

MAX32600

User's Guide
February 2016

Contents

1	Disclaimer and Revision History	2
1.1	Revision Information	2
2	Introduction	4
2.1	Overview	4
2.2	Core and Architecture	6
2.2.1	Core Parameters	6
2.2.2	Generic Memory Map	8
2.2.3	AHB Buses	9
2.2.4	APB Buses	9
2.2.5	Nested Vectored Interrupt Controller (NVIC)	9
2.2.6	ARM Debug	10
2.3	Power Supplies and Modes	10
2.3.1	Digital Supply Voltages	10
2.3.2	Analog Supply Voltage	10
2.3.3	Onboard Core Voltage Regulator	11
2.3.4	Onboard VUSB Voltage Regulator and Automatic Power Switching	11
2.3.5	VRTC Power Supply	11
2.3.6	Power Management Modes	11
2.3.7	Power Supply Monitoring	11
2.4	Clock Inputs	12
2.4.1	External High Frequency Crystal	12
2.4.2	32kHz Crystal Oscillator	12
2.4.3	48MHz USB Clock PLL	12
2.4.4	Internal 24MHz Trimmed Relaxation Oscillator	12
2.4.5	Cryptographic Internal Oscillator	13

2.5	Memory	13
2.5.1	Internal Flash Program Memory	13
2.5.2	2 KB Instruction Cache	13
2.5.3	Internal Data SRAM	14
2.5.4	Peripheral Management Unit (PMU)	14
2.5.5	Flash Information Block	15
2.5.6	Flash Memory Controller	15
2.6	Analog Peripherals	16
2.6.1	16-Bit ADC with PGA	16
2.6.2	ADC/DAC Internal/External Reference and Programmable Output Buffers	16
2.6.3	12-Bit Voltage Output DACs	16
2.6.4	8-Bit Voltage Output DACs	17
2.6.5	Uncommitted Op Amps with Comparator Mode	17
2.6.6	Uncommitted SPST Analog Switches	17
2.6.7	Temperature Sensor	17
2.7	Digital Peripherals	17
2.7.1	GPIO Pins w/Interrupt and Wakeup Capability	17
2.7.2	32-Bit Timer/Counters	18
2.7.3	Watchdog Timers	18
2.7.4	32-Bit Real Time Clock with Time of Day Alarm	19
2.7.5	SPI (three instances)	19
2.7.6	I ² C	19
2.7.7	USB 2.0 Device Slave with Integrated Transceiver	20
2.7.8	LCD Controller	20
2.8	Security Features	21
2.8.1	Trust Protection Unit (TPU)	21

2.8.2	AES Cryptographic Engine	21
2.8.3	Battery-Backed AES Secure Key Storage	21
2.8.4	Modular Arithmetic Accelerator (MAA)	22
2.8.5	CRC Hardware Block with CRC16 and CRC32	22
2.8.6	Code Scrambling	22
3	Memory, Register Mapping, and Access	23
3.1	Memory, Register Mapping, and Access Overview	23
3.2	Standard Memory Regions	24
3.2.1	Code Space	24
3.2.2	SRAM Space	24
3.2.3	Peripheral Space	25
3.2.4	External RAM Space	26
3.2.5	External Device Space	26
3.2.6	System Area (Private Peripheral Bus)	26
3.2.7	System Area (Vendor Defined)	26
3.3	Device Memory Instances	26
3.3.1	Main Program Flash Memory	27
3.3.2	Instruction Cache Memory	27
3.3.3	Information Block Flash Memory	27
3.3.4	System SRAM	27
3.3.5	AES Key and Working Space Memory	27
3.3.6	Modular Arithmetic Accelerator (MAA) Key and Working Space Memory	27
3.3.7	TPU Memory Secure Key Storage Area	27
3.4	AHB Bus Matrix and AHB Bus Interfaces	28
3.4.1	Core AHB Interface - I-Code	28
3.4.2	Core AHB Interface - D-Code	28

3.4.3	Core AHB Interface - System	28
3.4.4	AHB Master - Peripheral Management Unit (PMU)	28
3.4.5	AHB Master - USB Endpoint Buffer Manager	28
4	System Configuration and Management	29
4.1	Power Ecosystem and Operating Modes	29
4.1.1	Power Ecosystem	29
4.1.2	Low Power Modes (LP0: STOP and LP1: STANDBY)	31
4.1.2.1	Low Power Mode 0 (LP0: STOP)	31
4.1.2.2	Low Power Mode 1 (LP1: STANDBY)	31
4.1.2.3	Entering LP0: STOP or LP1: STANDBY	32
4.1.2.4	Wakeup Events from LP0: STOP and LP1: STANDBY	33
4.1.3	Low Power Modes (LP2: PMU and LP3: RUN)	34
4.1.3.1	Low Power Mode 2 (LP2: Peripheral Management Unit)	35
4.1.3.2	Low Power Mode 3 (LP3: RUN)	35
4.1.4	Power State Matrix Control Options	35
4.1.5	Power Domains	37
4.1.6	Power Manager	38
4.1.7	Power Sequencer	38
4.1.7.1	Power Mode Transitioning to Low Power Modes	38
4.1.7.2	Supply Voltage Monitoring During LP0: STOP and LP1: STANDBY	38
4.1.7.3	SVM Periodic Monitoring	39
4.1.7.4	First Boot Power Up	39
4.1.7.5	Brownout Detector	40
4.1.8	Trickle Charger	40
4.1.8.1	Trickle Charger Configuration	40
4.1.9	Low-Dropout Regulators (LDO)	41

4.1.9.1	1.8V LDO	41
4.1.9.2	3.3V USB LDO	41
4.1.10	Reset Pins	42
4.1.11	Power Pins	42
4.1.12	Registers (PWRMAN)	43
4.1.12.1	Module PWRMAN Registers	43
4.1.13	Registers (PWRSEQ)	70
4.1.13.1	Module PWRSEQ Registers	70
4.2	Interrupt Vector Table	100
4.3	Resets and Reset Sources	102
4.3.1	System Reset	102
4.3.2	Power-On Reset	103
4.3.3	RTC POR	103
4.4	Registers (IOMAN)	103
4.4.1	Module IOMAN Registers	103
4.4.1.1	IOMAN_WUD_REQ0	105
4.4.1.2	IOMAN_WUD_REQ1	106
4.4.1.3	IOMAN_WUD_ACK0	107
4.4.1.4	IOMAN_WUD_ACK1	108
4.4.1.5	IOMAN_ALI_REQ0	109
4.4.1.6	IOMAN_ALI_REQ1	110
4.4.1.7	IOMAN_ALI_ACK0	111
4.4.1.8	IOMAN_ALI_ACK1	112
4.4.1.9	IOMAN_SPI0_REQ	113
4.4.1.10	IOMAN_SPI0_ACK	116
4.4.1.11	IOMAN_SPI1_REQ	119

4.4.1.12	IOMAN_SPI1_ACK	122
4.4.1.13	IOMAN_SPI2_REQ	124
4.4.1.14	IOMAN_SPI2_ACK	127
4.4.1.15	IOMAN_UART0_REQ	130
4.4.1.16	IOMAN_UART0_ACK	131
4.4.1.17	IOMAN_UART1_REQ	132
4.4.1.18	IOMAN_UART1_ACK	133
4.4.1.19	IOMAN_I2CM0_REQ	134
4.4.1.20	IOMAN_I2CM0_ACK	135
4.4.1.21	IOMAN_I2CS0_REQ	136
4.4.1.22	IOMAN_I2CS0_ACK	136
4.4.1.23	IOMAN_LCD_COM_REQ	137
4.4.1.24	IOMAN_LCD_COM_ACK	137
4.4.1.25	IOMAN_LCD_SEG_REQ0	137
4.4.1.26	IOMAN_LCD_SEG_REQ1	145
4.4.1.27	IOMAN_LCD_SEG_ACK0	147
4.4.1.28	IOMAN_LCD_SEG_ACK1	155
4.4.1.29	IOMAN_CRNT_REQ	157
4.4.1.30	IOMAN_CRNT_ACK	158
4.4.1.31	IOMAN_CRNT_MODE	160
4.4.1.32	IOMAN_ALI_CONNECT0	161
4.4.1.33	IOMAN_ALI_CONNECT1	161
4.4.1.34	IOMAN_I2CM1_REQ	161
4.4.1.35	IOMAN_I2CM1_ACK	162
4.4.1.36	IOMAN_PADX_CONTROL	162

5 Pin Configurations, Packages, and Special Function Multiplexing

165

5.1	Pin Layout	165
5.2	Pin Function Mapping	166
5.2.1	Compact Package GPIO Mapping	166
5.2.2	Standard Package GPIO Mapping	168
5.3	General-Purpose I/O	173
5.4	GPIO Pins and Peripheral Mode Functions	175
5.5	Registers (GPIO)	180
5.5.1	Module GPIO Registers	180
5.5.1.1	GPIO_FREE_Pn	183
5.5.1.2	GPIO_OUT_MODE_Pn	184
5.5.1.3	GPIO_OUT_VAL_Pn	185
5.5.1.4	GPIO_FUNC_SEL_Pn	185
5.5.1.5	GPIO_IN_MODE_Pn	189
5.5.1.6	GPIO_IN_VAL_Pn	190
5.5.1.7	GPIO_INT_MODE_Pn	191
5.5.1.8	GPIO_INTFL_Pn	191
5.5.1.9	GPIO_INTEN_Pn	192
6	Peripheral Management Unit (PMU)	193
6.1	Overview	193
6.2	PMU Operation	195
6.2.1	PMU Channel Setup	195
6.2.2	PMU Channel Arbitration	195
6.3	PMU Programming Details	196
6.3.1	PMU Op Code: MOVE (0x00)	196
6.3.2	PMU Op Code: WRITE (0x01)	197
6.3.3	PMU Op Code: WAIT (0x02)	198

6.3.4	PMU Op Code: JUMP (0x03)	202
6.3.5	PMU Op Code: LOOP (0x04)	203
6.3.6	PMU Op Code: POLL (0x05)	204
6.3.7	PMU Op Code: BRANCH (0x06)	205
6.3.8	PMU Op Code: TRANSFER (0x07)	206
6.4	Registers (PMU)	208
6.4.1	Module PMU Registers	208
6.4.1.1	PMUn_DSCADR	210
6.4.1.2	PMUn_CFG	211
6.4.1.3	PMUn_LOOP	215
6.4.1.4	PMUn_OP	215
6.4.1.5	PMUn_DSC1	216
6.4.1.6	PMUn_DSC2	216
6.4.1.7	PMUn_DSC3	216
6.4.1.8	PMUn_DSC4	217
7	Communication Peripherals	218
7.1	I ² C	218
7.1.1	I ² C Overview	218
7.1.2	I ² C Features	218
7.1.3	I ² C Port and Pin Configurations	219
7.1.3.1	Compact Layout (7mm x 7mm) Configuration	219
7.1.3.2	Standard Layout (12mm x 12mm) Configuration	221
7.1.4	I ² C Master Operation	224
7.1.5	Protocol	225
7.1.6	Peripheral Clock Selection and Clock Gating	229
7.1.6.1	Peripheral Clock Frequency Selection	230

