating the minimum unsigned value prior to roll under.

If Output mode is set to 100, the OFLAG toggles while using alternating compare registers. In this variable frequency PWM mode, the TMRCMP2 value defines the desired pulse width of the ON time. The TMRCMP1 register defines the OFF time. The variable frequency PWM mode is defined for positive counting only.

**Note:** Use caution when changing TMRCMP1 and TMRCMP2 while the counter is active. If the counter has already passed the new value, it will count to \$FFFF or \$0000 roll-over, then begin counting toward the new value.

The check is:  $\text{Count} = \text{TMRCMPx}$ , *not*  $\text{Count} > \text{TMRCMP1}$  or  $\text{Count} < \text{TMRCMP2}$ .

With the use of the TMRCMPLD1 and TMRCMPLD2 registers to compare values will help to minimize this problem.

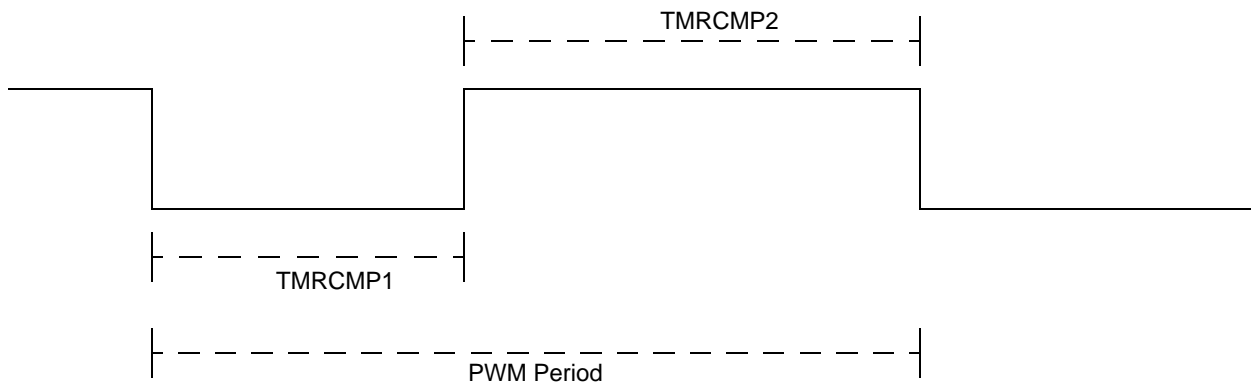
## 16.4.2 Compare Preload Registers

The TMRCMPLD1, TMRCMPLD2, and TMRCOMSCR offers a high degree of flexibility for loading compare registers with user-defined values on different compare events. To ensure correct functionality while using these registers, it is recommended to use the method described in this section.

The purpose of the Compare Preload feature is to allow quicker updating of the Compare registers. In prior families, a Compare register could be updated using interrupts. However, because of the latency between an interrupt event occurring and the service of that interrupt, there was the possibility the counter may have already counted past the new compare value by the time the Compare register is updated by the interrupt service routine. The counter would then continue counting until it rolled over and reached the new compare value.

To address this, the Compare registers are updated in hardware in the same way the Counter register is re-initialized to the value stored in the Load register. The Compare Preload feature allows calculation of new compare values to be stored in the Comparator Preload registers. When a compare event occurs, the new compare values in the Comparator Preload registers are written to the Compare registers, eliminating a software requirement to accomplish this.

The Compare Preload feature is intended to be used in a variable frequency PWM mode. The TMRCMP1 register determines the pulse width for the logic low part of OFLAG while TMRCMP2 determines the pulse width for the logic high part of OFLAG. The period of the waveform is determined by the sum of TMRCMP1 and TMRCMP2 values and the frequency of the primary clock source. Please see [Figure 16-2](#).



**Figure 16-2. Variable PWM Waveform**

If there is a desire to update the duty cycle or period of the above waveform, it is necessary to update the TMRCMP1 and TMRCMP2 values using the Compare Preload feature.

### 16.4.3 Capture Register Usage

The Capture register stores a copy of the counter's value when an input edge (positive, negative, or both) is detected. Once a capture event occurs, no further updating of the Capture register will occur until the Input Edge Flag (IEF) is cleared by writing a zero to the IEF.

## 16.5 Operating Modes

The TMR module design is always operating in Functional mode. The various counting modes are detailed in the following subsections.

Selected external count signals are sampled at the TMR's base clock rate (60MHz for the 8300 and 40MHz for the 8100 devices), then run through a transition detector. The maximum count rate is one-half of the base TMR's base clock rate. Internal clock sources can be used to clock the counters at the TMR's base clock rate. These signals are all at the same clock rate.

If a counter is programmed to count to a specific value and then stop, Count mode in CTLR is cleared when the count terminates.

### 16.5.1 Stop Mode

When Count mode field is set to 000, the counter is inert. No counting will occur.

## 16.5.2 Count Mode

When Count mode field is set to 001, the counter will count the rising edges of the selected clock source. This mode is useful for generating periodic interrupts for timing purposes, or counting external events such as *widgets* on a conveyor belt passing a sensor. If the selected input is inverted by setting the Input Polarity Select (IPS) bit, the negative edge of the selected external input signal is counted.

### Example 16-1. Count Pulses from External Source

```
//      (See Processor Expert PulseAccumulator bean.)
// This example uses TMRA1 to count pulse (actually counts rising edges of the
pulse)
// from an external source (TA3).
//
void Pulse_Init(void)
{
  /* TMRA1_CTRL: CM=0,PCS=3,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA1_CTRL,0x0600);          /* Set up mode */
  /* TMRA1_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
      Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA1_SCR,0x00);
  setReg(TMRA1_CNTR,0x00);           /* Reset counter register */
  setReg(TMRA1_LOAD,0x00);           /* Reset load register */
  setRegBitGroup(TMRA1_CTRL,CM,0x01); /* Run counter */
}
```

### Example 16-2. Generate Periodic Interrupt By Counting Internal Clocks

```
//      (See Processor Expert TimerInt bean.)
// This example generates an interrupt every 100ms,
// assuming the chip is operating at 60 MHz.
//
// It does this by using the IPBus clock divided by 128 as the counter clock source.
// The counter then counts to 46874 where it matches the CMP1 value.
// At that time an interrupt is generated, the counter is reloaded and
// the next CMP1 value is loaded from CMPLD1.
//
void TimerInt_Init(void)
{
    /* TMRA0_CTRL: CM=0,PCS=0,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
    setReg(TMRA0_CTRL,0x20);          /* Stop all functions of the timer */
    /* TMRA0_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
    setReg(TMRA0_SCR,0x00);
    setReg(TMRA0_LOAD,0x00);          /* Reset load register */
    setReg(TMRA0_CMP1,46874);         /* Set up compare 1 register */
    setReg(TMRA0_CMPLD1,46874);      /* Also set the compare preload register */
    /* TMRA0_COMSCR: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,TCF2EN=0,TCF1EN=1,
    TCF2=0,TCF1=0,CL2=0,CL1=1 */
    setReg(TMRA0_COMSCR,0x41);       /* Enable compare 1 interrupt and */
    /* compare 1 preload */
    setRegBitGroup(TMRA0_CTRL,PCS,0xF); /* Primary Count Source to IPBus clock / 128 */
    setReg(TMRA0_CNTR,0x00);         /* Reset counter register */
    setRegBitGroup(TMRA0_CTRL,CM,0x01); /* Run counter */
}

```

### 16.5.3 Edge Count Mode

When Count mode field is set to 010, the counter will count both edges of the selected external clock source. This mode is useful for counting the changes in the external environment such as a simple encoder wheel.

#### Example 16-3. Count Both Edges of External Source Signal

```
//      (See Processor Expert PulseAccumulator bean.)
// This example uses TMRA1 to count pulse (actually counts rising edges of the
pulse)
// from an external source (TA3).
//
void Pulse_Init(void)
{
    /* TMRA1_CTRL: CM=0,PCS=3,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
    setReg(TMRA1_CTRL,0x0600);       /* Set up mode */
    /* TMRA1_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
    setReg(TMRA1_SCR,0x00);
    setReg(TMRA1_CNTR,0x00);         /* Reset counter register */
    setReg(TMRA1_LOAD,0x00);         /* Reset load register */
    setRegBitGroup(TMRA1_CTRL,CM,0x02); /* Run counter */
}

```

## 16.5.4 Gated Count Mode

When Count mode field is set to 011, the counter will count while the selected secondary input signal is high. This mode is used to time the duration of external events. If the selected input is inverted by setting the Input Polarity Select (IPS) bit, the counter will count while the selected secondary input is low.

### Example 16-4. Capture Duration of External Pulse

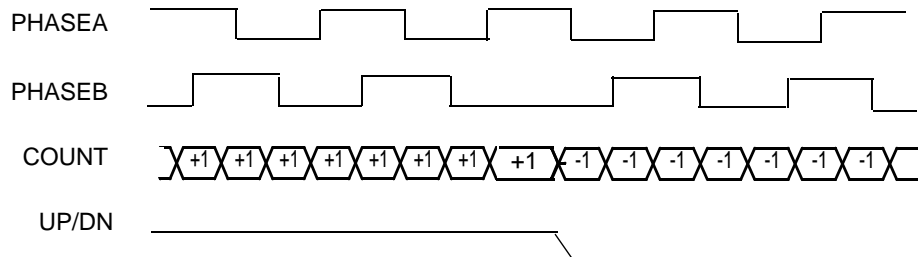
```
//      (See Processor Expert PulseAccumulator bean.)
// This example uses TMRA1 to determine the duration of an external pulse.
//
// The IPBus clock is used as the primary counter.  If the duration of the
// external pulse is longer than 0.001 seconds one of the other IPBus clock
// dividers can be used.  If the pulse duration is longer than 0.128 seconds
// an external clock source will have to be used as the primary clock source.
//
void Pulse1_Init(void)
{
    /* TMRA1_CTRL: CM=0,PCS=8,SCS=1,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
    setReg(TMRA1_CTRL,0x1080);          /* Set up mode */

    /* TMRA1_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
    setReg(TMRA1_SCR,0x00);
    setReg(TMRA1_CNTR,0x00);          /* Reset counter register */
    setReg(TMRA1_LOAD,0x00);         /* Reset load register */
    setReg Bit Group(TMRA1_CTRL,CM,0x03); /* Run counter */
}
```

## 16.5.5 Quadrature Count Mode

When Count mode field is set to 100, the counter will decode the primary and secondary external inputs as quadrature encoded signals. Quadrature signals are usually generated by rotary or linear sensors used to monitor movement of motor shafts or mechanical equipment. The quadrature signals are square waves, 90 degrees out-of-phase. The decoding of quadrature signal provides both count and direction information. A timing diagram illustrating the basic operation of a quadrature incremental position encoder is provided in [Figure 16-3](#).





**Figure 16-3. Quadrature Incremental Position Encoder**

### Example 16-5. Quadrature Count Mode Example

```

//      (See Processor Expert PulseAccumulator bean.)
//      This example uses TMRA0 for counting states of a quadrature position encoder.
//
//      Timer input 0 is used as the Primary Count Source (PHASEA).
//      Timer input 1 is used as the Secondary Count Source (PHASEB).
//
void Pulse_Init(void)
{
    /* TMRA0_CTRL: CM=0,PCS=0,SCS=1,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
    setReg(TMRC0_CTRL,0x80);          /* Set up mode */

    /* TMRA0_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
    setReg(TMRA0_SCR,0x00);

    setReg(TMRA0_CNTR,0x00);          /* Reset counter register */
    setReg(TMRA0_LOAD,0x00);         /* Reset load register */
    setReg(TMRA0_CMP1,0xFFFF);       /* Set up compare 1 register */
    setReg(TMRA0_CMP2,0x00);         /* Set up compare 2 register */

    /* TMRA0_COMSCR: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
    TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
    setReg(TMRA0_COMSCR,0x00);

    setRegBitGroup(TMRA0_CTRL,CM,0x04); /* Run counter */
}

```

## 16.5.6 Signed Count Mode

When Count mode field is set to 101, the counter counts the primary clock source while the selected secondary source provides the selected count direction (up/down).

### Example 16-6. Signed Count Mode Example

```

//      (See Processor Expert PulseAccumulator bean.)
//      This example uses TMRA0 for signed mode counting.
//
//      Timer input 2 is used as the Primary Count Source.
//      Timer input 1 is used to determine the count direction.
//
void Pulse_Init(void)
{
    /* TMRA0_CTRL: CM=0,PCS=2,SCS=1,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
    setReg(TMRA0_CTRL,0x0480);          /* Set up mode */

    /* TMRA0_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
       Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
    setReg(TMRA0_SCR,0x1000);

    setReg(TMRA0_CNTR,0x00);           /* Reset counter register */
    setReg(TMRA0_LOAD,0x00);          /* Reset load register */

    setRegBitGroup(TMRA0_CTRL,CM,0x05); /* Run counter */
}

```

## 16.5.7 Triggered Count Mode

When Count mode field is set to 110, the counter will begin counting the primary clock source after a positive transition (Negative Edge if IPS = 1) of the secondary input occurs. The counting will continue until a compare event occurs, or another positive input transition is detected. If a second input transition occurs before a terminal count was reached, counting will stop. Subsequent secondary input transitions will continue to cause the counting to restart and stop until a compare event occurs.

### Example 16-7. Triggered Count Mode Example

```

//      (See Processor Expert PulseAccumulator bean.)
//      This example uses TMRA1 for triggered mode counting.
//
//      Timer input 3 is used as the Primary Count Source.
//      Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
    /* TMRA1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
    setReg(TMRA1_CTRL,0x0700);          /* Set up mode */

    /* TMRA1_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
        Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
    setReg(TMRA1_SCR,0x1000);

    setReg(TMRA1_CNTR,0x00);           /* Reset counter register */
    setReg(TMRA1_LOAD,0x00);          /* Reset load register */
    setReg(TMRA1_CMP1,0x0012);        /* Set up compare 1 register */

    /* TMRA1_COMSCR: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
        TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
    setReg(TMRA1_COMSCR,0x00);

    setRegBitGroup(TMRA1_CTRL,CM,0x06); /* Run counter */
}

```

## 16.5.8 One-Shot Mode

This is a sub-mode of triggered event Count mode when Count mode field is set to 110 while:

- Count length (LENGTH) is set
- The OFLAG Output mode is set to 101
- ONCE bit of the Control Register (CTRL) is set to 1

Then the counter works in a One-Shot mode. An external event causes the counter to count. When terminal count is reached, the OFLAG output is asserted. This delayed output assertion can be used to provide timing delays. When used with timer C2 or C3, this one-shot mode may be used to delay the ADC acquisition of new samples until a specified period of time has passed since the Pulse Width Modulated (PWM) module asserted the synchronized signal.

### Example 16-8. One-Shot Mode Example

```

// (See Processor Expert PulseAccumulator bean.)
// This example uses TMR1 for one-shot mode counting.
//
// Timer input 3 is used as the Primary Count Source.
// Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
    /* TMR1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=5 */
    setReg(TMR1_CTRL,0x0725);          /* Set up mode */

    /* TMR1_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
    setReg(TMR1_SCR,0x1000);

    setReg(TMR1_CNTR,0x00);           /* Reset counter register */
    setReg(TMR1_LOAD,0x00);          /* Reset load register */
    setReg(TMR1_CMP1,0x0004);        /* Set up compare 1 register */

    /* TMR1_COMSCR: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
    TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
    setReg(TMR1_COMSCR,0x00);

    setRegBitGroup(TMR1_CTRL,CM,0x06); /* Run counter */
}

```

## 16.5.9 Cascade Count Mode

If Count mode field is set to 111, the counter's input is connected to the output of another selected counter. The counter will count up and down as compare events occur in the selected source counter. This Cascade, or Daisy-Chained mode, enables multiple counters to be cascaded in order to yield longer counter lengths. When operating in Cascade mode, a special high speed signal path is used between modules rather than the OFLAG output signal. If the selected source counter is counting up, and it experiences a compare event, the counter will be incremented. If the selected source counter is counting down and it experiences a compare event, the counter will be decremented. Up to four counters may be cascaded to create a 64-bit wide synchronous counter. Whenever any counter is read within a Counter module, all of the counters' values within the module are captured in their respective hold registers. This action supports the reading of a cascaded counter chain. First, read any counter of a cascaded counter chain, then read the Hold registers of the other counters in the chain. Cascaded Counter mode is synchronous.

**Note:** It is possible to connect counters together by using the other (non-cascade) Counter modes and selecting the outputs of other counters as a clock source. In this case, the counters are operating in a *ripple* mode, where higher order counters will transition a clock later than a purely synchronous design.

**Example 16-9. Generate Periodic Interrupt Cascading Two Counters**

```

//      (See Processor Expert TimerInt bean.)
// This example generates an interrupt every 30 seconds,
// assuming the chip is operating at 60 MHz.
//
// To do this, counter 2 is used to count 60,000 IPBus clocks, which means it
// will compare and reload every 0.001 seconds.
// Counter 3 is cascaded and used to count the 0.001 second ticks and
// generate the desired interrupt interval.
//
void TimerInt_Init(void)
{
// Set counter 2 to count IPBus clocks
/* TMRA2_CTRL: CM=0,PCS=8,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
setReg(TMRA2_CTRL,0x1020);          /* Stop all functions of the timer */

// Set counter 3 as cascaded and to count counter 2 outputs
/* TMRA3_CTRL: CM=7,PCS=6,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
setReg(TMRA3_CTRL,0xEC20);          /* Set up cascade counter mode */

/* TMRA3_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
setReg(TMRA3_SCR,0x00);
/* TMRA2_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
setReg(TMRA2_SCR,0x00);

setReg(TMRA3_CNTR,0x00);            /* Reset counter register */
setReg(TMRA2_CNTR,0x00);
setReg(TMRA3_LOAD,0x00);            /* Reset load register */
setReg(TMRA2_LOAD,0x00);
setReg(TMRA3_CMP1, 30000);          /* milliseconds in 30 seconds */
setReg(TMRA3_CMPLD1,30000);
setReg(TMRA2_CMP1, 60000);          // Set to cycle every milisecond
setReg(TMRA2_CMPLD1,60000);

/* TMRA3_COMSCR: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
TCF2EN=0,TCF1EN=1,TCF2=0,TCF1=0,CL2=0,CL1=1 */
setReg(TMRA3_COMSCR,0x41);          /* Enable compare 1 interrupt and */
/* compare 1 preload */
/* TMRA2_COMSCR: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=1 */
setReg(TMRA2_COMSCR,0x01);          /* Enable Compare 1 preload */

setRegBitGroup(TMRA2_CTRL,CM,0x01); /* Run counter */
}

```

**16.5.10 Pulse Output Mode**

The Counter will output a pulse stream of pulses with the same frequency of the selected clock source (can not be IPBus clock divided by 1) if the counter is setup for:

- Count mode (Mode = 001)
- The OFLAG Output mode is set to 111 (Gated Clock Output)
- The Count Once bit is set

The number of output pulses is equal to the compare value minus the initial value. This mode is useful for driving step motor systems.

**Note:** Primary Count Source must be set to one of the counter outputs for gated clock Output mode.

### Example 16-10. Pulse Outputs Using Two Counters

```
// (See Processor Expert PulseStream bean.)
// This example generates six 10ms pulses, from TA1 output.
// Assuming the chip is operating at 60 MHz.
//
// To do this, timer 3 is used to generate a clock with a period of 10ms.
//
// Timer 1 is used to gate these clocks and count the number of pulses that have
// been generated.
//
void PulseStream_Init(void)
{
// Select IPBus_clk/16 as the clock source for Timer A3
/* TMRA3_CTRL: CM=0,PCS=0x0C,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=3 */
setReg(TMRA3_CTRL,0x1823); /* Set up mode */
/* TMRA3_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
setReg(TMRA3_SCR,0x00);
setReg(TMRA3_LOAD,0x00); /* Reset load register */
setReg(TMRA3_CMP1,37500); /* (16 * 37500) / 60e6 = 0.01 sec */
/* TMRA3_COMSCR: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
setReg(TMRA3_COMSCR,0x00); /* Set up comparator control register */

// Timer 3 output is the clock source for this timer.
/* TMRA1_CTRL: CM=0,PCS=7,SCS=0,ONCE=1,LENGTH=1,DIR=0,Co_INIT=0,OM=7 */
setReg(TMRA1_CTRL,0x0E67); /* Set up mode */
/* TMRA1_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=1 */
setReg(TMRA1_SCR,0x01);
setReg(TMRA1_CNTR,0x00); /* Reset counter register */
setReg(TMRA1_LOAD,0x00); /* Reset load register */
setReg(TMRA1_CMP1,0x04); /* Set up compare 1 register */

// set to interrupt after the last pulse
/* TMRA1_COMSCR: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
TCF2EN=0,TCF1EN=1,TCF2=0,TCF1=0,CL2=0,CL1=0 */
setReg(TMRA1_COMSCR,0x40); /* Set up comparator control register */

// Finally, start the counters running
setReg(TMRA3_CNTR,0); /* Reset counter */

setRegBitGroup(TMRA3_CTRL,CM,0x01); /* Run source clock counter */
setRegBitGroup(TMRA1_CTRL,CM,0x01); /* Run counter */
}
```

### 16.5.11 Fixed Frequency PWM Mode

The Counter output yields a Pulse Width Modulated (PWM) signal with a frequency equal to the count clock frequency, divided by 65,536. It has a pulse width duty cycle equal to the compare value divided by 65,536, if the counter is setup for:

- Count mode (Mode = 001)
- Count through roll-over (Count Length = 0)
- Continuous count (Count Once = 0)
- OFLAG Output mode is 110 (set on compare, cleared on counter roll-over)

This mode of operation is often used to drive PWM amplifiers used to power motors and inverters.

#### Example 16-11. Fixed-Frequency PWM Mode Example

```
//      (See Processor Expert PWM bean.)
//      This example uses TMRA0 for Fixed-Frequency PWM mode timing.
//
//      The timer will count IPBus clocks continuously until it rolls over.
//      This results in a PWM period of 65536 / 60e6 = 1092.267 usec
//
//      Initially, an output pulse width of 25 usec is generated ( 1500 / 60e6 )
//      giving a PWM ratio of 1500 / 65536 = 2.289%
//      This pulse width can be changed by changing the CMP1 register value (using
//      CMPLD1).
//
void PWM1_Init(void)
{
    setReg(TMRA0_CNTR,0);          /* Reset counter */
    /* TMRA0_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
        Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=1,OPS=0,OEN=1 */
    setReg(TMRA0_SCR,0x05);       /* Enable output */
    setReg(TMRA0_CMP1,1500);      /* Store initial value to the duty-compare
register */

    /* TMRA0_CTRL: CM=1,PCS=8,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=6 */
    setReg(TMRA0_CTRL,0x3006);   /* Run counter */
}
```

### 16.5.12 Variable Frequency PWM Mode

If the counter is setup for:

- Continuous count (Count Once = 0)
- COUNT mode (mode = 001), count until compare (Count Length = 1)
- OFLAG Output mode is 100 (toggle OFLAG and alternate compare registers)

The counter output then yields a Pulse Width Modulated (PWM) signal whose frequency and pulse width is determined by the values programmed into the TMRCMP1 and TMRCMP2

registers and the input clock frequency. This method of PWM generation has the advantage of allowing almost any desired PWM frequency and/or constant on or off periods. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters. The TMRCMPLD1 and TMRCMPLD2 registers are especially useful for this mode because they permit time to calculate values for the next PWM cycle while the PWM current cycle is underway.

To setup the TMR to run in Variable Frequency PWM mode with compare preload please use the following setup for the specific counter you would like to use. When performing the setup, the CTLR is suggested to be updated *last* because the counter will start counting if Count mode is changed to any other value other than three bits with a value of 000, assuming the Primary Count Source is already active.

The following sections detail the timer settings required to implement variable frequency PWM operation.

### Example 16-12. Variable Frequency PWM Mode

```
//      (See Processor Expert PPG [Programmable Pulse Generator] bean.)
// This example starts with an 11 msec with a 31 msec cycle.
// Assuming the chip is operating at 60 MHz, the timer use IPBus_clk/32 as its
// clock source.
//
// Initial pulse period:  60e6/32 clocks/sec * 31 ms = 58125 total clocks in period
// Initial pulse width:   60e6/32 clocks/sec * 11 ms = 20625 clocks in pulse
//
//
// Once the initial values of CMP1/CMPLD1 and CMP2/CMPLD2 are set the pulse width
// can be varied by load new values of CMPLD1 and CMPLD2 on each compare
// interrupt.
// (See Section 16.4.2.)
//
void PPGL_Init(void)
{
    setReg(TMRA0_LOAD,0);           /* Clear load register */
    setReg(TMRA0_CNTR,0);          /* Clear counter */

    /* TMRA0_SCR: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=1,OPS=0,OEN=1 */
    setReg(TMRA0_SCR,5);           /* Set Status and Control Register */

    // Set compare preload operation and enable an interrupt on compare2 events.
    /* TMRA0_COMSCR: TCF2EN=1,TCF1EN=0,TCF2=0,TCF1=0,CL21=0,CL20=1,CL11=1,CL10=0 */
    setReg(TMRA0_COMSCR,0x86);     /* Set Comparator Status and Control Register
    */

    setReg(TMRA0_CMP1,20625);      /* Set the pulse width of the off time */
    setReg(TMRA0_CMPLD1,20625);   /* Set the pulse width of the off time */
    setReg(TMRA0_CMP2,58125-20625); /* Set the pulse width of the on time */
    setReg(TMRA0_CMPLD2,58125-20625); /* Set the pulse width of the on time */

    /* TMRA0_CTRL: CM=1,PCS=0xD,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=4 */
    setRegBits(TMRA0_CTRL,0x3A24); /* Set variable PWM mode and run counter */
}
```



## Timer Control Register (CTLR)

- Count Mode = 001 (count rising edges of primary source)
- Primary Count Source = 1000 (IPBus clock for best granularity for waveform timing)
- Secondary Count Source = Any (ignored in this mode)
- Count Once = 0 (want to count repeatedly)
- Count Length = 1 (want to count until compare value is reached and re-initialize counter register)
- Direction = Any (user's choice. The compare register values need to be chosen carefully to account for things like roll-under, etc.)
- Co-Channel Initialize = 0 (user can set this if they need this function)
- Output Mode = 100 (toggle OFLAG output using alternating compare registers)

## Timer Status and Control Register (TMRSCR)

- OEN = 1 (output enable to allow OFLAG output to be put on an external pin. Set this bit as needed)
- OPS = Any (user's choice)
- Make certain the rest of the bits are cleared for this register. Interrupts in the Comparator Status and Control register are enabled instead of this register.

## Comparator Status and Control Register (TMRCOMSCR)

- TCF2 enable = 1 (allow interrupt to be issued when TCF2 is set)
- TCF1 enable = 0 (do not allow interrupt to be issued when TCF1 is set)
- TCF1 = 0 (clear Timer Compare 1 Interrupt Source flag. This is set when Counter register equals Compare register 1 value and OFLAG is low)
- TCF2 = 0 (clear Timer Compare 2 Interrupt Source flag. This is set when Counter register equals Compare register 2 value and OFLAG is high)
- CL1[1:0] = 10 (load compare register when TCF2 is asserted)
- CL2[1:0] = 01 (load compare register when TCF1 is asserted)

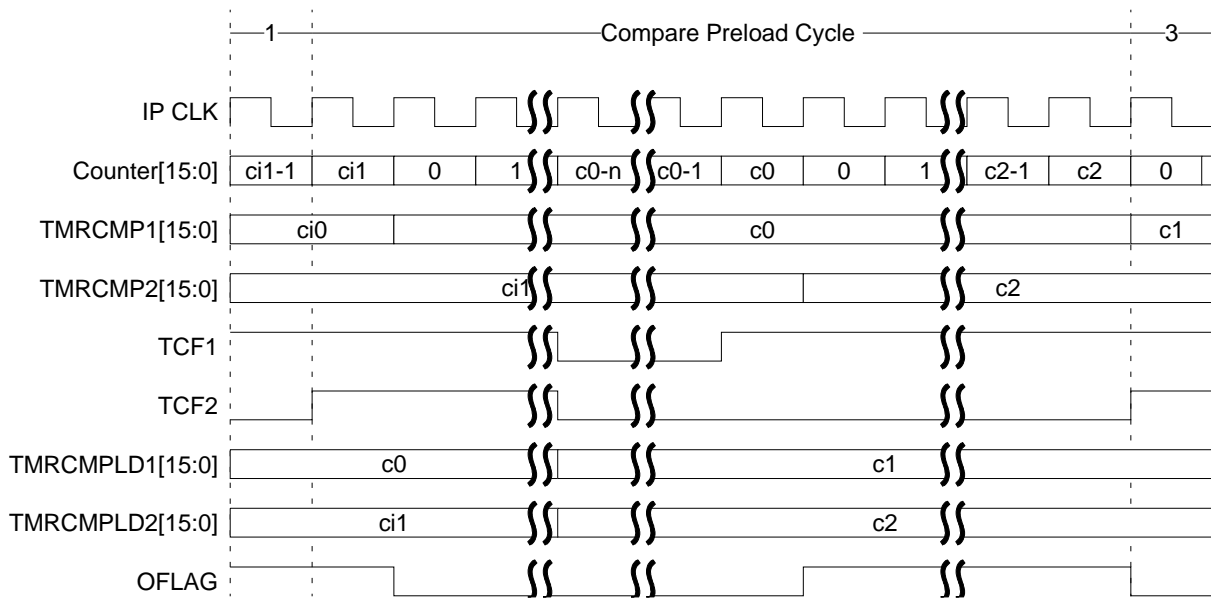
## Interrupt Service Routines

To service the TCF2 interrupts generated by the Timer, the Interrupt Controller must be configured, enabling the interrupts for the particular timer being used. Additionally is necessary to write an interrupt service routine to do the following at a minimum:

- Clear the TCF2 and TCF1 flags
- Calculate and write new values for both TMRCMPLD1 and TMRCMPLD2

## Timing

**Figure 16-4** contains the timing to use the Compare Preload feature. The Compare Preload cycle begins with a compare event on TMRCMP2, causing TCF2 to be asserted. TMRCMP1 is loaded with the value in the TMRCMPLD1 one IPBus clock later. Additionally, an interrupt is asserted by the timer and the interrupt service routine is executed. During this time both Comparator Load registers are updated with new values. When TCF1 is asserted, TMRCMP2 is loaded with the value in TMRCMPLD2. On the subsequent TCF2 event, TMRCMP1 is loaded with the value in TMRCMPLD1. The cycle starts over again as an interrupt is asserted and the interrupt service routine clears TCF1 and TCF2, calculating new values for TMRCMPLD1 and TMRCMPLD2.



**Figure 16-4. Compare Preload Timing**

## 16.6 Register Definitions

**Table 16-1. TMR Memory Map**

Device	Peripheral	Address
8100/8300	TMRA_BASE	\$00F040
	TMRB_BASE	\$00F080
	TMRC_BASE	\$00F0C0
	TMRD_BASE	\$00F100

A register address is the sum of a base address and an address offset. The base address is defined at the system level and the address offset is defined at the module level. TMR has 44 registers.

See the chip's data sheet for the base address of each timer module. Make certain to check which quad timer is available on the chip being used and which timer channels have external I/O.

**Table 16-2. TMR Register Summary**

Register Address Offsets	Register Acronym	Register Name	Access Type	Chapter Location
TMR_BASE + \$0, \$10, \$20, \$30	TMRCMP1	Compare Registers 1	Read/Write	<a href="#">Section 16.6.1</a>
TMR_BASE + \$1, \$11, \$21, \$31	TMRCMP2	Compare Registers 2	Read/Write	<a href="#">Section 16.6.2</a>
TMR_BASE + \$2, \$12, \$22, \$32	TMRCAP	Capture Registers	Read/Write	<a href="#">Section 16.6.3</a>
TMR_BASE + \$3, \$13, \$23, \$33	TMRLOAD	Load Registers	Read/Write	<a href="#">Section 16.6.4</a>
TMR_BASE + \$4, \$14, \$24, \$34	TMRHOLD	Hold Registers	Read/Write	<a href="#">Section 16.6.5</a>
TMR_BASE + \$5, \$15, \$25, \$35	TMRCNTR	Counter Registers	Read/Write	<a href="#">Section 16.6.6</a>
TMR_BASE + \$6, \$16, \$26, \$36	TMRCTRL	Control Registers	Read/Write	<a href="#">Section 16.6.7</a>
TMR_BASE + \$7, \$17, \$27, \$37	TMRSCR	Status and Control Registers	Read/Write	<a href="#">Section 16.6.8</a>
TMR_BASE + \$8, \$18, \$28, \$38	TMRCMPD1	Comparator Load Registers 1	Read/Write	<a href="#">Section 16.6.9</a>
TMR_BASE + \$9, \$19, \$29, \$39	TMRCMPD2	Comparator Load Registers 2	Read/Write	<a href="#">Section 16.6.10</a>
TMR_BASE + \$A, \$1A, \$2A, \$3A	TMRCOMSCR	Comparator Status and Control Register	Read/Write	<a href="#">Section 16.6.11</a>

Bit fields of each of the 44 registers are illustrated in [Figure 16-5](#). Details of each follow.