7.1.7	Communication and Data Transfer	231
7.1.7.1	FIFO-Based I ² C Master	231
7.1.8	Registers (I ² CM)	232
7.1.8.1	Module I ² CM Registers	232
7.1.9	Registers (I ² CS)	243
7.1.9.1	Module I ² CS Registers	243
7.2	SPI	258
7.2.1	Overview	258
7.2.2	SPI Port and Pin Configurations	259
7.2.2.1	Compact Layout (7mm x 7mm) Configuration	260
7.2.2.2	Standard Layout (12mm x 12mm) Configuration	262
7.2.3	Clock Selection and Configuration	264
7.2.3.1	Clock Gating	265
7.2.4	Configuration Modes Overview	265
7.2.4.1	Static Configuration	266
7.2.4.2	Dynamic Configuration	266
7.2.5	SPI Fast Mode	269
7.2.6	Communication and Data Transfer	269
7.2.7	Interrupts	270
7.2.8	SPI: FIFOs	270
7.2.9	Registers (SPI)	271
7.2.9.1	Module SPI Registers	271
7.3	UART	285
7.3.1	Overview	285
7.3.2	UART Port and Pin Configurations	285
7.3.2.1	Compact Layout Configuration	286

7.3.2.2	Standard Layout Configuration	286
7.3.3	Port Register Blocks	287
7.3.4	UART Clock Selection and Clock Gating	288
7.3.5	Format and Baud Rate Selection	288
7.3.6	Transferring and Receiving Data	290
7.3.7	UART: Interrupts	290
7.3.8	Hardware Flow Control	290
7.3.8.1	Clear to Send	290
7.3.8.2	Ready to Send	291
7.3.9	Registers (UART)	291
7.3.9.1	Module UART Registers	291
7.4	USB Device Interface	302
7.4.1	Overview	302
7.4.2	Operation	303
7.4.2.1	USB Reset Definitions	303
7.4.3	USB Endpoints	304
7.4.3.1	Endpoint Control Register	305
7.4.3.2	Endpoint Buffer Descriptor	306
7.4.4	Registers (USB)	307
7.4.4.1	Module USB Registers	307
8	Analog Front End	333
8.1	AFE Overview	333
8.1.1	AFE Analog Function Blocks	334
8.2	AFE Reconfiguration Matrix	335
8.2.1	AFE Reconfiguration Matrix Overview	335
8.2.1.1	DAC	343

8.2.1.2	Op Amps	343
8.2.1.3	Comparators	344
8.2.2	Registers (AFE)	347
8.2.2.1	Module AFE Registers	347
8.3	ADC	398
8.3.1	ADC Overview	398
8.3.2	ADC Architecture	398
8.3.3	ADC Operation	400
8.3.4	ADC Configuration	401
8.3.4.1	Peripheral Clock Configuration	402
8.3.4.2	Reference Voltage Configuration	404
8.3.4.3	Input Modes	404
8.3.4.4	Input Multiplexer	405
8.3.4.5	Scan Modes	408
8.3.4.6	Interrupts	409
8.3.4.7	Programmable Gain Amplifier	410
8.3.5	Sample Rate Calculation	412
8.3.5.1	Decimation Filter Modes	413
8.3.5.2	Register Abbreviations and Definitions	413
8.3.5.3	ADC Mode Explanations	415
8.3.5.4	Mode: PGA Enabled, High Performance, no Scan, with no Burst	417
8.3.5.5	Mode: PGA Enabled, High Performance, No Scan, with Burst and no decimation	417
8.3.5.6	Mode: PGA Enabled, High Performance, No Scan, with Burst and decimation, averaged data output to FIFO	418
8.3.5.7	Mode: PGA Enabled, High Performance, Scan, with no Burst	419
8.3.5.8	Mode: PGA Enabled, Low-Power, no Scan, with no Burst	420
8.3.5.9	Mode: PGA Enabled, Low-Power, Scan, with no Burst	421

8.3.5.10	Mode: PGA Enabled, Low-Power, no Scan, with Burst and decimation, only averaged data output to FIFO	422
8.3.5.11	Mode: PGA Enabled, Low-Power, Scan, with Burst and decimation, only averaged data output to FIFO	423
8.3.5.12	Mode: PGA Bypass, High Performance, no Scan, with no Burst	424
8.3.5.13	Mode: PGA Bypass, High Performance, No Scan, with Burst and no decimation	424
8.3.5.14	Mode: PGA Bypass, High Performance, No Scan, with Burst and decimation, only averaged data output to FIFO	425
8.3.5.15	Mode: PGA Bypass, High Performance, Scan, with no Burst	426
8.3.5.16	Mode: PGA Bypass, Low Power, no Scan, no Burst	427
8.3.5.17	Mode: PGA Bypass, Low Power, Scan, no Burst	428
8.3.5.18	Mode: PGA Bypass, Low-Power, no Scan, with Burst and decimation, only averaged data output to FIFO	429
8.3.5.19	Mode: PGA Bypass, Low-Power, Scan, with Burst and decimation, only averaged data output to FIFO	430
8.3.5.20	Start the Measurement	431
8.3.6	Registers (ADC)	431
8.3.6.1	Module ADC Registers	431
8.4	DAC	459
8.4.1	DAC Overview	459
8.4.2	DAC Interface	460
8.4.3	DAC Operation	461
8.4.3.1	Common DAC Configuration Settings	462
8.4.3.2	Individual DAC Configuration Settings	462
8.4.3.3	DAC Voltage Generation	467
8.4.4	Additional Topics	468
8.4.4.1	Reduced Power Level Modes for 12-bit DAC Instances	468
8.4.4.2	Correcting for Distortion at High Output Frequency	469
8.4.5	Registers (DAC)	470
8.4.5.1	Module DAC Registers	470
8.5	LED	479

8.5.1	LED Overview	479
8.5.1.1	LED Driver Details	479
8.5.2	LED Configuration	480
8.5.2.1	Basic LED Configuration	481
8.5.2.2	Multiple LED Configuration	482
8.5.2.3	H-bridge LED Configuration	483
8.5.2.4	Independent Loop H-bridge LED Configuration	484
8.5.2.5	Integrated Feedback Loops LED Configuration	485
8.5.2.6	Internal Feedback with Common Sense Resistor LED Configuration	486
8.5.2.7	Double H-bridge LED Configuration	487
8.5.2.8	Multiple High Voltage LED Control Configuration	488
8.5.2.9	3x3 LED Matrix Configuration	489
8.5.3	Register Configurations	490
8.5.3.1	Current Mode Request	490
8.5.4	Control Loop Setup	493
8.5.5	Fault Detection Setup	494
8.5.6	Timing Configuration	495
8.5.7	Considerations	495
8.5.7.1	Design Current	495
8.5.7.2	Average and Peak Current	496
9	Pulse Train Engine	497
9.1	Pulse Train Engine (PTE) Overview	497
9.2	Output Mode Selection	497
9.3	Pulse Train Peripheral Clock Rate Configuration	497
9.4	Enabling and Disabling Pulse Train Outputs	498
9.4.1	Master Enable/Disable	498

9.4.2	Enabling and Disabling Individual Pulse Train Outputs	498
9.5	Pulse Train Engine Modes	498
9.5.1	Pulse Train Mode	498
9.5.2	Square Wave Mode	500
9.6	Synchronization	500
9.7	Registers (PT)	500
9.7.1	Module PT Registers	500
9.7.1.1	PTG_CTRL	502
9.7.1.2	PTG_RESYNC	502
9.7.1.3	PTn_RATE_LENGTH	504
9.7.1.4	PTn_TRAIN	504
10	System Clock, Timers/Counters, Watchdog Timers and Real Time Clock	506
10.1	System Clock	506
10.1.1	System Clocks Overview	506
10.1.1.1	External High-Frequency Crystal Oscillator	507
10.1.1.2	32kHz Crystal Oscillator	508
10.1.1.3	48MHz USB Clock PLL	508
10.1.1.4	48MHz Internal 24MHz Trimmed Relaxation Oscillator	508
10.1.1.5	Cryptographic Internal Oscillator	508
10.1.2	System Clock Configuration	509
10.1.3	System Clock Sources	509
10.1.3.1	External Clock	510
10.1.3.2	External High Frequency Clock (Crystal or Resonator)	510
10.1.3.3	External 32kHz Clock (Crystal or Resonator)	512
10.1.3.4	Phase Lock Loop	513
10.1.3.5	Relaxation Oscillator	515

10.1.3.6	Crypto Clock Relaxation Oscillator	516
10.1.4	ADC Clock Source Configuration	517
10.1.5	Registers (CLKMAN)	518
10.1.5.1	Module CLKMAN Registers	518
10.2	Watchdog Timers	563
10.2.1	Watchdog Timers Overview	563
10.2.2	Clock Source Selection and Gating	563
10.2.3	Watchdog Timer Configuration	564
10.2.3.1	Locking and Unlocking the Watchdog Timer Configuration	566
10.2.3.2	Enabling and Disabling the Watchdog Timer Counter	566
10.2.4	Watchdog Timer Operation	567
10.2.5	Registers (WDT)	567
10.2.5.1	Module WDT Registers	567
10.3	Real Time Clock (RTC)	572
10.3.1	Real Time Clock Overview	572
10.3.1.1	Real Time Clock Features	573
10.3.2	RTC Resets	573
10.3.3	RTC Interrupts	574
10.3.4	RTC Configuration	575
10.3.5	Registers (RTCTMR)	578
10.3.5.1	Module RTCTMR Registers	578
10.3.6	Registers (RTCCFG)	592
10.3.6.1	Module RTCCFG Registers	592
10.4	Timers/Counters	595
10.4.1	Timers/Counters Overview	595
10.4.2	32-bit Mode Timer Operation	598

10.4.2.1	One-Shot Mode	598
10.4.2.2	Continuous Mode	599
10.4.2.3	Counter Mode	600
10.4.2.4	PWM Mode	601
10.4.2.5	Capture Mode	603
10.4.2.6	Compare Mode	603
10.4.2.7	Gated Mode	604
10.4.2.8	Capture/Compare Mode	605
10.4.3	16 bit Mode Timer Operation	606
10.4.3.1	One-Shot Mode	607
10.4.3.2	Continuous Mode	607
10.4.4	Registers (TMR)	608
10.4.4.1	Module TMR Registers	608
11	Trust Protection Unit (TPU)	618
11.1	AES Cryptographic Engine	618
11.1.1	Registers (AES)	618
11.1.1.1	Module AES Registers	618
11.2	Modular Arithmetic Accelerator (MAA)	627
11.2.1	Registers (MAA)	628
11.2.1.1	Module MAA Registers	628
11.3	Registers (TPU)	635
11.3.1	Module TPU Registers	635
11.3.1.1	TPU_PRNG_USER_ENTROPY	636
11.3.1.2	TPU_PRNG_RND_NUM	636
11.3.1.3	TPU_TSR_STATUS	636
11.3.1.4	TPU_TSR_CTRL0	637