Add. Offset	Register Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Multi	TMRCMP1	R W	COMPARISON 1															
Multi	TMRCMP2	R W	COMPARISON 2															
Multi	TMRCAP	R W	CAPTURE															
Multi	TMRLOAD	R W	LOAD															
Multi	TMRHOLD	R W	HOLD															
Multi	TMRCNTR	R W	COUNTER															
Multi	TMRCTRL	R W	CM			PCS				SCS		ONCE	LENGTH	DIR	Co INIT	OM		
Multi	TMRSCR	R W	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE MODE		MSTR	EEOF	VAL	0 FORCE	OPS	OEN
Multi	TMRCMPD1	R W	COMPARATOR LOAD 1															
Multi	TMRCMPD2	R W	COMPARATOR LOAD 2															
Multi	TMRCOMSCR	R W	0	0	0	0	0	0	0	0	TCF2 EN	TCF1 EN	TCF2	TCF1	CL2		CL1	

R	0	Read as 0
W		Reserved

Figure 16-5. TMR Register Map Summary

### 16.6.1 Timer Compare Registers 1 (TMRCMP1)

These read/write registers store the value used for comparison with counter value. There are four Timer Compare1 (TMRCMP1) registers. Their addresses are:

TMR0\_CMP1 (Channel 0 Compare 1)—Address: TMR\_BASE + \$0  
 TMR1\_CMP1 (Channel 1 Compare 1)—Address: TMR\_BASE + \$10  
 TMR2\_CMP1 (Channel 2 Compare 1)—Address: TMR\_BASE + \$20  
 TMR3\_CMP1 (Channel 3 Compare 1)—Address: TMR\_BASE + \$30

Base +	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COMPARISON 1															
Write	COMPARISON 1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-6. TMR Compare Registers 1 (TMRCMP1)

## 16.6.2 Timer Compare Registers 2 (TMRCMP2)

These read/write registers store the value used for comparison with counter value. There are four Timer Compare2 (TMRCMP2) registers. Their addresses are:

TMR0\_CMP2 (Channel 0 Compare 2)—Address: TMR\_BASE + \$1  
 TMR1\_CMP2 (Channel 1 Compare 2)—Address: TMR\_BASE + \$11  
 TMR2\_CMP2 (Channel 2 Compare 2)—Address: TMR\_BASE + \$21  
 TMR3\_CMP2 (Channel 3 Compare 2)—Address: TMR\_BASE + \$31

Base +	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COMPARISON 2															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 16-7. TMR Compare 2 Registers (TMRCMP2)**

## 16.6.3 Timer Capture Registers (TMRCAP)

These read/write registers store the values captured from the counters. There are four Timer Capture (TMRCAP) registers. Their addresses are:

TMR0\_CAP (Channel 0 Capture)—Address: TMR\_BASE + \$2  
 TMR1\_CAP (Channel 1 Capture)—Address: TMR\_BASE + \$12  
 TMR2\_CAP (Channel 2 Capture)—Address: TMR\_BASE + \$22  
 TMR3\_CAP (Channel 3 Capture)—Address: TMR\_BASE + \$32

Base +	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTURE															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 16-8. TMR Capture Registers (TMRCAP)**

## 16.6.4 Timer Load Registers (TMRLOAD)

These read/write registers store the value used to load the counter. There are four Timer Load (TMRLOAD) registers. Their addresses are:

TMR0\_LOAD (Channel 0 LOAD)—Address: TMR\_BASE + \$3  
 TMR1\_LOAD (Channel 1 LOAD)—Address: TMR\_BASE + \$13  
 TMR2\_LOAD (Channel 2 LOAD)—Address: TMR\_BASE + \$23  
 TMR3\_LOAD (Channel 3 LOAD)—Address: TMR\_BASE + \$33

Base +	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LOAD															
Write	LOAD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-9. TMR Load Registers (TMRLOAD)

## 16.6.5 Timer Hold Registers (TMRHOLD)

These read/write registers store the channel's value whenever any timer counter register (TMRCNTR) is read. There are four Timer Hold (TMRHOLD) registers. Their addresses are:

TMR0\_HOLD (Channel 0 HOLD)—Address: TMR\_BASE + \$4  
 TMR1\_HOLD (Channel 1 HOLD)—Address: TMR\_BASE + \$14  
 TMR2\_HOLD (Channel 2 HOLD)—Address: TMR\_BASE + \$24  
 TMR3\_HOLD (Channel 3 HOLD)—Address: TMR\_BASE + \$34

Base +	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HOLD															
Write	HOLD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-10. TMR Hold Registers (TMRHOLD)

## 16.6.6 Timer Counter Registers (TMRCNTR)

These are read/write count registers. There are four Timer Count Registers (TMRCNTR). Their addresses are:

TMR0\_CNTR (Channel 0 Counter)—Address: TMR\_BASE + \$5  
 TMR1\_CNTR (Channel 1 Counter)—Address: TMR\_BASE + \$15  
 TMR2\_CNTR (Channel 2 Counter)—Address: TMR\_BASE + \$25  
 TMR3\_CNTR (Channel 3 Counter)—Address: TMR\_BASE + \$35

Base +	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COUNTER															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-11. TMR Counter Registers (TMRCNTR)

## 16.6.7 Timer Control Registers (TMRCTRL)

There are four Timer Control (TMRCTRL) registers. Their addresses are:

TMR0\_CTRL (Channel 0 Control)—Address: TMR\_BASE + \$6  
 TMR1\_CTRL (Channel 1 Control)—Address: TMR\_BASE + \$16  
 TMR2\_CTRL (Channel 2 Control)—Address: TMR\_BASE + \$26  
 TMR3\_CTRL (Channel 3 Control)—Address: TMR\_BASE + \$36

Base +	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Read	CM			PCS				SCS		ONCE	LENGTH	DIR	Co INIT	OM			
Write																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 16-12. TMR Channel Control Registers (TMRCTRL)

### 16.6.7.1 Count Mode (CM)—Bits 15–13

These bits control the basic counting behavior of the counter.

- 000 = No operation
- 001 = Count rising edges of Primary Source<sup>1</sup>
  - Rising edges if IPS = 0
  - Falling edges if IPS = 1
- 010 = Count rising and falling edges of Primary Source
- 011 = Count rising edges of Primary Source while secondary input high active
- 100 = Quadrature Count mode, uses Primary and Secondary Sources
- 101 = Count edges of Primary Source
  - IPS = 0 then Secondary Input = 0; Count down on rising edges of Primary Input
  - IPS = 0 then Secondary Input = 1; Count up on rising edges of Primary Input
  - IPS = 1 then Secondary Input = 0; Count down on falling edge of Primary Input
  - IPS = 1 then Secondary Input = 1; Count up on falling edges of Primary Input
- 110 = Edge of Secondary Source triggers primary count until compare
- 111 = Cascaded Counter mode (up/down)<sup>2</sup>

### 16.6.7.2 Primary Count Source (PCS)—Bits 12–9

These bits select the Primary Count Source. Some timer modules have their external pins multiplexed with Quadrature Decoder input pins. For these timers, refer to [Table 12-1](#) and [Table 12.7.1.15](#) to see how the timer and Quadrature Decoder input interact.

- 0000 = Counter 0 pin
- 0001 = Counter 1 pin
- 0010 = Counter 2 pin
- 0011 = Counter 3 pin
- 0100 = Counter 0 OFLAG
- 0101 = Counter 1 OFLAG
- 0110 = Counter 2 OFLAG
- 0111 = Counter 3 OFLAG

---

1.If Primary Count Source is IPBus clock, divide by one, then only rising edges are counted regardless of IPS value.

2.Primary Count Source must be set to one of the counter outputs.



- 1000 = Prescaler (IPBus clock divide by 1)
- 1001 = Prescaler (IPBus clock divide by 2)
- 1010 = Prescaler (IPBus clock divide by 4)
- 1011 = Prescaler (IPBus clock divide by 8)
- 1100 = Prescaler (IPBus clock divide by 16)
- 1101 = Prescaler (IPBus clock divide by 32)
- 1110 = Prescaler (IPBus clock divide by 64)
- 1111 = Prescaler (IPBus clock divide by 128)

**Note:** A timer selecting its own output for input is not a legal choice. The result is no counting.

### 16.6.7.3 Secondary Count Source (SCS)—Bits 8–7

These bits identify the external input pin to be used as a count command. The selected input can trigger the timer to capture the current value of the TMRCNTR register. The selected input can also be used to specify the count direction. The polarity of the signal can be inverted by the IPS bit of the TMRSCR register.

- 00 = Counter 0 pin
- 01 = Counter 1 pin
- 10 = Counter 2 pin
- 11 = Counter 3 pin

### 16.6.7.4 Count Once (ONCE)—Bit 6

This bit selects Continuous or One-Shot Counting modes.

- 0 = Count repeatedly
- 1 = Count until compare and then stop. If *counting up*, successful compare occurs when the counter reaches a TMRCMP1 value. If *counting down*, successful compare occurs when the counter reaches a TMRCMP2 value. When the compare occurs, the timer module changes its Count Mode to *Stop Mode* (CM=0).

### 16.6.7.5 Count Length (LENGTH)—Bit 5

This bit determines whether the counter counts to the compare value and then reinitializes itself to the value specified in the Load register, or the counter continues counting past the compare value, the binary roll-over.

- 0 = Roll-over
- 1 = Count till compare, then reinitialized. If *counting up*, successful compare occurs when the counter reaches a TMRCMP1 value. If *counting down*, successful compare occurs when the counter reaches a TMRCMP2 value<sup>1</sup>

#### 16.6.7.6 Count Direction (DIR)—Bit 4

This bit selects either the normal count direction *up*, or the reverse *down* direction.

- 0 = Count up
- 1 = Count down

#### 16.6.7.7 Co-Channel Initialization (Co INIT)—Bit 3

This bit enables another counter/timer within the same module to force the reinitialization of this counter/timer when it has an active compare event. The Master channel forcing reinitialization has MSTR bit set. Please see [Section 16.6.8.10](#).

- 0 = Co-Channel counter/timers cannot force a reinitialization of this counter/timer
- 1 = Co-Channel counter/timers can force a reinitialization of this counter/timer

#### 16.6.7.8 Output Mode (OM)—Bits 2–0

This bit field determines the mode of operation for the OFLAG output signal.

- 000 = Assert OFLAG while counter is active
- 001 = Clear OFLAG output on successful compare
- 010 = Set OFLAG output on successful compare
- 011 = Toggle OFLAG output on successful compare
- 100 = Toggle OFLAG output using alternating compare registers<sup>1</sup>
- 101 = Set OFLAG on compare, cleared on secondary source input edge
- 110 = Set OFLAG on compare, cleared on counter roll-over
- 111 = Enable Gated Clock output while counter is active

**Note:** Unexpected results may occur if Output mode field is set to use alternating Compare registers (mode 100) and the Count Once (ONCE) bit is set.

---

1. When the Output mode 100 is used, alternating values of TMRCMP1 and TMRCMP2 are used to generate successful comparisons. For example, when the Output mode is 100, the counter counts until TMRCMP1 value is reached, reinitializes, then counts until TMRCMP2 value is reached, reinitializes, then counts until TMRCMP1 value is reached, and so on.

## 16.6.8 Timer Status and Control Registers (TMRSCR)

There are four Timer Status/Control Registers (TMRSCR). Their addresses are:

TMR0\_SCR (Channel 0 Status and Control)—Address: TMR\_BASE + \$7  
 TMR1\_SCR (Channel 1 Status and Control)—Address: TMR\_BASE + \$17  
 TMR2\_SCR (Channel 2 Status and Control)—Address: TMR\_BASE + \$27  
 TMR3\_SCR (Channel 3 Status and Control)—Address: TMR\_BASE + \$38

Base +	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE MODE		MSTR	EEOF	VAL	0	OPS	OEN
Write														FORCE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-13. TMR Status and Control Registers (TMRSCR)

### 16.6.8.1 Timer Compare Flag (TCF)—Bit 15

This bit is set when a successful compare occurs. Clear the bit by writing zero to it.

TCF asserts every time there is a compare event. (either when counter == cmp1, or counter == cmp2)

### 16.6.8.2 Timer Compare Flag Interrupt Enable (TCFIE)—Bit 14

When set, the timer compare interrupt is enabled.

### 16.6.8.3 Timer Overflow Flag (TOF)—Bit 13

This bit is set when the counter rolls over its maximum or minimum value \$FFFF or \$0000, depending on count direction. Clear the bit by writing zero to it.

### 16.6.8.4 Timer Overflow Flag Interrupt Enable (TOFIE)—Bit 12

When set, this bit enables interrupts when the TOF bit is set.

### 16.6.8.5 Input Edge Flag (IEF)—Bit 11

This bit is set when a positive input transition occurs on an input selected as a Secondary Count Source while the counter is enabled. Clear the bit by writing zero to it.

**Note:** Setting the Input Polarity Select (IPS) bit enables the detection of negative input edge transitions. Also, the Control register's Secondary Count Source determines which external input pin is monitored by the detection circuitry.

### 16.6.8.6 Input Edge Flag Interrupt Enable (IEFIE)—Bit 10

When set, this bit enables interrupts if the IEF bit is set

### 16.6.8.7 Input Polarity Select (IPS)—Bit 9

When set, this bit inverts the polarity of both the primary and secondary inputs.

### 16.6.8.8 External Input Signal (INPUT)—Bit 8

This *read-only* bit reflects the current state of the external input pin selected via the Secondary Count Source after application of the IPS bit.

### 16.6.8.9 Input Capture Mode (CAPTURE MODE)—Bits 7–6

These bits specify the operation of the Capture register as well as the operation of the Input Edge Flag (IEF). The input source is the Secondary Count Source.

**Table 16-3. Setting Combination of Capture Mode and IPS State**

Capture Mode	IPS	Action
00	x	Capture Disabled
01	0	Load on Rising Edge
01	1	Load on Falling Edge
10	0	Load on Falling Edge
10	1	Load on Rising Edge
11	x	Load on Both Edges

### 16.6.8.10 Master Mode (MSTR)—Bit 5

When set, this bit enables the Compare register function output to be broadcasted to the other counter/timers in the module. This signal then can be used to reinitialize the other counters and/or force their OFLAG signal outputs. Other timer channels within the Quad Timer accept the reinitialization signal when their COINIT bit is set. Please see [Section 16.6.7.7](#).

### 16.6.8.11 Enable External OFLAG Force (EEOF)—Bit 4

When set, this bit enables the Compare register from another counter/timer within the same module to force the state of this counter's OFLAG output signal.

### 16.6.8.12 Forced OFLAG Value (VAL)—Bit 3

This bit determines the value of the OFLAG output signal when a software triggered FORCE command occurs, i.e. when FORCE = 1, discussed in [Section 16.6.8.13](#).

### 16.6.8.13 Force OFLAG Output (FORCE)—Bit 2

This *write-only* bit forces the current value of the VAL bit to be written to the OFLAG output. This bit is always read as a zero. The VAL and FORCE bits can be written simultaneously in a single write operation. Write to the FORCE bit only if the counter is disabled.

**Note:** Setting this bit while the counter is enabled may yield unpredictable results.

#### 16.6.8.14 Output Polarity Select (OPS)—Bit 1

This bit determines the polarity of the OFLAG output signal.

- 0 = True polarity
- 1 = Inverted polarity

#### 16.6.8.15 Output Enable (OEN)—Bit 0

When set, this bit determines the direction of the external pin.

- 0 = The external pin is configured as an input.
- 1 = OFLAG output signal will be driven on the external pin. Other timer groups using this external pin as their input will see the driven value. The polarity of the signal will be determined by the OPS bit.

### 16.6.9 Timer Comparator Load Registers 1 (TMRCMPLD1)

This read/write register is the Comparator 1 preload value for the TMRCMP1 register of the corresponding channel in a timer module. Please see [Section 16.4.2](#) for additional information.

There are four Timer Comparator1 (TMRCMPLD1) registers. Their addresses are:

TMR0\_CMPLD1 (Channel 0 CMPLD1)—Address: TMR\_BASE + \$8  
 TMR1\_CMPLD1 (Channel 1 CMPLD1)—Address: TMR\_BASE + \$18  
 TMR2\_CMPLD1 (Channel 2 CMPLD1)—Address: TMR\_BASE + \$28  
 TMR3\_CMPLD1 (Channel 3 CMPLD1)—Address: TMR\_BASE + \$38

Base +	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COMPARATOR LOAD 1															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 16-14. TMR Comparator Load Registers 1 (TMRCMPLD1)**

### 16.6.10 Timer Comparator Load Registers 2 (TMRCMPLD2)

This read/write register is the Comparator 2 preload value for the TMRCMP2 register of the corresponding channel in a timer module. There are four Timer Comparator 2 (TMRCMPLD2) registers. Please see [Section 16.4.2](#) for additional information. Their addresses are:

TMR0\_CMPLD2 (Channel 0 CMPLD2)—Address: TMR\_BASE + \$9  
 TMR1\_CMPLD2 (Channel 1 CMPLD2)—Address: TMR\_BASE + \$19  
 TMR2\_CMPLD2 (Channel 2 CMPLD2)—Address: TMR\_BASE + \$29  
 TMR3\_CMPLD2 (Channel 3 CMPLD2)—Address: TMR\_BASE + \$39

Base +	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COMPARATOR LOAD 2															
Write	COMPARATOR LOAD 2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 16-15. TMR Comparator Load Registers 2 (TMR CPLD2)**

### 16.6.11 Timer Comparator Status/Control Registers (TMR COMSCR)

This read/write register is the Timer Comparator Status and Control Register (TMR COMSCR). There are four TMR COMSCRs. Their addresses are:

TMR0\_COMSCR (Channel 0 COMSCR)—Address: TMR\_BASE + \$A  
 TMR1\_COMSCR (Channel 1 COMSCR)—Address: TMR\_BASE + \$1A  
 TMR2\_COMSCR (Channel 2 COMSCR)—Address: TMR\_BASE + \$2A  
 TMR3\_COMSCR (Channel 3 COMSCR)—Address: TMR\_BASE + \$3A

Base +	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 16-16. TMR Comparator Status/Control Registers (TMR COMSCR)**

#### 16.6.11.1 Reserved—Bits 15–8

This bit field is reserved or not implemented. It is read as 0 and cannot be modified by writing.

#### 16.6.11.2 Timer Compare 2 Interrupt Enable (TCF2EN)—Bit 7

An interrupt is issued when both this bit and the TCF2 bit are set.

#### 16.6.11.3 Timer Compare 1 Interrupt Enable (TCF1EN)—Bit 6

An interrupt is issued when both this bit and the TCF1 bit are set.

#### 16.6.11.4 Timer Compare Flag 2 (TCF2)—Bit 5

When set, this bit indicates a successful comparison of the timer and TMR CMP2 register has occurred. This bit is sticky, and will remain set until explicitly cleared by writing zero to this bit location.

### 16.6.11.5 Timer Compare Flag 1 (TCF1)—Bit 4

When set, this bit indicates a successful comparison of the timer and TMRCMP1 register has occurred. This bit is sticky, and will remain set until explicitly cleared by writing zero to this bit location.

### 16.6.11.6 Compare Load Control 2 (CL2)—Bit 3–2

These bits control when TMRCMP2 is preloaded with the value from TMRCMPLD2.

**Table 16-4. Values for Compare Preload Control 2**

Value	Meaning
00	Never Preload
01	Load Upon Successful Compare with the Value in TMRCMP1
10	Load Upon Successful Compare with the Value in TMRCMP2
11	Reserved

### 16.6.11.7 Compare Load Control 1 (CL1)—Bit 1–0

These bits control when TMRCMP1 is preloaded with the value from TMRCMPLD1.

**Table 16-5. Values for Compare Preload Control 1**

Value	Meaning
00	Never preload
01	Load Upon Successful Compare with the Value in TMRCMP1
10	Load Upon Successful Compare with the Value in TMRCMP2
11	Reserved

## 16.7 Clocks

The Timer operates from the IPBus Clock.

## 16.8 Interrupts

Each of the four timers in a timer module can generate an interrupt, serviced by the Interrupt Service Routine (ISR) identified by the Interrupt Vector table. Please see the chip's data sheet for more information on the Vector table. The ISR will have to check the Timer Status and Control Register (TMRSCR) and the Timer Comparator Status/Control Register (TMRCOMSCR) for which of the five interrupt flag bits are high.

**Table 16-6** describes each of these flag bits.

**Table 16-6. Timer Interrupt Flags**

Acronym	Name	Located In		Location
		TMRSCR	TMRCOMSCR	
TCF	Timer Compare Flag	x	—	<a href="#">Section 16.6.8.1</a>
TOF	Timer Overflow Flag	x	—	<a href="#">Section 16.6.8.3</a>
IEF	Input Edge Flag	x	—	<a href="#">Section 16.6.8.5</a>
TCF1	Timer Compare Flag 1	—	x	<a href="#">Section 16.6.11.5</a>
TCF2	Timer Compare Flag 2	—	x	<a href="#">Section 16.6.11.4</a>

The ISR should reset each set flag bit by writing zero to the bit.



---

# Chapter 17

## Voltage Regulator (VREG)

**Document Revision History for Chapter 17, Voltage Regulator (VREG)**

<b>Version History</b>	<b>Description of Change</b>
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 5.0	Converting chapter to Freescale design standards

## 17.1 Introduction

This module provides an on-chip mechanism to regulate an external 3.3V supply down to 2.5V levels for use with the internal core logic. On-board regulators are stable and have sufficient capacity and dynamic response allowing the host chip to operate correctly at all times and under all combinations of specified loads, frequencies, voltages, temperatures, and process variations.

## 17.2 Features

Qualities of the Linear Voltage Regulator contain:

- Provide a 2.5V  $\pm$ 10 percent accuracy
- Provide an average current of at least 250mA for the larger regulator
- Provide an average current of at least 1mA for the smaller regulators

## 17.3 Block Diagram

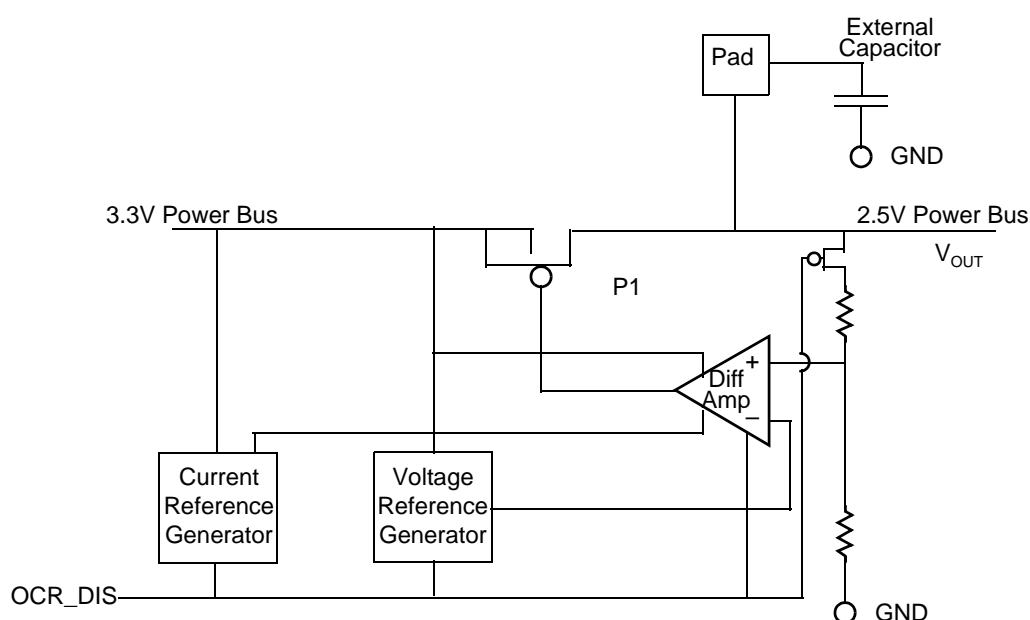


Figure 17-1. Voltage Regulator Block Diagram

## 17.4 Functional Description

Internal regulators (typically one for logic, and two for analog) are added to regulate down from 3.3 to 2.5V. Each of these regulators looks like the circuit illustrated in [Figure 17-1](#).

The regulator used is a linear series-pass low drop-out regulator. It uses a very large p-channel MOSFET as the pass device and an equally sized overload protection device. It takes an input of  $3.3V \pm 10$  percent and provides a regulated 2.5V output at a maximum current level of 250mA. It has a power down feature to facilitate the use of an external regulator on devices with  $4V_{CAP}$  pins and an OCR\_DIS pin available. When the internal regulator is disabled through OCR\_DIS, the  $V_{CAP}$  pins will be used to provide the 2.5V  $V_{DD\_CORE}$  core voltage. For devices with  $2V_{CAP}$  pins and no OCR\_DIS pin, the internal regulator must be used.

The regulator requires external  $2.2\mu F$  capacitors to be tied to the  $V_{CAP}$  pins to help maintain stability. For optimal transient performance, the  $2.2\mu F$  filter caps should be low ESR Multi-Layer Ceramic Chip (MLCC) caps and they should be placed as close as possible to the chip's  $V_{CAP}$  pins. We discourage the use of electrolytic capacitors because of their high ESR. Use of these capacitors cause the regulator to have poor transient load regulation performance. Filter capacitor values larger than  $2.2\mu F$  are allowed, improving the transient load regulation performance of the regulator. However, values less than  $2.2\mu F$  are not allowed.

## 17.5 Operating Modes

This is an analog part excited by the introduction of the input voltage. Its only mode of operation is ON.

On some parts, the large regulator for the on-chip digital logic can be disabled, or turned OFF. This can be useful if an external regulated 2.5V supply is provided by the system design. When the on-chip regulator is disabled the  $V_{CAP}$  pins are used to provide the regulated 2.5V  $V_{DD\_CORE}$  to the core logic.

## 17.6 Memory Map

This device has no memory mapped registers.

## 17.7 Pin Descriptions

**Table 17-1. Signal Properties**

Name	I/O Type	Function	Reset State	Notes
V <sub>OUT</sub>	DC Output	Output Voltage Supply	N/A	—
V <sub>IN</sub>	DC Source	Input Voltage Supply	N/A	—
V <sub>CAP1</sub>	—	Capacitor	N/A	Not included on the smaller regulators
V <sub>CAP2</sub>	—	Capacitor	N/A	Not included on the smaller regulators
V <sub>CAP3</sub>	—	Capacitor	N/A	Not included on the smaller regulators
V <sub>CAP4</sub>	—	Capacitor	N/A	Not included on the smaller regulators
OCR_DIS	Input	On-Chip Regulator Disable	N/A	Tie to V <sub>SS</sub> or V <sub>DD</sub>

### 17.7.1 Output Voltage (V<sub>OUT</sub>)

V<sub>OUT</sub> is the resultant 2.5V supplied to the core logic, Oscillator, and PLL.

### 17.7.2 Input Voltage (V<sub>IN</sub>)

V<sub>IN</sub> is the input voltage of 3.3V typically required by the regulator to convert to 2.5V.

### 17.7.3 Capacitor Pins (V<sub>CAP1</sub>, V<sub>CAP2</sub>, V<sub>CAP3</sub>, and V<sub>CAP4</sub>)

2.2μF capacitor pins are necessary on the larger regulator for proper operation.

### 17.7.4 On-Chip Regulator Disable (OCR\_DIS)

OCR\_DIS is available on devices which have 4 V<sub>CAP</sub> pins. This pin should be tied to either V<sub>SS</sub> or V<sub>DD</sub> at power-up.

- Tie this pin to V<sub>SS</sub> to enable the on-chip regulator
- Tie this pin to V<sub>DD</sub> to disable the on-chip regulator

**Note:** This pin is intended to be a static DC signal from power-up to shut down. One should not try and toggle this pin for power savings during operation.

## 17.8 Clocks

There are no clocks used by this module.

## **17.9 Resets**

There are no resets for this device.

## **17.10 Interrupts**

There are no interrupts generated by this module.



# Appendix A Glossary

## Document Revision History

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 5.0	Converted to Freescale design standards



## A.1 Glossary

This glossary is intended to reduce confusion potentially caused by the use of many acronyms and abbreviations throughout this manual.

<b>ACIM</b>	A/C Induction Motors
<b>A/D</b>	Analog-to-Digital
<b>ADC</b>	Analog-to-Digital Converter
<b>ADC Clock</b>	The ADC blocks run at a slower speed than the rest of the system. A separate ADC clock is derived from the system IPBus clock based upon the divisor specified in the ADC Control Register 2 (ADCR2).
<b>ADCR</b>	ADC Control Register
<b>ADDR</b>	Address
<b>ADHLMT</b>	ADC High Limit Registers
<b>ADLLMT</b>	ADC Low Limit Registers
<b>ADLST</b>	ADC Channel List Registers
<b>ADLSTAT</b>	ADC Limit Status Register
<b>ADM</b>	Application Development Module
<b>ADDFS</b>	ADC Offset Registers
<b>ADPOWER</b>	ADC Power Control Register
<b>ADR PD</b>	Address Bus Pull-up Disable
<b>ADRSLT</b>	ADC Result Registers
<b>ADSDIS</b>	ADC Sample Disable Register
<b>ADSTAT</b>	ADC Status Register
<b>ADZCC</b>	ADC Zero Crossing Control Register
<b>ADZCSTAT</b>	ADC Zero Crossing Status Register
<b>API</b>	Application Program Interface
<b>Array Write</b>	An array write is a write from the Hawk V2 to the HFM array space. The Program bus can write to the HFM Program or Boot array space only. The Primary Data bus can only write to the HFM Data array. All writes are via the cdbw[31:0] program bus but write addresses come from either program or primary address buses. The address and data are registered as part of the Command sequence prior to accessing the flash arrays.
<b>Banked Register</b>	A register operating on either Program and Boot Flash or Data Flash. Banked registers share the same register address offset as the equivalent registers for the other flash blocks. The active register bank is selected by a bank-select bit field in the unbanked register space. Flash Memory contains two sets of banked registers.

<b>BCR</b>	Bus Control Register in the EMI peripheral
<b>BDC</b>	Brush DC Motor
<b>BE</b>	Breakpoint Enable
<b>BFLASH</b>	Boot Flash
<b>BK</b>	Breakpoint Configuration Bit
<b>BLDC</b>	Brushless DC Motor
<b>BLKSZ</b>	Base Address and Block Size Register in the EMI peripheral
<b>BOTNEG</b>	Bottom-side PWM Polarity Bit
<b>BS</b>	Breakpoint Selection
<b>BSDL</b>	Boundary Scan Description Language
<b>BSR</b>	Boundary Scan Register in the JTAG peripheral
<b>CAN</b>	Controller Area Network
<b>CC</b>	Condition Codes
<b>CAP</b>	Capture
<b>CDBR</b>	Core Data Bus Read
<b>CDBW</b>	Core Data Bus Write
<b>CEN</b>	COP Enable Bit
<b>CFG</b>	Configuration
<b>CGM</b>	Clock Generator Module
<b>CGMDB</b>	Clock Generator Module Divide-By Register in the OCCS Module
<b>CGMTOD</b>	Clock Generator Module Time of Day Register in the OCCS Module
<b>CGMTST</b>	Clock Generator Module Test Register in the OCCS Module
<b>CHCNF</b>	Channel Configure
<b>CID</b>	Chip Identification Register in the JTAG peripheral
<b>CKDIVISOR</b>	Clock Divisor
<b>CLKO</b>	Clock Output pin
<b>CLKOSEL</b>	CLKO Select
<b>CLKOSR</b>	Clock Select Register
<b>CMOS</b>	Complementary metal oxide semiconductor. (A form of digital logic that is characterized by low power consumption, wide power supply range, and high noise immunity.)
<b>CMP</b>	Compare
<b>CNT</b>	Count
<b>CNTR</b>	Counter

<b>Codec</b>	Coder/Decoder
<b>Command Sequence</b>	A three-step hybrid controller instruction sequence to program or erase the Flash.
<b>Common Register</b>	A register operating and controlling all flash blocks. (See Unbanked Register)
<b>COP</b>	Computer Operating Properly
<b>COP/RTI</b>	Computer Operating Properly/Real Time Interface
<b>COPCTL</b>	COP Control
<b>COPCTR</b>	COP Count Register
<b>COPDIS</b>	COP Timer Disable
<b>COPR</b>	COP Reset
<b>COPTO</b>	COP Timeout
<b>CP</b>	Charge Pump
<b>CPHA</b>	Clock Phase
<b>CPOL</b>	Clock Polarity
<b>CRC</b>	Cyclic Redundancy Code
<b>CS</b>	Generic reference to Chip Select
<b>CSBAR</b>	Chip Select Base Address Registers in the EMI peripheral
<b>CSEN</b>	COP Stop Enable
<b><math>\overline{\text{CSn}}</math></b>	Reference to any of the Chip Selects
<b>CSOR</b>	Chip Select Option Register in the EMI peripheral
<b>CSTC</b>	Chip Select Timing Control Registers in the EMI peripheral
<b>CTL</b>	Control
<b>CTRL PD</b>	Control signal Pull-up Disable
<b>CVR</b>	Command Vector Register
<b>CWEN</b>	COP Wait Enable Bit
<b>CWP</b>	COP Write Protect
<b>DAC</b>	Digital to Analog Converter
<b>DDR</b>	Data Direction Register
<b>DEC</b>	Quadrature Decoder Module
<b>DECCR</b>	Decoder Control Register in the Quad Decoder peripheral
<b>DFLASH</b>	Data Flash
<b>DIE</b>	Watchdog Timeout Interrupt Enable
<b>DIRQ</b>	Watchdog Timeout Interrupt Request

<b>DR</b>	Data Register
<b>DRV</b>	Drive Control Bit
<b>DSO</b>	Data Shift Order
<b>DSP</b>	Digital Signal Processor (Hybrid Controller)
<b>DSP-Command Mode</b>	Functional test mode using the DSP (Hybrid Controller) to write individual commands to the Flash controller
<b>EDG</b>	Edge- or Center-Aligned PWMs
<b>EE</b>	Erase Enable
<b>EEOF</b>	Enable External OFLAG Force
<b>EM</b>	Event Modifier
<b>EMI</b>	External Memory Interface operates from the internal system bus, <i>not</i> from the IPBus
<b>EN</b>	Enable
<b>ENCR</b>	Encoder Control Register
<b>EOnCE</b>	Enhanced On-Chip Emulation (unit)
<b>EOSI</b>	End of Scan Interrupt
<b>EOSIE</b>	End of Scan Interrupt Enable
<b>ERASE</b>	Erase Cycle
<b>ERRIE</b>	Error Interrupt Enable
<b>ESR</b>	Equivalent Series Resistance
<b>EXTBOOT</b>	External Boot
<b>FAULT</b>	Fault Input to PWM
<b>FE</b>	Framing Error Flag
<b>FIR</b>	Filter Interval Register in the Quad Decoder peripheral
<b>FLAGx</b>	FAULT X Pin Flag
<b>Flash Erase Page</b>	Eight rows of 64 bytes (512 bytes) for all by 16-bit Flash modules. This will result in 1k bytes for interleaved blocks and 512 bytes for all other blocks being erased due to a page erase command.
<b>FIEx</b>	Faultx Pin Interrupt Enable
<b>Flash Module</b>	Includes Bus Interface, Command Control, and the Flash physical blocks
<b>FMODEx</b>	FAULTx Pin Clearing Mode
<b>FOSC</b>	Oscillator Frequency
<b>FPINx</b>	FAULT X Pin
<b>FREF</b>	Reference Frequency

<b>FTACKx</b>	FAULT X Pin Acknowledge
<b>GPIO</b>	General Purpose Input/Output
<b>Harvard Architecture</b>	A microprocessor architecture using separate buses for program and data. This architecture is typically used on Hybrid Controllers to optimize the data throughput.
<b>HLMTI</b>	High Limit Interrupt Bit
<b>HLMTIE</b>	High Limit Interrupt Enable Bit
<b>HOLD</b>	Hold Register
<b>HOME</b>	Home Switch Input
<b>IA</b>	Interrupt Assert
<b>IC</b>	Integrated Circuit
<b>IE</b>	Interrupt Enable
<b>IEF</b>	Input Edge Flag
<b>IEFIE</b>	Input Edge Flag Interrupt Enable
<b>IENR</b>	Interrupt Enable Register
<b>IES</b>	Interrupt Edge Sensitive
<b>IMR</b>	Input Monitor Register in the Quad Decoder peripheral
<b>INDEP</b>	Independent or Complimentary Pair Operation
<b>INDEX</b>	Index Input
<b>INIT</b>	Initialize Bit
<b>INPUT</b>	External Input Signal
<b>ITCN</b>	Interrupt Controller
<b>I/O</b>	Input/Output
<b>IP</b>	Intellectual Property
<b>IP</b>	Interrupt Pending
<b>IPBus</b>	Intellectual Properties Bus. This proprietary bus architecture accommodates various intellectual properties operating at slower speed than the system bus.
<b>IPBus Clock</b>	System peripheral clock
<b>IPE</b>	Intelligent Program Enable
<b>IPOL</b>	Current Polarity
<b>IPOLR</b>	Interrupt Polarity Register
<b>IPBB</b>	Interrupt Pending Bus Bridge
<b>IPR</b>	Interrupt Pending Register (in GPIO)
<b>IPR</b>	Interrupt Priority Register (in the Core)