11.3.1.5	TPU_TSR_CTRL1	639
11.3.1.6	TPU_TSR_SKS0	639
11.3.1.7	TPU_TSR_SKS1	639
11.3.1.8	TPU_TSR_SKS2	639
11.3.1.9	TPU_TSR_SKS3	640
12	CRC16 and CRC32 Hardware Accelerator	641
12.1	Overview	641
12.2	CRC Clock Gating	641
12.3	CRC Operation	641
12.4	CRC-16-CCITT Example Calculation	642
12.5	CRC-32 Example Calculation	642
12.6	Registers (CRC)	642
12.6.1	Module CRC Registers	642
12.6.1.1	CRC_RESEED	643
12.6.1.2	CRC_SEED16	643
12.6.1.3	CRC_SEED32	644
12.6.1.4	CRC_DATA_VALUE16	644
12.6.1.5	CRC_DATA_VALUE32	644
13	LCD Controller	645
13.1	LCD Overview	645
13.2	LCD Operation	646
13.3	LCD Configuration	647
13.3.1	LCD Clock	647
13.3.2	LCD Power Control	647
13.3.3	LCD Configuration Registers	647

13.3.4	LCD Internal Register Adjust	648
13.3.5	LCD Port Configuration	650
13.3.6	LCD Memory Configuration	652
13.4	Registers (LCD)	652
13.4.1	Module LCD Registers	652
13.4.1.1	LCD_LCFG	652
13.4.1.2	LCD_LCRA	654
13.4.1.3	LCD_LPCF	656
13.4.1.4	LCD_LCADDR	656
13.4.1.5	LCD_LCDATA	657
13.4.1.6	LCD_LPWRCTRL	657
14	Flash Controller and Instruction Cache	660
14.1	Registers (FLC)	660
14.1.1	Module FLC Registers	660
14.1.1.1	FLC_FADDR	661
14.1.1.2	FLC_FCKDIV	661
14.1.1.3	FLC_CTRL	661
14.1.1.4	FLC_INTR	665
14.1.1.5	FLC_FDATA	666
14.1.1.6	FLC_PERFORM	666
14.1.1.7	FLC_STATUS	667
14.1.1.8	FLC_SECURITY	668
14.1.1.9	FLC_BYPASS	669
14.1.1.10	FLC_USER_OPTION	670
14.1.1.11	FLC_CTRL2	671
14.1.1.12	FLC_INTFL1	672

14.1.1.13 FLC_INTEN1	672
14.1.1.14 FLC_DISABLE_XR0	673
14.1.1.15 FLC_DISABLE_XR1	674
14.1.1.16 FLC_DISABLE_XR2	674
14.1.1.17 FLC_DISABLE_XR3	675
14.1.1.18 FLC_DISABLE_WE0	675
14.1.1.19 FLC_DISABLE_WE1	675
14.1.1.20 FLC_DISABLE_WE2	676
14.1.1.21 FLC_DISABLE_WE3	676
14.2 Registers (ICC)	677
14.2.1 Module ICC Registers	677
14.2.1.1 ICC_ID	677
14.2.1.2 ICC_MEM_CFG	678
14.2.1.3 ICC_CTRL_STAT	678
14.2.1.4 ICC_INVDT_ALL	679
15 Trademarks and Service Marks	680

1 Disclaimer and Revision History

Disclaimer

LIFE SUPPORT POLICY

MAXIM'S PRODUCTS ARE NOT DESIGNED, INTENDED OR AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT, LIFE SUSTAINING, DEVICES OR SYSTEMS OR APPLICATIONS, INCLUDING BUT NOT LIMITED TO, NUCLEAR, TRANSPORTATION OPERATING SYSTEMS, IN WHICH THE FAILURE OF SUCH GOODS COULD REASONABLY BE EXPECTED TO RESULT IN PERSONAL INJURY, LOSS OF LIFE OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF MAXIM INTEGRATED PRODUCTS, INC.

As used herein

Life support, life sustaining devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

©2016 by Maxim Integrated Products, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. MAXIM INTEGRATED PRODUCTS, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. MAXIM ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering or registered trademarks of Maxim Integrated Products, Inc. All other product or service names are the property of their respective owners.

1.1 Revision Information

Version	Section	Changes
1.2		Initial Public Release
1.3	2.2.6	Added the JTAG TAP device ID value.
	4.4.1.36	IOMAN_PADX_CONTROL: Changed "three-state" to "High Impedance" when describing pin driver output states.

Version	Section	Changes
	6.4.1.3	PMUn_LOOP: Corrected descriptions for counter_0 and counter_1 fields.
	8.2.1	Graphic and label corrections to Figures 8.1, 8.2, 8.3, and 8.4.
	8.2.2.1.4	AFE_CTRL2: Corrected field and value descriptions for dacout_en0, dacout_en1, dacout_en2, and dacout_en3.
2.0	1,2	Updated trademark and document disclaimer.
	2.1	Corrected LCD capacity to 160 segments.
	5.5.1.5	Corrected default values for fields in GPIO_IN_MODE_Pn to 10b (returns 0 regardless of input pin voltage).
	6.4	Corrected names and descriptions for PMU channel registers.
	8.2.1	Corrected note following "Top Level Multiplexers" section.
	8.2.1.1	Corrected dac_orN name description to 0,1,2,3.
	8.2.1.2	Cleanup and elaboration of section under "Op Amp Control".
	8.2.1.3	Added section describing hysteresis modes in more detail.
	8.2.1.3	Corrected hysteresis description under "Low Power Comparator Control".
	8.2.1.3	Corrected entries in "Wakeup Comparator Set Modes" table.
	8.2.2.1.3	In AFE_CTRL1, corrected and expanded descriptions of fields bgextsel, dacrefsel, and adcrefsel.
	8.3.5	Extensive rewrite of this section; corrected sample rate calculation formulas have been provided.
	8.3.6.1.1	Corrected description of ADC_CTRL0.avg_mode; the "output average plus raw data mode" is for testing only.
	8.4.1	Added note in DAC overview about why interpolation mode would be useful to an application.
	8.4.4.1	Corrected formula and other minor edits.
	10.1.5.1.2	Expanded description for rtos_mode field.

2 Introduction

The **MAX32600** User Guide is targeted to hardware, embedded firmware and application developers. This guide provides information on how to use and configure the **MAX32600** memory, peripherals and registers. For ordering information, complete feature sets, package information, and electrical specifications, refer to the [MAX32600 data sheet](#).

Related Documents

- [ARM[®] SM Cortex[®]-M3 Technical Reference Manual](#) available from www.arm.com
- [MAX32600 data sheet](#)

2.1 Overview

The **MAX32600** is a low-power, mixed-signal microcontroller that can be used for a variety of applications, including integration in wearable medical devices, pulse oximetry measurement, galvanic skin response measurement, and blood glucose metering. It is based on the ARM Cortex-M3 32-bit core targeted for a maximum operating frequency of 24MHz.

Application code on the **MAX32600** runs from an onboard program flash memory (64 KB to 256 KB), with 16 KB to 32 KB SRAM available for general application use. A 2 KB instruction cache improves execution throughput, and a transparent code scrambling interface is used to protect customer intellectual property residing in the program flash memory.

Key analog peripherals on the **MAX32600** include a 16-bit ADC with an onboard PGA/MUX front end. This ADC is designed to accept inputs from up to sixteen single-ended pins or eight differential pairs. For analog output functions, there are two 12-bit voltage output DACs, two 8-bit voltage output DACs, four uncommitted op amps, four SPST switches, and four uncommitted ground switches.

The **MAX32600** includes a wide variety of digital communications and interface peripherals. An onboard LCD controller (available on the standard 12mm x 12mm package) can be used to direct drive up to 160 segments. Other communication peripherals include a USB 2.0 slave interface, three SPI master ports, master and slave I²C interfaces, and two UART ports.

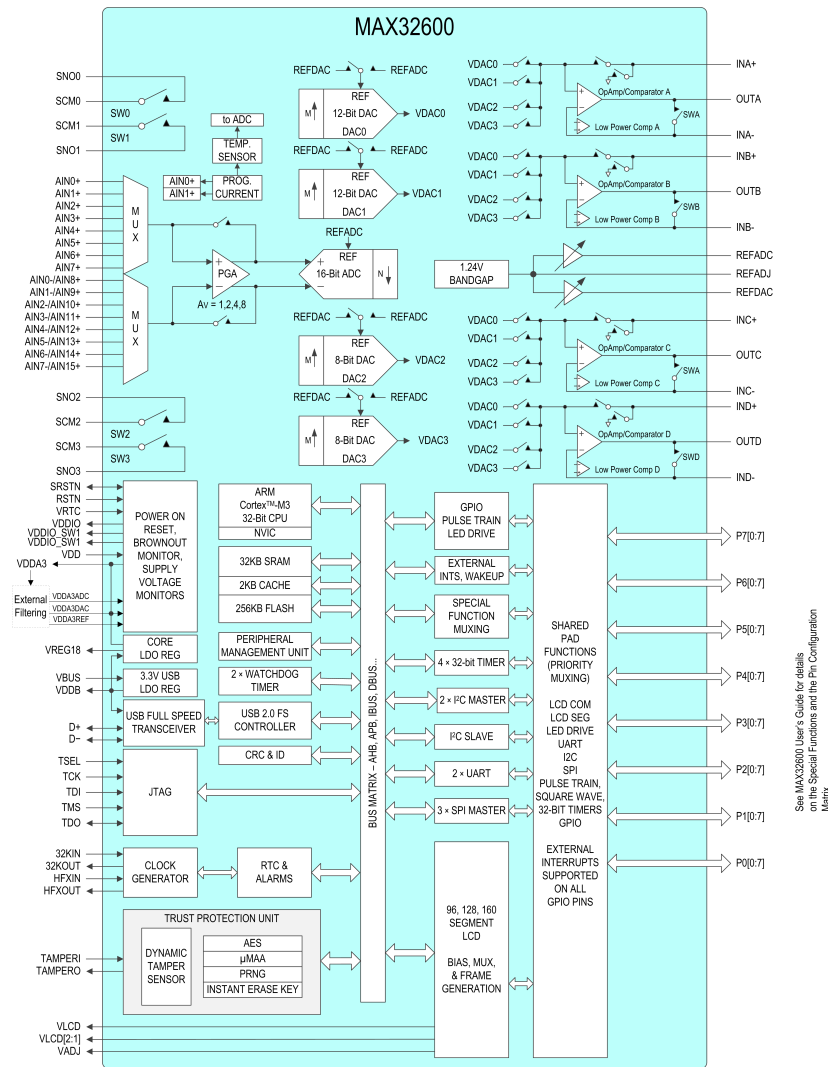


Figure 2.1: Block Diagram

See MAX32600 User's Guide for details on the Special Functions and the Pin Configuration Matrix.