<b>IPS</b>	Input Polarity Select
<b>IRQ</b>	Interrupt Request
<b>JTAG</b>	Joint Test Action Group
<b>JTAGBR</b>	JTAG Bypass Register
<b>JTAGIR</b>	JTAG Instruction Register
<b>LC</b>	Loop Count Register (in Core)
<b>LCK</b>	Loss of Lock
<b>LDOK</b>	Load OK
<b>LIR</b>	Lower Initialization Register in the Quad Decoder peripheral
<b>LLMTI</b>	Low Limit Interrupt
<b>LLMTIE</b>	Low Limit Interrupt Enable
<b>LOAD</b>	Load Register
<b>LOCI</b>	Loss of Clock Interrupt
<b>LOCIE</b>	Loss of Clock Interrupt Enable
<b>LOLI</b>	PLL Lock of Lock Interrupt
<b>LOOP</b>	Loop Select Bit
<b>LPOS</b>	Lower Position Counter Register
<b>LPOSH</b>	Lower Position Hold Register in the Quad Decoder peripheral
<b>LSB</b>	Least Significant Bit
<b>LSH_ID</b>	Least Significant Half of JTAG_ID
<b>LVI</b>	Low Voltage Interrupt
<b>LVIE</b>	Low Voltage Interrupt Enable
<b>MA</b>	Mode A
<b>MAC</b>	Multiply and Accumulate
<b>MB</b>	Mode B
<b>MCU</b>	Microcontroller Unit
<b>MHz</b>	Megahertz
<b>MISO</b>	Master In/Slave Out
<b>MODF</b>	Mode Fault Error
<b>MODFEN</b>	Mode Fault Enable
<b>MOSI</b>	Master Out/Slave In
<b>MSB</b>	Most Significant Bit

<b>MSH_ID</b>	Most Significant Half of JTAG ID
<b>MSTR</b>	Master Mode
<b>MUX</b>	Multiplexer
<b>NF</b>	Noise Flag
<b>OCCS</b>	On-Chip Clock Synthesis
<b>OEN</b>	Output Enable
<b>OMR</b>	Operating Mode Register
<b>OnCE</b>	On-Chip Emulation (unit)
<b>OPS</b>	Output Polarity Select
<b>OR</b>	Overrun
<b>OSCTL</b>	Oscillator Control Register in the OCCS peripheral
<b>OVRF</b>	Overflow
<b>PAB</b>	Program Address Bus
<b>PD</b>	Power Down, Pull-Up Disable
<b>PDB</b>	Program Data Bus
<b>PE</b>	Peripheral Enable
<b>PE</b>	Parity Enable Bit
<b>PER</b>	Peripheral Enable Register
<b>PF</b>	Parity Error Flag
<b>PFLASH</b>	Program Flash
<b>PLL</b>	Phase Locked Loop Module
<b>PLLCID</b>	PLL Clock In Divide
<b>PLLCOD</b>	PLL Clock Out Divide
<b>PLLDB</b>	PLL Divide-By Register in the OCCS peripheral
<b>PLLCR</b>	PLL Control Register in the OCCS peripheral
<b>PLLPDN</b>	PLL Power Down
<b>PLLSR</b>	PLL Status Register in the OCCS peripheral
<b>PMCCR</b>	PWM Channel Control Register
<b>PMCFG</b>	PWM Configuration Register
<b>PMCNT</b>	PWM Counter Register
<b>PMCTL</b>	PWM Control Register
<b>PMDEADTM</b>	PWM Deadtime Register

<b>PMDISMAP</b>	PWM Disable Mapping Registers
<b>PMICCR</b>	PWM Internal Correction Control Register
<b>PMFCTL</b>	PWM Fault Control Register
<b>PMFSA</b>	PWM Fault Status Acknowledge
<b>PMOUT</b>	PWM Output Control Register
<b>PMPORT</b>	PWM Port Register
<b>PWMCM</b>	PWM Counter Modulo Register
<b>PWMVAL</b>	PWM Value Registers
<b>POL</b>	Polarity
<b>POR</b>	Power on Reset
<b>POSD</b>	Position Difference Counter Register in the Quad Decoder peripheral
<b>POSDH</b>	Position Difference Counter Hold Register in the Quad Decoder peripheral
<b>POR_LOV</b>	Module name for the Power Supervisor_Low Voltage
<b>PRAM</b>	Program RAM
<b>PT</b>	Parity Type
<b>PUR</b>	Pull-up Enable Register
<b>PWM</b>	Pulse Width Modulator
<b>PWMEN</b>	PWM Enable
<b>PWMF</b>	PWM Reload Flag
<b>PWMRIE</b>	PWM Reload Interrupt Enable
<b>PWMVAL</b>	PWM Value Registers
<b>RAF</b>	Receiver Active Flag
<b>RAM</b>	Random Access Memory
<b>RDRF</b>	Receive Data Register Full
<b>RE</b>	Receiver Enable
<b>REIE</b>	Receive Error Interrupt Enable
<b>REV</b>	Revolution Counter Register in the Quad Decoder peripheral
<b>REVH</b>	Revolution Hold Register in the Quad Decoder peripheral
<b>RIDLE</b>	Receiver Idle Line
<b>RIE</b>	Receiver Full Interrupt Enable
<b>ROM</b>	Read Only Memory
<b>RSRC</b>	Receiver Source Bit




<b>RWU</b>	Receiver Wake up
<b>RXDF</b>	Receive Data Register Full Bit
<b>SBK</b>	Send Break
<b>SBR</b>	SCI Baud Rate
<b>SCI</b>	Serial Communications Interface
<b>SCIBR</b>	SCI Baud Rate Register in the SCI peripheral
<b>SCICR</b>	SCI Control Register in the SCI peripheral
<b>SCIDR</b>	SCI Data Register in the SCI peripheral
<b>SCISR</b>	SCI Status Register in the SCI peripheral
<b>SCLK</b>	Serial Clock
<b>SCR</b>	Status and Control
<b>S/H</b>	Sample and Hold
<b>Shift-DR</b>	This scan path captures and loads data into other core JTAG registers
<b>Shift-IR</b>	This scan path is used to capture and load store JTAG instructions
<b>SHUTDOWN</b>	Shutdown Register in the EMI peripheral
<b>SIM</b>	System Integration Module
<b>SPDRR</b>	SPI Data Receive Register in the SPI peripheral
<b>SPDSR</b>	SPI Data Size Register in the SPI peripheral
<b>SPDTR</b>	SPI Data Transmit Register in the SPI peripheral
<b>SP</b>	Stack Pointer
<b>SPI</b>	Serial Peripheral Interface
<b>SPMSTR</b>	SPI Master
<b>SPRF</b>	SPI Receiver Full
<b>SPRIE</b>	SPI Receiver Interrupt Enable
<b>SPSCR</b>	SPI Status Control Register in the SPI peripheral
<b>SPTE</b>	SPI Transmitter Empty
<b>SR</b>	Status Register
<b>SRM</b>	Switched Reluctance Motor
<b><math>\overline{SS}</math></b>	Slave Select
<b>SWAI</b>	Stop in Wait Mode
<b>SYS_CNTL</b>	System Control Register

<b>System Clock</b>	A free running clock generated by the PLL and oscillators. In reality, each system component will receive a unique version of this clock generated by the Clock Generation Module and controlled by the System Integration Module.
<b>SYS_STS</b>	System Status Register
<b>TAP</b>	Test Access Port
<b>TCF</b>	Timer Compare Flag
<b>TCFIE</b>	Timer Compare Flag Interrupt Enable
<b>TCK</b>	TAP Clock
<b>TDI</b>	TAP Data In
<b>TDO</b>	TAP Data Out
<b>TDRE</b>	Transmit Data Register Empty
<b>TE</b>	Transmitter Enable
<b>TEIE</b>	Transmitter Empty Interrupt Enable
<b>TSENSOR_CTL</b>	Temperature Sensor Control Register
<b>TIDLE</b>	Transmitter Idle
<b>TIIE</b>	Transmitter Idle Interrupt Enable
<b>TLM</b>	TAP Linking Module
<b>TMR</b>	Quadrature Timer
<b>TMRCAP</b>	Timer Capture Registers in the Quad Timer peripheral
<b>TMRCMP</b>	Timer Compare Registers in the Quad Timer peripheral
<b>TMRCMPLD</b>	Timer Comparator Load Registers in the Quad Timer peripheral
<b>TMRCNTR</b>	Timer Counter Registers in the Quad Timer peripheral
<b>TMRCOMSCR</b>	Timer Comparator Status and Control Register in the Quad Timer peripheral
<b>TMRCTRL</b>	Timer Control Register in the Quad Timer peripheral
<b>TMRLOAD</b>	Timer Load Registers in the Quad Timer peripheral
<b>TMRHOLD</b>	Timer Hold Registers in the Quad Timer peripheral
<b>TMR PD</b>	Timer I/O Pull-up Disable
<b>TMRSCR</b>	Timer Status and Control Register in the Quad Timer peripheral
<b>TOF</b>	Timer Overflow Flag
<b>TOFIE</b>	Timer Overflow Flag Interrupt Enable
<b>TOPNEG</b>	Top side PWM Polarity Bit
<b>TSENSOR_CONTROL</b>	Temperature Sensor Control Register

<b>UIR</b>	Upper Initialization Register in the Quad Decoder peripheral
<b>Unbanked Register</b>	A register operating and controlling all flash blocks (See Common Register)
<b>UPOS</b>	Upper Position Counter Register in the Quad Decoder peripheral
<b>UPOSH</b>	Upper Position Hold Register in the Quad Decoder peripheral
<b>VCO</b>	Voltage Controlled Oscillator
<b>V<sub>DD</sub></b>	Power
<b>V<sub>DDA</sub></b>	Analog Power
<b>VEL</b>	Velocity Counter Register
<b>VELH</b>	Velocity Hold Register
<b>VLMODE</b>	Value Register Load Mode
<b>V<sub>REF</sub></b>	Voltage Reference
<b>VRM</b>	Variable Reluctance Motor
<b>V<sub>SS</sub></b>	Ground
<b>V<sub>SSA</sub></b>	Analog Ground
<b>WAKE</b>	Wake up Condition
<b>WDE</b>	Watchdog Enable
<b>WP</b>	Write Protect
<b>WTR</b>	Watchdog Timeout Register in the Quad Decoder peripheral
<b>XDB2</b>	X Data Bus
<b>XIE</b>	Index Pulse Interrupt Enable
<b>XIRQ</b>	Index Pulse Interrupt Request
<b>XNE</b>	Use Negative Edge of Index Pulse
<b>XRAM</b>	Data RAM
<b>ZCI</b>	Zero Crossing Interrupt
<b>ZCIE</b>	Zero Crossing Interrupt Enable
<b>ZCS</b>	Zero Crossing Status





# **Appendix B**

## **Programmer's Sheets**

## Document Revision History for Appendix B

Version History	Description of Change
Rev 1.0	Pre-Release version, Alpha customers only
Rev 2.0	Initial Public Release
Rev 3.0	Correcting cross references and minor non technical additional edits
Rev 5.0	Converted this appendix to Freescale design standards Page B-13, Clock DIV; added statement delineating the 8100 differences Page B-39, PLLDB; added Note delineating the 8100 differences
Rev 8.0	Minor formatting edits.

## B.1 Introduction

The following pages provide a set of reference tables and programming sheets intended to simplify programming the MC56F8300 Family. The programming sheets provide room to add the value of each bit and the hexadecimal value for each register. These pages may be photocopied.

For complete instruction set details, refer to Chapter 4 of the *DSP56800E Reference Manual*.

## B.2 Programmer's Sheets

The following pages provide programmer's sheets summarizing functions of the bits in various registers found in this manual. The programmer's sheets provide room to write-in the value of each bit and the hexadecimal value for each register. Programmers may photocopy these sheets.

The programmer's sheets are arranged in the same order as the sections in this document.

**Table B-1** lists the programmer's sheets by module, the registers in each module, and the pages in this appendix where the programmer's sheets are located.

**Note:** Reserved bits should only be set to 0 unless otherwise stated.

**Note:** Please see the appropriate device Data Sheet for register information for references to *Data Sht* in the following table.

**Table B-1. List of Programmer's Sheets**

Register Type	Register	Page/ Figure
<b>CPU Registers</b>		
Bus Control Register	BCR	<b>B-35</b>
Operating Mode Register	OMR	Data Sht
Interrupt Priority Register	IPR	Data Sht
<b>Interrupt Control (ITCN)</b> ITCN_BASE: 56F8300 = \$00F1A0		
Interrupt Priority Register 0-9	IPR0-9	Data Sht
Vector Base Address Register	VBA	Data Sht
Fast Interrupt Match Register 0	FIM0	Data Sht
Fast Interrupt Vector Address Low 0 Register	FIVAL0	Data Sht
Fast Interrupt Vector Address High 0 Register	FIVAH0	Data Sht
Fast Interrupt Match Register 1	FIM1	Data Sht
Fast Interrupt Vector Address High 1 Register	FIVAL1	Data Sht

**Table B-1. List of Programmer's Sheets (Continued)**

Register Type	Register	Page/ Figure
Fast Interrupt Vector Address Low 1 Register	FIVAH1	Data Sht
IRQ Pending Register 0-5	IRQP0-5	Data Sht
Interrupt Control Register	CTRL	Data Sht

Enhanced On Chip Emulation (EOnCE)		EOnCE_BASE = \$FFF8A-\$FFFFFF
External Signal Control Register	OESCR	Data Sht
Breakpoint [0] Unit Counter	OBCNTR	Data Sht
Breakpoint 1 Unit [0] Mask Register	OBMSK (32 Bits)	Data Sht
Breakpoint 2 Unit [0] Address Register	OBAR2 (32 Bits)	Data Sht
Breakpoint 1 Unit [0] Address Register	OBAR1 (24 Bits)	Data Sht
Breakpoint Unit [0] Control Register	OBCR (24 Bits)	Data Sht
Trace Buffer Register Stages	OTB (21-24 Bits/stage)	Data Sht
Trace Buffer Pointer Register	OTBPR (8 Bits)	Data Sht
Trace Buffer Control Register	OTBCR	Data Sht
Peripheral Base Address Register	OBASE (8 Bits)	Data Sht
Status Register	OSR	Data Sht
Instruction Step Counter	OXCNTR (24 Bits)	Data Sht
Control Register	OCR	Data Sht
Core Lock/Unlock Status Register	OCLSR (8 Bits)	Data Sht
Transmit and Receive Status and Control Register	OTRXSR (8 Bits)	Data Sht
Transmit Register/Receive Register	OTX/ORX (32 Bits)	Data Sht
Transmit Register Upper Word/Receive Register Upper Word	OTX1/ORX1	Data Sht

System Integration Module (SIM)		SIM_BASE: 56F8300 = \$00F350
Control Register	SIM_CNTL	Data Sht
Reset Status Register	SIM_RSTSTS	Data Sht
Software Control Register 0-3	SIM_SCR0-3	Data Sht
Most Significant Half of JTAG ID	SIM_MSH_ID	Data Sht
Least Significant Half of JTAG ID	SIM_LSH_ID	Data Sht
Pull-Up Disable Register	SIM_PUDR	Data Sht
Clock Out Select Register	SIM_CLKOSR	Data Sht
Quad Decoder 1/Timer B/SPI1 Select Register	SIM_GPS	Data Sht
Peripheral Clock Enable Register	SIM_PCE	Data Sht
I/O Short Address Location High Register	SIM_ISALH	Data Sht
I/O Short Address Location Low Register	SIM_ISALL	Data Sht

Analog to Digital Converter (ADC)		ADCA_BASE: 56F8300 = \$00F200
-----------------------------------	--	-------------------------------



**Table B-1. List of Programmer's Sheets (Continued)**

Register Type	Register	Page/ Figure
<b>ADCB_BASE: 56F8300 = \$00F240</b>		
Control Register 1	ADCR1	B-11
Control Register 2	ADCR2	B-14
Zero Crossing Control Register	ADZCC	B-15
Channel List Register 1	ADLST1	B-16
Channel List Register 2	ADLST2	B-17
Sample Disable Register	ADSDIS	B-18
Status Register	ADSTAT	B-19
Limit Status Register	ADLSTAT	B-20
Zero Crossing Status Register	ADZCSTAT	B-21
Result Registers 0-7	ADRSLT0-7	B-22
Low Limit Register 0-7	ADLLMT0-7	B-23
High Limit Register 0-7	ADHLMT0-7	B-23
Offset Registers 0-7	ADOF0-7	B-24
Power Control Register	ADPOWER	B-25
Calibration Register	ADC-CAL	B-28

<b>Computer Operating Properly (COP)</b>		<b>COP_BASE: 56F8300 = \$00F2C0</b>
Control Register	COPCTL	B-29
Timeout Register	COPTO	B-30
Counter Register	COPCTR	B-31

<b>External Memory Interface (EMI)</b>		<b>EMI_BASE = \$00F020</b>
Chip Select Base Address Register 0-7	CS0-7	B-32
Chip Select Option Register 0-7	CSOR0-7	B-33
Chip Select Timing Control Register 0-7	CSTC0-7	B-34
Bus Control Register	BCR	B-35

<b>On Chip Clock Synthesis (OCCS)</b>		<b>CLKGEN_BASE: 56F801/803/805 = \$00F2D0</b>
Control Register w/o Relax Oscillator	PLLCR	B-36
Control Register w/ Relax Oscillator	PLLCR	B-38
Divide-By Register	PLLDDB	B-40
Status Register	PLLSR	B-41
Shutdown Register	SHUTDOWN	B-42
Oscillator Control Register w/o Relax	OSCTL	B-43

**Table B-1. List of Programmer's Sheets (Continued)**

Register Type	Register	Page/ Figure
Oscillator Control Register w/ Relax	OSCTL	B-44
<b>Flash Module (FM)</b> <span style="float: right;"><b>FM_BASE: 56F8300 = \$00F400</b></span>		
Clock Divider Register	FMCLKD	B-45
Configuration Register	FMCR	B-46
Security High Register	FMSECH	B-47
Security Low Register	FMSECL	B-47
Optional Data Register 0-2	FMOPT0-2	B-48
Protection Register	FMPROT	B-49
Protection Boot Register	FMPROTB	B-50
User Status Register	FMUSTAT	B-51
Command Buffer Register	FMCMD	B-52
<b>FlexCAN (FC)</b> <span style="float: right;"><b>FC_BASE: 56F8300 = \$00F800</b></span>		
Module Configuration Register	FCMCR	B-53
Control Register 0	FCCTL0	B-55
Control Register 1	FCCTL1	B-57
Free-Running Timer Register	FCTIMER	B-58
Maximum Message Buffer Register	FCMAXMB	B-59
Receive Global Mask High Register	FCRXMASK_H	B-60
Receive Global Mask Low Register	FCRXMASK_L	B-60
Receive Buffer 14 Mask High Register	FCRX14MASK_H	B-61
Receive Buffer 14 Mask Low Register	FCRX14MASK_L	B-61
Receive Buffer 15 Mask High Register	FCRX15MASK_H	B-62
Receive Buffer 15 Mask Low Register	FCRX15MASK_L	B-62
Error and Status Register	FCSTATUS	B-63
Interrupt Mask Register 1	FCIMASK1	B-65
Interrupt Flag Register 1	FCIFLAG1	B-66
Error Counters Register	FCERR_CNTRS	B-67
<b>General Purpose Input/Output (GPIO)</b> <span style="float: right;"><b>GPIOA_BASE: 56F8300 = \$00F2E0</b></span>		
<b>GPIOB_BASE: 56F8300 = \$00F300</b>		
<b>GPIOC_BASE: 56F8300 = \$00F310</b>		
<b>GPIOD_BASE: 56F8300 = \$00F320</b>		
<b>GPIOE_BASE: 56F8300 = \$00F330</b>		
<b>GPIOF_BASE: 56F8300 = \$00F340</b>		
Pull-Up Enable Register	GPIO <sub>n</sub> _PUR	B-68

**Table B-1. List of Programmer's Sheets (Continued)**

Register Type	Register	Page/ Figure
Data Register	GPIO <sub>n</sub> _DR	B-69
Data Direction Register	GPIO <sub>n</sub> _DDR	B-70
Peripheral Enable Register	GPIO <sub>n</sub> _PER	B-71
Interrupt Assert Register	GPIO <sub>n</sub> _IAR	B-72
Interrupt Enable Register	GPIO <sub>n</sub> _IENR	B-73
Interrupt Polarity Register	GPIO <sub>n</sub> _IPOLR	B-74
Interrupt Pending Register	GPIO <sub>n</sub> _IPR	B-75
Interrupt Edge Sensitive Register	GPIO <sub>n</sub> _IESR	B-76
Push/Pull Output Mode Control Register	GPIO <sub>n</sub> _PPMODE	B-77
Raw Data Register	GPIO <sub>n</sub> _RAWDATA	B-78

<b>Power Supervisor (PS)</b>		<b>PS_BASE = \$00F360</b>
Control Register	LVICTLR	B-79
Status Register	LVISR	B-80

<b>Pulse Width Module (PWM)</b>		<b>PWMA_BASE: 56F8300 = \$00F140</b>
<b>PWMB_BASE: 56F8300 = \$00F160</b>		
Control Register	PMCTL	B-81
Fault Control Register	PMFCTL	B-83
Fault Status and Acknowledge Register	PMFSA	B-84
Output Control Register	PMOUT	B-85
Counter Register	PMCNT	B-86
Counter Modulo Register	PMCM	B-87
Value Registers	PMVAL0-5	B-88
Deadtime Register	PMDEADTIME	B-89
Disable Mapping Register 1	PMDISMAP1	B-90
Disable Mapping Register 2	PMDISMAP2	B-90
Configuration Register	PMCFG	B-91
Channel Control Register	PMCCR	B-93
Port Register	PMPORT	B-95
Internal Correction Control Register	PMICCR	B-96

<b>Quadrature Decoder (DEC)</b>		<b>DEC0_BASE: 56F8300 = \$00F180</b>
<b>DEC1_BASE: 56F8300 = \$00F190</b>		
Control Register	DECCR	B-97
Filter Delay Register	FIR	B-101

**Table B-1. List of Programmer's Sheets (Continued)**

Register Type	Register	Page/ Figure
Watchdog Timer Register	WTR	B-102
Position Difference Counter Register	POSD	B-103
Position Difference Hold Register	POSDH	B-106
Revolution Counter Register	REV	B-106
Revolution Hold Register	RE VH	B-106
Upper Position Counter Register	UPOS	B-107
Lower Position Counter Register	LPOS	B-108
Upper Position Hold Register	UPOSH	B-108
Lower Position Hold Register	LPOSH	B-108
Upper Initialization Register	UIR	B-109
Lower Initialization Register	LIR	B-109
Input Monitor Register	IMR	B-110

<b>Serial Communication Interface (SCI)</b>		<b>SCI0_BASE: 56F8300 = \$00F280</b>
		<b>SCI1_BASE: 56F8300 = \$00F290</b>
Baud Rate Register	SCIBR	B-111
Control Register	SCICR	B-112
Status Register	SCISR	B-116
Data Register	SCIDR	B-117

<b>Serial Peripheral Interface (SPI)</b>		<b>SPI0_BASE: 56F8300 = \$00F2A0</b>
		<b>SPI1_BASE: 56F8300 = \$00F2B0</b>
Status and Control Register	SPSCR	B-118
Data Size and Control Register	SPDSR	B-121
Data Receive Register	SPDRR	B-122
Data Transmit Register	SPDTR	B-123

<b>TSENSOR (TSEN)</b>		<b>TSENSOR_BASE = \$00F270</b>
Control Register	TSENSOR_CTRL	B-124

<b>Quad Timer (TMR)</b>		<b>TMRA_BASE: 56F8300 = \$00F040</b>
		<b>TMRB_BASE: 56F8300 = \$00F080</b>
		<b>TMRC_BASE: 56F8300 = \$00F0C0</b>
		<b>TMRD_BASE: 56F8300 = \$00F100</b>
Compare Registers 1	TMRCMP1	B-125
Compare Registers 2	TMRCMP2	B-126

**Table B-1. List of Programmer's Sheets (Continued)**

<b>Register Type</b>	<b>Register</b>	<b>Page/ Figure</b>
Capture Registers	TMRCAP	<a href="#">B-127</a>
Load Registers	TMRLOAD	<a href="#">B-128</a>
Hold Registers	TMRHOLD	<a href="#">B-129</a>
Counter Registers	TMRCNTR	<a href="#">B-130</a>
Control Registers	TMRCTRL	<a href="#">B-131</a>
Status and Control Registers	TMRSCR	<a href="#">B-132</a>
Comparator Load Registers 1	TMRCMPLD1	<a href="#">B-136</a>
Comparator Load Registers 2	TMRCMPLD2	<a href="#">B-137</a>
Comparator Status/Control Registers	TMRCOMSCR	<a href="#">B-138</a>



Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Control Register 1 (ADCR1)

Please see the following page for continuation of this register

Bits	Name	Description
14	<b>STOP</b>	<b>Stop</b>
		When selected, the current conversion process is stopped. Any further sync pulses or modifications to the START bit are ignored until the STOP bit has been cleared. After the ADC is in Stop mode, the results registers can be modified by the processor. <b>Note:</b> This is not the same as Stop mode for the whole chip.
		0 Normal operation
		1 Stop command issued
13	<b>START</b>	<b>Start</b>
		The conversion process is started by a modification to the <i>write-only</i> START bit. Once a start command is issued, the conversion process is synchronized and begun on the positive edge of the ADC clock. A resumed START bit is ignored while the ADC is in a conversion cycle. The ADC must be idle for it to recognize a new start-up.
		0 No action
		1 Start command issued
12	<b>SYNC</b>	<b>Synchronization</b>
		A conversion can be initiated by the SYNC input or by a write to the START bit. A SYNC pulse restarted while the ADC is in a conversion cycle is ignored. The ADC must be idle for it to recognize a new start-up.
		0 Conversion is initiated by a write to START bit only
		1 Use the sync input or START bit to initiate a conversion
11	<b>EOSIE</b>	<b>End of Scan Interrupt Enable</b>
		This bit enables the generation of an interrupt upon completion of any scan and convert sequence, except in Loop Sequential or Loop Simultaneous modes. This bit is ignored when the ADC is configured for either Loop modes. (There is no <i>end</i> of a scan in Loop mode.)
		0 Interrupt disabled
		1 Interrupt enabled

ADC Control Register 1 (ADCR1) BASE+\$0		Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		0	STOP	0	SYNC	EOSIE	ZCIE	LLMTIE	HLMTIE	CHNCRG				0	SMODE			
Write				START														
Reset		0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Control Register 1 (ADCR1) Continued

Please see the following page for continuation of this register

Bits	Name	Description
10	ZCIE	<b>Zero Crossing Interrupt Enable</b>
		This bit enables the generation of an optional interrupt if the current result value has a sign change from the previous result, configured by the ADZCC register.
		0 Interrupt disabled
		1 Interrupt enabled
9	LLMTIE	<b>Low Limit Interrupt Enable</b>
		This bit enables the generation of an optional interrupt when the current result value is less than the Low Limit register value. The raw result (i.e. without offset) value is compared to the ADC Low Limit (ADLLMT $n$ ) register corresponding to the sample.
		0 Interrupt disabled
		1 Interrupt enabled
8	HLMTIE	<b>High Limit Interrupt Enable</b>
		This bit enables the generation of an optional interrupt if the current result value is greater than the High Limit register value. The raw result (i.e. without offset) value is compared to the ADC High Limit (ADHLMT $n$ ) register corresponding to the sample.
		0 Interrupt disabled
		1 Interrupt enabled
7 - 4	CHNCRG	<b>Channel Configure</b>
		Each bit in the four bit field determines the configuration of an analog pin input pair as either single ended or differential.
		xxx1 Inputs AN0–AN1 configured as differential inputs (AN0 is + and AN1 is -)
		xxx0 Inputs AN0–AN1 configured as single ended inputs
		xx1x Inputs AN2–AN3 configured as differential inputs (AN2 is + and AN3 is -)
		xx0x Inputs AN2–AN3 configured as singled ended inputs
		x1xx Inputs AN4–AN5 configured as differential inputs (AN4 is + and AN5 is -)
		x0xx Inputs AN4–AN5 configured as singled ended inputs
		1xxx Inputs AN6–AN7 configured as differential inputs (AN6 is + and AN7 is -)
		0xxx Inputs AN6–AN7 configured as singled ended inputs

ADC Control Register 1 (ADCR1) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	STOP	0	SYNC	EOSIE	ZCIE	LLMTIE	HLMTIE	CHNCRG				0	SMODE			
	Write			START														
	Reset	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1

Reserved Bits



Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# ADC

## ADC Control Register 1 (ADCR1) Continued

Bits	Name	Description
2 - 0	<b>SMODE</b>	<b>ADC Mode Control</b>
		Determines ADC scan mode. Mode can be either Once, Loop, or Triggered and Sequential (I) or Simultaneous (I/Q).
	000	Once sequential
	001	Once simultaneous
	010	Loop sequential
	011	Loop simultaneous
	100	Triggered sequential
	101	Triggered simultaneous (default)
	110	Reserved use
	111	Reserved use

ADC Control Register 1 (ADCR1) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	STOP	0	SYNC	EOSIE	ZCIE	LLMTIE	HLMTIE	CHNCRG				0	SMODE		
	Write			START													
	Reset	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Control Register 2 (ADCR2)

Bits	Name	Description
4 - 0	DIV	<b>Clock Divisor Select</b>
<p>ADC_CLK must be run at a slower rate than the system clock. The divider circuit provides a clock of period 2N of system clock where <math>N = \text{DIV}[3:0] + 1</math>. A divisor must be selected so the ADC clock is within specified limits. For an IPBus clock frequency of 60MHz and a desired ADC_CLK frequency of 5MHz, a DIV minimum value of 5 is required. Five is the default value for DIV.</p> <p>For an IPBus clock frequency of 40MHz and a desired ADC_CLK frequency of 5MHz, a DIV minimum value of 3 is required. Five is the default value which will result in an ADC_CLK frequency of 3.33MHz.</p>		

ADC Control Register 2 (ADCR2) BASE+\$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	DIV				
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Reserved Bits

**Appendix B - Programmer's Sheets, Rev. 10 Draft A**

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# ADC

## ADC Zero Crossing Control Register (ADZCC)

Bits	Name	Description
15-0	ZCE $n$	<b>Zero Crossing Enable <math>n</math></b>
		Representing the channel number, $n$ , setting the ZCE $n$ field allows detection of the indicated zero crossing condition.
	00	Zero crossing disabled
	01	Zero crossing enabled for positive to negative sign change
	10	Zero crossing enabled for negative to positive sign change
	11	Zero crossing enabled for any sign change

ADC Zero Crossing Control Register (ADZCC) BASE+\$2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0	
	Write	ZCE7		ZCE6		ZCE5		ZCE4		ZCE3		ZCE2		ZCE1		ZCE0	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Channel List Registers (ADLST1 and ADLST2)

Please see the following page for continuation of this register

Bits	Name	Description
14-12	SAMPLE3	<b>Sample3</b>
		For single-ended inputs, using Sequential (I) samplings, each SAMPLE $n$ register contains the numeric value corresponding to the analog input pin assigned to that sample. For Simultaneous (I/Q) sampling and differential inputs, please see <a href="#">Section 2.11.1</a> .
10-8	SAMPLE2	<b>Sample2</b>
		For single-ended inputs, using Sequential (I) samplings, each SAMPLE $n$ register contains the numeric value corresponding to the analog input pin assigned to that sample. For Simultaneous (I/Q) sampling and differential inputs, please see <a href="#">Section 2.11.1</a> .
6-4	SAMPLE1	<b>Sample1</b>
		For single-ended inputs, using Sequential (I) samplings, each SAMPLE $n$ register contains the numeric value corresponding to the analog input pin assigned to that sample. For Simultaneous (I/Q) sampling and differential inputs, please see <a href="#">Section 2.11.1</a> .
2-0	SAMPLE0	<b>Sample0</b>
		For single-ended inputs, using Sequential (I) samplings, each SAMPLE $n$ register contains the numeric value corresponding to the analog input pin assigned to that sample. For Simultaneous (I/Q) sampling and differential inputs, please see <a href="#">Section 2.11.1</a> .

ADC Channel List	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register 1 (ADLST1) BASE+\$3	Read	0	SAMPLE3			0	SAMPLE2			0	SAMPLE1			0	SAMPLE0		
	Write																
	Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

 Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# ADC

## ADC Channel List Registers (ADLST1 and ADLST2) Continued

Bits	Name	Description
14-12	SAMPLE7	<b>Sample7</b>
		For single-ended inputs, using Sequential (I) samplings, each SAMPLE $n$ register contains the numeric value corresponding to the analog input pin assigned to that sample. For Simultaneous (I/Q) sampling and differential inputs, please see <a href="#">Section 2.11.1</a> .
10-8	SAMPLE6	<b>Sample6</b>
		For single-ended inputs, using Sequential (I) samplings, each SAMPLE $n$ register contains the numeric value corresponding to the analog input pin assigned to that sample. For Simultaneous (I/Q) sampling and differential inputs, please see <a href="#">Section 2.11.1</a> .
6-4	SAMPLE5	<b>Sample5</b>
		For single-ended inputs, using Sequential (I) samplings, each SAMPLE $n$ register contains the numeric value corresponding to the analog input pin assigned to that sample. For Simultaneous (I/Q) sampling and differential inputs, please see <a href="#">Section 2.11.1</a> .
2-0	SAMPLE4	<b>Sample4</b>
		For single-ended inputs, using Sequential (I) samplings, each SAMPLE $n$ register contains the numeric value corresponding to the analog input pin assigned to that sample. For Simultaneous (I/Q) sampling and differential inputs, please see <a href="#">Section 2.11.1</a> .