2.2 Core and Architecture

ARM® Cortex®-M3 Core

The **MAX32600** is based on the ARM Cortex-M3 32-bit RISC CPU, which implements the ARMv7-M architectural profile. The implementation of the Cortex-M3 core used in the **MAX32600** is targeted for a maximum operating frequency of 24MHz and provides the following features.

- 32-bit data path with mixed 16-bit and 32-bit instructions (Thumb-2 instruction set)
- Single cycle multiplication and hardware-based division operations
- Nested vectored interrupt controller (NVIC) with multiple interrupt priority levels and nested interrupt support
- 4GB total memory space, shared by code memory, data memory, and peripheral registers
- Low power, highly energy efficient core reduces power consumption
- Built-in debug functionality and tracing with JTAG port (connects to internal Debug Access Port)
- Power saving sleep mode(s)

2.2.1 Core Parameters

When the Cortex-M3 core is instantiated in a design, values must be selected for configurable parameters in the core. For the **MAX32600** design, core parameters have been selected as shown below.

Parameter	Value	Description
NUM_IRQ	48	Number of Interrupts supported by the Cortex-M3
LVL_WIDTH	3	Specifies the number of bits of interrupt priority levels supported. At a width of three, there are eight levels supported. At a width of eight (the maximum allowed), 256 levels are supported.
MPU_PRESENT	0	The MPU (memory protection unit) is not included on this device.
BB_PRESENT	1	Bit-banding (memory mapped bit) operations are supported on this device.
AHB_CONST_CTRL	1	Specifies whether the external AHB-Lite buses maintain control information during wait stated transfers.
DEBUG_LVL	3	Full debug with data matching. All debug functionality is present including data matching for watchpoint generation.
TRACE_LVL	0	Standard trace. ITM, TPIU, and DWT triggers and counters are present. ETM and HTM port are not present.

Parameter	Value	Description
RESET_ALL_REGS	1	Registers are set to a known reset state.
JTAG_PRESENT	1	JTAG Debug Access Port is included on this design.
CLKGATE_PRESENT	1	Architectural gates are included to minimize dynamic power dissipation.
OBSERVATION	0	Additional features to observe processor internal state are not included.
WIC_PRESENT	0	The Wakeup Interrupt Controller (WIC) block is included on this design.

2.2.2 Generic Memory Map

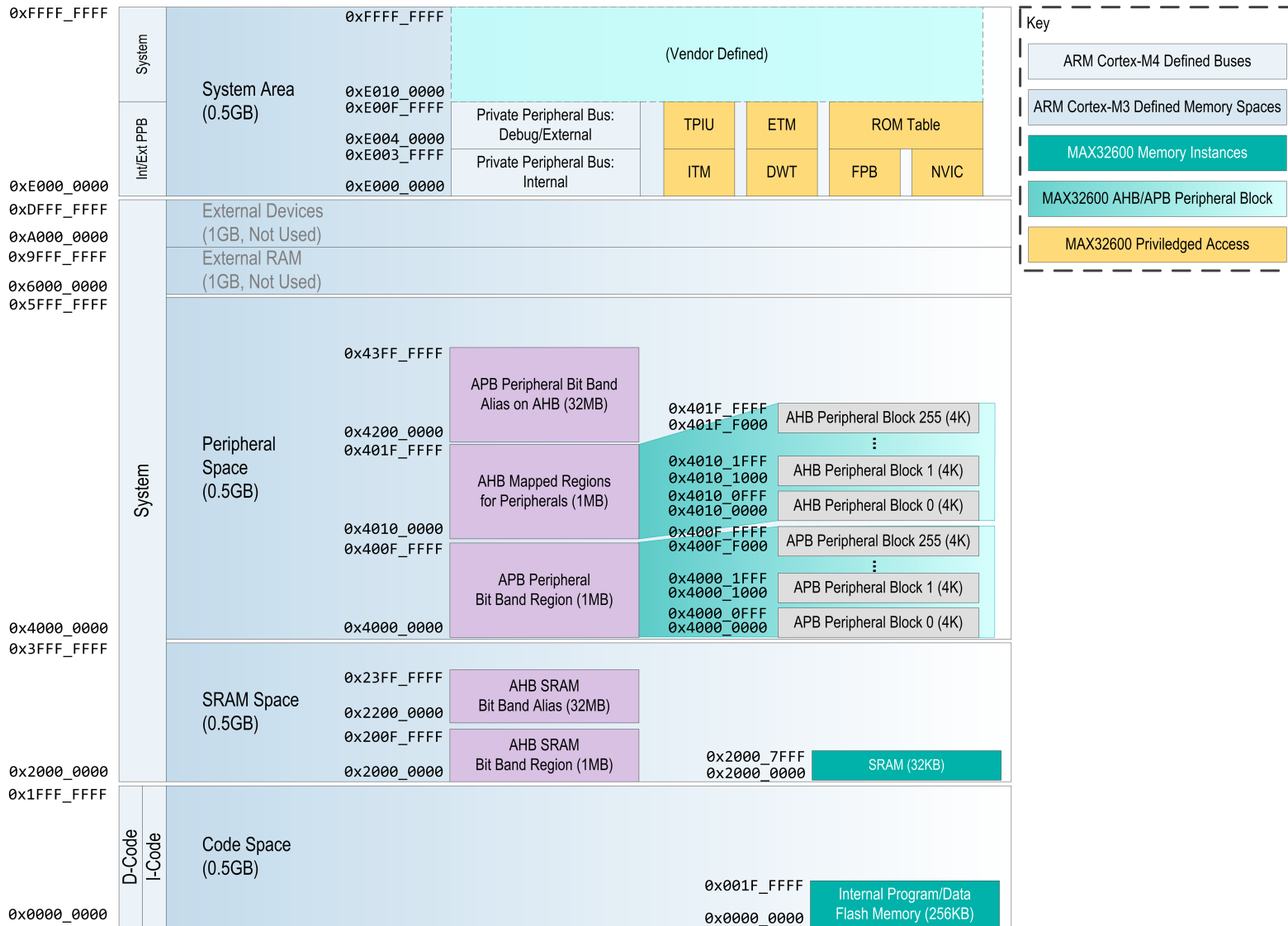


Figure 2.2: Memory Map

2.2.3 AHB Buses

The standard ARM Advanced High Performance Bus (AHB-Lite version) is used for several different system bus masters on the **MAX32600**. All buses are 32-bits in width.

- **I-Code:** Performs instruction fetches from internal code memory regions. On the **MAX32600**, accesses to internal program flash memory (for instruction decoding purposes) are cached to improve execution throughput.
- **D-Code:** Performs data fetches from program flash memory; this includes literal local constant fetches. These data fetches are not cached, unlike instruction code fetches.
- **System:** Performs data read/write and bit-band operations on internal SRAM, and data read/write operations on peripherals (including bit-band operations) and vendor defined expansion devices in the system area.

Note Bit-band operations are translated internally by the ARM core into a read-modify-write sequence. Only the core itself (when performing instruction execution) can read or write to locations using the bit-banding function. The bit-banding alias areas, although they are shown on the memory map, do not exist as separate logical mapped areas; they cannot be accessed by other AHB masters (e.g., the DMA master or the JTAG/PTP master), since they do not exist at this layer.

2.2.4 APB Buses

The External PPB (private peripheral bus) is a 32-bit bus based on the APB (Advanced Peripheral Bus) standard. It is intended for adding components to the private peripheral bus area which are not intended for general application use, since privileged operating mode is required to access this area. The **MAX32600** does not currently map any non-core components to this bus area.

The majority of the digital and analog peripherals on the **MAX32600** are controlled by registers that are memory mapped into the Peripheral region from address 0x4000_0000 to 0x400F_FFFF (in the bit-banding enabled region). These peripherals are connected to the CPU core using a lower-speed APB peripheral bus (connected to the System AHB-Lite bus through an AHB-to-APB bridge).

Peripherals which require higher speed access for large data transfers have control/buffer regions mapped to the AHB bus from address 0x4010_0000 to 0x401F_FFFF. These regions are designed to allow more rapid data transfer directly through the AHB bus, without having to go through the AHB-to-APB bridge. Peripherals using this type of interface include: SPI, I²C, DAC, ADC, AES, Micro MAA, CRC, and USB.

2.2.5 Nested Vectored Interrupt Controller (NVIC)

The **MAX32600** includes the standard Nested Vectored Interrupt Controller (NVIC) as implemented for the Cortex-M3 ARM core. The NVIC supports high-speed, deterministic interrupt response, interrupt masking, and multiple interrupt sources. External interrupts support rising or falling edge trigger mode as well as level triggered mode.

With the core instantiation parameters given above, the NVIC will support up to 48 distinct interrupt sources (including internal and external interrupts), with programmable interrupt priority of eight priority levels (3-bit width).

2.2.6 ARM Debug

The **MAX32600** includes the standard JTAG debug engine as implemented for the Cortex-M3 ARM core. The JTAG TAP interface is supported, but the serial wire interface is not.

Since this is a standard ARM core component, it is instantiated 'as-is' along with the rest of the Cortex-M3 core and cannot be modified to support special requirements for the **MAX32600** design. Accordingly, there are two separate JTAG TAP engines on the **MAX32600** - the ARM Debug JTAG and the Maxim Test JTAG. The two JTAG TAP ports share the same pins; the TSEL pin is used to switch access between the two of them.

The JTAG TAP device address for the **MAX32600** is 0x4BA00477.

Standard features supported by the ARM JTAG debug engine include the ability to set up to six breakpoints and two watchpoints, access main system memory and peripheral registers even when the CPU is running, and pause, trace, or reset the CPU. Because the debug engine is coupled to the CPU only for clocking and reset purposes, if the debug engine pauses the CPU it is important to note that other peripherals and functions on the **MAX32600** are not paused and continue to operate normally.

2.3 Power Supplies and Modes

2.3.1 Digital Supply Voltages

The **MAX32600** operates from a main digital supply voltage of 1.8V to 3.6V (V_{DD}). For portable electronic systems, this supply is typically provided by a battery. Generated supplies V_{DDIO} and V_{REG18} are switched automatically to draw from V_{DDB} when the USB power supply is present, and from V_{DD} when USB is not powered/connected. V_{DDIO} is externally available to power GPIO and GPIO-multiplexed peripheral I/O. V_{REG18} is internally regulated to 1.8V and provides power to the digital core; it is also an external option to power GPIO. The POR, power-fail reset, and power fail warning functions all operate from the switched digital supply.

LCD outputs operate from their own dedicated boost converter output rail (V_{LCD}).

2.3.2 Analog Supply Voltage

The analog functions on the **MAX32600** (such as the ADC, DACs, operational amplifiers, and SPST switches) operate from a separate analog supply voltage rail which may vary from 2.3V to 3.6V (V_{DDA3}). This supply is switched automatically and is provided from either V_{DDB} (when USB is present) or V_{DD} (when USB is disconnected/unpowered). Four secondary analog power supplies (V_{DD3ADC} , $V_{DDA3ADC}$, $V_{DDA3DAC}$, $V_{DDA3REF}$) are normally derived from V_{DDA3} . This topology enables optional external filtering circuits.

2.3.3 Onboard Core Voltage Regulator

The **MAX32600** includes a dedicated onboard digital supply voltage regulator with a 1.8V nominal output voltage (V_{REG18}). This internal regulator is used to provide a fixed low-voltage digital supply for the internal CPU core. This regulator derives its voltage output from the main digital supply voltage (V_{DD}) or V_{DDB} when USB is present.

2.3.4 Onboard VUSB Voltage Regulator and Automatic Power Switching

The **MAX32600** includes a second onboard digital supply voltage regulator which is used when a powered USB bus has been connected to the device. This regulator takes the 5V (typical) supply from V_{BUS} and regulates it down to a nominal 3.3V output. Automatic power switching features allow the device to switch automatically from the external digital supply voltage to the V_{BUS} -derived 3.3V supply voltage when USB power is available. This ensures that when the device is connected to USB, it uses the power supply available from that source, instead of running from the normal digital supply which will typically be a battery supply for portable applications.

2.3.5 VRTC Power Supply

A dedicated power supply is used to maintain the state and operation of the most critical components and memories, which can remain powered even when the rest of the device is shut down in the lowest possible power savings modes. The V_{RTC} supply ensures that the Real Time Clock can continue running (and can be used to wake the **MAX32600** after a preset time interval) even when everything else has been powered down. The V_{RTC} supply is also used to maintain certain critical battery-backed areas, such as the power sequencer and associated control registers, the tamper-detect-wiped "key of keys", and certain analog trim shadow registers.

2.3.6 Power Management Modes

Power management modes supported by the device include stop, standby, and active (PMU and run) modes. These four power modes are: **LP0: STOP**; **LP1: STANDBY**; **LP2: PMU**; and **LP3: RUN**. The Wakeup Interrupt Controller (WIC) can be used to wake the device from power saving modes.

Individual power down and clock gating controls are provided as well. This allows user application-specific reductions in power consumption by disabling/powering down functions that are not currently in use.

2.3.7 Power Supply Monitoring

Several programmable supply voltage monitors are provided, which allow a power fail warning interrupt and/or a system reset to be triggered when the supply voltage drops below a preset level (depending on the type of power supply and firmware settings). Supply voltage monitors are available to measure the following supplies: V_{DDA3} , V_{RTC} , V_{REG18} , and V_{DD3} .

2.4 Clock Inputs

Configuration of time delays are necessary to ensure that the **MAX32600** always switches to a valid clock; reference the [Clocks and Timers](#) section for detailed information.

2.4.1 External High Frequency Crystal

The **MAX32600** includes a high-frequency crystal oscillator circuit designed to operate with an external crystal. Fundamental mode crystals can be used with the oscillator circuit up to a frequency of 24MHz. External load capacitors are required depending on the crystal specifications.

An external clock source may also be used by the **MAX32600** in place of a high-frequency crystal. For this configuration, the external clock source (which must meet the electrical/timing requirements given in the datasheet) is connected to the part on the HFXIN pin and the HFXOUT pin is left floating.

2.4.2 32kHz Crystal Oscillator

A 32kHz crystal oscillator (with the 32kHz crystal connected between the 32KIN and 32KOUT pins) is used to generate the 32kHz clock that is used by the Real Time Clock module.

An external clock source may also be used by the **MAX32600** in place of a 32kHz crystal. For this configuration, the external clock source (which must meet the electrical/timing requirements given in the datasheet) is connected to the part on the 32KIN pin.

2.4.3 48MHz USB Clock PLL

The phase locked loop (PLL) clock generation circuit is used to generate a 48MHz clock which is required for proper operation of the USB device interface. If the USB interface will not be used, then use of the PLL is optional.

The PLL generates a 48MHz clock using a clock multiplier circuit, which has a 2X mode for use with a 24MHz crystal, a 4X mode for use with a 12MHz crystal, and a 6X mode for use with an 8MHz crystal. The output of the PLL can be used as a system clock source (after dividing it by two) with a 24MHz frequency.

2.4.4 Internal 24MHz Trimmed Relaxation Oscillator

An internal trimmed relaxation oscillator generates a 24MHz ($\pm 1\%$) clock which can be used as a system clock source. If the USB interface will be used, the PLL is used to generate a 48MHz ($\pm 0.25\%$) clock. The input clock to the PLL can be an external crystal, an external digital clock source, or the internal relaxation oscillator can be used if a 32kHz crystal is available to frequency trim the relaxation oscillator.

For optimal performance, the ADC requires a more stable clock source (with less jitter) than can be provided by the relaxation oscillator; the high frequency crystal oscillator must be used as a system clock source when the ADC is in use. However, in lower sample rate applications, the relaxation oscillator can be used as the ADC clock source.

2.4.5 Cryptographic Internal Oscillator

To reduce opportunities for timing-based analysis and fault injection (clock interference) attacks targeting the cryptographic and security functions of the device (such as the AES cryptographic engine, the uMAA, and other related functions), a dedicated on-chip oscillator is provided on the **MAX32600** which can be used to supply a separate, isolated clock for use by these functions.

Decoupling these functions from the main system clock allows encryption and decryption operations to take a constant amount of time regardless of the current system clock rate, and also provides a more variable (and not externally observable) clock to reduce the opportunity for an attacker to perform power or timing analysis attacks against the cryptographic and security functions.

2.5 Memory

2.5.1 Internal Flash Program Memory

The **MAX32600** includes from 64 KB to 256 KB (depending on the specific device production option) of internal flash program memory. Internally, the flash memory has a width of 64-bits. Flash memory must thus be programmed one 64-bit location at a time, which requires two programming operations of 32-bits each. The flash is divided into logical pages; when erasing the flash, it is possible to erase either a single page (page erase) or the entire flash array (mass erase) in one operation.

For read access, the program flash memory is mapped into the standard code space region beginning at address 0x0000_0000. Modifying the flash memory array (either program or erase operations) is handled by the flash controller peripheral.

The flash controller also provides the ability to directly program a flash location by writing a 32-bit value to the proper memory location using the AHB bus (as opposed to setting up the operation directly using the appropriate flash controller peripheral registers). Whether the write operation is triggered by an AHB memory write or by writing a control sequence to the flash controller registers, the same security and operational restrictions apply in either case. The operation proceeds in the same manner in both cases regardless of the method that was used to trigger it.

The beginning of program flash memory (starting at address 0x0000_0000) is also the default location for the ARM exception/interrupt vector table. Since this table contains initialization information such as the reset vector address which is required for the system to properly initialize, this table must be loaded into the flash in order for the **MAX32600** to execute any application code.

The **MAX32600** supports multiple size options for the flash memory within the maximum possible space of 256 KB. The actual size of the memory is controlled by a trim option loaded from the flash information block. When a size smaller than the maximum one is being used, the flash controller will respond to attempts to access out-of-range addresses within the 256 KB range by setting an interrupt flag and returning a fixed "invalid access" data result pattern.

2.5.2 2 KB Instruction Cache

The 2 KB (512 x 32 or 256 x 64) instruction cache is used to cache the content of recently accessed locations in the program flash array to improve execution throughput. Instruction fetches go to the instruction cache, which either returns the previously cached data (cache hit) or fetches the requested data from the flash (cache miss) and stores it for future access.

Fetches between the instruction cache and the flash memory go through the code descrambler, so that the content stored in the instruction cache has already been descrambled.

Code access to the internal data SRAM is not cached; instruction fetches from SRAM are always performed directly. Data fetches (D-Code, as opposed to I-Code fetches for the purposes of decoding instructions) are also not cached and are performed directly. This includes fetches of local constant literals that are used by certain ARM instruction op codes.

Firmware has the option to flush the instruction cache manually at any point using a control register. When code mapped into a cached instruction space is updated (for example, if an in-application programming modification is made to an executable area of program flash), the cache should be flushed to ensure that the latest version of the code will be accessible and that stale cache contents will not be used instead of the new flash programmed values.

2.5.3 Internal Data SRAM

The internal data SRAM on the **MAX32600** ranges from 16 KB to 32 KB in size and has a 32-bit internal width. It is mapped into the SRAM bit-banding access region beginning at address 0x2000_0000, and so it can be read/written either a full 32-bit word at a time, or a single bit at a time using the bit-band alias region (beginning at 0x2200_0000). The bit-banding function can only be used when the data SRAM is being accessed by the ARM core itself, since the ARM core handles the remapping from the bit-banding alias area to a read-modify-write sequence (or single read/mask/shift for a bit read function) of the standard memory area.

The data SRAM can be read or written freely by the application, and can be used for either code or data access. The contents of the data SRAM are not battery-backed. The data SRAM is also used to hold the stack.

The **MAX32600** supports multiple SRAM memory sizes within the maximum allowed address space of 32 KB. When a smaller size than the maximum is used (such as 16 KB) attempts to access out-of-range addresses within the 32 KB maximum range will generate an AHB bus error. The effect of this error condition will be determined by the AHB master accessing the bus; for example, the ARM core will respond to this error condition by generating a MemFault system exception. Other bus masters (e.g., the DMA AHB bus master or the JTAG/PTP bus master) may respond differently or ignore the AHB error altogether; the exact results of this condition are determined by the designer of the AHB master interface block.

2.5.4 Peripheral Management Unit (PMU)

The PMU controller on the **MAX32600** provides a generalized, flexible mechanism to perform automatic read and/or write sequences to peripherals and areas of internal SRAM memory. The PMU controller includes multiple channels which can be connected to different peripherals or memory areas and is capable of operation during sleep mode.

Peripherals which can be read from or written to using PMU channels (accessing AHB mapped memory areas) include:

- The ADC
- Any of the four DAC instances
- Any of the UART instances
- Any of the SPI instances

- Any of the USB endpoint buffers (since they are stored in the main SRAM area)
- The CRC engine

The PMU controller can also be used to read from the program flash memory or any peripheral register area which is normally accessible on the System bus. It does not, however, have access to the USB control register area mapped to the AHB bus.