ADC Channel List	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Register 2 (ADLST2) BASE+\$4</b>	Read	0	SAMPLE7			0	SAMPLE6			0	SAMPLE5			0	SAMPLE4		
	Write																
	Reset	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Sample Disable Register (ADSDIS)

Bits	Name	Description
15-14	TEST	<b>Test</b>
		The ADC core has a built-in test mode allowing the $V_{REFH}/2$ to be applied to AN0 and AN4.
		00 Normal mode
		01 Test mode, $V_{REFH}/2$ applied to AN0 and AN4
		10 Reserved
		11 Reserved
7-0	DS $n$	<b>Disable Sample7-0</b>
		The respective SAMPLE $n$ can be enabled or disabled where $n = 0-7$ .
		<b>Note:</b> When TEST is configured for Test mode, VREF is applied to AN0 and AN4 between the analog muxing and the ADC core. Only AN0 and AN4 will have valid results so only AN0 and AN4 should be sampled.
		0 Enable SAMPLE $n$
		1 Disable SAMPLE $n$

ADC Sample Disable Register (ADSDIS) BASE+\$5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TEST		0	0	0	0	0	0	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Status Register (ADSTAT)

Bits	Name	Description
15	<b>CIP</b>	<b>Conversion in Progress</b>
		This bit indicates when a scan is in progress.
		0 Idle state
		1 A scan cycle is in progress; the ADC will ignore all sync pulses or start commands
11	<b>EOSI</b>	<b>End of Scan Interrupt</b>
		This bit indicates if a scan of analog inputs has been completed since the last read of the status register, or since a reset. The EOSI bit is cleared by writing one to it. This bit cannot be set by software.
		0 Scan is not completed; no end of scan IRQ pending
		1 Scan cycle is completed; end of scan IRQ pending
10	<b>ZCI</b>	<b>Zero Crossing Interrupt</b>
		For each of the samples taken, if the corresponding Offset register has a value, 0000h < value < 7FF8h, then zero crossing checking is enabled. Outside of this range of values, a zero crossing is not possible.
		0 No ZCI IRQ
		1 Zero crossing encountered; IRQ pending if ZCIE is set.
9	<b>LLMTI</b>	<b>Low Limit Interrupt</b>
		If the respective Low Limit register is enabled by having a value other than 0000h, low limit checking is enabled.
		0 No low limit IRQ
		1 Low limit exceeded; IRQ pending if LLMTIE is set
8	<b>HLMTI</b>	<b>High Limit Interrupt</b>
		If the respective High Limit register is enabled by having a value less than 7FF8h, high limit checking is enabled.
		0 No high limit IRQ
		1 High limit exceeded; IRQ pending if HLMTIE is set
7-0	<b>RDY<sub>n</sub></b>	<b>Ready Sample7-0</b>
		These bits indicate whether samples seven through zero are ready to be read. These bits are cleared after a read from the respective results register.
		0 Sample not ready or has been read
		1 Sample ready to be read

<b>ADC Status Register (ADSTAT) BASE+\$6</b>	<b>Bits</b>	<b>15</b>	<b>14</b>	<b>13</b>	<b>12</b>	<b>11</b>	<b>10</b>	<b>9</b>	<b>8</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>	
	<b>Read</b>	CIP	0	0	0	EOSI	ZCI	LLMTI	HLMTI	RDY7	RDY6	RDY5	RDY4	RDY3	RDY2	RDY1	RDY0	
	<b>Write</b>																	
	<b>Reset</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Limit Status Register (ADLSTAT)

Bits	Name	Description
15-8	<b>HLS<sub>n</sub></b>	<b>High Limit Status<sub>n</sub></b>
		The Limit Status register latches in the result of the comparison between the result of the sample and the respective limit register, ADHLMT0-7 and ADLLMT0-7.  <b>Note:</b> The sample value used in the comparison is the value captured before the application of the offset value in ADOFS <sub>n</sub> register.
7-0	<b>LLS<sub>n</sub></b>	<b>Low Limit Status<sub>n</sub></b>
		The Limit Status register latches in the result of the comparison between the result of the sample and the respective limit register, ADHLMT0-7 and ADLLMT0-7.  <b>Note:</b> The sample value used in the comparison is the value captured before the application of the offset value in ADOFS <sub>n</sub> register.

ADC Limit Status Register (ADLSTAT) BASE+\$7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	HLS7	HLS6	HLS5	HLS4	HLS3	HLS2	HLS1	HLS0	LLS7	LLS6	LLS5	LLS4	LLS3	LLS2	LLS1	LLS0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Zero Crossing Status Register (ADZCSTAT)

Bits	Name	Description
7-0	ZCS $n$	<b>Zero Crossing Status</b>
		The zero crossing condition is determined by examining the ADC value after it is adjusted by the offset for the result register. Each bit of the register is cleared by writing 1 to the register bit.
	0	A sign change did not occur in a comparison between the current channel $n$ result and previous channel $n$ result, or Zero crossing control is disabled
	1	In a comparison between the current channel $n$ result and the previous channel $n$ result, a sign change occurred as defined in the ADZCC register.

ADC Zero Crossing Status Register (ADZCSTAT) BASE+\$8	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	ZCS7	ZCS6	ZCS5	ZCS4	ZCS3	ZCS2	ZCS1	ZCS0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Result Registers (ADRSLT0-7)

Bits	Name	Description
15	SEXT	<b>Sign Extend</b>
		SEXT is the sign-extend bit of the result. SEXT set to one implies a negative result. Set to zero, the implication is a positive one. If only positive results are required, then the respective offset register must be set to a value of zero.
14-3	RSLT	<b>Digital Result of the Conversion</b>
		ADRSLT can be interpreted as either a signed integer or a signed fractional number. As a signed fractional number, the ADRSLT can be used directly. As a signed integer, it is an option to right shift with sign extend (ASR) three places and interpret the number, or accept the number as presented, knowing there are missing codes. The lower three bits are always going to be zero.  Negative results (SEXT = 1) are always presented in two's complement format. If it is a requirement of application that the result registers always be positive, the offset registers must always be set to zero.
14-3	TEST_DATA	<b>Test Data</b>
		The eight Result registers contain the converted results from a scan. The SAMPLE0 result is loaded into ADRSLT0, SAMPLE1 result in ADRSLT1, and so on. In a Simultaneous Scan mode, the first channel pair designated by SAMPLE0 and SAMPLE4 in register ADLST1/2 are stored in ADRSLT0 and ADRSLT4, respectively.

ADC Result Registers (ADRSLT) BASE+\$9-\$10	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SEXT	RSLT											0	0	0	
	Write		TEST_DATA														
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Low and High Limit Registers (ADLLMT0-7 & ADHLMT0-7)

Bits	Name	Description
14-3	LLMT	<b>Low Limit</b>
		Both Limit registers are programmed with the value the result is compared against. The High Limit register is used for the comparison of <i>Result &gt; High Limit</i> . The Low Limit register is used for the comparison of the comparison of <i>Result &lt; Low Limit</i> . The limit checking can be disabled by programming the respective Limit register with \$FFF8 for the High Limit and \$0000 for the Low Limit register. The power-up default is limit checking disabled.
14-3	HLMT	<b>High Limit</b>
		Both Limit registers are programmed with the value the result is compared against. The High Limit register is used for the comparison of <i>Result &gt; High Limit</i> . The Low Limit register is used for the comparison of the comparison of <i>Result &lt; Low Limit</i> . The limit checking can be disabled by programming the respective Limit register with \$FFF8 for the High Limit and \$0000 for the Low Limit register. The power-up default is limit checking disabled.

<b>ADC Low Limit Registers (ADLMT0-7) BASE+\$11-\$18</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	LLMT											0	0	0	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

<b>ADC High Limit Registers (ADHLMT0-7) BASE+\$19-\$20</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	HLMT											0	0	0	
	Write																
	Reset	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

Reserved Bits

Application: \_\_\_\_\_ Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 14 of 18

# ADC

## ADC Offset Registers (ADOFs0-7)

Bits	Name	Description
14-3	OFFSET	Offset
		Value of the Offset register is used to correct the ADC result before it is stored in the ADRSLT registers. The offset value is subtracted from the ADC result. In order to obtain unsigned results, the respective offset register should be programmed with a value of \$0000, thus giving a result range of \$0000 to \$7FF8.

<b>ADC Offset Registers (ADOFs0-7) BASE+\$21-\$28</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	OFFSET											0	0	0	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Power Control Register (ADPOWER)

Please see the following page for continuation of this register

Bits	Name	Description
12	PSTS2	<b>Power Status 2</b>
		This <i>read-only</i> bit does not have a non-zero fixed ADC clock assertion time unlike PSTS0 and PSTS1. It simply reflects the current state of the control bit to the voltage reference circuit. Application software is responsible to ensure sufficient time being allocated for the reference to stabilize prior to beginning conversions.
	0	Voltage reference circuit is currently powered up
	1	Voltage reference circuit is currently powered down
11	PSTS1	<b>Power Status 1</b>
		This <i>read-only</i> bit is deasserted immediately following a write of one to PD1. It asserts PUDELAY cycles after a write of zero to PD1. Consequently, this bit can be read only as a status bit to determine when the ADC is ready for operation.
	0	ADC Converter 1 is currently powered up
	1	ADC Converter 0 is currently powered down
10	PSTS0	<b>Power Status 0</b>
		This <i>read-only</i> bit is deasserted immediately following a write of one to PD0. It asserts PUDELAY cycles after a write of zero to PD0. Consequently, this bit can be read only as a status bit to determine when the ADC is ready for operation.
	0	ADC Converter 1 is currently powered up
	1	ADC Converter 0 is currently powered down

ADC Power Control Register (ADPOWER) BASE+\$29	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read		0	0	0	PSTS2	PSTS1	PSTS0	PUDELAY						PSM	PD2	PD1	PD0
Write																	
Reset		0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Power Control Register (ADPOWER) Continued

Please see the following page for continuation of this register

Bits	Name	Description
9-4	PUDELAY	<b>Power Up Delay</b>
		This bit field determines the number of ADC clocks required for an ADC converter to exit from Power-Down mode or begin conversions after START/SYNC in Power Savings mode. In the latter case, it determines the delay in terms of ADC clocks between power-up initiated by a write to START or SYNC and initiation of an ADC conversion cycle. The default value is 13 ADC clocks. The binary value in this field is used to initialize a 6-bit count down counter. Counting begins when the START/SYNC occurs; the ADC conversion commences when the count hits zero.
3	PSM	<b>Power Savings Mode</b>
		This bit powers down both ADCs when written to a value of one. It modifies the operation of the SYNC and START bits in ACCR1 in the following way: instead of simply initiating a new conversion sequence, writing to START or a SYNC signal will power-up the ADC, wait a number of ADC clock cycles as determined by the PUIDELAY bits and then initiate a conversion sequence equivalent to that done when PSM is not active. At the end of the last conversion, the ADCs will be powered down again. A new SYNC or write to START will then begin the sequence over. SYNC pulses and writes to START during a conversion sequence are ignored.
		ADC converters are powered up in this mode only if needed for the programmed conversion. If only one ADC converter is required, then only that ADC converter is powered up.
		The voltage reference circuit remains powered up in Power Savings mode. This is because it normally requires 25 msec for the reference voltages to stabilize after power-up.
		0 Power Savings mode is not active
		1 Power Savings mode is active

ADC Power Control Register (ADPOWER) BASE+\$29	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	PSTS2	PSTS1	PSTS0	PUDELAY						PSM	PD2	PD1	PD0
	Write																
	Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# ADC

## ADC Power Control Register (ADPOWER) Continued

Bits	Name	Description
<b>2</b>	<b>PD2</b>	<b>Power-Down Voltage Reference 2</b>
		This bit can be used to power-down the voltage reference.
	0	ADC voltage reference is powered up
	1	Power-down voltage reference if all associated ADC converters are also powered down
<b>1</b>	<b>PD1</b>	<b>Power-Down Voltage Reference 1</b>
		This bit can be used to force ADC converter one to power-down.
	0	ADC converter one is powered up
	1	Power-down ADC converter one
<b>0</b>	<b>PD0</b>	<b>Power-Down Voltage Reference 0</b>
		This bit can be used to force ADC converter zero to power-down.
	0	ADC converter zero is powered up
	1	Power-down ADC converter zero

ADC Power Control Register (ADPOWER) BASE+\$29	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	PSTS2	PSTS1	PSTS0	PUDELAY						PSM	PD2	PD1	PD0
	Write																
	Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# ADC

## ADC Calibration Register (ADC-CAL)

Bits	Name	Description
3	CRS1	<b>Calibration Reference Select ADC 1</b>
		This bit selects which reference to select during ADC calibration.
		0 $V_{REFLO}$ is selected as the calibrate input to ADC1
		1 $V_{REFH}$ is selected as the calibrate input to ADC 1
2	CAL1	<b>Calibrate ADC 1</b>
		When this bit is set, ADC 1 enters Calibrate mode and $V_{REFLO}$ or $V_{REFH}$ is routed to the ADC input as selected by the CRS1 bit.
		0 ADC 1 Normal operation
		1 ADC 1 is placed in Calibrate mode
1	CRS0	<b>Calibration Reference Select ADC 0</b>
		This bit selects which reference to select during ADC calibration.
		0 $V_{REFH}$ is selected as the calibrate input to ADC 0
		1 $V_{REFLO}$ is selected as the calibrate input to ADC 0
0	CAL0	<b>Calibrate ADC 0</b>
		When this bit is set, ADC0 enters Calibrate mode and $V_{REFLO}$ or $V_{REFH}$ is routed to the ADC input as selected by the CRS0 bit.
		0 ADC 0 Normal operation
		1 ADC 0 is placed in Calibrate mode

ADC Calibration Register (ADC-CAL) BASE+\$2A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	CRS1	CAL1	CRS0	CAL0
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits



Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# COP

## COP Control Register (COPCTL)

Bits	Name	Description
4	<b>BYPS</b>	<b>Bypass OSCCLK</b>
		Setting this bit allows testing of the COP to be increased by routing the IPBus clock to the counter instead of the OSCCLK. This bit should not be set during normal operation of the chip because it will cause the COP Timeout to occur sooner. If this bit is utilized, it should only be changed while CEN = 0. This bit can only be changed when CWP is set to zero.
		0 Counter uses OSCCLK (default)
		1 Counter uses IPBus clock
3	<b>CSEN</b>	<b>COP Stop Mode Enable</b>
		This bit controls the operation of the COP counter in Stop mode. This bit can only be changed when CWP is set to zero.
		0 COP counter will stop in Stop mode (default)
		1 COP counter will run in Stop mode when CEN is set to one
2	<b>CWEN</b>	<b>COP Wait Mode Enable</b>
		This bit controls the operation of the COP counter in Wait mode. This bit can only be changed when CWP is set to zero.
		0 COP counter will stop in Wait mode (default)
		1 COP counter will run in Wait mode if CEN is set to one.
1	<b>CEN</b>	<b>COP Enable</b>
		This bit controls the operation of the Counter (COPCTR). It can only be changed when CWP is set to zero. This bit always reads as zero when the chip is in Debug mode.
		0 COP counter is disabled (default)
		1 COP counter is enabled
0	<b>CWP</b>	<b>COP Write Protect</b>
		This bit controls the write protection feature of both the COPCTL and Timeout (COPTO) registers. Once set, this bit can only be cleared by resetting the module.
		0 COPCTL and COPTO can be read and modified by writing (default)
		1 COPCTL and COPTO are <i>read-only</i>

COP Control Register (COPCTL) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	0	0	0	BYPS	CSEN	CWEN	CEN	CWP
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# COP

## COP Timeout Register (COPTO)

Bits	Name	Description
15-0	TIMEOUT	<b>Timeout</b>
<p>The value in this register determines the timeout period of the COP counter. TIMEOUT should be written before enabling COP. Once COP is enabled, the recommended procedure for changing TIMEOUT is to disable the COP, write to COPTO, then re-enable the COP. This procedure ensures the new TIMEOUT is loaded into the counter. Alternatively, the 16-bit controller can write to COPTO prior to writing the proper patterns to COPCTR, thereby causing the counter to reload with the new TIMEOUT value. The COPCTR is not reset by a write to COPTO. Changing TIMEOUT while the COP is enabled results in a timeout period differing from the expected value. These bits can be changed only when CWP is set to zero.</p>		

<b>COP Timeout Register (COPTO) BASE + \$1</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TIMEOUT															
	Write	TIMEOUT															
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# COP

## COP Counter Register (COPCTR)

Bits	Name	Description
15-0	<b>COUNT</b>	<b>Count</b>
		This is the current value of the COP counter as it counts down from the timeout value to zero. A reset is issued when this count reaches zero.
15-0	<b>SERVICE</b>	<b>Service</b>
		When enabled, the COP requires a service sequence be performed periodically in order to clear its counter and prevent a reset from being issued. This routine consists of writing \$5555 to the COPCTR followed by writing \$AAAA before the timeout period expires. The writes to COPCTR must be performed in the correct order, but any number of other instructions (and writes to other registers) may be executed between the two writes.

<b>COP Counter Register (COPCTR) BASE+\$2</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COUNT															
	Write	COPSERV															
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 1 of 4

# EMI

## Chip Select Base Address Registers 0-7 (CS0-7)

Bits	Name	Description
15-4	ADR23-12	<b>Chip Select Base Addresses</b>
		Active address range of a given Chip Select (CS <sub>n</sub> .)
3-0	BLKSZ	<b>Block Size</b>
		This field determines the size of the memory map covered by the CS <sub>n</sub> . This field also determines which of the address bits to use when specifying the base address of the CS <sub>n</sub> .

Chip Select Base Address Registers 0-7 (CS0-7) BASE+\$0 to \$7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ADR23	ADR22	ADR21	ADR20	ADR19	ADR18	ADR17	ADR16	ADR15	ADR14	ADR13	ADR12	BLKSZ			
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 2 of 4

# EMI

## Chip Select Option Registers 0-7 (CSOR0-7)

Bits	Name	Description
15-11	RWS	<b>Read Wait States</b>
		This field specifies the number of additional system clocks, 0-30 (31 is invalid) to delay for read access to the selected memory mapped device.
10-9	BYTE_EN	<b>Upper/Lower Byte Option</b>
		This field specifies whether the memory mapped device is 16 bits wide or one byte wide. If the device is a byte wide, the option of upper or lower byte of a 16-bit word is selectable.
8-7	R/W	<b>Read/Write Enable</b>
		This field determines the read/write capabilities of the associated memory mapped devices.
6-5	PS/DS	<b>Program/Data Space Select</b>
		This field determines the mapping of a chip select to program and/or data space.
4-0	WWS	<b>Write Wait States</b>
		This field specifies the number of additional system clocks 0-30 (31 is invalid) to delay for write access to the selected memory mapped device.

Chip Select Option Registers 0-7 (CSOR0-7) BASE+\$8 to \$F	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	RWS				BYTE_EN		R/W		PS/DS		WWS					
	Write	RWS				BYTE_EN		R/W		PS/DS		WWS					
	Reset	1	0	1	1	1	0	0	0	0	0	0	0	1	0	1	1

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

Sheet 3 of 4

# EMI

## Chip Select Timing Control Registers 0-7 (CSTC0-7)

Bits	Name	Description
15-14	<b>WWSS</b>	<b>Write Wait States Setup Delay</b>
		This field affects the write cycle timing diagram. It specifies the number of additional system clocks to provide between the assertion of $\overline{CSn}$ and address lines and the assertion of $\overline{WR}$ .
13-12	<b>WWSH</b>	<b>Write Wait States Hold Delay</b>
		This field affects the write cycle timing diagram. It specifies the number of additional system clocks to hold the address, data, and $\overline{CSn}$ signals after the $\overline{WR}$ signal is deasserted.
11-10	<b>RWSS</b>	<b>Read Wait States Setup Delay</b>
		This field affects the read cycle timing diagram. It specifies the number of additional system clocks to provide between the assertion of $\overline{CSn}$ and address lines and the assertion of $\overline{RD}$ .
9-8	<b>RWSH</b>	<b>Read Wait States Hold Delay</b>
		This field affects the read cycle timing diagram. It specifies the number of additional system clocks to hold the address, data, and $\overline{CSn}$ signals after the $\overline{RD}$ signal is deasserted.
2-0	<b>MDAR</b>	<b>Minimal Delay After Read</b>
		This field specifies the number of system clocks to delay between reading from a memory mapped device in a $\overline{CSn}$ controlled space and reading from another device. A read followed by write also triggers this delay.

Chip Select Timing Control Registers 0-7 (CSTC0-7) BASE+\$10 to \$17	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	WWSS		WWSH		RWSS		RWSH		0	0	0	0	0	MDAR		
	Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Reserved Bits

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# EMI

## Bus Control Register (BCR)

Bits	Name	Description
15	DRV	<b>Drive</b>
		This bit is used to specify what occurs on the external memory port pins when no external access is performed (whether the pins remain driven or are placed in tri-state).
14-12	BMDAR	<b>Base Minimal Delay After Read</b>
		This bit field specifies the number of system clocks to delay after reading from memory mapped device not in $\overline{CS}$ controlled space and reading from one in a $\overline{CSn}$ controlled space. A read followed by write also triggers this delay.
9-5	BWWS	<b>Base Write Wait States</b>
		This field specifies the number of additional system clocks 0-30 (31 is invalid) to delay for write access to the selected memory mapped device when the address does not fall within $\overline{CS}$ controlled range.
4-0	BRWS	<b>Base Read Write States</b>
		This field specifies the number of additional system clocks 0-30 (31 is invalid) to delay for read access to the selected memory mapped device when the address does not fall within $\overline{CS}$ controlled range.

Bus Control Register (BCR) BASE+\$18	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	DRV	BMDAR				0	0	BWWS				BRWS				
	Write																
	Reset	0	0	0	0	0	0	1	0	0	1	0	1	1	1	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# OCCS

## PLL Control Register w/o Relaxation Oscillator (PLLCR)

Please see the following page for continuation of this register.

Bits	Name	Description
15-14	PLLIE1	<b>PLL Interrupt Enable 1</b>
		An optional interrupt can be generated when the Fine PLL Lock (LCK1) status bit in the PLLSR changes:
	00	Disable interrupt
	01	Enable interrupt on any rising edge of LCK1
	10	Enable interrupt on falling edge of LCK1
	11	Enable interrupt on any edge change of LCK1
13-12	PLLIE0	<b>PLL Interrupt Enable 0</b>
		An optional interrupt can be generated if the Coarse PLL Lock (LCK0) status bit in the PLLSR changes:
	00	Disable interrupt
	01	Enable interrupt on any rising edge of LCK0
	10	Enable interrupt on falling edge of LCK0
	11	Enable interrupt on any edge change of LCK0
11	LOCIE	<b>Loss of Reference Clock Interrupt Enable</b>
		Loss of the reference clock circuit monitors the output of the selected clock source. In the event of reference clock loss, an optional interrupt can be generated.
	0	Interrupt disabled
	1	Interrupt enabled
7	LCKON	<b>Lock Detector On</b>
	0	Lock detector disabled
	1	Lock detector enabled

PLL Control Register w/o Relaxation (PLLCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	CHPMPTRI	0	PLLPD	0	0	ZSRC	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Reserved Bits



Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# OCCS

## PLL Control Register w/o Relaxation Oscillator (PLLCR) Cont.

Bits	Name	Description
6	CHPMPTRI	<b>Charge Pump Tri-State</b>
		During normal chip operation, the CHPMPTRI bit should be set to a zero value. In the event of loss of reference clock, the CHPMPTRI bit must be set to value of one.
	0	Normal operation
	1	Isolates the charge pump from the loop filter allowing the PLL output to slowly drift, thereby providing enough time to shutdown the chip. Activating this bit will render the PLL inoperable and should not be executed during standard operation of the chip.
4	PLLPD	<b>PLL Power-Down</b>
		The PLL can be turned off by setting the PLLPD bit. A four IPBus clock delay is created from changing the bit to signaling the PLL. When the PLL is powered down, the gear shifting logic automatically switches to ZSRC=1, preventing loss of reference clock to the core.
	0	PLL is enabled
	1	PLL powered down
1-0	ZSRC	<b>Clock Source</b>
		The clock source determines the SYS_CLK_X2 source to the SIM module. In turn it generates divided down versions of this signal for use by memories and IPBus. ZSRC is automatically set to one during Stop mode, or when the PLLPD is set, preventing loss of the reference clock to the core.
	00	Reserved
	01	Prescaler output
	10	Postscaler output
	11	Reserved

PLL Control Register w/o Relaxation (PLLCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	CHPMPTRI	0	PLLPD	0	0	ZSRC	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# OCCS

## PLL Control Register w/ Relaxation Oscillator (PLLCR)

Please see the following page for continuation of this register.

Bits	Name	Description
15-14	PLLIE1	<b>PLL Interrupt Enable 1</b>
		An optional interrupt can be generated when the Fine PLL Lock (LCK1) status bit in the PLLSR changes:
	00	Disable interrupt
	01	Enable interrupt on any rising edge of LCK1
	10	Enable interrupt on falling edge of LCK1
	11	Enable interrupt on any edge change of LCK1
13-12	PLLIE0	<b>PLL Interrupt Enable 0</b>
		An optional interrupt can be generated if the Coarse PLL Lock (LCK0) status bit in the PLLSR changes:
	00	Disable interrupt
	01	Enable interrupt on any rising edge of LCK0
	10	Enable interrupt on falling edge of LCK0
	11	Enable interrupt on any edge change of LCK0
11	LOCIE	<b>Loss of Reference Clock Interrupt Enable</b>
		Loss of the reference clock circuit monitors the output of the selected clock source. In the event of reference clock loss, an optional interrupt can be generated.
	0	Interrupt disabled
	1	Interrupt enabled
7	LCKON	<b>Lock Detector On</b>
	0	Lock detector disabled
	1	Lock detector enabled

PLL Control Register w/Relaxation (PLLCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	CHPMPTRI	0	PLLPD	0	PRECS	ZSRC		
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# OCCS

## PLL Control Register w/ Relaxation Oscillator (PLLCR) Cont.

Bits	Name	Description
6	CHPMPTRI	<b>Charge Pump Tri-State</b>
		During normal chip operation, the CHPMPTRI bit should be set to a zero value. In the event of loss of reference clock, the CHPMPTRI bit must be set to value of one.
		0 Normal operation
		1 Isolates the charge pump from the loop filter allowing the PLL output to slowly drift, thereby providing enough time to shutdown the chip. Activating this bit will render the PLL inoperable and should not be executed during standard operation of the chip.
4	PLLPD	<b>PLL Power-Down</b>
		The PLL can be turned off by setting the PLLPD bit. A four IPBus clock delay is created from changing the bit to signaling the PLL. When the PLL is powered down, the gear shifting logic automatically switches to ZSRC=1, preventing loss of reference clock to the core.
		0 PLL is enabled
		1 PLL powered down
2	PRECS	<b>Prescaler Clock Select</b>
		This bit is reserved with a zero value. It is set only if the crystal is enabled in the GPIO. The clock source to the Prescaler can be selected to be either the Internal Relaxation Oscillator or Crystal Oscillator.
		0 Relaxation Oscillator selected (reset value)
		1 Crystal Oscillator selected
1-0	ZSRC	<b>Clock Source</b>
		The clock source determines the SYS_CLK_X2 source to the SIM module. In turn it generates divided down versions of this signal for use by memories and IPBus. ZSRC is automatically set to one during Stop mode, or when the PLLPD is set, preventing loss of the reference clock to the core.
		00 Reserved
		01 Prescaler output
		10 Postscaler output
		11 Reserved

PLL Control Register w/Relaxation (PLLCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PLLIE1		PLLIE0		LOCIE	0	0	0	LCKON	CHPMPTRI	0	PLLPD	0	PRECS	ZSRC	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# OCCS

## PLL Divide-By Register (PLLDB)

Bits	Name	Description
15-12	LORTP	<b>Loss of Reference Timer Period</b>
		This bit field controls the amount of time required for the loss of reference clock interrupt to be generated. This failure detection time is $LORTP \times 10 \times \text{PLL-clock-time-period}$ , where $\text{PLL-clock-time-period} = 2/F_{OUT}$ .
11-10	PLLCOD	<b>PLL Clock Out Divide or Postscaler</b>
		The PLL output clock can be divided down by a 2-bit postscaler. The output of the postscaler is a selectable clock source for the hybrid controller core as determined by the ZSRC bits in the PLLCR.
		00 Divide-by one
		01 Divide-by two
		10 Divide-by four
		11 Divide-by eight
9-8	PLLCID	<b>PLL Clock in Divide or Prescaler</b>
		The PLL input clock (EXTAL_CLK) can be divided down by a 2-bit prescaler.
		00 Divide-by one
		01 Divide-by two
		10 Divide-by four
		11 Divide-by eight
6-0	PLLDB	<b>PLL Divide-By</b>
		In part, the output frequency of the PLL is controlled by the PLLDB register. The plus one value written to this register is used by the PLL to directly multiply the input frequency and present it at its output. For example, if the input frequency is 8MHz and the PLLDB register is set to default 29, the PLL output frequency is 240MHz ( $= F_{OUT} = ((PLLDB + 1) \times 8\text{MHz})$ ). This yields a 120MHz SYS_CLK_2X ( $F_{OUT}/2$ ). Dividing this formula by two (hard coded into SIM) results in a 60MHz system clock assuming both prescaler and postscaler are set to one.  Before changing the divide-by value, the core clock must first be switched to the prescaler clock.  <b>Note:</b> For the 8100 family of devices, the PLLDB should be changed from the default such that the chip operates at 40MHz instead of 60MHz. For example, assuming an 8MHz input clock; PLLDB should be set to 19. This yields a PLL frequency of 160MHz with an 80MHz SYS_CLK_2X ( $F_{OUT}/2$ ) and a SYS_CLK frequency of 40MHz.

PLL Divide-By Register (PLLDB) BASE+\$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LORTP				PLLCOD		PLLCID		0	PLLDB						
	Write	LORTP				PLLCOD		PLLCID			PLLDB						
	Reset	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# OCCS

## PLL Status Register (PLLSR)

Bits	Name	Description
15	LOLI1	<b>PLL Loss of Lock Interrupt 1</b>
		LOLI1 displays the status of the lock detector state from LCK1 circuit. The interrupt is cleared by writing a one to LOLI1.
		0 PLL locked
		1 PLL unlocked
14	LOLI0	<b>PLL Loss of Lock Interrupt</b>
		LOLI0 shows the status of the lock detector state from LCK0 circuit. The interrupt is cleared by writing a one to LOLI0.
		0 PLL locked
		1 PLL unlocked
13	LOCI	<b>Loss of Reference Clock Interrupt</b>
		LOCI shows the status of the reference clock detection circuit.
		0 Oscillator clock normal
		1 Lost oscillator clock
6	LCK1	<b>PLL Fine Lock</b>
		0 PLL is unlocked
		1 PLL is locked (fine)
5	LCK0	<b>PLL Coarse Lock</b>
		<b>Either LCK0 or LCK1 may be set first to indicate a loss of lock. There is no guaranteed sequence.</b>
		0 PLL is unlocked
		1 PLL is locked (course)
4	PLLPDS	<b>PLL Power-Down Status</b>
		PLL power-down status is delayed by four IPBus clocks from the PLLPD bit in the PLLCR.
		0 PLL not powered-down
		1 PLL powered down
1-0	ZSRCS	<b>Clock Source Status</b>
		ZSRC indicates the current SYS_CLK_X2 clock source. Since the synchronizing circuit switches the system clock source, ZSRC takes more than one IPBus clock to indicate the new selection.
		00 Synchronizing in progress
		01 Prescaler output
		10 Postscaler output
		11 Synchronizing in progress

PLL Status Register (PLLSR) BASE+\$2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LOLI1	LOLI0	LOCI	0	0	0	0	0	0	LCK1	LCK0	PLLPDS	0	0	ZSRCS	
	Write																
	Reset	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# OCCS

## PLL Shutdown Register (SHUTDOWN)

Bits	Name	Description
15-0	SHUTDOWN	<b>Shutdown</b>
		<p>This register can be used to shutdown all system clocks. SYS_CLK_X2, SYS_CLK, and SYS_CLK_DIV2 will be left in a low state.</p> <p><b>Note:</b> This is a terminal condition. The only recovery is to assert the RESET.</p> <p>Shutdown can be asserted only under the following conclusions: The Loss of Reference (LOR) interrupt is asserted the CHPMPTRI bit in the PLLCR has been set. The value OXDEAD is written to this register.</p>

Shutdown Register (SHUTDOWN) BASE+\$4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Write	SHUTDOWN																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 8 of 9

# OCCS

## PLL Oscillator Control Register w/o Relaxation (OSCTL)

Bits	Name	Description
13	COHL	<b>Crystal Oscillator High/Low Power Level</b>
		This bit controls the power usage of the crystal oscillator. It is reset to the high power state, allowing use of either a crystal or resonator.
	0	High Power mode is required when a resonator is used
	1	Low Power mode is the desired mode when a crystal is used.

PLL Oscillator Control Register w/o Relaxation (OSCTL) BASE+\$5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	COHL	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# OCCS

## PLL Oscillator Control Register w/ Relaxation (OSCTL)

Bits	Name	Description
15	ROPD	<b>Relaxation Oscillator Power-Down</b>
		This bit is used to power-down the relaxation oscillator if the crystal oscillator is being used. It can prevent loss of clock to the core or the PLL only if the prescaler clock source has been changed to the crystal oscillator by setting the PRECS bit in PLLCR.
	0	Relaxation oscillator enabled
	1	Relaxation oscillator powered down
13	COHL	<b>Crystal Oscillator High/Low Power Level</b>
		This bit is used to control the power of the crystal oscillator. It is reset to the high power state, allowing either a crystal or resonator to be used.
	0	High Power mode is required when a resonator is used
	1	Low Power mode is desired when a crystal is used
12	CLKMODE	<b>Crystal Oscillator Clock Mode</b>
		This bit is used to control the crystal/resonator clock selection.
	0	Crystal oscillator enabled
	1	Direct clock mode. Setting this bit shuts down the crystal oscillator, allowing an external clock source on the XTAL pin to drive the clock input to the chip directly.
7-0	TRIM	<b>Internal Relaxation Oscillator</b>
		This bit field changes the size of the internal capacitor used by the internal relaxation oscillator. By testing the frequency of the internal clock and incrementing or decrementing this factor accordingly, the accuracy of the internal clock can be improved by two percent. Incrementing these bits by one decreases the frequency by 0.23 percent of the unadjusted value. Decrementing this register by one increases the frequency by 0.23 percent. Reset sets these bits to \$80, centering the range of possible adjustment.

PLL Oscillator Control Register w/ Relaxation (OSCTL) BASE+\$5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ROPD	0	COHL	CLKMODE	0	0	TRIM									
	Write																
	Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

Reserved Bits



Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# FLASH

## FLASH Clock Divider Register (FMCLKD)

Bits	Name	Description
7	DIVLD	<b>Clock Divider Loaded</b>
		Writing to this bit has no effect.
		0 Register has not been written
		1 Register has been written to since the last reset.
6	PRDIV8	<b>Enable Prescaler Divide-By Eight</b>
		0 The input clock is directly fed into the FMCLKD Divider
		1 Enables a prescaler x 8 to divide the FM input clock before feeding it into the FMCLKD Divider
5-0	DIV	<b>Clock Divider</b>
		The combination of PRDIV8 and DIV effectively divides the FM input clock down to a frequency of 150kHz–200kHz. The input clock frequency range is 150kHz < input clock < 102.4MHz.