2.5.5 Flash Information Block

The flash information block on the **MAX32600** allows production trim values and other nonvolatile information that will be written during the production process (e.g., device configuration and test details / logging data) to be stored in a separate dedicated instance of internal flash.

The flash information block can be mapped to the AHB bus in the code space area, beginning at byte address location 0x0004_0000 (which is immediately following the maximum possible flash memory address). This address location is constant even if a smaller flash memory size option is being used by the device.

When the flash information block is mapped to this memory region (between 0x0004_0000 and 0x0004_07FF), it can be accessed in the same manner as the main flash; that is, it can be read from in code space (although accesses to it are not cached under any circumstances, and its contents are not subject to the scrambling/descrambling function used by the contents of the main program flash memory). It is possible to write to locations in the flash information block using either direct AHB writes (which will trigger a 32-bit write operation in the flash controller) or by writing the appropriate sequence directly to the flash controller registers.

However, this mapping of the flash information block (and direct read/write access to its contents) is only intended for testing and trimming purposes during the factory production test sequence. Once production test of the **MAX32600** has completed (or at least the last stage where trim operations are performed has been completed), a lock option setting will be set in the information block to prevent future modifications to the trim and option settings (except for those which are explicitly allowed to be set later by the user, such as the DSB access key and the auto-lock security option). Setting the info block lockout setting also removes the information block from the memory map, which means that the only way to view its contents after that point is by reading the copies of the info block settings that have been copied by the hardware into the trim shadow registers (which are mapped to a different area in the APB peripheral region).

The flash controller automatically loads trim and other configuration values from the appropriate locations in the flash information block following any system reset. To prevent misconfiguration of trim and other option settings in the case of a random (i.e., unprogrammed), blank, or corrupted information block, an "info block valid" field is checked by the flash controller before the remaining locations in the information block memory are scanned. If the info block valid field (which is the first addressed location in the information block array) does not match the predetermined "valid" key value (which is 0x12345678), the information block is presumed to be invalid, and instead of the trim registers being set to copies of the data contained in the information block, they are loaded with predetermined "default" values to prevent undesired or erratic system performance. These default values are designed to allow maximum access (with a minimum of security settings applied) for ease of testing and configuration.

2.5.6 Flash Memory Controller

The flash memory controller on the **MAX32600** handles control and timing signals for programming and erase operations on both the main program flash memory array and the flash information block. The flash information block is normally written during production test only, and is not generally intended to be modified by the

user application. Two exceptions to this are the Destructive Security Bypass access key value and the Auto-Lock option value, which may be set by the user by means of a special procedure even after access to the rest of the information block has been locked out.

Functions provided by the flash controller for use in normal operation include:

- Mass erase of main program flash array
- Page erase of one page in the main program flash array
- Write to one location (programmed 32-bits at a time) in the main program flash array
- Special one-time writes by the user to the DSB Access Key and/or the Auto-Lock option values in the information block

2.6 Analog Peripherals

2.6.1 16-Bit ADC with PGA

The **MAX32600** includes a 16-bit analog-to-digital converter (ADC) with a 16-channel analog input multiplexer, to allow selection of input from one of 16 input lines (single-ended mode) or two of eight input pairs (differential mode). The differential mode supports fully differential signal inputs.

The front end PGA allows programmable gain settings of 1X, 2X, 4X, and 8X before the input sample is converted.

The ADC reference voltage is selectable between V_{AVDD} and the dedicated ADC reference level. The ADC reference level can be set by software to one of four output levels - 1.024V, 1.5V, 2.048V, and 2.5V - based on the 1.23V reference bandgap.

2.6.2 ADC/DAC Internal/External Reference and Programmable Output Buffers

Two programmable reference levels (one used by the ADC, one used by the DACs) are included, and each can be individually set to one of four output levels: 1.024V, 1.5V, 2.048V, and 2.5V.

An external reference can also be provided at the REFADJ pin; if this feature is used, the external reference voltage will be used in place of the 1.23V bandgap output, and the programmable output levels for the ADC and DAC references will shift accordingly.

2.6.3 12-Bit Voltage Output DACs

The **MAX32600** includes two 12-bit voltage output DACs (DAC0, DAC1) which output single-ended voltages. The reference used by these DACs is selectable between the DAC reference level and the ADC reference level.

Each DAC instance includes PMU channel access to allow output values to be loaded to the DAC directly from memory.

2.6.4 8-Bit Voltage Output DACs

The **MAX32600** includes two 8-bit voltage output DACs (DAC2, DAC3) which output single-ended voltages. The reference used by these DACs is selectable between the DAC reference level and the ADC reference level.

Each DAC instance includes PMU channel access to allow output values to be loaded to the DAC directly from memory.

2.6.5 Uncommitted Op Amps with Comparator Mode

The **MAX32600** contains four uncommitted operational amplifiers. Any unused op amp should not be enabled. Each op amp may be switched between amplifier and comparator mode under software control.

Each op amp contains an integrated internal switch that can be used to short the negative/inverting input pin to the output pin of the op amp under software control, putting the op amp in a voltage follower mode. In this configuration, the op amp can be used as an output buffer for any of the four DAC outputs.

Any of the four DAC outputs may optionally be internally connected to the inverting or noninverting inputs of one or more of the four op amps, under software control.

2.6.6 Uncommitted SPST Analog Switches

The device contains four uncommitted SPST analog switches which can be opened and closed under software or pulse train control. All SPST switches are open by default following any reset or power-on reset.

The SPST switches support input voltages from ground to V_{AVDD} .

2.6.7 Temperature Sensor

The device includes an internal temperature sensor which can be read using the ADC. The **MAX32600** also supports a mode for an external temperature sensor.

2.7 Digital Peripherals

2.7.1 GPIO Pins w/Interrupt and Wakeup Capability

The device includes up to eight GPIO ports with eight pins per port for a total of 64 GPIO pins. Pins may be multiplexed with digital peripheral functions and/or LCD segment output lines. GPIO pins may be individually switched between GPIO and secondary/tertiary peripheral functions using the appropriate control registers.

All GPIO pins can be configured individually by firmware to act as external interrupt sources. All GPIO pins also have the option (which is separate from configuring a GPIO pin to act as an external interrupt source) to be configured to provide wakeup source inputs to the Wakeup Interrupt Controller (WIC) to wake the device from power-saving modes.

All GPIO pins support standard I/O operating modes - input/output mode select, strong drive high/low when in output mode, and selection of weak pullup or high impedance (three-state) when in input mode.

Certain GPIO pin pairs have current sink capability that allows them to drive an open drain output (optical LED drive) based on a DAC output.

2.7.2 32-Bit Timer/Counters

The device includes four 32-bit timer/counter modules with the following features:

- 32-bit up/down count with auto reload mode
- One-shot or continuous operation mode
- Programmable 16-bit prescaler
- PWM output generation mode
- Capture/compare modes
- External input pin for timer input, clock gating or capture, limited to an input frequency of 1/4 of the peripheral clock
- Timer output pin
- Timer interrupt

Each 32-bit timer/counter module also has the option to be split into two separate 16-bit timers (dual 16-bit timer mode) for a possible total eight timers in the system. Each 16-bit timer in this pair has the following features:

- 16-bit up/down count with auto reload mode
- One-shot or continuous operation mode
- Programmable 16-bit prescaler (setting is shared by both 16-bit timers in the pair)
- Timer interrupt (separate interrupt for each 16-bit timer in the pair)

2.7.3 Watchdog Timers

The **MAX32600** includes two independent watchdog timers (WDT) with window support. The watchdog timers run independently from each other and the processor and have multiple clock source options for ensuring system stability. The watchdog uses a 32-bit timer with prescaler to generate the watchdog reset. When enabled, the watchdog timers must be fed/reset prior to timeout or within a specified window of time if window mode is enabled. Failure to do so before the watchdog times out will result in a watchdog reset event.

The first watchdog instance (WDT0) can be configured to trigger a system reset (reset of digital core) when it generates a watchdog reset. The second watchdog instance (WDT1) can be configured to generate a system reboot (equivalent to digital POR event) when it generates a watchdog reset.

The two watchdogs may be configured independently to use either the currently selected system clock or an external clock as the watchdog clock source. The external clock is selected separately (in system manager) and is the same for both watchdog instances.

2.7.4 32-Bit Real Time Clock with Time of Day Alarm

A binary real-time clock (RTC) keeps the time of day in absolute seconds with 1/256-second resolution. The 32-bit second counter can count up to approximately 140 years and be translated to calendar format by application software. A time-of-day alarm and independent sub-second alarm can cause an interrupt or wake the device from stop mode.

The independent sub-second alarm runs from the same RTC and allows the application to support interrupts with a minimum interval of approximately 3.9ms. This creates an additional timer that can be used to measure long periods of time without performance degradation.

2.7.5 SPI (three instances)

The integrated SPI controller provides an independent master-mode-only serial communication channel that communicates synchronously with peripheral devices in a single or multiple slave system. Depending on the other peripherals and GPIO pins that are in use by the application, up to two separate SPI ports are available for general use, with a third SPI instance reserved for Bluetooth® module communication.

The SPI controllers support half- or full-duplex communications with single, dual, or quad data transmission modes, and can be operated in master mode only. In master mode, the SPI can transfer data at up to 24 MHz depending on the clock source. In addition, the SPI module supports configuration of active SSEL state (active low or active high) through the slave active select. DMA is supported for both the transmit and receive buffers.

2.7.6 I²C

The microcontroller integrates an internal I²C bus master/slave for communication with a wide variety of other I²C-enabled peripherals. The I²C bus is a two-wire, bidirectional bus using a ground line and two bus lines: the serial data line (SDA) and the serial clock line (SCL). Both the SDA and SCL lines must be driven as open-collector/drain outputs. External resistors (R_P) are required to pull the lines to a logic-high state.

The device supports both the master and slave protocols. In the master mode, the device has ownership of the I²C bus, drives the clock, and generates the START and STOP signals. This allows it to send data to a slave or receive data from a slave as required. In slave mode, the device relies on an externally generated clock to drive SCL and responds to data and commands only when requested by the I²C master device.

There are two instances of the I²C master interface and one instance of the I²C slave interface supported.

2.7.7 USB 2.0 Device Slave with Integrated Transceiver

The integrated USB controller is compliant with the USB 2.0 specification, providing full-speed operation as a USB peripheral device. Integrating the USB physical interface (PHY) allows direct connection to the USB cable, reducing board space and overall system cost. An integrated voltage regulator enables smart switching between the main supply and V_{BUS} when connected to a USB host controller.

The USB Controller contains an integrated AHB bus master which it uses to write to and read from the buffers for each supported/active endpoint. These buffers are located in the standard system SRAM (in user-configurable locations), so they can also be accessed by firmware directly as well as by the USB DMA engine. A total of seven endpoint buffers are supported with configurable selection of IN or OUT in addition to endpoint 0, which is read-only.

2.7.8 LCD Controller

The **MAX32600** incorporates an LCD controller with a boost regulator that interfaces to common low-voltage displays in the standard 12mm x 12mm package. By incorporating the LCD controller into the microcontroller, the design requires only an LCD glass rather than a considerably more expensive LCD module. Every character in an LCD glass is composed of one or more segments, each of which is activated by selecting the appropriate segment and common signal.

The microcontroller can multiplex combinations of up to 40 segment outputs (SEG0 to SEG39) and four common signal outputs (COM0 to COM3). Unused segment outputs can be used as general-purpose port pins. The segments are easily addressed by writing to dedicated display memory. Once the LCD controller settings and display memory have been initialized, the 21-byte display memory is periodically scanned, and the segment and common signals are generated automatically at the selected display frequency. No additional processor overhead is required while the LCD controller is running. Unused display memory can be used for general-purpose storage.

The design is further simplified and cost reduced by the inclusion of software-adjustable internal voltage-dividers to control display contrast, using either V_{DDIO} or an external voltage. If desired, contrast can also be controlled with an external resistor network.

The features of the LCD controller include the following:

- Automatic LCD segment and common-drive signal generation
- Integrated boost regulator ensures LCD operation down to 2V
- Flexible LCD clock source selection
- Adjustable frame frequency
- Internal voltage-divider resistors eliminate requirement for external components
- Internal adjustable resistor allows contrast adjustment without external components

Four display modes are supported by the LCD controller:

- Static (COM0)

- 1/2 duty multiplexed with 1/2 bias voltages (COM [0:1])
- 1/3 duty multiplexed with 1/3 bias voltages (COM [0:2])
- 1/4 duty multiplexed with 1/3 bias voltages (COM [0:3])

Note Since the voltages available for LCD drive are V_{LCD} , $V_{LCD} \times \frac{2}{3}$, $V_{LCD} \times \frac{1}{3}$, and V_{ADJ} , the $\frac{1}{2}$ -bias mode (which requires an output level of $V_{LCD} \times \frac{1}{2}$) will require two of the LCD voltage output pins (V_{LCD2} and V_{LCD1}) to be shunted together externally.

2.8 Security Features

2.8.1 Trust Protection Unit (TPU)

The **MAX32600** includes several cryptographic and security peripherals which are grouped together to form the Trust Protection Unit, or TPU. The TPU architecture includes cryptographic peripherals (such as the AES engine or the Micro MAA) as well as security features (such as the dynamic tamper sensor) which help to form a secure cryptographic boundary for protecting critical user information within the device.

2.8.2 AES Cryptographic Engine

Another component found in the TPU is an Advanced Encryption Standard (AES) cryptographic engine. This cryptographic engine allows 256-bit AES encryption and decryption operations to be performed in hardware without processor intervention.

The AES control register selects encryption or decryption mode, controls interrupt notification of the processor upon an operation completion, and selects whether or not the key expansion (generation of first/last round key) is performed before the encryption or decryption operation begins. For multiple-block encryption or decryption operations using a single key, the key expansion is performed on the first block only, allowing subsequent operations to complete more quickly by reusing the previously generated round key information.

The working AES key, cryptographic working space, and input and output parameters (plaintext to ciphertext or vice versa) are stored in a dedicated internal AES memory. This memory is automatically cleared (rapid zeroization) in the event of a tamper response.

2.8.3 Battery-Backed AES Secure Key Storage

The battery-backed (by the V_{RTC} supply voltage) RTC module contains a dedicated set of registers which can be used to store a master AES key or other critical data. This master AES key can be generated by the device using a PRNG random number sequence, and is stored in a dedicated location in the RTC register area which will be automatically cleared (rapid zeroization) in the event of a tamper response.

This key is intended for use in encrypting other sensitive information that might be stored in other locations on the device (such as the main system SRAM or the program flash memory) that will not be automatically wiped in the event of a tamper response. However, if this sensitive information has been encrypted before

storage using an AES master key, then once a tamper response occurs and this master key has been deleted, the sensitive information will not be recoverable by an attacker at that point; even if the ciphertext version of the information can be recovered in some way, the key that was used to encrypt that data no longer exists.

Another potential use of this key would be to encrypt larger keys that might be stored in long-term, nonvolatile storage on the device (such as a public key, private key, or certificate, which might be stored in the main program flash). By encrypting the larger keys with the AES master key, they may safely be stored in a nonsecure location such as the main SRAM or program flash memory; if a tamper response occurs, the AES key will be wiped, effectively destroying the other encrypted keys or data as well since the encrypted information is now useless.

The Secure Key Storage area (which has a capacity of four 32-bit registers or 128 bits total) does not have to be used for an AES key, however; it can be used for any type of secure data that must be immediately erased in the event of a tamper response.

2.8.4 Modular Arithmetic Accelerator (MAA)

The MAA cryptographic module is considered to be another component of the TPU. It allows firmware to perform 512-bit large number modular arithmetic operations which can be used in turn to implement cryptographic algorithms such as RSA. The RSA set of cryptographic operations can be used for public/private cryptography operations.

The MAA operates from a dedicated internal register file memory which is used to store keys, input and output parameters, and for scratch working space. The MAA does not use the general purpose system SRAM to store data.

2.8.5 CRC Hardware Block with CRC16 and CRC32

A CRC hardware module is included to provide fast calculations and integrity checking of application software and data. The CRC module supports both CRC-16--CCITT and CRC-32 polynomial modes. The CRC-16 operation completes in two clock cycles, while the CRC-32 operation requires four cycles.

Additional features of the CRC module include:

- Programmable start seed
- Programmable start address
- Programmable length
- Direct load or PMU-based memory load support

2.8.6 Code Scrambling

All application code and data loaded into the main program flash memory is scrambled in both content and location by hardware before it is stored in the flash. When data is retrieved from the flash, it is descrambled before arriving at the program cache (for instruction fetches) or the main data bus (for data fetches). Both the scrambling and descrambling operations are transparent to the end user.

3 Memory, Register Mapping, and Access

3.1 Memory, Register Mapping, and Access Overview

The ARM Cortex-M3 architecture defines a standard memory space for unified code and data access. This memory space is addressed in units of single bytes but is most typically accessed in 32-bit (4-byte) units. It may also be accessed, depending on the implementation, in 8-bit (1-byte) or 16-bit (2-byte) widths. The total range of the memory space is 32-bits in width (4GB addressable total), from addresses 0x0000_0000 to 0xFFFF_FFFF.

It is important to note, however, that the architectural definition does not require the entire 4GB memory range to be populated with addressable memory instances.

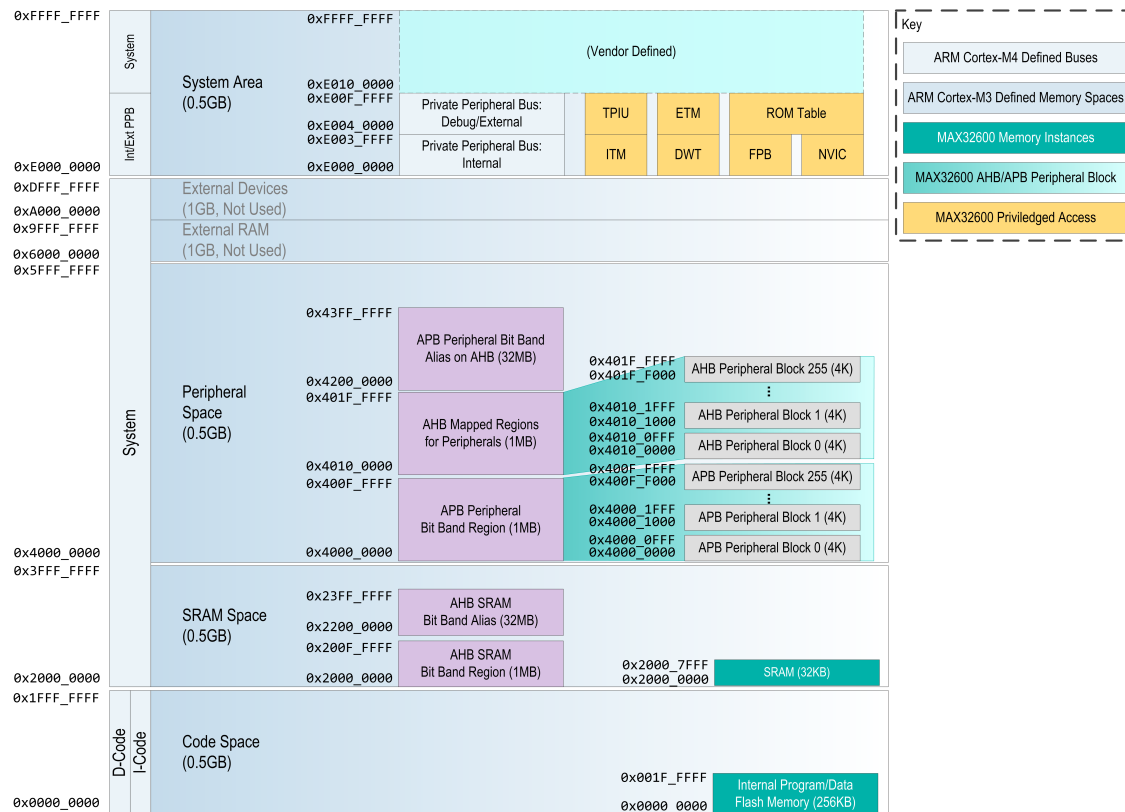


Figure 3.1: Memory Map Diagram

3.2 Standard Memory Regions

A number of standard memory regions are defined for the ARM Cortex-M3 architecture; the use of many of these is optional for the system integrator. At a minimum, the **MAX32600**, a Cortex-M3-based device, must contain some code and data memory for application code and variable/stack use, as well as certain components which are part of the instantiated core.

3.2.1 Code Space

The code space area of memory is designed to contain the primary memory used for code execution by the device. This memory area is defined from byte address range 0x0000_0000 to 0x1FFF_FFFF (0.5GB maximum). Two different standard core bus masters are used by the Cortex-M3 core and ARM debugger to access this memory area. The I-Code AHB bus master is used for instruction decode fetching from code memory, while the D-Code AHB bus master is used for data fetches from code memory. This is arranged so that data fetches avoid interfering with instruction execution.

On the **MAX32600**, the code space memory area contains the main program flash memory, which holds the majority of the instruction code that will be executed on the device. The program flash is mapped from 0x0000_0000 to 0x0003_FFFF. This program memory area must also contain the default system vector table (located initially at address 0x0000_0000), which contains the reset vector for the device and the initial settings for all system exception handlers and interrupt handlers.

The code space memory on the **MAX32600** also contains the mapping for the flash information block, from 0x0004_0000 to 0x0004_07FF. However, this mapping is generally only present during production test; it is disabled once the information block has been loaded with valid data and the info block lockout option has been set. This memory is accessible for data reads only and cannot be used for code execution.