FLASH Clock Divider Register (FMCLKD) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	D6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	DIVLD	PRDIV8	DIV					
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# FLASH

## FLASH Configuration Register (FMCR)

Bits	Name	Description
10	LOCK	<b>Write Lock Control</b>
		This bit can always be read. It may be set only once and is cleared at chip reset.
		0 The FMPROT and FMPROTB bits can receive writing.
		1 The FMPROT and FMPROTB bits are write-locked.
8	AEIE	<b>Access Error Interrupt Enable</b>
		This read/write bit enables an interrupt in case of an ACCERR flag being set.
		0 ACCERR interrupts disabled
		1 An interrupt will be requested whenever the ACCERR flag is set.
7	CBEIE	<b>Command Buffer Empty Interrupt Enable</b>
		This read/write bit enables an interrupt in case of an empty command buffer in the Flash.
		0 Command Buffer Empty interrupts disabled
		1 An interrupt will be requested whenever the CBEIF flag is set.
6	CCIE	<b>Command Complete Interrupt Enable</b>
		This read/write bit enables an interrupt in case of all commands being completed in the Flash.
		0 Command Complete interrupts disabled
		1 An interrupt will be requested whenever the CCIF flag is set.
5	KEYACC	<b>Enable Security Key Writing</b>
		This bit can be read; however, it can only receive writing if the KEYEN bit in the FMSECH register is set.
		0 Flash writes are interpreted as the start of a program or erase sequence.
		1 Writes to Flash array are interpreted as keys to open the back door
1-0	BKSEL	<b>Bank Select</b>
		This read/write bit field selects which register bank is addressed.
		00 Program or Boot
		01 Data
		10 Program or Boot
		11 Data

FLASH Configuration Register (FMCR) BASE+\$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	LOCK	0	AEIE	CBEIE	CCIE	KEYACC	0	0	0	BKSEL	
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# FLASH

FLASH Security High Register (FMSECH)

FLASH Security Low Register (FMSECL)

Bits	Name	Description
15	KEYEN	Enable Back Door Key to Security
	0	Back door to Flash is disabled
	1	Back door to Flash is enabled
14	SECSTAT	Security Status
	0	Flash Security is disabled
	1	Flash Security is enabled

FLASH Security High Register (FMSECH) BASE+\$3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	KEYEN	SECSTAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Write																	
	Reset	F <sup>1</sup>	F <sup>2</sup>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset state loaded from Configuration Field (SECH\_VALUE) during reset.
2. Reset state determined by value in FMSECL. If FMSECL=0xE70A, the bit is set and the chip is secured.

Bits	Name	Description
15-0	SEC	Security
		This bit field defines the security state of the device. 0xE07A value was selected because it represents an illegal instruction on the 56800E core, making it unlikely user compiled code accidentally programmed in the SECL_VALUE in the FM Configuration field location would unintentionally secure the Flash.
	0xE70A	Flash secured
	All Other Combinations	Flash unsecured

FLASH Security Low Register (FMSECL) BASE+\$4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	SEC																
	Write																	
	Reset	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>

1. Reset state loaded from Configuration Field (SECL\_VALUE) during reset.

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# FLASH

FLASH Optional Data Register 0 (FMOPT0)

FLASH Optional Data Register 1 (FMOPT1)

FLASH Optional Data Register 2 (FMOPT2)

Bits	Name	Description
11-0	FMOPT0	<b>TSENSOR Hot Temp</b>
		This bit field contains the ADC voltage conversion value for the Temperature Sensor at Hot Temperature (150°C). This value is referenced as VHOT in the Temperature Sensor section.

FLASH Optional Data Register (FMOPT0) BASE+\$1A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	TSENSOR HOT TEMP												
	Write																	
	Reset	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>

- Reset state loaded from Flash information block at reset.

Bits	Name	Description
9-0	FMOPT1	<b>Relaxation Trim 8MHz</b>
		This bit field contains the value required to trim the internal relaxation oscillator to 8MHz as measured at the factory. Refer to the chip specific data sheet to determine if the chip in question has an internal relaxation oscillator.

FLASH Optional Data Register (FMOPT0) BASE+\$1B	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	RELAXATION TRIM 8MHz										
	Write																	
	Reset	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>

- Reset state loaded from Flash information block at reset.

Bits	Name	Description
11-0	FMOPT2	<b>TSENSOR Room Temp</b>
		This bit field contains the ADC voltage conversion value for the Temperature Sensor at Room Temperature (25°C). This value is referenced as VR in the Temperature Sensor section.

FLASH Optional Data Register (FMOPT0) BASE+\$1C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	TSENSOR ROOM TEMP												
	Write																	
	Reset	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>

- Reset state loaded from Flash information block at reset.

Reserved Bits

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# FLASH

## FLASH Protection Register (FMPROT)

Bits	Name	Description
15-0	PROTECT	<b>Protection</b>
		Each Program or Data Flash sector can be protected from program and erase by setting PROTECT[M] bit.
	0	PROTECT[M] Array sector M is not protected
	1	PROTECT[M] Array sector M is protected

FLASH Protection Register (FMPROT) BASE+\$10	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PROTECT															
	Write	PROTECT															
	Reset	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>

1. Reset state loaded from Flash Configuration Field during reset.

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 6 of 8

# FLASH

## FLASH Protection Boot Register (FMPROTB)

Bits	Name	Description
3-0	PROTECT	<b>Protection Boot</b>
		Each Boot Flash sector can be protected from program and erase by setting PROTECT[M] bit.
	0	PROTECT[M] Array sector M is not protected
	1	PROTECT[M] Array sector M is protected

FLASH Protection Boot Register (FMPROTB) BASE+\$11	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	0	0	0	PROTECT			
	Write																
	Reset	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>	F <sup>1</sup>

- Reset state loaded from Flash Configuration Field during reset.

 Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# FLASH

## FLASH User Status Register (FMUSTAT)

Bits	Name	Description
7	<b>CBEIF</b>	<b>Command Buffer Empty Interrupt Flag</b>
		This bit indicates the address, data, and command buffers are empty so a new command sequence can be started. The CBEIF flag is cleared by writing 1 to it. Writing 0 has no effect on CBEIF; however, writing 0 to the CBEIF bit can be used to abort a command sequence.
	0	Buffers are full
	1	Buffers are ready to accept a new command
6	<b>CCIF</b>	<b>Command Complete Interrupt Flag</b>
		This bit indicates there are no pending commands. The CCIF flag is set and cleared automatically upon start and completion of a command. Writing to the CCIF bit has no effect. The CCIF bit can generate an interrupt if the CCIE bit in the Configuration register is set.
	0	Command in progress
	1	All commands are completed
5	<b>PVIOL</b>	<b>Protection Violation</b>
		This bit indicates an attempt was made to program, or erase an address in a protected Flash memory area. Clear the PVIOL flag by writing 1 to the bit. Writing 0 to the bit has no effect on PVIOL. While the PVIOL flag is set in any banked register, it is not possible to launch another command.
	0	No failure
	1	A protection violation has occurred
4	<b>ACCERR</b>	<b>Access Error</b>
		This bit indicates an illegal access to the Flash Memory array, or registers caused by a bad program, or an erase sequence. Clear the ACCERR flag by writing 1 to the bit. Writing 0 to the ACCERR bit has no effect. While the ACCERR flag is set in any banked register, it is not possible to launch another command. Please see <a href="#">Section 6.5.3.4</a> for ACCERR flag setting details.
	0	No failure
	1	Access error has occurred
2	<b>BLANK</b>	<b>Block Verified Erased</b>
		This bit indicates an Erase Verify command (RDARY1) has checked the Flash block and found it to be blank. Clear the BLANK flag by writing 1 to the bit. Writing 0 to the bit has no effect.
	0	If an Erase Verify command is requested, the CCIF flag is set. A zero in BLANK indicates block is not erased
	1	Flash block verifies as erased

FLASH User Status Register (FMUSTAT) BASE+\$13	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# FLASH

## FLASH Command Buffer Register (FMCMD)

Bits	Name	Description
6-0	CMD	Command
Valid commands are listed here. Writing a command other than those listed will set the ACCERR flag in the User Status register.		
\$05	RDARY1	Erase Verify (All Ones)
\$20	PGM	Word Program
\$40	PGERS	Page Erase
\$41	MASERS	Mass Erase

FLASH Command Buffer Register (FMCMD) BASE+\$14	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	CMD						
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 Reserved Bits



Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# FLEX

## FlexCAN Module Configuration Register (FCMCR)

Please see the following page for continuation of this register.

Bits	Name	Description
15	<b>STOP</b>	<b>Stop</b>
		This bit is a Low Power Sleep mode. It may be set by the device. The bit can be cleared either by the device or by FlexCAN only if the SELF_WAKE bit in FCMCR is set.
		0   Enable FlexCAN clocks to run
		1   Shutdown FlexCAN clocks
14	<b>FRZ1</b>	<b>Freeze Enable</b>
		The FRZ bit specifies the FlexCAN module response to the HALT bit in FCMCR being asserted. This bit is initialized to 1 during reset. FRZ0 is not used in this FlexCAN module.
		0   Ignore FRZ1/HALT
		1   Refer to HALT
12	<b>HALT</b>	<b>Halt FlexCAN SClOCK</b>
		This bit is initialized to 1. When HALT and FRZ1 are set to 1 FlexCAN enters Debug mode.
		0   FlexCAN operates normally
		1   Enter Debug mode if FRZ1 = 1
11	<b>NOT_RDY</b>	<b>Not Ready</b>
		This <i>read-only</i> bit indicates FlexCAN is either in Stop or in Debug states.
		This bit is cleared once the FlexCAN module has exited Debug mode either by synchronization to the bus (11 recessive bits), or by a self-wake mechanism.
10	<b>WAKE_MASK</b>	<b>Wake-Up Interrupt Mask</b>
		This bit enables the Wake-Up interrupt generation.
		0   Wake-Up Interrupt is disabled
		1   Wake-Up Interrupt is enabled

FlexCAN Module Configuration Register (FCMCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	STOP	FRZ1	0	HALT	NOT_RDY	WAKE_MASK	SOFT_RST	FREEZ_ACK	0	SELF_WAKE	AUTO_PWR_SAVE	STOP_ACK	0	0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# FLEX

## FlexCAN Module Configuration Register (FCMCR) Continued

Bits	Name	Description
9	<b>SOFT_RST</b>	<b>Soft Reset</b>
		When this bit is asserted FlexCAN resets its Internal State Machines (Sequencer, Error Counters, Error Flags, Timer) and the interface registers (FCMCR, FCIMASK1, FCIFLAG1, FCIMAXMB).
8	<b>FREEZ_ACK</b>	<b>FlexCAN Disabled</b>
		The read-only bit provides status of when FlexCAN is in Debug mode, stopping its prescaler.
		0 FlexCAN exited Debug mode; the prescaler is enabled
		1 FlexCAN entered Debug mode; the prescaler is disabled
6	<b>SELF_WAKE</b>	<b>Self Wake-Up</b>
		This bit enables the Self Wake-Up of FlexCAN after Stop, without device intervention. If this bit is set when entering Stop, the FlexCAN module will be looking for a dominant bit on the bus during Stop. If a transition from Recessive-to-Dominant is detected, the FlexCAN module immediately clears the STOP bit, resuming the clocks.
		If a write to FCMCR with SELF_WAKE set occurs at the same time a Recessive-to-Dominant edge appears on the CAN bus, the bit will not be set and the module clocks will not stop. Verify this bit was set by reading FCMCR. DO NOT set this bit if eventually the LPStop command is executed.
5	<b>AUTO_PWR_SAVE</b>	<b>Auto Power Save</b>
		This bit enables FlexCAN to automatically shut-off its clocks, saving power when it has no process to execute. Those same clocks automatically resume when it has a task to execute, without any device intervention. IPBus clocks are not stopped, enabling device access. The Auto Power Save does not depend on the SELF_WAKE or WAKE_MASK bit values.
		0 Auto Power Save mode is not active; clocks are running normally
		1 Auto Power Save mode is active; clocks stop and resume as required
4	<b>STOP_ACK</b>	<b>FlexCAN Stopped</b>
		FlexCAN is in Stop mode with its main clocks stop. This is a read-only bit. This bit has a value of one when the FlexCAN module enters Stop mode and its clocks are stopped, but a value of zero when Stop mode is cleared and the FlexCAN module's clocks are running again.
		0 FlexCAN exited Stop mode
		1 FlexCAN entered Stop mode

FlexCAN Module Configuration Register (FCMCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	STOP	FRZ1	0	HALT	NOT_RDY	WAKE_MASK	SOFT_RST	FREEZ_ACK	0	SELF_WAKE	AUTO_PWR_SAVE	STOP_ACK	0	0	0	0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# FLEX

## FlexCAN Control Register 0 (FCCTL0)

Please see the following page for continuation of this register.

Bits	Name	Description
15	<b>BOFF_MASK</b>	<b>Busoff Mask</b>
		This bit provides a mask for the Busoff Interrupt.
		0   Interrupt disabled
		1   Interrupt enabled
14	<b>ERR_MASK</b>	<b>Error Mask</b>
		This bit provides a mask for the Error Interrupt.
		0   Interrupt disabled
		1   Interrupt enabled
7	<b>SAMP</b>	<b>Sampling Mode</b>
		The sample bit determines whether the FlexCAN module will sample each received bit once or three times to determine its value.
		0   (Reset Value) One sample is used to determine the value of received bit.
		1   Three samples are used to determine the value of received bit, the normal one (sample point) and two preceding periods of SClOCK, using a majority rule.
6	<b>LOOPB</b>	<b>Loop Back Mode</b>
		This bit can only be set with in Test mode, but only when the TEST_EN bit in FCMAXMB register is set.
		0   Loop Back mode is disabled
		1   Loop Back mode is enabled
5	<b>TSYNC</b>	<b>Timer Synchronization Mode</b>
		This bit enables a mechanism to reset or clear the Free-Running Timer each time a message is received in MB0.
		0   Timer Sync is disabled
		1   Timer Sync is enabled

FlexCAN Control Register (FCCTL0) BASE + \$3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	BOFF_MASK	ERR_MASK	0	0	0	0	0	0	0	SAMP	LOOPB	TSYNC	LBUF	LOM	PROPSEG		
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# FLEX

## FlexCAN Control Register 0 (FCCTL0) Continued

Bits	Name	Description
4	LBUF	<b>Lowest Buffer Transmitted First</b>
		This bit defines the transmit-first scheme.
	0	Lowest ID is transmitted first
	1	Lowest number buffer is transmitted first
3	LOM	<b>Listen-Only Mode</b>
		This control bit configures FlexCAN to be in Listen-Only mode thereby being able to receive messages without acknowledging or being active the bus for diagnosis.
	0	Regular operation Listen-Only mode is off
	1	Enable Listen-Only mode
2-0	PROPSEG	<b>Propagation Segment</b>
		This bit field defines the length of the Propagation Segment in the bit time with values from 0-7.

FlexCAN Control Register (FCCTL0) BASE + \$3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	BOFF_MASK	ERR_MASK	0	0	0	0	0	0	0	SAMP	LOOPB	TSYNC	LBUF	LOM	PROPSEG		
	Write	MASK	MASK															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# FLEX

## FlexCAN Control Register 1 (FCCTL1)

Bits	Name	Description
15-8	PRES_DIV	<b>Prescaler Divide Factor</b> This bit field determines the ratio between the system clock frequency and the Serial Clock (SClock).
7-6	RJW	<b>Re synchronization Jump Width</b> This bit field defines the maximum number of time quanta a bit may be changed by one re sync. Valid programmed values are 0-3.
5-3	PSEG1	<b>Phase Segment 1</b> This bit defines the length of Phase Buffer Segment 1 in the bit time. Valid programmed values are 0-7.
2-0	PSEG2	<b>Phase Segment 2</b> This bit defines the length of Phase Buffer Segment 2 in the bit time. Valid programmed values are 0-7.

FlexCAN Control Register 1 (FCCTL1) BASE+\$4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PRES_DIV								RJW		PSEG1			PSEG2		
	Write	PRES_DIV								RJW		PSEG1			PSEG2		
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 6 of 15

# FLEX

## FlexCAN Free-Running Timer Register (FCTIMER)

Bits	Name	Description
15-0	TMR	<b>Free-Running Timer</b>
		This bit field is read/write by the device. The timer begins from 0 after Reset, counting linearly to \$FFFF, then wraps around.

FlexCAN Free-Running Timer Register (FCTIMER) BASE+\$5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TMR															
	Write	TMR															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_



**FlexCAN Maximum Message Buffer Register (FCMAXMB)**

Bits	Name	Description
7	TEST_EN	<b>Test Enable</b>
		This <i>write-only</i> bit is set prior to setting the LOOPB bit in the FCCTL0 register.
3-0	MAXMB	<b>Maximum Message Buffer</b>
		This bit field defines the maximum MB configuration of the module. The reset value of these bits is \$F for a 16-bit configuration.

FlexCAN Maximum Message Buffer Register (FCMAXMB) BASE+\$6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	0	0	0	MAXMB				
	Write									TEST_EN								
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# FLEX

FlexCAN Receive Global Mask High Register (FCRXMASK\_H)

FlexCAN Receive Global Mask Low Register (FCRXMASK\_L)

Bits	Name	Description
31-21	MID28-MID18	<b>Mask Identification 28 to 18</b>
		These bits are the same mask bits for the Standard and Extended formats.
20	RESERVED	<b>See Special Note Below</b>
		This bit of a received frame is never compared to the corresponding bit in the MB ID field. Remote frames (RTR=1) are never received into MBs. RTR mask bits locations in the mask (bits 20 and 0) are always read as 0 regardless of any device write to these bits.
19	RESERVED	<b>See Special Note Below</b>
		The Identification Extended (IDE) bit of a Received Frame is always compared. Its location in the mask is always 1 regardless of any device write to this bit.
18-1	MID17-MID0S	<b>Mask Identification 17-1</b>
		These bits are used to mask comparison only in Extended format.

FlexCAN Receive Global Mask Low Register BASE+\$8 (FCRXGMASK_H)	Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Read												0	1			
	Write	MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18			MID17	MID16	MID15
	Reset	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

FlexCAN Receive Global Mask High Register BASE+\$9 (FCRXGMASK_L)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read																0
	Write	MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Reserved Bits



Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# FLEX

Receive Buffer 14 Mask High Register (FCRX14MASK\_H)

Receive Buffer 14 Mask Low Register (FCRX14MASK\_L)

Bits	Name	Description
31-21	MID28-MID18	<b>Mask Identification 28 to 18</b>
		These bits are the same mask bits for the Standard and Extended formats.
20	RESERVED	<b>See Special Note Below</b>
		This bit of a received frame is never compared to the corresponding bit in the MB ID field. Remote frames (RTR=1) are never received into MBs. RTR mask bits locations in the mask (bits 20 and 0) are always read as 0 regardless of any device write to these bits.
19	RESERVED	<b>See Special Note Below</b>
		The Identification Extended (IDE) bit of a Received Frame is always compared. Its location in the mask is always 1 regardless of any device write to this bit.
18-1	MID17-MID0S	<b>Mask Identification 17-1</b>
		These bits are used to mask comparison only in Extended format.

FlexCAN Receive Buffer 14 Mask High Register BASE+\$0A (FCRXG14MASK_H)	Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Read	MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
	Write																
	Reset	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

FlexCAN Receive Buffer 14 Mask Low Register BASE+\$0B (FCRXG14MASK_L)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
	Write																
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# FLEX

## Receive Buffer 15 Mask High Register (FCRX15MASK\_H)

## Receive Buffer 15 Mask Low Register (FCRX15MASK\_L)

Bits	Name	Description
31-21	MID28-MID18	<b>Mask Identification 28 to 18</b>
		These bits are the same mask bits for the Standard and Extended formats.
20	RESERVED	<b>See Special Note Below</b>
		This bit of a received frame is never compared to the corresponding bit in the MB ID field. Remote frames (RTR=1) are never received into MBs. RTR mask bits locations in the mask (bits 20 and 0) are always read as 0 regardless of any device write to these bits.
19	RESERVED	<b>See Special Note Below</b>
		The Identification Extended (IDE) bit of a Received Frame is always compared. Its location in the mask is always 1 regardless of any device write to this bit.
18-1	MID17-MID0S	<b>Mask Identification 17-1</b>
		These bits are used to mask comparison only in Extended format.

FlexCAN Receive Buffer 15 Mask High Register BASE+\$0C (FCRXG15MASK_H)	Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	Read												0	1				
	Write	MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18			MID17	MID16	MID15	
	Reset	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1

FlexCAN Receive Buffer 15 Mask Low Register BASE+\$0D (FCRXG15MASK_L)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read																0
	Write	MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# FLEX

## FlexCAN Error and Status Register (FCSTATUS)

Please see the following page for continuation of this register.

Bits	Name	Description
15	BIT1_ERR	<b>Bit 1 Error</b>
		This bit is not set by a transmitter in case of arbitration field or Acknowledge (ACK) slot, or in case of a node sending a passive edge flag detects dominate bits.
		0 No errors seen
		1 At least one bit sent as recessive is received as dominant
14	BIT0_ERR	<b>Bit 0 Error</b>
		0 No errors seen
		1 At least one bit sent as dominant is received as recessive
13	ACK_ERR	<b>Acknowledge Error</b>
		0 No occurrence
		1 An ACK error occurred since the last read of this register
12	CRC_ERR	<b>Cyclic Redundancy Code Error</b>
		0 No occurrence
		1 A CRC error occurred since the last read for this register
11	FORM_ERR	<b>Form Error</b>
		0 No occurrence
		1 A FORM error occurred since the last read for this register
10	STUFF_ERR	<b>Stuff Error</b>
		0 No occurrence
		1 A STUFF error occurred since the last read for this register
9	TX_WARN	<b>Transmit Warning</b>
		0 CANTX_ERR_Counter < 96
		1 CANTX_ERR_Counter >= 96
8	RX_WARN	<b>Receive Warning</b>
		0 CANRX_ERR_Counter < 96
		1 CANRX_ERR_Counter >= 96

FlexCAN Error and Status Register (FCSTATUS) BASE+\$10	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	BIT1_ERR	BIT0_ERR	ACK_ERR	CRC_ERR	FORM_ERR	STUFF_ERR	TX_WARN	RX_WARN	IDLE	TX/RX	FCS			0	BOFF_INT	ERR_INT	WAKE_INT
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# FLEX

## FlexCAN Error and Status Register (FCSTATUS) Continued

Bits	Name	Description
7	IDLE	<b>Idle</b>
		0 The CAN bus is not idle. See CANTX/CANRX bit
		1 The CAN bus is idle
6	TX/RX	<b>Transmission Receive</b>
		0 This FlexCAN is receiving a message if IDLE = 0
		1 This FlexCAN is transmitting a message if IDLE = 0
5-4	FCS	<b>Fault Confinement State</b>
		This two-bit field describes the state of the device.
		00 Error Active
		01 Error PAssive
		1X Busoff
2	BOFF_INT	<b>Busoff Interrupt</b>
		This bit is set each time the FlexCan module state changes to Busoff. If the FCCTL0 BOFF_MASK bit is set, an interrupt is generated to the core. This interrupt is not generated after the reset. This bit must be cleared to zero. To clear this bit, read the Error and Status register, then write 1 to the bit. Writing 0 has no effect.
1	ERR_INT	<b>Error Interrupt</b>
		This bit is set anytime one of the error bits in this register is set. This is true even if an error bit is already set. If the FCCTL0 ERR_MASK bit is set, an interrupt is generated to the core. This bit must be cleared to zero. To clear this bit, read the Error and Status register, then write 1 to the bit. Writing 0 has no effect.
0	WAKE_INT	<b>Wake Interrupt</b>
		This bit is set when the FlexCAN module is in Stop mode and a Recessive-to-Dominant transition is detected on the CAN bus. If the FCMCR WAKE_MASK bit is set, an interrupt is generated to the core. To clear this bit, read the Error and Status register, then write 1 to the bit. Writing 0 has no effect.

FlexCAN Error and Status Register (FCSTATUS) BASE+\$10	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	BIT1_ERR	BIT0_ERR	ACK_ERR	CRC_ERR	FORM_ERR	STUFF_ERR	TX_WARN	RX_WARN	IDLE	TX/RX	FCS			0	BOFF_INT	ERR_INT	WAKE_INT
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_



**FlexCAN Interrupt Mask Register 1 (FCIMASK1)**

Bits	Name	Description
15-0	BUF <sub>n</sub> M	<b>Interrupt Mask Register 1</b>
		This register contains interrupt mask bit per MB. It enables FlexCAN to determine which buffer will generate an interrupt after a successful transmission/reception when the corresponding FCIFLAG1 bit is set.
	0	The corresponding buffer interrupt is disabled
	1	The corresponding buffer interrupt is enabled
		<b>Note:</b> Setting or clearing a bit in the FC1MASK1 register can assert, or clear an interrupt request respectively.

FlexCAN Interrupt Mask Register 1 (FCIMASK1) BASE+\$11	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	BUF1	BUF1	BUF1	BUF1	BUF1	BUF1	BUF9	BUF8	BUF7	BUF6	BUF5	BUF4	BUF3	BUF2	BUF1	BUF0
	Write	5M	4M	3M	2M	1M	0M	M	M	M	M	M	M	M	M	M	M
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# FLEX

## FlexCAN Interrupt Flag Register 1 (FCIFLAG1)

Bits	Name	Description
15-0	BUF <i>n</i>	<b>Interrupt Flag Register 1</b>
		This register contains one interrupt flag bit per MB. Each successful transmission/reception sets the corresponding BUF <i>n</i> bit and if the corresponding FCIMASK1 bit is set, will generate an interrupt.
	0	The corresponding buffer did not successfully complete transmission or reception
	1	The corresponding buffer successfully completed transmission or reception

FlexCAN Interrupt Flag Register 1 (FCIFLAG1) BASE+\$12	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	BUF15	BUF14	BUF13	BUF12	BUF11	BUF10	BUF9	BUF8	BUF7	BUF6	BUF5	BUF4	BUF3	BUF2	BUF1	BUF0
	Write	5	4	3	2	1	0										
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# FLEX

## FlexCAN Error Counters Register (FCERR\_CNTRS)

Bits	Name	Description
15-8	CANRX_ERR_CNTR	<b>Receive Error Counters</b>
		There are two error counters in the FlexCAN module. 1. Receive error counter 2. Transmit error counter <b>Note:</b> Both counters are <i>read-only</i> except for Freeze/Halt modes.
7-0	CANTX_ERR_CNTR	<b>Transmit Error Counters</b>
		Each counter is comprised of: <ul style="list-style-type: none"> <li>• 8-bit up/down counter</li> <li>• Increment by eight (CANRX_ERR_CNTR also by one)</li> <li>• Decrement by one</li> <li>• Avoid decrement when equal to zero</li> <li>• CANRX_ERR_COUNTER preset to a value <math>119 &lt; x &lt; 127</math></li> <li>• Value after reset = zero</li> <li>• Detect values for error passive, busoff and error active transitions and to alert the core</li> <li>• Cascade usage of CANRX_ERR_CNTR with an internal other counter to detect the 128 occurrences of 11 consecutive recessive bits to determine move from Busoff into Error Active</li> </ul>

FlexCAN Error Counters Register (FC_ERR_CNTRS) BASE+\$13	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CANRX_ERR_CNTR								CANTX_ERR_CNTR							
	Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_ Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# GPIO

## GPIO Pull-Up Enable Register (PUR)

Bits	Name	Description
7-0	PU	<b>Pull-Up Enable</b>
		These read/write bits enable/disable the pull-up on each GPIO pin.
	0	Pull-up is disabled
	1	Pull-up is enabled (when DDR=0 and PER=0)

GPIO Pull-Up Enable Register (PUR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	PU							
	Write																
	Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Reserved Bits



Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# GPIO

## GPIO Data Register (DR)

Bits	Name	Description
7-0	D	Data
		These read/write bits hold data coming either from the pin or the IPBus. That ability makes the DR data interface between the pin and IPBus.

GPIO Data Register (DR) BASE+\$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	D							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# GPIO

## GPIO Data Direction Register (DDR)

Bits	Name	Description
7-0	DD	<b>Data Direction</b>
		These read/write bit configures the state of the pin as either an input or output when PER is set to zero. When DDR is set to zero, the pin is an input. When DDR is set to one, the pin is an output.
	0	Pin is an input
	1	Pin is an output

GPIO Data Direction Register (DDR) BASE+\$2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	DD							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# GPIO

## GPIO Peripheral Enable Register (PER)

Bits	Name	Description
7-0	PE	<b>Peripheral Enable</b>
		These read/write bits determine the configuration of the GPIO. When PER value is one, the GPIO module is configured for Normal mode. In this mode a peripheral masters the GPIO pin. The master condition included configuring the GPIO pin as a required input (with or without pull-up), or output depending on the status of the peripheral output disable. It also includes data transfers from the pin to the peripheral.
		0 Pin is for GPIO (GPIO mode)
		1 Pin is for peripheral (Normal mode)

GPIO Peripheral Enable Register (DDR) BASE+\$3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	PE							
	Write																
	Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 5 of 11

# GPIO

## GPIO Interrupt Assert Register (IAR)

Bits	Name	Description
7-0	IA	<b>Interrupt Assert</b>
		These read/write bits are used for software testing only. An interrupt is asserted when the IAR is set to one. It is cleared by writing zeros into the IAR register.

<b>GPIO Interrupt Assert Register (IAR) BASE+\$4</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	IA							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# GPIO

## GPIO Interrupt Enable Register (IENR)

Bits	Name	Description
7-0	IEN	<b>Interrupt Enable</b>
		These read/write bits enables or disables the edge detection for any incoming interrupt from the pin. Bits are set to one for interrupt detection. The interrupts are recorded in the GPIO_X_IPR.
	0	Interrupt is disabled
	1	Interrupt is enabled

GPIO Interrupt Enable Register (IENR) BASE+\$5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	IEN							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# GPIO

## GPIO Interrupt Polarity Register (IPOLR)

Bits	Name	Description
7-0	IPOL	<b>Interrupt Polarity</b>
		These read/write bits are used for polarity detection caused by any external interrupts. The interrupt at the pin is active low when this register is set to one (falling edge causes the interrupt). The interrupt seen at the pin is active high when these bits are set to zero (rising edge causes the interrupt). This is true only when the IENR is set at one. There is no effect on the interrupt if the IENR is set to zero.
	0	Interrupt is active high
	1	Interrupt is active low

GPIO Interrupt Polarity Register (IPOLR) BASE+\$6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	0	IPOL						
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# GPIO

## GPIO Interrupt Pending Register (IPR)

Bits	Name	Description
7-0	IP	<b>Interrupt Pending</b>
		These <i>read-only</i> bits are used for polarity detection caused by any external interrupts. The interrupt at the pin is active low when this register is set to one (falling edge causes the interrupt). The interrupt seen at the pin is active high when these bits are set to zero (rising edge causes the interrupt). This is true only when the IENR is set at one. There is no effect on the interrupt if the IENR is set to zero.
0		Interrupt is active high
1		Interrupt is active low

GPIO Interrupt Pending Register (IPR) BASE+\$7	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	IP																
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_ Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 9 of 11

# GPIO

## GPIO Interrupt Edge Sensitive Register (IESR)

Bits	Name	Description
7-0	IES	<b>Interrupt Edge Sensitive</b>
		These read/write bits clears the corresponding IPR bit field by writing one to a bit in the IESR. Writing zero to an IESR bit is ignored. When an edge is detected by the edge detector circuit, and IENR is set to one, IESR records the interrupt.
	0	Interrupt is active high
	1	Interrupt is active low

GPIO Interrupt Edge Sensitive Register (IESR) BASE+\$8	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	IES							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits



Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# GPIO

## GPIO Push/Pull Output Mode Control Register (PPMODE)

Bits	Name	Description
7-0	OEN	<b>Output Enable</b>
		These read/write bits explicitly sets each output driver to either push/pull or Open Drain mode.
	0	Open Drain mode
	1	Push/Pull mode

GPIO Push/Pull Output Mode Control Register (PPMODE) BASE+\$9	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	OEN							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_ Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 11 of 11

# GPIO

## GPIO Raw Data Register (RAWDATA)

Bits	Name	Description
7-0	RAW DATA	Raw Data
		These <i>read-only</i> bits allow they hybrid controller direct access to the logic values on each GPIO pin even when pins are not in the GPIO mode. The value at reset is unknown.
	0	Logic zero present on GPIO pin
	1	Logic one present on GPIO pin

GPIO Raw Data Register (RAWDATA) BASE+\$A	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	RAWDATA								
	Write																	
	Reset	0	0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U

Reserved Bits

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

**PS**

**Power Supervisor Control Register (LVICTLR)**

Bits	Name	Description
1	LVIE27	<b>2.7V Low Voltage Interrupt Enable</b>
		0   Interrupt is disabled
		1   Interrupt is enabled
0	LVIE22	<b>2.2V Low Voltage Interrupt Enable</b>
		0   Interrupt is disabled
		1   Interrupt is enabled

Power Supervisor Control Register (LVICTLR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LVIE27	LVIE22
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# PS

## Power Supervisor Status Register (LVISR)

Bits	Name	Description	
4	LVI	<b>Low Voltage Interrupt</b>	
		0   Interrupt is not asserted	
		1   Interrupt is asserted	
3	LVIS27S	<b>Sticky 2.7V Low Voltage Interrupt Source</b>	
		0   2.7V interrupt threshold not passed	
		1   3.0V supply has dropped below 2.7V interrupt threshold	
2	LVIS22S	<b>Sticky 2.2V Low Voltage Interrupt Source</b>	
		When this bit is set, it must be cleared explicitly by writing one to it. Writing zero has no effect.	
		0   2.2V interrupt threshold has not been passed	
1	LVIS27	<b>Non-Sticky 2.7V Low Voltage Interrupt Source</b>	
		This bit may reset itself if the supply voltage raises above the comparator threshold point. The bit cannot be modified. This bit is set whenever the LVIT27 bit, illustrated in <a href="#">Figure 10-1</a> , is one long enough to pass through the glitch filter.	
		0   2.7V interrupt threshold has not been passed	
0	LVIS22	<b>Non-Sticky 2.2V Low Voltage Interrupt Source</b>	
		This <i>read-only</i> bit will reset itself if the supply voltage raises above the comparator threshold point. This bit cannot be modified. This bit is set whenever the LVIT22 bit, illustrated in <a href="#">Figure 10-1</a> , is one long enough to pass through the glitch filter.	
		0   2.2V interrupt threshold has not been passed	
1	LVIS22	1   2.5V supply is below the 2.2V interrupt threshold	

Power Supervisor Status Register (LVISR) BASE+\$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	0	0	0	LVI	LVIS27S	LVIS22S	LVIS27	LVIS22
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# PWM

## PWM Control Register (PMCTL)

Please see the following page for continuation of this register.

Bits	Name	Description
15-12	LDFQ	<b>Load Frequency</b>
		These buffered read/write bits select the PWM load frequency. Reset clears the LDFQ bits, selecting loading every PWM opportunity. The occurrence of a PWM opportunity is determined by the half bit.  <b>Note:</b> The LDFQx bits take effect when the current load cycle is complete, regardless of the state of the Load Okay (LDOK) bit. Reading the LDFQx bits reads the buffered values and not necessarily the values currently in effect. See <a href="#">Table 11-10</a> .
11	HALF	<b>Half Cycle Reload</b>
		This read/write bit enables half cycle reloads in Center-Aligned PWM mode. This bit has no effect on Edge-Aligned PWMs.
		0 Half cycle reloads disabled
		1 Half cycle reloads enabled
10	IPOL2	<b>Current Polarity 2</b>
		This buffered read/write bit selects the PWM value register for the PWM4 and PWM5 pins in top/bottom manual deadtime correction.
		0 PWM value register four in next PWM cycle
		1 PWM value register five in next PWM cycle
9	IPOL1	<b>Current Polarity 1</b>
		This buffered read/write bit selects the PWM value register for the PWM2 and PWM3 pins in top/bottom manual deadtime correction.
		0 PWM value register two in next PWM cycle
		1 PWM value register three in next PWM cycle
8	IPOL0	<b>Current Polarity 0</b>
		This buffered read/write bit selects the PWM value register for the PWM0 and PWM1 pins in top/bottom manual deadtime correction.
		0 PWM value register zero in next PWM cycle
		1 PWM value register one in next PWM cycle

PWM Control Register (PMCTL) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	LDFQ				HALF	IPOL2	IPOL1	IPOL0	PRSC	PWMRIE	PWMF	ISENS		LDOK	PWMEN		
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# PWM

## PWM Control Register (PMCTL) Continued

Bits	Name	Description				
7-6	<b>PRSC</b>	<b>Prescaler</b>				
<p>These buffered read/write bits select the PWM clock frequency.</p> <p><b>Note:</b> Reading the PRSC<math>n</math> bits reads the buffered values and not necessarily the values currently in effect. The PRSC<math>n</math> bits take effect at the beginning of the next PWM cycle and only when the LDOK bit is set.</p>						
5	<b>PWMRIE</b>	<b>PWM Reload Interrupt Enable</b>				
<p>This read/write bit enables the PWMF flag to generate interrupt requests.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 5%; text-align: center;">0</td> <td>PWMF interrupt request disabled</td> </tr> <tr> <td style="text-align: center;">1</td> <td>PWMF interrupt request enabled</td> </tr> </table>			0	PWMF interrupt request disabled	1	PWMF interrupt request enabled
0	PWMF interrupt request disabled					
1	PWMF interrupt request enabled					
4	<b>PWMF</b>	<b>PWM Reload Flag</b>				
<p>This read/write flag is set at the beginning of every reload cycle regardless of the state of the LDOK bit. Clear PWMF by reading it, then write a Logic 0 to it. If another reload occurs before the clearing sequence is complete, writing Logic 0 to PWMF has no effect.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 5%; text-align: center;">0</td> <td>No new reload cycle since last PWMF clearing</td> </tr> <tr> <td style="text-align: center;">1</td> <td>New reload cycle since last PWMF clearing</td> </tr> </table>			0	No new reload cycle since last PWMF clearing	1	New reload cycle since last PWMF clearing
0	No new reload cycle since last PWMF clearing					
1	New reload cycle since last PWMF clearing					
3-2	<b>ISENS</b>	<b>Current Status</b>				
<p>These read/write bits select the top/bottom correction scheme.</p> <p>These read/write bits select the top/bottom correction scheme, illustrated in <a href="#">Table 11-2</a>. Reset clears the ISENS<math>n</math> bits. This selects manual correction or no correction, or automatic deadtime correction.</p> <p><b>Note:</b> The ISENS<math>n</math> bits are not buffered. Changing the current status sensing method can affect the present PWM cycle.</p>						
1	<b>LDOK</b>	<b>Load Okay</b>				
<p>This read/write bit loads the prescaler bits of PMCTL and the entire PWMCM register and PWMVAL registers into a set of buffers.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 5%; text-align: center;">0</td> <td>No action is taken</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Load prescaler, PWM modulus and PWM values into buffers</td> </tr> </table>			0	No action is taken	1	Load prescaler, PWM modulus and PWM values into buffers
0	No action is taken					
1	Load prescaler, PWM modulus and PWM values into buffers					
0	<b>PWMEN</b>	<b>PWM Enable</b>				
<p>This read/write bit enables the PWM generator. When PWMEN equals zero, the PWM outputs are in their inactive states unless OUTCTL<math>n</math> equals one.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 5%; text-align: center;">0</td> <td>PWM generator and PWM outputs disabled unless OUTCTRL = 1</td> </tr> <tr> <td style="text-align: center;">1</td> <td>PWM generator and PWM outputs enabled</td> </tr> </table>			0	PWM generator and PWM outputs disabled unless OUTCTRL = 1	1	PWM generator and PWM outputs enabled
0	PWM generator and PWM outputs disabled unless OUTCTRL = 1					
1	PWM generator and PWM outputs enabled					

<b>PWM Control Register (PMCTL) BASE+\$0</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	LDFQ				HALF	IPOL2	IPOL1	IPOL0	PRSC	PWMRIE	PWMF	ISENS		LDOK	PWMEN		
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# PWM

## PWM Fault Control Register (PMFCTL)

Bits	Name	Description
7,5,3,1	Fine	<b>Fault Pin Interrupt Enable</b>
		These read/write bits enable the interrupt request generated by the Fault pin.
		0   Fault <sub>n</sub> interrupt request disabled
		1   FAULT <sub>n</sub> interrupt request enabled
6,4,2,0	FMODE <sub>n</sub>	<b>FAULT<sub>n</sub> Pin Clearing Mode</b>
		These read/write bits select automatic or manual clearing of FAULT <sub>n</sub> pin faults.
		0   Manual fault clearing of FAULT <sub>n</sub> pin faults
		1   Automatic fault clearing of FAULT <sub>n</sub> pin faults

PWM Fault Control Register (PMFCTL) BASE+\$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	FIE3	FMODE3	FIE2	FMODE2	FIE1	FMODE1	FIE0	FMODE0
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# PWM

## PWM Fault Status and Acknowledge Register (PMFSA)

Bits	Name	Description
15, 13, 11, 9	FPIN $n$	<b>Fault<math>n</math> Pin</b>
		These <i>read-only</i> bits reflect the current state of the filtered FAULT $n$ bit. Reset has no effect on FPIN $n$ .
	0	Logic 0 on the FAULT $n$ bit
	1	Logic 1 on the FAULT $n$ bit
14, 12, 10, 8	FFLAG $n$	<b>Fault<math>n</math> Pin Flag</b>
		These <i>read-only</i> flags are set within two IPBus cycles after a rising edge on the FAULT $n$ pin. Clear these bits by writing Logic 1 to the FTACK $n$ bit in the PMFSA register. Reset clears FFLAG $n$ .
	0	No fault on the FAULT $n$ bit
	1	Fault on the FAULT $n$ bit
6, 4, 2, 0	FTACK $n$	<b>Fault<math>n</math> Pin Acknowledge</b>
		Writing a Logic 1 to FTACK $n$ clears FFLAG $n$ . Writing a Logic 0 has no effect. Reading these bits reads the appropriate DT $n$ bit. Please see <a href="#">Section 11.10.3.5</a> . Reset clears FTACK $n$ . The fault protection is enabled even when the PWM is not enabled; therefore if a fault is latched, it must be cleared prior to enabling the PWM to prevent an unexpected interrupt.
5 - 0	DT $n$	<b>Deadtimen</b>
		These are <i>read-only</i> bits. The DT $n$ bits are grouped in pairs, DT0 and DT1, DT2 and DT3, DT4 and DT5. Each pair reflects the corresponding IS $n$ pin value as sampled during deadtime period. A reset clears these bits.

PWM Fault Status & Acknowledge Register (PMFSA) BASE+\$2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	FPIN3	FFLAG3	FPIN2	FFLAG2	FPIN1	FFLAG1	FPIN0	FFLAG0	0	0	DT5	DT4	DT3	DT2	DT1	DT0
	Write											FTACK3		FTACK2		FTACK1	
Reset	U	0	U	0	U	0	U	0	0	0	0	0	0	0	0	0	0

Reserved Bits



Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# PWM

## PWM Output Control Register (PMOUT)

Bits	Name	Description
15	PAD_EN	<b>Output Pad Enable</b>
		The PWM output pads can be enabled or disabled by setting this bit. The power-up default has the pads disabled. This bit does not affect the functionality of the PWM, so the PWM module can be energized with the output pads disabled. This enable is to power-up with a safe default value for the PWM drivers.
		0 Output pads disabled (tri-stated)
		1 Output pads enabled (not tri-stated)
13-8	OUTCTRL5-0	<b>Output Control Enables</b>
		These read/write bits enable software control of their corresponding PWM pin. When OUTCTRL $n$ is set, the OUT $n$ bit activates and deactivates the PWM $n$ output. A reset clears the OUTCTRL bits. When operating the PWM in Complementary mode, these bits must be switched in pairs for proper operation. Please see <a href="#">Section 11.5</a> for details.
		0 Software control disabled
		1 Software control enabled
5-0	OUT0-5	<b>Output Control0-5</b>
		When corresponding OUTCTRL bit is set, these read/write bits control the PWM pins. See <a href="#">Table 11-12</a> .

PWM Output Control Register (PMOUT) BASE+\$3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	PAD_EN	0	OUT CTL5	OUT CTL4	OUT CTL3	OUT CTL2	OUT CTL1	OUT CTL0	0	0	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 6 of 16

# PWM

## PWM Counter Register (PMCNT)

Bits	Name	Description
14-0	CNT	Counter
		These <i>read-only</i> bits display the state of the 15-bit PWM counter.

<b>PWM Counter Register (PMCNT)</b> <b>BASE+\$4</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	CNT															
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# PWM

## PWM Counter Modulo Register (PWMCM)

Bits	Name	Description
14-0	CM	<b>Counter Modulo</b>
		The 15-bit unsigned value written to this buffered read/write register defines the number of PWM clocks to use in determining the PWM period. Do not write a modulus value of zeros into these bits.  <b>Note:</b> The PWM Counter Modulo register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins. Reading PWMCM reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

<b>PWM Counter Modulo Register (PWMCM) BASE+\$5</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	CM														
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# PWM

## PWM Value Registers (PWMVAL0-5)

Bits	Name	Description
15-0	VAL	Value
<p>The PWM Value registers are buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins. Reading PWMVAL<math>n</math> reads the value in a buffer and not necessarily the value the PWM generator is currently using. A PWM value less than, or equal to zero, deactivates the PWM output for the entire PWM period. A PWM value greater than, or equal to the modulus, activates the PWM output for the entire PWM period. Please see <a href="#">Table 11-1</a>.</p> <p><b>Note:</b> The terms activate and deactivate refer to the high and low logic states of the PWM outputs.</p>		

PWM Value Registers (PWMVAL0-5) BASE+\$6 to \$B	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	VAL															
	Write	VAL															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# PWM

## PWM Deadtime Register (PMDEADTIME)

Bits	Name	Description
11-0	PWMDT	<b>PWM Deadtime</b>
		<p>The 12-bit value written to this write-protectable register is the number of PWM clock cycles in complementary channel operation. A reset sets the PWM Deadtime register to a default value of 0x0FFF, selecting a deadtime of 4096-PWM clock cycles minus one IPBus clock cycle. This register is write protected after the Write Protect (WP) bit in the PWM configuration register is set. Please refer to <a href="#">Section 11.10.10</a>.</p> <p><b>Note:</b> Deadtime is affected by changes to the prescaler value. The deadtime duration is determined as follows: <math>DT = P \times PWMDT - 1</math> IPBus clocks, where DT is deadtime, P is the prescaler value, PWMDT is the programmed value of deadtime. For example: if the prescaler is programmed for a divide-by-two and PWMDT is set to five, then <math>P = 2</math> and the deadtime value is equal to:</p> <p style="text-align: center;"><math>DT = 2 \times 5 - 1 = 9</math> IPBus clock cycles.</p>

PWM Deadtime Register (PMDEADTIME) BASE+\$C	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	PWMDT											
	Write	0	0	0	0	PWMDT											
	Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# PWM

PWM Disable Mapping Register 1 (PMDISMAP1)

PWM Disable Mapping Register 2 (PMDISMAP2)

Bits	Name	Description
23-0	DISMAP $n$	Disabled Mapping $n$
		These write-protectable registers determine which PWM pins are disabled by fault protection inputs, provided in Table 11-6. Reset sets all of the bits used in the PWM disable mapping registers. These registers are write-protected after the WP bit in the PWM Configure register is set. Register bits 15-8 in the PMDISMAP2 register cannot be modified. The bits are read as 0.

PWM Disable Mapping Register (PMDISMAP1) BASE+\$D	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	DISMAP[15:0]															
	Write	DISMAP[15:0]															
	Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PWM Disable Mapping Register (PMDISMAP2) BASE+\$E	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	DISMAP[23:16]							
	Write									DISMAP[23:16]							
	Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Reserved Bits

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# PWM

## PWM Configure Register (PMCFG)

Please see the following page for continuation of this register.

Bits	Name	Description
14	DBG_EN	<b>Debug Enable</b>
		When this <i>write-protectable</i> bit is set to one, the PWM will continue to run while the chip is in EOnCE Debug mode. If the device enters the EOnCE mode and this bit is zero, the PWM outputs will be switched to their inactive state until the EOnCE mode is exited. At that point, the PWM pins will resume operation as programmed in the PWM registers. PWM parameters updates do not occur in EOnCE Debug mode.
13	WAIT_EN	<b>Wait Enable</b>
		When this <i>write-protectable</i> bit is set to one, the PWM will continue to run while the chip is in the Wait mode. In this mode, the peripheral clock continues to run; however, the core clock does not. If the device enters the Wait mode and this bit is zero, the PWM outputs will be switched to their inactive state until the Wait mode is exited. At the point of exit the PWM pins will resume operation as programmed in the PWM registers. PWM parameter updates do not occur in Wait mode.
12	EDG	<b>Edge-Aligned or Center-Aligned PWMs</b>
		This <i>write-protectable</i> bit determines whether all PWM channels will use Edge-Aligned or Center-Aligned wave forms.
	0	Center-Aligned PWMs
	1	Edge-Aligned PWMs
10-8	TOPNEG	<b>Top-Side PWM Polarity</b>
		This <i>write-protectable</i> bit determines the polarity for the top-side PWMs.
	0	Positive top-side polarity
	1	Negative top-side polarity
6-4	BOTNEG	<b>Bottom-side PWM Polarity</b>
		This <i>write-protectable</i> bit determines the polarity for the bottom-side PWMs.
	0	Positive bottom-side polarity
	1	Negative bottom-side polarity

PWM Configure Register (PMCFG) BASE+\$F	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	DBG_EN	WAIT_EN	EDG	0	TOP_NEG45	TOP_NEG23	TOP_NEG01	0	BOT_NEG45	BOT_NEG23	BOT_NEG01	INDEP_45	INDEP_45	INDEP_01	WP
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# PWM

## PWM Configure Register (PMCFG) Continued

Bits	Name	Description
<b>6-4</b>	<b>BOTNEG</b>	<b>Bottom-side PWM Polarity</b>
		This <i>write-protectable</i> bit determines the polarity for the bottom-side PWMs.
	0	Positive bottom-side polarity
	1	Negative bottom-side polarity
<b>3-1</b>	<b>INDEP</b>	<b>Independent or Complement Pair Operation</b>
		This <i>write-protectable</i> bit determines if the PWM channels will be independent PWMs or complementary PWM pairs.
	0	Complementary PWM pair
	1	Independent PWMs
<b>0</b>	<b>WP</b>	<b>Write Protect</b>
		This bit enables write-protection for all <i>write-protectable</i> registers. While clear, WP allows write-protected registers and bits to be written. When set, WP prevents writes to <i>write-protectable</i> registers and bits. Once set, WP can be cleared only by a reset. <i>Write-protectable</i> registers and bits include: PMDISMAP1–2, PMDEADTM, PMCFG, and the ENHA bit in the Channel Control Register (PMCCR). The VLMODE, SWP45, SWP23, and SWP01 bits in the PMCCR are protected when the Enable Hardware Acceleration (ENHA) bit is set to zero in the PMCCR. ENHA is in turn, protected by setting the WP bit in the PMCFG.
	0	<i>Write-protectable</i> registers may be written
	1	<i>Write-protectable</i> registers are write protected

PWM Configure Register (PMCFG) BASE+\$F	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	DBG_EN	WAIT_EN	EDG	0	TOP_NEG45	TOP_NEG23	TOP_NEG01	0	BOT_NEG45	BOT_NEG23	BOT_NEG01	INDEP_45	INDEP_45	INDEP_01	WP
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits



Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# PWM

## PWM Channel Control Register (PMCCR)

Please see the following page for continuation of this register.

Bits	Name	Description
15	ENHA	<b>Enable Hardware Acceleration</b>
		This bit enables writing to the nBX, VLMODE, SWP45, SWP23, and SWP01 bits. The bit is <i>write-protectable</i> by WP bit in the PMCFG register.
	0	Disable writing to nBX, VLMODE, SWP45, SWP23, and SWP01 bits
	1	Enable writing to nBX, VLMODE, SWP45, SWP23, and SWP12 bits
14	nBX	<b>56F80x Compatibility</b>
		This bit is used to enable/disable improved SWAP and MASK operations. If it is cleared, SWAP/MASK operates identical to the 56F80x version of this module. Please see 56F80x User's Manual for details.
	0	SWAP and MASK provide 56F80x compatible operation
	1	SWAP <sub>n</sub> and MASK <sub>n</sub> provide new functionality described in this manual
13-8	MASK5-0	<b>Mask5-0</b>
		This bit field determines the mask for each of the PWM logical channels.
	0	Unmasked
	1	Masked, channel deactivated
5-4	VLMODE	<b>Value Register Load Mode</b>
		These bits determine the way the PWM value register are being are being loaded. These bits are write-protected when ENHA is zero.
	00	Each PWM value register is accessed independently
	01	Writing to PWM value register zero also writes to PWM value registers one to five
	10	Writing to PWM value register zero also writes to PWM value registers one to three
	11	Reserved

PWM Channel Control Register (PMCCR) BASE+\$10	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ENHA	nBX	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	0	0	VLMODE		0	SWP 45	SWP 23	SWP 01
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# PWM

## PWM Channel Control Register (PMCCR) Continued

Bits	Name	Description
5-4	<b>VLMODE</b>	<b>Value Register Load Mode</b>
		These bits determine the way the PWM value register are being are being loaded. These bits are write-protected when ENHA is zero.
		00 Each PWM value register is accessed independently
		01 Writing to PWM value register zero also writes to PWM value registers one to five
		10 Writing to PWM value register zero also writes to PWM value registers one to three
		11 Reserved
2	<b>SWP45</b>	<b>Swap45</b>
		This bit is write-protected when ENHA is zero.
		0 No swap
		1 Channels four and five are swapped
1	<b>SWP23</b>	<b>Swap23</b>
		This bit is write-protected when ENHA is zero.
		0 No swap
		1 Channels two and three are swapped
0	<b>SWP01</b>	<b>Swap01</b>
		This bit is write-protected when ENHA is zero.
		0 No swap
		1 Channels zero and one are swapped

PWM Channel Control Register (PMCCR) BASE+\$10	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	ENHA	nBX	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	0	0	VLMODE		0	SWP 45	SWP 23	SWP 01
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# PWM

## PWM Port Register (PMPORT)

Bits	Name	Description
6-0	PORT	Port
These are <i>read-only</i> bits; therefore, any writes to them are not affected.		

PWM Port Register (PMPORT) BASE+\$11	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	0	IS2	IS1	IS0	FAULT3	FAULT2	FAULT1	FAULT0
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	U	U	U	U	U	U	U

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# PWM

## PWM Internal Correction Control Register (PMICCR)

Bits	Name	Description
2	ICC2	<b>Internal Current Control 2</b>
		This bit controls PWM4/PWM5 pair.
	0	PWM value register to use is determined by the ISENS[0:1] setting. The current direction sensed is captured in the DT4 and DT5 bits of the PMFSA register.
	1	Use PWMVAL4 register when the PWM counter is counting up. Use PWMVAL5 register when counting down.
1	ICC1	<b>Internal Current Control 1</b>
		This bit controls PWM2/PWM3 pair.
	0	PWM value register to use is determined by the ISENS[0:1] setting. The current direction sensed is captured in the DT2 and DT3 bits of the PMFSA register.
	1	Use PWMVAL2 register when the PWM counter is counting up. Use PWMVAL3 register when counting down.
0	ICC0	<b>Internal Current Control 0</b>
		This bit controls PWM0/PWM1 pair.
	0	PWM value register to use is determined by the ISENS[0:1] setting. The current direction sensed is captured in the DT0 and DT1 bits of the PMFSA register.
	1	Use PWMVAL0 register when the PWM counter is counting up. Use PWMVAL1 register when counting down.

PWM Internal Correction Control Register (PMICCR) BASE+\$12	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ICC2	ICC1	ICC0
	Write																	
Reset		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# DEC

## Decoder Control Register (DECCR)

Please see the following page for continuation of this register.

Bits	Name	Description
15	HIRQ	<b>Home Signal Transition Interrupt Request</b>
		This bit is set when a transition on the HOME signal occurs according to the HNE bit. A HOME interrupt occurs if HIRQ and HIE bits are set. HIRQ bit remains set until it is cleared by software. To clear, write 1 to this bit.
		0 No interrupt
		1 HOME signal transition interrupt request
14	HIE	<b>Home Interrupt Enable</b>
		This read/write bit enables HOME signal interrupts.
		0 Disable HOME interrupts
		1 Enable HOME interrupts
13	HIP	<b>Enable HOME to Initialize Position Counters UPOS and LPOS</b>
		This read/write bit allows the position counter to be initialized by the HOME signal.
		0 No action
		1 HOME signal initializes the position counter
12	HNE	<b>Use Negative Edge of HOME Input</b>
		This read/write bit determines the edge of the HOME input used to initialize the position counter.
		0 Use positive transition edge of HOME pulse
		1 Use negative transition edge of HOME pulse
11	SWIP	<b>Software Triggered Initialization of Position Counters UPOS and LPOS</b>
		Writing 1 to this bit will transfer the UIR and LIR contents to UPOS and LPOS respectively. This bit is always read as 0.
		0 No action
		1 Initialize position counter

Decoder Control Register (DECCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE		
	Write					SWIP												
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# DEC

## Decoder Control Register (DECCR) Continued

Please see the following page for continuation of this register.

Bits	Name	Description
10	REV	<b>Enable Reverse Direction Counting</b>
		This read/write bit reverses the interpretation of the quadrature signal, changing the direction of count.
	0	Count normally
	1	Count in the reverse direction
9	PH1	<b>Enable Signal Phase Count Mode</b>
		This read/write bit bypasses the Quadrature Decoder logic.
	0	Use standard Quadrature Decoder where PHASEA and PHASEB represents a two-phase quadrature signal.
	1	Bypass the Quadrature Decoder, implement pulse accumulator, a positive transition of the PHASEA input generates a count signal. The PHASEB input and the REV bit controls the counter direction. -IF REV = 0, PHASEB = 0, then count up -IF REV = 0, PHASEB = 1, then count down -IF REV = 1, PHASEB = 0, then count down -IF REV = 1, PHASEB = 1, then count up
8	XIRQ	<b>INDEX Pulse Interrupt Request</b>
		This bit is set when an INDEX interrupt occurs. It remains set until cleared by software. To clear, write a one to this bit. An INDEX pulse will cause this bit to become set regardless of the XIE bit value.
	0	No interrupt has occurred
	1	INDEX pulse interrupt has occurred
7	XIE	<b>INDEX Pulse Interrupt Enable</b>
		This read/write bit enables INDEX interrupts. This bit gates INDEX pulse interrupts before they reach the interrupt controller.
	0	INDEX pulse interrupt is disabled
	1	INDEX pulse interrupt is enabled

Decoder Control Register (DECCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE		
	Write					SWIP												
Reset		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# DEC

## Decoder Control Register (DECCR) Continued

Please see the following page for continuation of this register.

Bits	Name	Description
6	XIP	<b>INDEX Triggered Initialization of Position Counters UPOS and LPOS</b>
		This read/write bit enables the position counter to be initialized by the INDEX pulse.
	0	No action
	1	INDEX pulse initializes the position counter
5	XNE	<b>Use Negative Edge of INDEX Pulse</b>
		This read/write bit determines the edge of the INDEX pulse used to initialize the position counter.
	0	Use positive transition edge of INDEX pulse
	1	Use negative transition edge of INDEX pulse
4	DIRQ	<b>Watchdog Timeout Interrupt Request</b>
		This bit is set when a Watchdog Timeout interrupt occurs. It remains set until it is cleared by software. To clear, write 1 to this bit.
	0	No interrupt has occurred
	1	Watchdog Timeout interrupt has occurred

Decoder Control Register (DECCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
	Write					SWIP											
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# DEC

## Decoder Control Register (DECCR) Continued

Bits	Name	Description
3	<b>DIE</b>	<b>Watchdog Timeout Interrupt Enable</b>
		This read/write bit enables Watchdog Timeout interrupts
		0 Watchdog timer interrupt is disabled
		1 Watchdog timer interrupt is enabled
2	<b>WDE</b>	<b>Watchdog Enable</b>
		This bit operates the Watchdog timer monitoring the PHASEA and PHASEB inputs for shaft movement.
		0 Watchdog timer is disabled
		1 Watchdog timer is enabled
1-0	<b>MODE</b>	<b>Switch Matrix Mode</b>
		This read/write bit field selects the Switch Matrix mode, connecting inputs to the Timer module.
		00 Mode 0: Input captures connected to the four input pins
		01 Mode 1: Input captures connected to the filtered versions of the four input pins
		10 Mode 2: PHASEA input connected to both channels 0/1. PHASEB input connected to both channels 2/3 of the timer.
		11 Mode 3: Reserved

Decoder Control Register (DECCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE		MODE	
	Write					SWIP												
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits



Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# DEC

## Decoder Filter Interval Register (DEC FIR)

Bits	Name	Description
7-0	DELAY	<b>Delay</b>
		The value of DELAY plus one represents the filter interval period in increments of the IPBus clock period.

Decoder Filter Interval Register (FIR) BASE+\$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	0	DELAY							
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 6 of 14

# DEC

## Decoder Watchdog Timer Register (DECWTR)

Bits	Name	Description
15-0	COUNT	Count
		COUNT is a binary representation of the number of IPBus clock cycles plus one.

Decoder Watchdog Timer Register (WTR) BASE+\$2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COUNT															
	Write	COUNT															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# DEC

## Decoder Position Difference Counter Register (DECPOSD)

Bits	Name	Description
15-0	POSD	<b>Position Difference Counter</b>
		This read/write register contains the position change in value occurring between each read of the position register. The value of the Position Difference Counter (POSD) register can calculate velocity.

<b>Decoder Position Difference Counter Register (POSD) BASE+\$3</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	POSD															
	Write	POSD															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 8 of 14

**DEC**

**Decoder Position Difference Counter Hold Register (POSDH)**

Bits	Name	Description
15-0	POSDH	<b>Position Difference Counter Hold</b>
		This read/write register contains a snapshot of the value of the POSD register. When the Position register is read, the Position Difference Counter's contents are copied into the POSDH register and the Position Difference Counter is cleared. The value of the POSDH can be used to calculate velocity.

Decoder Position Difference Counter Hold Register (POSDH) BASE+\$4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	POSDH															
Write																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# DEC

## Decoder Revolution Counter Register (DECREV)

Bits	Name	Description
15-0	REV	<b>Revolution Counter</b>
This read/write register contains the current value of the revolution counter.		

Decoder Revolution Counter Register (REV) BASE+\$5	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	REV															
	Write	REV															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 10 of 14

# DEC

## Decoder Revolution Hold Register (DECREVH)

Bits	Name	Description
15-0	REVH	<b>Revolution Hold</b>
		This read/write register contains a snapshot of the value of the REV register.

Decoder Revolution Hold Register (REVH) BASE+\$6	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	REVH															
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# DEC

**Decoder Upper Position Counter Register (DECUPOS)**

**Decoder Lower Position Counter Register (DECLPOS)**

Bits	Name	Description
31-16	UPOS	<b>Upper Position Counter</b>
		This read/write register contains the upper and most significant half of the position counter. This is the binary count from the position counter.
15-0	LPOS	<b>Lower Position Counter</b>
		This read/write register contains the lower and least significant half of the current value of the position counter. It is the binary count from the position counter.

<b>Decoder Upper Position Counter Register (UPOS) BASE+\$7</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	POS[31:16]															
	Write	POS[31:16]															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

<b>Decoder Lower Position Counter Register (LPOS) BASE+\$8</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	POS[15:0]															
	Write	POS[15:0]															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 12 of 14

# DEC

Decoder Upper Position Hold Register (DECUPOSH)

Decoder Lower Position Hold Register (DECLPOSH)

Bits	Name	Description
31-16	UPOSH	<b>Upper Position Hold</b>
		This read/write register contains a snapshot of the UPOS register.
15-0	LPOSH	<b>Lower Position Hold</b>
		This read/write register contains a snapshot of the LPOS register.

<b>Decoder Upper Position Hold Register (UPOSH) BASE+\$9</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	UPOSH[31:16]															
	Write	UPOSH[31:16]															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

<b>Decoder Lower Position Hold Register (LPOSH) BASE+\$A</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LPOSH[15:0]															
	Write	LPOSH[15:0]															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# DEC

**Decoder Upper Initialization Register (DECUIR)**

**Decoder Lower Initialization Register (DECLIR)**

Bits	Name	Description
31-16	UIV	<b>Upper Initialization Value</b>
		This read/write register contains the value to initialize the upper half of UPOS.
15-0	LIV	<b>Lower Initialization Value</b>
		This read/write register contains the value to be used to initialize the lower half of LPOS.

<b>Decoder Upper Initialization Register (UIR) BASE+\$B</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	INITIALIZATION VALUE[31:16]															
	Write	INITIALIZATION VALUE[31:16]															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

<b>Decoder Lower Initialization Register (LIR) BASE+\$C</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	INITIALIZATION VALUE[15:0]															
	Write	INITIALIZATION VALUE[15:0]															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# DEC

## Decoder Input Monitor Register (DECIMR)

Bits	Name	Description
7	<b>FPHA</b>	<b>Filtered PHASEA</b> This is the filtered version of PHASEA input.
6	<b>FPHB</b>	<b>Filtered PHASEB</b> This is the filtered version of PHASEB input.
5	<b>FIND</b>	<b>Filtered Index</b> This is the filtered version of INDEX input.
4	<b>FHOM</b>	<b>Filtered Home</b> This is the filtered version of HOME input.
3	<b>PHA</b>	<b>Phase A Input</b> This is the raw PHASEA input.
2	<b>PHB</b>	<b>Phase B Input</b> This is the raw PHASEB input.
1	<b>INDEX</b>	<b>Index Input</b> This is the raw INDEX input.
0	<b>HOME</b>	<b>Home Switch Input</b> This is the raw HOME input.

Decoder Input Monitor Register (IMR) BASE+\$D	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	FPHA	FPHB	FIND	FHOM	PHA	PHB	INDEX	HOME
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	U	U	U	U	U	U	U	U

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# SCI

## SCI Baud Rate Register (SCIBR)

Bits	Name	Description
12-0	SBR	SCI Baud Rate
		These bits can be modified by writing at any time. Contents have a value of 1 to 81191.

<b>SCI Baud Rate Register (SCIBR)</b> <b>BASE+\$0</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	SBR												
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

 Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# SCI

## SCI Control Register (SCICR)

Please see the following page for continuation of this register.

Bits	Name	Description
15	LOOP	<b>Loop Select</b>
		This bit enables loop operation. This operation disconnects the RXD pin from the SCI and transmitter output goes into the receiver input.
		0 Normal operation enabled
		1 Loop operation enabled
14	SWAI	<b>Stop in Wait Mode</b>
		The SWAI bit disables the SCI in Wait mode.
		0 SCI enabled in Wait mode
		1 SCI disabled in Wait mode
13	RSRC	<b>Receiver Source</b>
		When LOOP=1, the RSRC bit determines the internal feedback path for the receiver.
		0 Receiver input internally connected to transmitter output
		1 Receiver input connected to TXD pin
12	M	<b>Data Format Mode</b>
		This bit determines whether data characters are eight or nine bits long.
		0 One START bit, eight data bits, one STOP bit
		1 One START bit, nine data bits, one STOP bit
11	WAKE	<b>Wake-Up Condition</b>
		This bit determines which condition wakes up the SCI.
		0 Idle line wake-up
		1 Address mark wake-up (a Logic 1 in the MSB position of a receive data character)
10	POL	<b>Polarity</b>
		This bit determines whether to invert the data as it goes from the transmitter to the TXD pin and from the RXD to the receiver. All bits START, DATA, and STOP are inverted as they leave Transmit Shift register and before they enter the Receive Shift register.
		0 Does not invert transmit and receive data bits (Normal mode)
		1 Invert transmit and receive data bits (Inverted mode)

SCI Control Register (SCICR) BASE+\$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIIE	RFIE	REIE	TE	RE	RWU	SBK
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# SCI

## SCI Control Register (SCICR) Continued

Please see the following page for continuation of this register.

Bits	Name	Description
9	PE	<b>Parity Enable</b>
		This bit enables the parity function. When enabled, the parity function replaces the MSB of the data character with a parity bit.
	0	Parity function disabled
	1	Parity function enabled
8	PT	<b>Parity Enable</b>
		This bit determines whether the SCI generates and checks for even or odd parity of the data bits. When even parity, and even number of ones clears the PARITY bit, while an odd number of ones sets the PARITY bit. However, with odd parity, an odd number of ones clears the PARITY bit, while an even number of ones sets the PARITY bit.
	0	Even parity
	1	Odd parity
7	TEIE	<b>Transmitter Empty Interrupt Enable</b>
		This bit enables the TDRE flag to generate interrupt requests.
	0	TDRE interrupt requests disabled
	1	TDRE interrupt requests enabled
6	TIEE	<b>Transmitter Idle Interrupt Enable</b>
		This bit enables the TIDLE flag to generate interrupt requests.
	0	TIDLE interrupt requests disabled
	1	TIDLE interrupt requests enabled
5	RFIE	<b>Receiver Full Interrupt Enable</b>
		This bit enables the RDRF flag or the OR flag to generate interrupt requests.
	0	RDRF and OR interrupt requests disabled
	1	RDRF and OR interrupt requests enabled

SCI Control Register (SCICR) BASE+\$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIEE	RFIE	REIE	TE	RE	RWU	SBK
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# SCI

## SCI Control Register (SCICR) Continued

Bits	Name	Description
4	REIE	<b>Receiver Error Interrupt Enable</b>
		This bit enables the Receive Error (RE) flags (NF, PF, Fe, and OR) to generate interrupt requests. The status bits can be checked during the error interrupt process.
	0	Error interrupt requests disabled
	1	Error interrupt requests enabled
3	TE	<b>Transmitter Enable</b>
		This bit enables the SCI transmitter and configures the TXD pin as the SCI transmitter output. The TE bit can be used to queue an idle preamble.
	0	Transmitter disabled
	1	Transmitter enabled
2	RE	<b>Receiver Enable</b>
		This bit enables the SCI Receiver.
	0	Receiver disabled
	1	Receiver enabled
1	RWU	<b>Receiver Wake-Up</b>
		This bit enables the wake-up function, inhibiting further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU.
	0	Normal operation
	1	Standby state
0	SBK	<b>Send Back</b>
		Setting SBK sends one break character (all Logic 0s include START, DATA, STOP). As long as SBK is set, transmitter sends uninterrupted break characters.
	0	No break characters
	1	Transmit break characters

SCI Control Register (SCICR) BASE+\$1	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIIE	RFIE	REIE	TE	RE	RWU	SBK
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# SCI

## SCI Status Register (SCISR)

Please see the following page for continuation of this register.

Bits	Name	Description
15	TDRE	<b>Transmit Data Register Empty Flag</b>
		This bit is set when the Transmit Shift register receives a character from the SCIDR. Clear TDRE by reading SCISR, then write to the SCIDR.
	0	No character transferred to Transmit Shift register
	1	Character transferred to Transmit Shift register, TDRE
14	TIDLE	<b>Transmitter Idle Flag</b>
		This bit is set when the TDRE flag is set and not data, preamble, or break character is being transmitted. When TIDLE is set, the TXD pin becomes idle (Logic 1). Clear TIDLE by reading the SCISR, then write to the SCIDR.
	0	Transmission in progress
	1	No transmission in progress
13	RDRF	<b>Receive Data Register Full Flag</b>
		This bit is set when the data in the Receive Shift register transfers to the SCIDR. Clear RDRF by reading the SCISR, then read the SCIDR.
	0	Data not available in SCIDR
	1	Received data available in SCIDR
12	RIDLE	<b>Receiver Idle Line Flag</b>
		This bit is set when ten consecutive Logic 1s (if M=0) or eleven consecutive Logic 1s (if M=1) appear on the receiver input. Once the RIDLE flag is cleared by the receiver detecting a Logic 0, a valid frame must again set the RDRF flag before an idle condition can set the RIDLE flag.
	0	Receiver input is either active now or has never become active since the RIDLE flag was last cleared by reset
	1	Receiver input has become idle (after receiving a valid frame)

SCI Status Register (SCISR) BASE+\$3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	TDRE	TIDLE	RDRF	RIDLE	OR	NF	FE	PF	0	0	0	0	0	0	0	0	RAF
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 6 of 7

# SCI

## SCI Status Register (SCISR) Continued

Bits	Name	Description
11	OR	<b>Overrun Flag</b>
		This bit is set when software fails to read the SCIDR before the Receive Shift register receives the next frame. The data in the Shift register is lost, but the data already in the SCIDR is not affected. Clear OR by reading the SCISR, then write the SCISR with any value.
		0 No overrun
		1 Overrun
10	NF	<b>Noise Flag</b>
		This bit is set when the SCI detects noise on the receiver input. The NF bit is set during the same cycle as the RDRF flag, but it is not set in the case of an overrun. Clear NF by reading the SCISR, then write the SCISR with any value.
		0 No noise
		1 Noise
9	FE	<b>Framing Error Flag</b>
		This bit is set when a Logic 0 is accepted as the STOP bit. The FE bit is set during the same cycle as the RDRF flag but it is not set in the case of an overrun. FE inhibits further data reception until it is cleared. Clear FE by reading the SCISR, then write the SCISR with any value.
		0 No framing error
		1 Framing error
8	PF	<b>Parity Error Flag</b>
		This bit is set when the Parity Enable (PE) bit is set and the parity of the received data does not match its parity bit. Clear PF by reading the SCISR, then write the SCISR with any value.
		0 No parity error
		1 Parity error
0	RAF	<b>Receiver Active Flag</b>
		This bit 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 false start bits (usually from noise or baud rate mismatch) or when the receiver detects a preamble.
		0 No reception in progress
		1 Reception in progress

SCI Status Register (SCISR) BASE+\$3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	TDRE	TICLE	RDRF	RIDLE	OR	NF	FE	PF	0	0	0	0	0	0	0	0	RAF
	Write																	
Reset		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits



Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# SCI

## SCI Data Register (SCIDR)

Bits	Name	Description
8-0	RECEIVE DATA	<b>Receive Data</b>
		Writing to this bit field loads the transmit data. Reading the bit field accesses the receive data.
8-0	TRANSMIT DATA	<b>Transmit Data</b>
		Writing to this bit field loads the transmit data. Reading the bit field accesses the receive data.

SCI Data Register (SCIDR) BASE+\$4	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	0	0	0	0	0	0	0	RECEIVE DATA								
	Write								TRANSMIT DATA								
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# SPI

## SPI Status and Control Register (SPSCR)

Please see the following page for continuation of this register.

Bits	Name	Description
15-13	SPR	<b>SPI Baud Rate Select</b>
		While in the Master mode, these read/write bits select one of eight baud rates depicted in <a href="#">Table 14-5</a> . SPR[2:0] have no effect in Slave mode. Use the formula below to calculate the SPI baud rate.
12	DSO	<b>Data Shift Order</b>
		This read/write bit determines whether the MSB or LSB bit is transmitted or received first. Both Master and Slave SPI modules must transmit and receive the same length packets. Regardless how this bit is set, when reading from the SPDRR or writing to the SPDTR, the LSB will always be at bit location zero. If the data length is less than 16 bits, the data will be zero padded on the upper bits.
		0 MSB transmitted first
		1 LSB transmitted first
11	ERRIE	<b>Error Interrupt Enable</b>
		This read/write bit enables the MODF, if MODFEN is also set, and OVRF bits to generate interrupt requests.
		0 MODF and OVRF cannot generate interrupt requests
		1 MODF and OVRF can generate interrupt requests
10	MODFEN	<b>Mode Fault Enable</b>
		This read/write bit, when set to one, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN does not clear the MODF flag.
9	SPRIE	<b>SPI Receiver Interrupt Enable</b>
		This read/write bit enables interrupt requests generated by the SPRF bit. The SPRF bit is set when data transfers from the Shift register to the Receive Data register.
		0 SPRF interrupt requests disabled
		1 SPRF interrupt requests enabled
8	SPMSTR	<b>SPI Master</b>
		This read/write bit selects Master or Slave modes operation.
		0 Slave mode
		1 Master mode

SPI Status/Control Register (SPSCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	SPR			DSO	ERRIE	MODFEN	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE	SPRF	OVRF	MODF	SPTIE	
	Write	SPR			DSO	ERRIE	MODFEN	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE	SPRF	OVRF	MODF	SPTIE	
	Reset	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0	1	

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# SPI

## SPI Status and Control Register (SPSCR) Continued

Please see the following page for continuation of this register.

Bits	Name	Description
7	CPOL	<b>Clock Polarity</b>
		This read/write bit determines the logic state of the SCLK pin between transmissions. To transmit data between SPI modules, the SPI modules must have identical CPOL values.
		0 Falling edge of SCLK starts transmission
		1 Rising edge of SCLK starts transmission
6	CPHA	<b>Clock Phase</b>
		This read/write bit controls the timing relationship between the serial clock and SPI data. Please see <a href="#">Figure 14-4</a> and <a href="#">Figure 14-6</a> . To transmit data between SPI modules, the SPI modules must have identical CPHA values. When CPHA = 0, the SS pin of the Slave SPI module must be set to Logic 1 between data transmissions, illustrated in <a href="#">Figure 14-5</a> .
5	SPE	<b>SPI Enable</b>
		This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. When setting/clearing this bit, no other bits in the SPSCR should be changed. Failure to following this statement may result in spurious clocks.
		0 SPI module disabled
		1 SPI module enabled
4	SPTIE	<b>SPI Transmit Interrupt Enable</b>
		This read/write bit enables interrupt requests generated by the SPTE bit. SPTE is set when data transfers from the SPDTR to the Shift register.
		0 SPTE interrupt requests disabled
		1 SPTE interrupt requests enabled

SPI Status/Control Register (SPSCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SPR			DSO	ERRIE	MODFEN	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE	SPRF	OVRF	MODF	SPTE
	Write																
	Reset	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0	1

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# SPI

## SPI Status and Control Register (SPSCR) Continued

Bits	Name	Description
3	<b>SPRF</b>	<b>SPI Receiver Full</b>
		This <i>read-only</i> flag is set each time data transfers from the Shift register to the SPDRR. SPRF generates an interrupt request if the SPRIE bit in the SPI Control register is set. This bit is cleared by reading the SPDRR.
	0	Data Receive Register (SPDRR) not full
	1	Data Receive Register (SPDRR) full
2	<b>OVRF</b>	<b>Overflow</b>
		This <i>read-only</i> flag is set if software does not read the data in the SPDRR before the next full data enters the Shift register. In an overflow condition, the data already in the SPDRR is unaffected, and the data shifted in last is lost. Clear the OVRF bit by reading the SPSCR and then reading the SPDRR. OVRF generates a Receiver/Error interrupt if the EERIE bit is set.
	0	No overflow
	1	Overflow
1	<b>MODF</b>	<b>Mode Fault</b>
		This <i>read-only</i> flag is set in a slave SPI if the SS pin goes high during a transmission with the MODFEN bit set. In a master SPI, the MODF flag is set if the SS pin goes low at any time with the MODFEN bit set. Clear the MODF bit by writing a one to the MODF bit when it is set. MODF generates a Receiver/Error interrupt if the EERIE bit is set.
	0	$\overline{\text{SS}}$ pin at appropriate logic level
	1	$\overline{\text{SS}}$ pin at inappropriate logic level
0	<b>SPTE</b>	<b>SPI Transmitter Empty</b>
		This <i>read-only</i> flag is set each time the SPDTR transfers data into the Shift register. SPTE generates an interrupt request if the SPTIE bit in the SPI Control register is set. This bit is cleared by writing to the SPDTR.
	0	Data Transmit Register (SPDTR) not empty
	1	Data Transmit Register (SPDTR) empty

SPI Status/Control Register (SPSCR) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	SPR			DSO	ERRIE	MODFEN	SPRIE	SPMSTR	CPOL	CPHA	SPE	SPTIE	SPRF	OVRF	MODF	SPTE
	Write																
	Reset	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0	1

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# SPI

## SPI Data Size and Control Register (SPDSR)

Bits	Name	Description
15	<b>WOM</b>	<b>Wired OR Mode</b>
		This control bit is used to select the nature of the SPI pins. When the WOM bit is set, the SPI pins are configured as open-drain drivers with the pull-ups disabled. When the WOM bit is cleared, the SPI pins are configured as push-pull drivers.
	0	Wired OR mode disabled
	1	Wired OR mode enabled
3-0	<b>DS</b>	<b>Data Size</b>
		<b>Transmission Data</b>
	\$0	Not Allowed
	\$1	2 Bits
	\$2	3 Bits
	\$3	4 Bits
	\$4	5 Bits
	\$5	6 Bits
	\$6	7 Bits
	\$7	8 Bits

<b>SPI Data Size/ Control Register (SPDSR) BASE+\$1</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	WOM	0	0	0	0	0	0	0	0	0	0	0	0	DS			
	Write																	
	Reset		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 5 of 6

# SPI

## SPI Data Receive Register (SPDRR)

Bits	Name	Description
15-0	<i>Rn</i>	<b>Data Receive Number</b>
These <i>read-only</i> bits shows the last data word received after a complete transmission. The SPRF bit is set when new data is transferred to this register.		

SPI Data Receive Register (SPDRR) BASE+\$2	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# SPI

## SPI Data Transmit Register (SPDTR)

Bits	Name	Description
15-0	$T_n$	<b>Data Transmit Number</b>
<p>These <i>write-only</i> bits holds data to be transmitted. When the SPTE bit is set, new data should be written to this register. If new data is not written while in the Master mode, a new transaction will not be initiated until this register is written. When in Slave mode, the old data will be re-transmitted. All data should be written with the LSB at bit zero.</p>		

SPI Data Transmit Register (SPDTR) BASE+\$3	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Write	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved Bits

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 1 of 1

# TSENSOR

## TSensor Control Register (TSEN\_CTRL)

Bits	Name	Description
0	PWR	<b>Power-On</b>
		This bit is used to power current source I <sub>C</sub> ON or OFF.
	0	OFF (default)
	1	On

TSensor Control Register (TSENSOR_CTRL) BASE+\$0	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PWR
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 Reserved Bits



Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# TMR

## TMR Compare Registers 1 (TMRCMP1)

Bits	Name	Description
15-0	COMPARISON 1	<b>Comparison 1</b>
		These read/write bits store the value used for comparison with counter value. There are four Timer Compare1 (TMRCMP1) registers.
		TMR0_CMP1 (Channel 0 Compare 1)—Address: TMR_BASE + \$0 TMR1_CMP1 (Channel 1 Compare 1)—Address: TMR_BASE + \$10 TMR2_CMP1 (Channel 2 Compare 1)—Address: TMR_BASE + \$20 TMR3_CMP1 (Channel 3 Compare 1)—Address: TMR_BASE + \$30

TMR Compare Registers 1 (TMRCMP1)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COMPARISON 1															
	Write	COMPARISON 1															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base + \$0, \$10, \$20, \$30

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 2 of 14

# TMR

## TMR Compare Registers 2 (TMRCMP2)

Bits	Name	Description
15-0	COMPARISON 2	<b>Comparison 2</b>
		These read/write bits store the value used for comparison with counter value. There are four Timer Compare2 (TMRCMP2) registers.
		TMR0_CMP2 (Channel 0 Compare 2)—Address: TMR_BASE + \$1 TMR1_CMP2 (Channel 1 Compare 2)—Address: TMR_BASE + \$11 TMR2_CMP2 (Channel 2 Compare 2)—Address: TMR_BASE + \$21 TMR3_CMP2 (Channel 3 Compare 2)—Address: TMR_BASE + \$31

<b>TMR Compare Registers 2 (TMRCMP2)</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COMPARISON 2															
	Write	COMPARISON 2															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base + \$1, \$11, \$21, \$31

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# TMR

## TMR Capture Registers (TMRCAP)

Bits	Name	Description
15-0	CAPTURE	<b>Capture</b>
		These read/write registers store the values captured from the counters. There are four Timer Capture (TMRCAP) registers.
		TMR0_CAP (Channel 0 Capture)—Address: TMR_BASE + \$2 TMR1_CAP (Channel 1 Capture)—Address: TMR_BASE + \$12 TMR2_CAP (Channel 2 Capture)—Address: TMR_BASE + \$22 TMR3_CAP (Channel 3 Capture)—Address: TMR_BASE + \$32

<b>TMR Capture Registers (TMRCAP)</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CAPTURE															
	Write	CAPTURE															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base + \$2, \$12, \$22, \$32

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 4 of 14

# TMR

## TMR Load Registers (TMRLOAD)

Bits	Name	Description
15-0	LOAD	<b>Load</b>
These read/write registers store the value used to load the counter. There are four Timer Load (TMRLOAD) registers.		
TMR0_LOAD (Channel 0 LOAD)—Address: TMR_BASE + \$3 TMR1_LOAD (Channel 1 LOAD)—Address: TMR_BASE + \$13 TMR2_LOAD (Channel 2 LOAD)—Address: TMR_BASE + \$23 TMR3_LOAD (Channel 3 LOAD)—Address: TMR_BASE + \$33		

<b>TMR Load Registers (TMRLOAD)</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	LOAD															
	Write	LOAD															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base + \$3 \$13, \$23, \$33

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# TMR

## TMR Hold Registers (TMRHOLD)

Bits	Name	Description
15-0	HOLD	<b>Hold</b>
		These read/write registers store the channel's value whenever any timer counter register (TMRCNTR) is read. There are four Timer Hold (TMRHOLD) registers.
		TMR0_HOLD (Channel 0 HOLD)—Address: TMR_BASE + \$4 TMR1_HOLD (Channel 1 HOLD)—Address: TMR_BASE + \$14 TMR2_HOLD (Channel 2 HOLD)—Address: TMR_BASE + \$24 TMR3_HOLD (Channel 3 HOLD)—Address: TMR_BASE + \$34

TMR Hold Registers (TMRHOLD)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	HOLD															
	Write	HOLD															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base + \$4 \$14, \$24, \$34

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 6 of 14

# TMR

## TMR Counter Registers (TMRCNTR)

Bits	Name	Description
15-0	COUNTER	Counter
		These are read/write count registers. There are four Timer Count Registers (TMRCNTR).
		TMR0_CNTR (Channel 0 Counter)—Address: TMR_BASE + \$5 TMR1_CNTR (Channel 1 Counter)—Address: TMR_BASE + \$15 TMR2_CNTR (Channel 2 Counter)—Address: TMR_BASE + \$25 TMR3_CNTR (Channel 3 Counter)—Address: TMR_BASE + \$35

<b>TMR Counter Registers (TMRCNTR)</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COUNTER															
	Write	COUNTER															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base + \$5 \$15, \$25, \$35

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet 7 of 14

# TMR

## TMR Control Registers (TMRCTRL)

Please see the following page for continuation of this register.

Bits	Name	Description
15-13	CM	<b>Count Mode</b>
		This bit field controls the basic counting behavior of the counter.
		000 No operation
		001 Count rising edges of Primary Source –Rising edges if IPS =0 –Falling edges if IPS=1
		010 Count rising and falling edges of Primary Source
		011 Count rising edges of Primary Source while secondary input high active
		100 Quadrature Count mode uses Primary and Secondary Sources
		101 Count edges of Primary Source –IPS=0 then Secondary Input=0; Count down on rising edges of Primary Input –IPS=0 then Secondary Input=1; Count up on rising edges of Primary Input –IPS=1 then Secondary Input=0; Count down on falling edge of Primary Input –IPS=1 then Secondary Input=1; Count up on falling edges of Primary Input
		110 Edge of Secondary Source triggers primary count until compare
		111 Cascaded Counter mode (up/down)
12-9	PCS	<b>Primary Count Source</b>
		This bit field selects the primary count source.
		0000 Counter 0 input pin
		0001 Counter 1 input pin
		0010 Counter 2 input pin
		0011 Counter 3 input pin
		0100 Counter 0 output (OFLAG0)
		0101 Counter 1 output (OFLAG1)
		0110 Counter 2 output (OFLAG2)
		0111 Counter 3 output (OFLAG3)
		1000 Prescaler (IPBus clock divide by 1)
		1001 Prescaler (IPBus clock divide by 2)
		1010 Prescaler (IPBus clock divide by 4)
		1011 Prescaler (IPBus clock divide by 8)
		1100 Prescaler (IPBus clock divide by 16)
		1101 Prescaler (IPBus clock divide by 32)
		1110 Prescaler (IPBus clock divide by 64)
		1111 Prescaler (IPBus clock divide by 128)

TMR Control Registers (TMRCTRL)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	CM			PCS				SCS		ONCE	LENGTH	DIR	Co INIT	OM			
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base + \$6 \$16, \$26, \$36

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# TMR

## TMR Control Registers (TMRCTRL) Continued

Please see the following page for continuation of this register.

Bits	Name	Description
8-7	SCS	<b>Secondary Count Source</b>
		These bits identify the external input pin to be used as a count command. The selected input can trigger the timer to capture the current value of the TMRCNTR register. The selected input can also be used to specify the count direction. The polarity of the signal can be inverted by the IPS bit of the TMRSCR register.
	00	Counter 0 input pin
	01	Counter 1 input pin
	10	Counter 2 input pin
	11	Counter 3 input pin
6	ONCE	<b>Count Once</b>
		This bit selects continuous or one-shot counting modes
	0	Count repeatedly
	1	Count until compare and then stop. If counting up, successful compare occurs when the counter reaches a TMRCMP1 value. If counting down, successful compare occurs when the counter reaches a TMRCMP2 value.
5	LENGTH	<b>Count Length</b>
		This bit determines whether the counter counts to the compare value and then re initializes itself to the value specified in the Load register, or the counter continues counting past the compare value, the binary roll-over.
	0	Roll-over
	1	Count till compare, then re initialized. If counting up, successful compare occurs when the counter reaches a TMRCMP1 value. If counting down, successful compare occurs when the counter reaches a TMRCMP2 value

TMR Control Registers (TMRCTRL)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	CM			PCS				SCS		ONCE	LENGTH	DIR	Co INIT	OM			
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base + \$6 \$16, \$26, \$36



Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# TMR

## TMR Control Registers (TMRCTRL) Continued

Bits	Name	Description
4	DIR	<b>Count Direction</b>
		This bit selects either the normal count direction up, or the reverse down direction.
		0 Count-up
		1 Count-down
3	CoINIT	<b>Co-Channel Initialization</b>
		This bit enables another counter/timer within the same module to force the re initialization of this counter/timer when it has an active compare event. The Master channel forcing re initialization has MSTR bit set.
		0 Co-Channel counter/timers cannot force a re initialization of this counter/timer
		1 Co-Channel counter/timers can force a re initialization of this counter/timer
2-0	OM	<b>Output Mode</b>
		This bit field determines the mode of operation for the OFLAG output signal.
		000 Assert OFLAG while counter is active
		001 Clear OFLAG output on successful compare
		010 Set OFLAG output on successful compare
		011 Toggle OFLAG output on successful compare
		100 Toggle OFLAG output using alternating compare registers
		101 Set OFLAG on compare, cleared on secondary source input edge
		110 Set OFLAG on compare, cleared on counter rollover
		111 Enable Gated Clock output while counter is active
		<b>Note:</b> Unexpected results may occur if Output mode field is set to use alternating Compare registers (mode 100) and the Count Once (ONCE) bit is set.

TMR Control Registers (TMRCTRL)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	CM			PCS				SCS		ONCE	LENGTH	DIR	Co INIT	OM		
	Write																
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base + \$6 \$16, \$26, \$36

Application: \_\_\_\_\_

Date: \_\_\_\_\_

\_\_\_\_\_

Programmer: \_\_\_\_\_

# TMR

## TMR Status and Control Registers (TMRSCR)

Please see the following page for continuation of this register.

Bits	Name	Description
15	TCF	<b>Timer Compare Flag</b>
		This bit is set when a successful compare occurs. Clear the bit by writing zero to it. TCF asserts every time there is a compare event. (either when counter == cmp1, or counter == cmp2)
14	TCFIE	<b>Timer Compare Flag Interrupt Enable</b>
		When set, the timer compare interrupt is enabled.
13	TOF	<b>Timer Overflow Flag</b>
		This bit is set when the counter rolls over its maximum or minimum value \$FFFF or \$0000, depending on count direction. Clear the bit by writing zero to it.
12	TOFIE	<b>Timer Overflow Flag Interrupt Enable</b>
		When set, this bit enables interrupts when the TOF bit is set.
11	IEF	<b>Input Edge Flag</b>
		This bit is set when a positive input transition occurs on an input selected as a secondary count source while the counter is enabled. Clear the bit by writing zero to it.  Note: Setting the INput Polarity Select (IPS) bit enables the detection of negative input edge transitions. Also, the Control register's secondary count source determines which external input pin is monitored by the detection circuitry.
10	IEFIE	<b>Input Edge Flag Interrupt Enable</b>
		When set, this bit enables interrupts if the IEF bit is set.
9	IPS	<b>Input Polarity Select</b>
		When set, this bit inverts the input signal polarity.

TMR Status/Control Registers (TMRSCR)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE MODE	MSTR	EEOF	VAL	0	OPS	OEN		
	Write													FORCE				
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base + \$7 \$17, \$27, \$37

Application: \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# TMR

## TMR Status and Control Registers (TMRSCR) Continued

Bits	Name	Description
8	<b>INPUT</b>	<b>External Input Signal</b> This <i>read-only</i> bit reflects the current state of the external input pin selected via the secondary count source after application of the IPS bit.
7-6	<b>CAPTURE MODE</b>	<b>Capture Mode</b> This bit field specify the operation of the Capture register as well as the operation of the Input Edge Flag (IEF). The input source is the secondary count source. See <a href="#">Table16-3</a> .
5	<b>MSTR</b>	<b>Master Mode</b> When set, this bit enables the Compare register function output to be broadcast to the other counter/timers in the module.
4	<b>EEOF</b>	<b>Enable External OFLAG Force</b> When set, this bit enables the Compare register from another counter/timer within the same module to force the state of this counter's OFLAG output signal.
3	<b>VAL</b>	<b>Forced OFLAG Value</b> This bit determines the value of the OFLAG output signal when a software triggered FORCE command occurs, i.e. when FORCE = 1.
2	<b>FORCE</b>	<b>Force OFLAG Output</b> This <i>write-only</i> bit forces the current value of the VAL bit to be written to the OFLAG output. This bit is always read as a zero. The VAL and FORCE bits can be written simultaneously in a single write operation. Write to the FORCE bit only if the counter is disabled.
1	<b>OPS</b>	<b>Output Polarity Select</b> This bit determines the polarity of the OFLAG output signal. 0 True polarity 1 Inverted polarity
0	<b>OEN</b>	<b>Output Enable</b> When set, this bit determines the direction of the external pin. 0 The external pin is configured as an input 1 OFLAG output signal will be driven on the external pin. Other timer groups using this external pin as their input will see the driven value. Polarity of the signal is determined by the OPS bit.

TMR Status/Control Registers (TMRSCR)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CAPTURE MODE		MSTR	EEOF	VAL	0	OPS	OEN
	Write														FORCE		
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base + \$7 \$17, \$27, \$37

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# TMR

## TMR Comparator Load Registers 1 (TMRCMPLD1)

Bits	Name	Description
15-0	<b>COMPARATOR LOAD 1</b>	<b>Comparator Load 1</b>
		This read/write register is the Comparator 1 preload value for the TMRCMP1 register of the corresponding channel in a timer module. There are four Timer Comparator1 (TMRCMPLD1) registers.
		TMR0_CMPLD1 (Channel 0 CMPLD1)—Address: TMR_BASE + \$8 TMR1_CMPLD1 (Channel 1 CMPLD1)—Address: TMR_BASE + \$18 TMR2_CMPLD1 (Channel 2 CMPLD1)—Address: TMR_BASE + \$28 TMR3_CMPLD1 (Channel 3 CMPLD1)—Address: TMR_BASE + \$38

<b>TMR Comparator Load Registers 1 (TMRCMPLD1)</b>	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COMPARATOR LOAD 1															
	Write	COMPARATOR LOAD 1															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Base + \$8 \$18, \$28, \$38**

Application: \_\_\_\_\_  
 \_\_\_\_\_

Date: \_\_\_\_\_  
 Programmer: \_\_\_\_\_

# TMR

## TMR Comparator Load Registers 2 (TMRCMPLD2)

Bits	Name	Description
15-0	COMPARATOR LOAD 2	<b>Comparator Load 2</b>
		This read/write register is the Comparator 2 preload value for the TMRCMP2 register of the corresponding channel in a timer module. There are four Timer Comparator 2 (TMRCMPLD2) registers.
		TMR0_CMPLD2 (Channel 0 CMPLD2)—Address: TMR_BASE + \$9 TMR1_CMPLD2 (Channel 1 CMPLD2)—Address: TMR_BASE + \$19 TMR2_CMPLD2 (Channel 2 CMPLD2)—Address: TMR_BASE + \$29 TMR3_CMPLD2 (Channel 3 CMPLD2)—Address: TMR_BASE + \$39

TMR Comparator Load Registers 2 (TMRCMPLD2)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read	COMPARATOR LOAD 2															
	Write	COMPARATOR LOAD 2															
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Base + \$9 \$19, \$29, \$39

Application: \_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

# TMR

## TMR Comparator Status/Control Registers (TMRCOMSCR)

Bits	Name	Description								
7	TCF2EN	<b>Timer Compare 2 Interrupt Enable</b> An interrupt is issued when both this bit and the TCF2 bit are set.								
6	TCF1EN	<b>Timer Compare 1 Interrupt Enable</b> An interrupt is issued when both this bit and the TCF1 bit are set.								
5	TCF2	<b>Timer Compare Flag 2</b> When set, this bit indicates a successful comparison of the timer and TMRCMP2 register has occurred. This sticky bit will remain set until explicitly cleared by writing zero to this bit location.								
4	TCF1	<b>Timer Compare Flag 1</b> When set, this bit indicates a successful comparison of the timer and TMRCMP1 register has occurred. This sticky bit will remain set until explicitly cleared by writing zero to this bit location.								
3-2	CL2	<b>Compare Load Control 2</b> These bits control when TMRCMP2 is preloaded with the value from TMRMPLD2. <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>00</td><td>Never preload</td></tr> <tr><td>01</td><td>Load upon successful compare with the value in TMRCMP1</td></tr> <tr><td>10</td><td>Load upon successful compare with the value in TMRCMP2</td></tr> <tr style="background-color: #cccccc;"><td>11</td><td>Reserved</td></tr> </table>	00	Never preload	01	Load upon successful compare with the value in TMRCMP1	10	Load upon successful compare with the value in TMRCMP2	11	Reserved
00	Never preload									
01	Load upon successful compare with the value in TMRCMP1									
10	Load upon successful compare with the value in TMRCMP2									
11	Reserved									
1-0	CL1	<b>Compare Load Control 1</b> These bits control when TMRCMP1 is preloaded with the value from TMRMPLD1. <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>00</td><td>Never preload</td></tr> <tr><td>01</td><td>Load upon successful compare with the value in TMRCMP1</td></tr> <tr><td>10</td><td>Load upon successful compare with the value in TMRCMP2</td></tr> <tr style="background-color: #cccccc;"><td>11</td><td>Reserved</td></tr> </table>	00	Never preload	01	Load upon successful compare with the value in TMRCMP1	10	Load upon successful compare with the value in TMRCMP2	11	Reserved
00	Never preload									
01	Load upon successful compare with the value in TMRCMP1									
10	Load upon successful compare with the value in TMRCMP2									
11	Reserved									

TMR Comparator Status/Control Registers (TMRCOMSCR)	Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Read	0	0	0	0	0	0	0	0	0	TCF2 EN	TCF1 EN	TCF2	TCF1	CL2		CL1	
	Write																	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

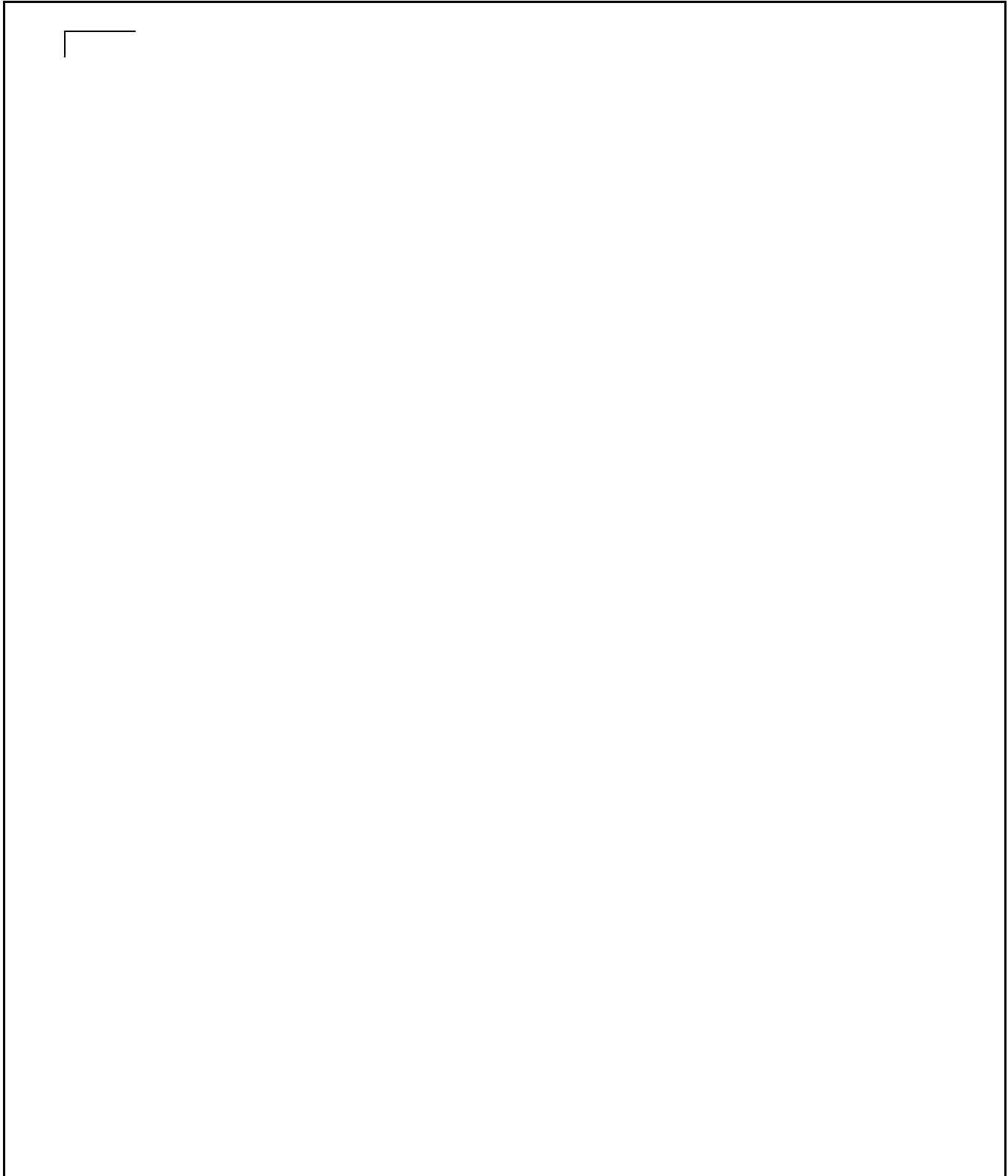
Base + \$A \$1A, \$2A, \$3A

Application: \_\_\_\_\_  
\_\_\_\_\_

Date: \_\_\_\_\_

Programmer: \_\_\_\_\_

Sheet







# INDEX

## Numerics

56800E  
Core 1-4

## A

A/D [A-3](#)  
ACIM [A-3](#)  
ADC [A-3](#)  
ADC STOP bit in ADCR1 [2-16](#)  
ADZCSTAT register [2-39](#)  
Calibration [2-17](#)  
Ceramic capacitors [2-27](#)  
Channel List register [2-35](#)  
Conversion sequence [2-32](#)  
Conversions [2-43](#)  
Converters required for conversion [2-30](#)  
Differential signals [2-25](#)  
Dual ADC configuration [2-43](#)  
EOSI [2-37](#)  
High Limit register [2-38](#)  
HLMTI and LLMTI [2-37](#)  
Internal clock [2-47](#)  
IPBbus Bridge [2-47](#)  
Limit registers [2-41](#)  
Limit Status register [2-39](#)  
Loop Sequential [2-33](#)  
Loop Simultaneous [2-33](#)  
Low Limit register [2-38](#)  
Modes of normal operation [2-9](#)  
Offset register [2-42](#)  
Offset value [2-43](#)  
PSM assertion [2-45](#)  
RDYn [2-37](#)  
Reference source [2-25](#)  
Reference voltage [2-25](#)  
Register  
ADC\_CAL [2-46](#)  
ADCPower [2-43](#)  
ADCTL1 [2-29](#)  
ADCTL2 [2-33](#)  
ADHLMT0-7 [2-42](#)  
ADLLMT0-7 [2-41](#), [2-42](#)  
ADLST1 [2-35](#)  
ADLST2 [2-35](#)  
ADLSTAT [2-39](#)  
ADDFS0-7 [2-42](#)  
ADDFS0-7 [2-42](#)  
ADRSLT0-7 [2-40](#)  
ADSDIS [2-36](#)  
ADSTAT [2-37](#)

ADZCC [2-34](#)  
ADZCSTA [2-39](#)  
AHHLMT0-7 [2-41](#)  
ZCSn [2-40](#)

Result registers [2-40](#), [B-22](#)  
Sample Disable [2-36](#)  
Scan Sequence Control [2-13](#)  
Sequential or simultaneous conversion [2-12](#)  
START conversion [2-30](#)  
Synchronize conversions [2-47](#)  
Triggered Sequential [2-33](#)  
Triggered Simultaneous (default) [2-33](#)

ADC (Analog-to-Digital Converter) [2-1](#)

ADCR [A-3](#)  
ADDLMT [A-3](#)  
ADDR [A-3](#)  
ADHLMT [A-3](#)  
ADLST [A-3](#)  
ADLSTAT [A-3](#)  
ADM [A-3](#)  
ADDFS [A-3](#)  
ADR PD [A-3](#)  
ADRSLT [A-3](#)  
ADSDIS [A-3](#)  
ADSTAT [A-3](#)  
ADZCC [A-3](#)  
ADZCSTAT [A-3](#)  
AGU [A-3](#)  
ALU [A-3](#)  
API [A-3](#)

## B

Baud Rate Generation  
SCI [13-6](#)

BCR [A-4](#)  
BDC [A-4](#)  
BE [A-4](#)  
BFLASH [A-4](#)  
BK [A-4](#)  
BLDC [A-4](#)  
BOTNEG [A-4](#)  
BS [A-4](#)  
BSDL [A-4](#)  
BSR [A-4](#)

## C

CAN [A-4](#)  
CAP [A-4](#)  
CC [A-4](#)  
CEN [A-4](#)  
Ceramic Resonator

---

- Minimize output distortion [5-20](#)
- CFG [A-4](#)
- CGDB [A-4](#)
- CHCNF [A-4](#)
- CKDIVISOR [A-4](#)
- CLKO [A-4](#)
- CLKOSEL [A-4](#)
- CLKOSR [A-4](#)
- CMOS [A-4](#)
- CMP [A-4](#)
- CNT [A-4](#)
- CNTR [A-4](#)
- Codec [A-5](#)
- Computer Operating Properly (COP) [3-3](#)
- COP [A-5](#)
  - After reset [3-8](#)
  - COP\_RST signal [3-8](#)
  - Counter [3-6](#)
  - Counter in Stop mode [3-5](#)
  - Debug mode [3-8](#)
  - Determined time out period [3-6](#)
  - Increased testing [3-5](#)
  - Prescaler [3-7](#)
  - Register
    - COPCTL [3-5](#)
    - COPCTR [3-7](#)
    - COPTO [3-6](#)
  - Stop mode [3-8](#)
  - System reset [3-8](#)
  - Timeout [3-7](#)
  - Timer base [3-8](#)
  - Wait Mode [3-5](#)
  - Wait mode [3-8](#)
  - Write protection [3-6](#)
- COP (Computer Operating Properly) [3-1](#)
- COP/RTI [A-5](#)
- COPCTL [A-5](#)
- COPDIS [A-5](#)
- COPR [A-5](#)
- COPSRV [A-5](#)
- COPTO [A-5](#)
- CPHA [A-5](#)
- CPOL [A-5](#)
- CPU [A-5](#)
- CRC [A-5](#)
- Crystal oscillator [5-10](#)
- CSEN [A-5](#)
- CTRL [A-5](#)
- CTRL PD [A-5](#)
- CWEN [A-5](#)
- CWP [A-5](#)

## D

- DAC [A-5](#)
- DAT [A-5](#)
- Data Frame Format
  - SCI [13-5](#)
- DDA [A-5](#)
- DDR [A-5](#)
- DEC [A-5](#)
  - Decoder Control [12-12](#)
  - Filter Interval [12-15](#)
  - Holding and Initializing Registers [12-6](#)
  - Input Monitor [12-20](#)
  - Lower Initialization [12-20](#)
  - Lower Position Counter [12-18](#)
  - Lower Position Hold [12-19](#)
  - Position Counter [12-5](#)
  - Position Difference Counter [12-6](#), [12-17](#)
  - Position Difference Hold [12-17](#)
  - Positive Vs. Negative Direction [12-5](#)
  - Prescaler for Slow or Fast Speed
    - Measurement [12-7](#)
  - Quadrature Decoder [12-1](#)
  - Register
    - DECCR [12-12](#)
    - FIR [12-15](#)
    - IMR [12-20](#)
    - LIR [12-20](#)
    - LPOS [12-18](#)
    - LPOSH [12-19](#)
    - POSD [12-17](#)
    - POSDH [12-17](#)
    - REV [12-18](#)
    - REXH [12-18](#)
    - UIR [12-19](#)
    - UPOS [12-18](#)
    - UPOSH [12-19](#)
    - WTR [12-16](#)
  - Revolution Counter [12-18](#)
  - Revolution counter [12-6](#)
  - Revolution Hold [12-18](#)
  - Upper Initialization [12-19](#)
  - Upper Position Counter [12-18](#)
  - Upper Position Hold [12-19](#)
  - Watchdog Timeout [12-16](#)
- DFIU [A-5](#)
- DFLASH [A-5](#)
- DIRQ [A-5](#)
- DR [A-6](#)
- DRV [A-6](#)
- DSO [A-6](#)
- DSP [A-6](#)

---

## E

- EDG [A-6](#)
- EE [A-6](#)
- EEOF [A-6](#)
- EM [A-6](#)
- EMI (External Memory Interface) [4-1](#)
- EN [A-6](#)
- ENCR [A-6](#)
- EOSI [A-6](#)
- EOSIE [A-6](#)
- ERASE [A-6](#)
- ERRIE [A-6](#)
- ESR Multi-Layer Ceramic Chip [17-4](#)
- EXTBOOT [A-6](#)
- External Clock Source
  - Recommended method of connecting [5-21](#)
- EXTR [A-6](#)

## F

- FAULT [A-6](#)
- FE [A-6](#)
- FH [A-6](#)
- FIEx [A-6](#)
- FIR [A-6](#)
- Flash
  - Access Error flag [6-29](#)
  - Back door access [6-16](#)
  - Banked registers
    - share register address offset [6-24](#)
  - Block Verified Erased flag [6-30](#)
  - Boot Flash read access [6-4](#)
  - Boot memories [6-5](#)
  - Changing Flash Protection [6-26](#)
  - CMD User Mode [6-30](#)
  - Command Buffer Empty flag [6-29](#)
  - Command Complete flag [6-29](#)
  - Command operation completion [6-12](#)
  - Command-Write sequence [6-11](#)
  - Common register [6-17](#)
  - Configuration field modification
    - reset [6-27](#)
  - Data Flash memories [6-5](#)
  - Data Flash read access [6-4](#)
  - DIV setting [6-9](#)
  - Enable/disable Back Door Access scheme [6-7](#)
  - Enable/disable Security [6-7](#)
  - Erase sensitive information [6-16](#)
  - erased bit [6-5](#)
  - Flash Memory Clock Divider (FMCLKD) [6-18](#)

- Flash Memory Configuration Register (FMCR) [6-19](#)
- FM Configuration Field [6-15](#)
- FM Security feature [6-15](#)
- FMPROT register [6-26](#)
- FMPROTB register [6-27](#)
- Mass Erase
  - unsecured [6-16](#)
- Optional Data registers [6-23](#)
- PRDIV8 setting [6-9](#)
- Program Flash read access [6-4](#)
- Program memories [6-5](#)
- Program/erase algorithms [6-9](#)
- Program/erase protected [6-17](#)
- Programmed bit [6-5](#)
- PROTB\_VALUE [6-27](#)
- Protection and access restriction [6-7](#)
- Protection Boot [6-28](#)
- Protection Register modified after Reset [6-17](#)
- Protection Violation flag [6-29](#)
- Read access [6-4](#)
- Read operation [6-9](#)
- read-while-write operations [6-3](#)
- Register
  - FMCLKD [6-9](#), [6-18](#)
  - FMCMD [6-30](#)
  - FMCR [6-19](#)
  - FMOPT0 [6-23](#)
  - FMOPT1 [6-23](#)
  - FMOPT2 [6-24](#)
  - FMOPTn [6-23](#)
  - FMPROT [6-25](#)
  - FMPROTB [6-27](#)
  - FMSECH [6-21](#)
  - FMSECL [6-21](#)
  - FMUSTAT [6-28](#)
- Resets [6-32](#)
- Security function [6-23](#)
- Security words [6-7](#)
- Smallest memory block [6-4](#)
- Special memory locations [6-6](#)
- Unbanded Register [6-17](#)
- Unbanked FMSECH and FMSECL registers [6-21](#)
- User Mode Valid Commands
  - Erase Verify [6-14](#)
  - Mass Erase [6-14](#)
  - Page Erase [6-14](#)
  - Program [6-14](#)
  - Write operation [6-9](#)
- Flash Memory (FM) [6-1](#)
- FlexCAN
  - Acceptance Masks [7-32](#)

- Auto Power Save
  - optimizing power savings 7-20
- Bit timing 7-15
- Busoff Interrupt 7-38
- CODE 7-7
- Control Register
  - Control0, Control1, and FCTIMER registers 7-28
- DATA 7-7
- End-of-frame (EOF) 7-14
- Error and Status 7-36
- Error Counters 7-40
- Error Interrupt 7-28, 7-38
- FlexCAN disabled 7-26
- FlexCan Stopped 7-27
- Format frames 7-6, 7-8
- Free-Running Timer 7-31
- Identifier (ID) bits 7-8
- Information Processing Time (IPT) 7-16
- Intermission 7-14
- Internal State Machines 7-26
- Interrupt Flag 7-39
- Interrupt Mask 7-39
- LENGTH (Receive) 7-7
- LENGTH (Transmit) 7-7
- Listen-Only mode 7-30
- Low Power Sleep mode 7-25
- Lowest Buffer
  - Transmitted First 7-29
- Maximum Message Buffer 7-32
- Message Buffer (MB) 7-6
- Message buffers 7-12
  - BUSY 7-7
  - EMPTY 7-7
  - extended format frames 7-6, 7-8
  - FULL 7-7
  - handling
    - receive deactivation 7-13
    - transmit deactivation 7-12
  - NOT ACTIVE 7-7
  - overload frames 7-14
  - OVERRUN 7-7
  - receive
    - codes 7-7
  - remote frames 7-13
  - self-received frames 7-11
  - standard format frames 7-8
  - structure 7-6
  - time stamp 7-14
  - transmit
    - codes 7-7
- Operation
  - bit timing configuration 7-16
  - Phase Buffer Segment 1 7-31
  - Phase Buffer Segment 2 7-31
  - Prescaler Divide Factor 7-30
  - Propagation Segment 7-30
  - Receive Buffer 14 7-35
  - Receive Buffer 15 7-35
  - Receive process 7-10
  - Register
    - FC\_ERR\_CNTRS 7-40
    - FCCTL0 7-28
    - FCCTL1 7-30
    - FCIFLAG1 7-39
    - FCIMASK1 7-39
    - FCMAXMB 7-32
    - FCMCR 7-25
    - FCRX14MASK\_H 7-35
    - FCRX14MASK\_L 7-35
    - FCRX15MASK\_H 7-35
    - FCRX15MASK\_L 7-35
    - FCRXGMASK\_H 7-33
    - FCRXGMASK\_L 7-33
    - FCSTATUS 7-36
    - FCTMR 7-31
  - Resynchronization Jump Width 7-30
  - Sample Mode 7-29
  - Start-of-frame (SOF) 7-16
  - Stop Mode for Power Saving 7-18
  - Stop Mode Operation Notes 7-19
  - Time stamp 7-7
  - Timer Synchronization 7-29
  - Transmit process 7-9
  - Two Global Masks 7-32
  - Wake Interrupt 7-38
  - Wake-Up bit enables interrupt generation 7-26
- FlexCAN (FC) 7-1
- FLOLI A-6
- FMODEx A-6
- Fout
  - 8MHz clock 5-12
  - Substitution 5-13
  - System clock 5-12
- FPINx A-6
- Frequency Lock Detector Block 5-18
- FTACKx A-7

## G

- General Purpose Input/Output (GPIO) module 8-3
- GPIO A-7
  - Clocks 8-13
  - Configurations 8-5

---

- Data [8-9](#)
- Data Direction [8-9](#)
- Data Transfers Between I/O Pin and IPBus [8-6](#)
- General Purpose Input/Output [8-1](#)
- Interrupt Assert [8-10](#)
- Interrupt Edge Sensitive [8-12](#)
- Interrupt Enable [8-10](#)
- Interrupt Pending [8-11](#)
- Interrupt Polarity [8-11](#)
- Interrupts [8-13](#)
  - Latched Edge (or level) sensitive hardware [8-13](#)
  - Software interrupt [8-13](#)
- Logic diagram [8-4](#)
- Modes of operation [8-4](#)
- Peripheral Controlled Mode [8-4](#)
- Peripheral Enable [8-9](#)
- Pull-Up [8-8](#)
- Push/Pull Output Mode Control [8-12](#)
- Raw Data [8-13](#)
- Register
  - DDR [8-9](#)
  - DR [8-9](#)
  - IAR [8-10](#)
  - IENR [8-10](#)
  - IESR [8-12](#)
  - IPOLR [8-11](#)
  - IPR [8-11](#)
  - PER [8-9](#)
  - PPMODE [8-12](#)
  - PUR [8-8](#)
  - RAWDATA [8-13](#)
- Resets [8-13](#)
- GPIO Mode [8-4](#)
- GPR [A-7](#)

## H

- Harvard architecture [A-7](#)
- HLMTI [A-7](#)
- HLMTIE [A-7](#)
- HOLD [A-7](#)
- HOME [A-7](#)

## I

- I/O [A-7](#)
- IA [A-7](#)
- IC [A-7](#)
- IE [A-7](#)
- IEE [A-7](#)
- IEF [A-7](#)
- IEFIE [A-7](#)

- IENR [A-7](#)
- IES [A-7](#)
- IMR [A-7](#)
- INDEP [A-7](#)
- INDEX [A-7](#)
- INPUT [A-7](#)
- Internal Crystal Oscillator
  - Ceramic resonator [5-20](#)
- Internal regulators [17-4](#)
- Internal Relaxation Oscillator [5-10](#)
- IP [A-7](#)
- IPBus [A-7](#)
- IPE [A-7](#)
- IPOL [A-7](#)
- IPOLR [A-7](#)
- IPR [A-7](#)
- IPS [A-8](#)
- IRQ [A-8](#)
- IS [A-8](#)

## J

- JTAG [A-8](#)
  - Block Diagram [9-4](#)
  - BYPASS instruction [9-5](#)
  - Capture Data [9-9](#)
  - Capture Instruction [9-10](#)
  - Clocks [9-12](#)
    - TCK [9-12](#)
  - Exit1 Data [9-9](#)
  - Exit1 Instruction [9-10](#)
  - Exit2 Data [9-10](#)
  - Exit2 Instruction [9-11](#)
  - External Test (EXTEST) instruction [9-5](#)
  - Flash Erase [9-6](#)
  - IDCODE instruction [9-6](#)
  - Joint Test Action Group Port [9-1](#)
  - Operation TAP Controller [9-8](#)
  - Pause Data [9-9](#)
  - Pause Instruction [9-10](#)
  - Register
    - Boundary Scan [9-12](#)
  - Resets [9-13](#)
    - TRST [9-13](#)
  - Run-Test-Idle [9-9](#)
  - SAMPLE/PRELOAD instruction [9-6](#)
  - Select-Data [9-9](#)
  - Select-Instruction [9-9](#)
  - Shift Data [9-9](#)
  - Shift Instruction [9-10](#)
  - Signal summaries [9-11](#)
  - TAP Block Diagram [9-4](#)

---

- TAP Controller [9-7](#)
- Test-Logic-Reset [9-8](#)
- TLM\_SEL instruction [9-6](#)
- TRST [9-11](#)
- Update Data [9-10](#)
- Update Instruction [9-11](#)

JTAGBR [A-8](#)  
JTAGIR [A-8](#)

## L

LCK [A-8](#)

- Frequency Lock Detector [5-18](#)
  - set [5-18](#)

LDOK [A-8](#)  
LIR [A-8](#)  
LLMTI [A-8](#)  
LLMTIE [A-8](#)  
LOAD [A-8](#)  
LOCI [A-8](#)  
LOCIE [A-8](#)  
Lock Time

- Critical design parameters [5-17](#)
- Parametric influences [5-18](#)

LOLI [A-8](#)  
LOOP [A-8](#)  
Loss of Lock Interrupt

- cleared [5-27](#)

Loss of Reference Clock Interrupt

- cleared [5-27](#)

LPOS [A-8](#)  
LPOSH [A-8](#)  
LSB [A-8](#)  
LSH\_ID [A-8](#)  
LVI Interrupt Service Routines [10-8](#)  
LVIE [A-8](#)  
LVIS [A-8](#)

## M

MA [A-8](#)  
MAC [A-8](#)  
MAS [A-8](#)  
MB [A-8](#)  
MCU [A-8](#)  
MHz [A-8](#)  
MIPS [A-8](#)  
MISO [A-8](#)  
MLCC [17-4](#)  
MODF [A-8](#)  
MODFEN [A-8](#)  
MOSI [A-8](#)  
MSB [A-8](#)

MSH\_ID [A-9](#)  
MSTR [A-9](#)  
MUX [A-9](#)

## N

NL [A-9](#)  
Normal Mode

- GPIO [8-4](#)

## O

OCCS [5-1, A-9](#)

- Clock multiplexer [5-7](#)
- Clock shutdown [5-30](#)
- Crystal parameters [5-20](#)
- Minimize distortion [5-20](#)
- Prescaler clock [5-7](#)
- Stabilization time [5-20](#)
- Usable registers [5-22](#)

OCCS (On-Chip Clock Synthesis) [5-1](#)  
OCNTR [A-9](#)  
OCR\_DIS [17-4, 17-5](#)  
OEN [A-9](#)  
OMAC [A-9](#)  
OMR [A-9](#)  
On-board regulators [17-3](#)  
OnCE [A-9](#)

- Enhanced On-Chip Emulation Module [1-9](#)

On-Chip Clock Synthesis (OCCS) [5-4](#)  
OP [A-9](#)  
OPABDR [A-9](#)  
OR [A-9](#)  
Oscillator

- Frequency variance [5-10](#)

OSHR [A-9](#)  
OVRF [A-9](#)

## P

PAB [A-9](#)  
PD [A-9](#)  
PDB [A-9](#)  
PE [A-9](#)  
PER [A-9](#)  
PF [A-9](#)  
PFLASH [A-9](#)  
PGDB [A-9](#)  
PLL [A-9](#)

- Clock Source Status [5-29](#)
- Critical design parameters [5-17](#)
- Divide-by ratio [5-7](#)
- Frequency Lock Detector Block [5-18](#)

---

- Interrupt enable [5-24](#)
- Legal combinations of settings [5-26](#)
- Lock Time [5-17](#)
- Lock time [5-7](#)
- Loss of reference clock [5-24](#)
- Loss of Reference Clock Detector [5-19](#)
- Output frequency [5-27](#)
- Postscaler [5-7](#)
- Power-Down [5-25](#)
- Power-down status [5-28](#)
- Powered down state [5-17](#)
- Prescaler [5-7](#)
- Recommended Range of Operation [5-12](#)
- Register
  - OSCTL [5-30](#), [5-31](#)
  - PLLCR [5-23](#)
  - PLLDB [5-26](#)
  - PLLSR [5-27](#)
  - SHUTDOWN [5-29](#)
- VCO output clock [5-18](#)
- PLLCID [A-9](#)
- PLLCOD [A-9](#)
- PLLCR [A-9](#)
- PLLDB [A-9](#)
- PLLPDN [A-9](#)
- PLR [A-9](#)
- PMCCR [A-9](#)
- PMCFG [A-9](#)
- PMCNT [A-9](#)
- PMCTL [A-9](#), [A-10](#)
- PMDEADTM [A-9](#)
- PMDISMAP [A-10](#)
- PMFCTL [A-10](#)
- PMFSA [A-10](#)
- PMOUT [A-10](#)
- PMPORT [A-10](#)
- POL [A-10](#)
- POR [A-10](#)
  - Power-On Reset [10-4](#)
- Power Supervisor
  - Low Voltage Interrupt [10-7](#)
  - LVI Interrupt Service Routines [10-8](#)
  - Non-Sticky 2.2V Low Voltage Interrupt [10-8](#)
  - Non-Sticky 2.7V Low Voltage Interrupt [10-8](#)
  - POR versus low voltage detect circuits [10-5](#)
  - Register
    - LVICTLR [10-6](#)
    - LVISR [10-7](#)
  - Sticky 2.2V Low Voltage Interrupt [10-7](#)
  - Sticky 2.7V Low Voltage Interrupt [10-7](#)
- Power Supervisor Control Register [10-6](#)
- Power-On Reset (POR) [10-4](#)
- PRAM [A-10](#)
- PROG [A-10](#)
- PS (Power Supervisor) [10-1](#)
- PT [A-10](#)
- PTM [A-10](#)
- PUR [A-10](#)
- PWD [A-10](#)
- PWM [A-10](#)
  - Asymmetric PWM Output [11-17](#)
  - Automatic Deadtime Correction [11-16](#)
  - Block Diagram [11-4](#)
  - Channel Control Register [11-44](#)
  - Complementary Channel Operation [11-8](#)
  - Configuration [11-3](#)
  - Configure Register [11-42](#)
  - Control Register [11-32](#)
  - Counter Modulo Register [11-39](#)
  - Counter Register [11-39](#)
  - Deadtime Generators [11-9](#)
  - Deadtime Register [11-40](#)
  - Determining a PWM period [11-5](#)
  - Disable Mapping Registers [11-41](#)
  - Edge Detect State Machine [12-7](#)
  - Edge-Aligned operation [11-6](#)
  - Fault Control Register [11-36](#)
  - Fault Flags [11-48](#)
  - Fault Protection [11-27](#)
  - Fault Status and Acknowledge Register [11-36](#)
  - Generator Loading [11-22](#)
  - Glitch Filter [12-7](#)
  - HOME signal transition [12-5](#)
  - Independent Channel Operation [11-8](#)
  - INDEX puls [12-5](#)
  - INDEX signal transition [12-5](#)
  - Internal Correction Control Register [11-46](#)
  - Operating Modes [11-29](#), [12-8](#)
  - Output Control Enable [11-38](#)
  - Output Control Register [11-37](#)
  - Output Polarity [11-18](#)
  - PHASEA signal [12-5](#)
  - PHASEB signal [12-5](#)
  - Pin Descriptions [11-30](#), [12-8](#)
  - Port Register [11-46](#)
  - Pulse Accumulator Functionality [12-7](#)
  - Register
    - PMCCR [11-44](#)
    - PMCFG [11-42](#)
    - PMCNT [11-39](#)
    - PMCTL [11-32](#)
    - PMDEADTM [11-40](#)
    - PMDISMAP1 [11-41](#)
    - PMDISMAP1-2 [11-41](#)

---

- PMDISMAP2 [11-41](#)
- PMFCTL [11-36](#)
- PMFSA [11-36](#)
- PMICCR [11-46](#)
- PMOUT [11-37](#)
- PMPORT [11-46](#)
- PWMCM [11-39](#)
- PWMVAL0–5 [11-40](#)
- Reload Flag [11-48](#)
- Software-triggered event [12-5](#)
- Top/Bottom Deadtime Correction [11-11](#)
- Up-counter [11-6](#)
- Value Registers [11-40](#)
- Watchdog Timer [12-8](#)
- PWM (Pulse Width Modulator) [11-1](#)
- PWMEN [A-10](#)
- PWMF [A-10](#)
- PWMRIE [A-10](#)
- PWMVAL [A-10](#)

## Q

- QE [A-10](#)
- Quad Timer (TMR) [16-3](#)

## R

- RAF [A-10](#)
- RAM [A-10](#)
- RDRF [A-10](#)
- RE [A-10](#)
- REIE [A-10](#)
- Relaxation Oscillator
  - Clock Choices [5-9](#)
  - Switching to the Crystal Oscillator Clock [5-11](#)
- REV [A-10](#)
- REXH [A-10](#)
- RIDLE [A-10](#)
- RIE [A-10](#)
- ROM [A-10](#)
- RPD [A-10](#)
- RSRC [A-10](#)
- RWU [A-11](#)

## S

- SA [A-11](#)
- SBK [A-11](#)
- SBO [A-11](#)
- SBR [A-11](#)
- SCI [A-11](#)
  - Baud Rate Generation [13-6](#)
  - Baud Rate Register [13-19](#)

- Baud rate tolerance [13-13](#)
- Block Diagram [13-4](#)
- Control Register [13-20](#)
- Data Frame Format [13-5](#)
- Data Register [13-25](#)
- Framing Errors [13-13](#)
- Functional Description [13-4](#)
- Interrupt Sources [13-26](#)
- Loop Operation [13-17](#)
- Receive Error Interrupt [13-27](#)
- Receiver Full Interrupt [13-26](#)
- Register
  - SCIBR [13-19](#)
  - SCICR [13-20](#)
  - SCIDR [13-25](#)
  - SCISR [13-23](#)
- Single-Wire Operation [13-16](#)
- START bit [13-8](#)
- Status Register [13-23](#)
- STOP bit [13-8](#)
- Transmission [13-8](#)
- Transmitter Block Diagram [13-7](#)
- Transmitter Empty Interrupt [13-26](#)
- Transmitter Idle Interrupt [13-26](#)
- SCI (Serial Communications Interface) [13-1](#)
- SCIBR [A-11](#)
- SCICR [A-11](#)
- SCIDR [A-11](#)
- SCISR [A-11](#)
- SCLK [A-11](#)
- Sclock [7-30](#)
- SCR [A-11](#)
- SD [A-11](#)
- Serial Communications Interface (SCI) [13-3](#)
- SIM [A-11](#)
- SP [A-11](#)
- SPDRR [A-11](#)
- SPDSR [A-11](#)
- SPDTR [A-11](#)
- SPI [A-11](#)
  - Block Diagram [14-4](#)
  - Clock Phase and Polarity Controls [14-9](#)
  - Data Receive Register [14-25](#)
  - Data Shift Ordering [14-9](#)
  - Data Size and Control Register [14-23](#)
  - Data Transmission Length [14-9](#)
  - Data Transmit Register [14-25](#)
  - Error Conditions [14-15](#)
  - Master In/Slave Out (MISO) [14-7](#)
  - Master Mode [14-5](#)
  - Master Out/Slave In (MOSI) [14-8](#)
  - Mode Fault Error [14-17](#)



---

Operation Modes [14-4](#)  
Overflow Error [14-15](#)  
Pin Descriptions [14-7](#)  
Register  
    SPDRR [14-25](#)  
    SPDSR [14-23](#)  
    SPDTR [14-25](#)  
    SPSCR [14-19](#)  
Serial Clock (SCLK) [14-8](#)  
Slave Mode [14-6](#)  
Slave Select SS [14-8](#)  
Status and Control Register [14-19](#)  
Transmission Data [14-13](#)  
Transmission Format When CPHA = 0 [14-10](#)  
Transmission Format When CPHA = 1 [14-11](#)  
Transmission Formats [14-9](#)  
Transmission Initiation Latency [14-12](#)  
Wired OR Mode [14-6](#)  
SPI (Serial Peripheral Interface) [14-1](#)  
SPMSTR [A-11](#)  
SPRF [A-11](#)  
SPRIE [A-11](#)  
SPSCR [A-11](#)  
SPTE [A-11](#)  
SR [A-11](#)  
SRM [A-11](#)  
SS [A-11](#)  
SSI [A-11](#)  
Stop Mode Operation  
    Notes [7-19](#)  
SWAI [A-11](#)  
Switching to crystal oscillator [5-11](#)  
Switching to relaxation oscillator [5-11](#)  
SYS\_CNTL [A-12](#)  
SYS\_STS [A-12](#)  
System Bus Controller (SBC) [1-9](#)

## T

TAP [A-12](#)  
TAP Block Diagram [9-4](#)  
TCE [A-12](#)  
TCF [A-12](#)  
TCFIE [A-12](#)  
TCSR [A-12](#)  
TDRE [A-12](#)  
TE [A-12](#)  
TEIE [A-12](#)  
Temperature Sensor module [15-3](#)  
TERASEL [A-12](#)  
Test Clock Input pin (TCK) [9-11](#)  
Test Data Input pin (TDI) [9-11](#)

Test Data Output pin (TDO) [9-11](#)  
Test Mode Select Input pin (TMS) [9-11](#)  
Test Reset/Debug Event pin [9-11](#)  
TESTR [A-12](#)  
TIDLE [A-12](#)  
TIIE [A-12](#)  
TIRQ [A-12](#)  
TM [A-12](#)  
TMR  
    Block Diagram [16-4](#)  
    Capture Register Usage [16-7](#)  
    Cascade Count Mode [16-14](#)  
    Compare Preload Registers [16-6](#)  
    Compare Registers Usage [16-5](#)  
    Count Mode [16-8](#)  
    Edge Count Mode [16-9](#)  
    Fixed Frequency PWM Mode [16-17](#)  
    Gated Count Mode [16-10](#)  
    One-Shot Mode [16-13](#)  
    Operation Modes [16-7](#)  
    Pulse Output Mode [16-15](#)  
    Quadrature Count Mode [16-10](#)  
    Register  
        CTLR [16-19](#)  
        TMRCAP [16-23](#)  
        TMRCMP1 [16-22](#)  
        TMRCMP2 [16-23](#)  
        TMRCMPLD1 [16-31](#)  
        TMRCMPLD2 [16-31](#)  
        TMRCNTR [16-25](#)  
        TMRCOMSCR [16-19, 16-32](#)  
        TMRCTRL [16-25](#)  
        TMRHOLD [16-24](#)  
        TMRLOAD [16-24](#)  
        TMRSCR [16-19, 16-29](#)  
    Signed Count Mode [16-11](#)  
    Stop Mode [16-7](#)  
    Timer Capture Registers [16-23](#)  
    Timer Comparator Load Registers 1 [16-31](#)  
    Timer Comparator Load Registers 2 [16-31](#)  
    Timer Comparator Status and Control  
        Registers [16-32](#)  
    Timer Compare Registers 1 [16-22](#)  
    Timer Compare Registers 2 [16-23](#)  
    Timer Control Registers [16-25](#)  
    Timer Hold Registers [16-24](#)  
    Timer Load Registers [16-24](#)  
    Timer Status/Control Registers [16-29](#)  
    Triggered Count Mode [16-12](#)  
    Variable Frequency PWM Mode [16-17](#)  
TMR (Quad Timer) [16-1](#)  
TMR PD [A-12](#)

---

TNVHL [A-12](#)  
TO [A-12](#)  
TOFIE [A-12](#)  
TRIM  
    Internal Relaxation Oscillator [5-32](#)  
TSENSOR  
    ADC [15-3](#), [15-7](#)  
    Block Diagram [15-4](#)  
    Constant current source [15-4](#)  
    Inverting amplifier [15-4](#)  
    Monotonic [15-3](#)  
    Operating Modes [15-5](#)  
    Pin Descriptions [15-6](#)  
    Register  
        TSSENSOR\_CTRL [15-6](#)  
    Reset [15-7](#)  
    Temperature Sensor Control Register [15-6](#)  
TSENSOR (Temperature Sensor System) [15-1](#)

## U

UIR [A-13](#)  
UPOS [A-13](#)  
UPOSH [A-13](#)

## V

Vcap1 [17-5](#)  
Vcap2 [17-5](#)  
Vcap3 [17-5](#)  
Vcap4 [17-5](#)  
VDD [A-13](#)  
VDDA [A-13](#)  
VEL [A-13](#)  
VELH [A-13](#)  
VIN [17-5](#)  
VLMODE [A-13](#)  
VOUT [17-5](#)  
VREF [A-13](#)  
VREFH [2-25](#)  
VREFLO [2-25](#)  
VREFMID [2-25](#)  
VREFN [2-25](#)  
VREFP [2-25](#)  
VREG  
    Block Diagram [17-3](#)  
    OCR\_DIS [17-4](#)  
    Regulators [17-4](#)  
VREG (Voltage Regulator) [17-1](#)  
VRM [A-13](#)  
VSS [A-13](#)  
VSSA [A-13](#)

## W

WAKE [A-13](#)  
WDE [A-13](#)  
WP [A-13](#)  
WSPM [A-13](#)  
WSX [A-13](#)  
WTR [A-13](#)  
WWW [A-13](#)

## X

XDB2 [A-13](#)  
XE [A-13](#)  
XIE [A-13](#)  
XIRQ [A-13](#)  
XNE [A-13](#)  
XRAM [A-13](#)

## Y

YE [A-13](#)

## Z

ZCI [2-37](#), [A-13](#)  
ZCIE [A-13](#)  
ZCS [A-13](#)



## **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.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku,  
Tokyo 153-0064, Japan  
0120 191014 or +81 3 5437 9125  
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 2666 8080  
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

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

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

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



Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. This product incorporates SuperFlash® technology licensed from SST.

© Freescale Semiconductor, Inc. 2005. All rights reserved.

MC56F8300UM  
Rev. 10  
10/2007