3.2.2 SRAM Space

The SRAM area of memory is intended to contain the primary SRAM data memory of the device and is defined from byte address range 0x2000_0000 to 0x3FFF_FFFF (0.5GB maximum). This memory can be used for general purpose variable and data storage, code execution, and the ARM Cortex-M3 stack.

This memory area contains the main system SRAM on the **MAX32600**, which is mapped from 0x2000_0000 to 0x2000_7FFF.

The entirety of the SRAM memory space on the **MAX32600** is contained within the dedicated ARM Cortex-M3 SRAM bit-banding region from 0x2000_0000 to 0x200F_FFFF (1MB maximum for bit-banding). This means that the entire SRAM can be accessed using bit-banding operations when executing core instructions. This allows any single bit of 32-bit SRAM location to be set, cleared, or read individually by reading from or writing to a specified location in the bit-banding alias area.

The alias area for the SRAM bit-banding is located beginning at 0x2200_0000 and is a total of 32MB maximum, which allows the entire 1MB bit banding area to be accessed. Each location in the bit-banding alias area translates into a single bit access (read or write) in the bit-banding primary area. Reading from the location performs a single bit read, while writing either a 1 or 0 to the location performs a single bit set or clear.

Note The ARM Cortex-M3 core translates the access in the bit-banding alias area into the appropriate read cycle (for a single bit read) or a read-modify-write cycle (for a single bit set or clear) of the bit-banding primary area. This means that bit-banding is a core function (i.e., not a function of the SRAM memory interface layer or the AHB bus layer), and thus is only applicable to accesses generated by the core itself. Reads/writes to the bit-banding alias area by other (non-ARM-core) bus masters such as the PMU AHB bus master will not trigger a bit-banding operation and will instead result in an AHB bus error.

The SRAM area on the **MAX32600** can be used to contain executable code. Code stored in the SRAM is accessed directly for execution (using the system bus) and is not cached or code scrambled.

The SRAM is also where the ARM Cortex-M3 stack must be located, as it is the only general-purpose SRAM memory on the device. A valid stack location inside the SRAM must be set by the system exception table stored in the main program flash.

The general purpose PMU engine and the function specific AHB bus master included in the USB peripheral block can both access the SRAM to use as general storage or working space. Specifically in the case of the USB interface, SRAM memory area can be used to store the descriptor table for the endpoint buffers as well as the endpoint buffers themselves.

3.2.3 Peripheral Space

The peripheral space area of memory is intended for mapping of control registers, internal buffers/working space, and other features needed for the firmware control of non-core peripherals. It is defined from byte address range 0x4000_0000 to 0x5FFF_FFFF (0.5GB maximum). On the **MAX32600**, this includes such functions as the [ADC/DAC](#), [TPU](#) ([MAA](#), [AES](#), etc.), [UART](#) interfaces, [I²C](#), [USB](#), etc.

As with the SRAM region, there is a dedicated 1MB area at the bottom of this memory region (from 0x4000_0000 to 0x400F_FFFF) that is used for bit-banding operations by the ARM core. Read/write operations in the peripheral bit-banding alias area (32MB in length, from 0x4200_0000 to 0x43FF_FFFF) are translated by the core into read/mask/shift or read/modify/write operation sequences to the appropriate location in the bit-banding area.

Note The bit-banding operation within peripheral memory space is, like bit-banding function in SRAM space, a core remapping function, and it is only applicable to operations performed directly by the ARM core. If another memory bus master accesses the peripheral bit-banding region (e.g., the JTAG/PTP AHB master or the PMU AHB master), the bit-banding operation will not take place, and the bit-banding alias region will appear to be a non-implemented memory area (causing an AHB bus error).

On the **MAX32600**, access to the region that contains most peripheral registers (0x4000_0000 to 0x400F_FFFF) goes from the AHB bus through an AHB-to-APB bridge. This allows the peripheral blocks to operate on the slower, easier to handle APB bus matrix while also ensuring that peripherals with slower response times do not tie up bandwidth on the AHB bus, which must necessarily have a faster response time since it handles main application instruction and data fetching.

A secondary region within the peripheral memory space is set aside for other peripherals that require more rapid data transfer to implement direct access AHB slave instances (0x4010_0000 to 0x401F_FFFF). This area is used so that peripherals which have FIFOs or other functions requiring large amounts of data to be transferred quickly (such as the ADC and DACs) can benefit from the more rapid transfers available on the AHB bus.

3.2.4 External RAM Space

The external RAM space area of memory is intended for use in mapping off-chip external memory and is defined from byte address range 0x6000_0000 to 0x9FFF_F_FFFF (1GB maximum). The **MAX32600** does not support external RAM space.

3.2.5 External Device Space

The external device space area of memory is intended for use in mapping off-chip device control functions onto the AHB bus and is defined from byte address range 0xA000_0000 to 0xDFFF_FFFF (1GB maximum). The **MAX32600** does not implement this memory area.

3.2.6 System Area (Private Peripheral Bus)

The system area (private peripheral bus) memory space contains register areas for functions that are only accessible by the ARM core itself (and the ARM debugger, in certain instances). It is defined from byte address range 0xE000_0000 to 0xE00F_FFFF. This APB bus is restricted to core access; it cannot be accessed by other AHB memory masters, such as the DMA or the JTAG/PTP bus master.

In addition to being restricted to the core, application code is only allowed to access this area when running in the privileged execution mode (as opposed to the standard user thread execution mode). This helps ensure that critical system settings controlled in this area are not altered inadvertently or by errant code that should not have access to this area.

Core functions controlled by registers mapped to this area include the SysTick timer, debug and tracing functions, the NVIC (interrupt handler) controller, and the Flash Breakpoint controller.

3.2.7 System Area (Vendor Defined)

The system area (vendor defined) memory space is reserved for vendor (system integrator) specific functions that are not handled by another memory area. It is defined from byte address range 0xE010_0000 to 0xFFFF_FFFF. The **MAX32600** does not include this memory region.

3.3 Device Memory Instances

This section details physical memory instances on the **MAX32600** (including main program flash and SRAM instances) that are accessible as standalone memory regions using either the AHB or APB bus matrix. Memory areas which are only accessible via FIFO interfaces, or memory areas consisting of only a few registers for a particular peripheral, are not covered here.

3.3.1 Main Program Flash Memory

The main program flash memory is 256KB in size and consists of 2KB (or 512 instruction words) logical pages.

3.3.2 Instruction Cache Memory

The instruction cache is 2KB in size (64 rows x 256 bits) and is used to cache instructions from the main flash memory after they have been descrambled. Note that the cache is used for instruction fetches only; data fetches from the flash will always go directly to the flash contents, bypassing the Instruction Cache Memory.

3.3.3 Information Block Flash Memory

The information block is a separate flash instance with a single 2KB page. It is used to store trim settings (option configuration and analog trim) as well as other device-specific information designed to be readable by firmware that needs to be preserved across main application flash load/erase cycles.

3.3.4 System SRAM

The system SRAM is 32KB in size and can be used for general purpose data storage, the ARM system stack, USB data transfers (endpoints), and code execution if desired.

3.3.5 AES Key and Working Space Memory

The AES key memory and working space for AES operations (including input and output parameters) are located in a dedicated register file memory tied to the AES engine block. This AES memory is mapped into AHB space for rapid firmware access. In the event of a tamper detection, the AES memory will be automatically erased by hardware.

3.3.6 Modular Arithmetic Accelerator (MAA) Key and Working Space Memory

The MAA also contains a dedicated memory for key storage, input and output parameters for operations, and working space. It is mapped into the AHB memory space for ease of loading and unloading.

3.3.7 TPU Memory Secure Key Storage Area

The **MAX32600** contains a specialized 128-bit memory that is designed to preserve a critical key (such as an AES key) even when the device is in the lowest power-saving state. As long as the RTC power supply is still available, this key will be retained, even if the AES block and the main SRAM are shut down completely. In the event of a tamper response, this key will be automatically erased by hardware.

The Secure Key Storage Area consists of four V_{RTC}-backed 32-bit registers: [TPU_TSR_SKS0](#), [TPU_TSR_SKS1](#), [TPU_TSR_SKS2](#), and [TPU_TSR_SKS3](#).

3.4 AHB Bus Matrix and AHB Bus Interfaces

This section details memory accessibility on the AHB bus matrix and the organization of AHB master and slave instances.

3.4.1 Core AHB Interface - I-Code

This AHB master is used by the ARM core for instruction fetching from the code space. This bus master has access to the main flash program memory and the main system SRAM.

3.4.2 Core AHB Interface - D-Code

This AHB master is used by the ARM core for data fetches from the code space. This bus master has access to the main flash program memory, the information block (when it is not locked), and the main system SRAM.

3.4.3 Core AHB Interface - System

This AHB master is used by the ARM core for other data read and write operations involving the system SRAM, the APB mapped peripherals (through the AHB-to-APB bridge), and AHB mapped peripheral and memory areas.

3.4.4 AHB Master - Peripheral Management Unit (PMU)

The PMU bus master has access to all off-core memory areas (equivalent to the System plus D-Code) with the exception of the USB AHB memory mapped area. The PMU bus master does not have access to the ARM Private Peripheral Bus area.

3.4.5 AHB Master - USB Endpoint Buffer Manager

The USB AHB bus master is used to manage endpoint buffers in the main system SRAM. It has access to the main system SRAM and flash main memory.

4 System Configuration and Management

4.1 Power Ecosystem and Operating Modes

4.1.1 Power Ecosystem

The **MAX32600** has multiple operating modes with many user configurable options offering significant flexibility in total power consumption. These options are stored in the data retention power domain registers and are continuously powered across all modes of operation. The registers dictate which analog and digital peripherals are intended to remain enabled during low power modes. Likewise, there are dedicated system registers that dictate the configuration of features during run modes.

The **MAX32600** supports four power modes, **LP0: STOP**; **LP1: STANDBY**; **LP2: PMU**; and **LP3: RUN**. The [Power State Diagram](#) shows a state diagram of these power modes.

The low power modes, **LP0: STOP** and **LP1: STANDBY**, are under the control of the Power Sequencer while **LP2: PMU** is controlled by the **PMU**, and the **LP3: RUN** mode is controlled by the ARM core.

The V_{RTC} power pad (powered by battery or super cap) ensures that this domain is always on during battery change or other loss-of-power events on the main V_{DD} supply.

When a wakeup event is detected, the **MAX32600** exits the low power mode (**LP0: STOP** or **LP1: STANDBY**) and always enters **LP3: RUN** where firmware takes over control of the system and power states.

Note Power mode transition restrictions dictate measurement sequences. Further transitioning information is found in the power mode sections below. The following is a typical measurement sequence: LP0/LP1 → LP3 → LP2 → LP3 → LP0/LP1

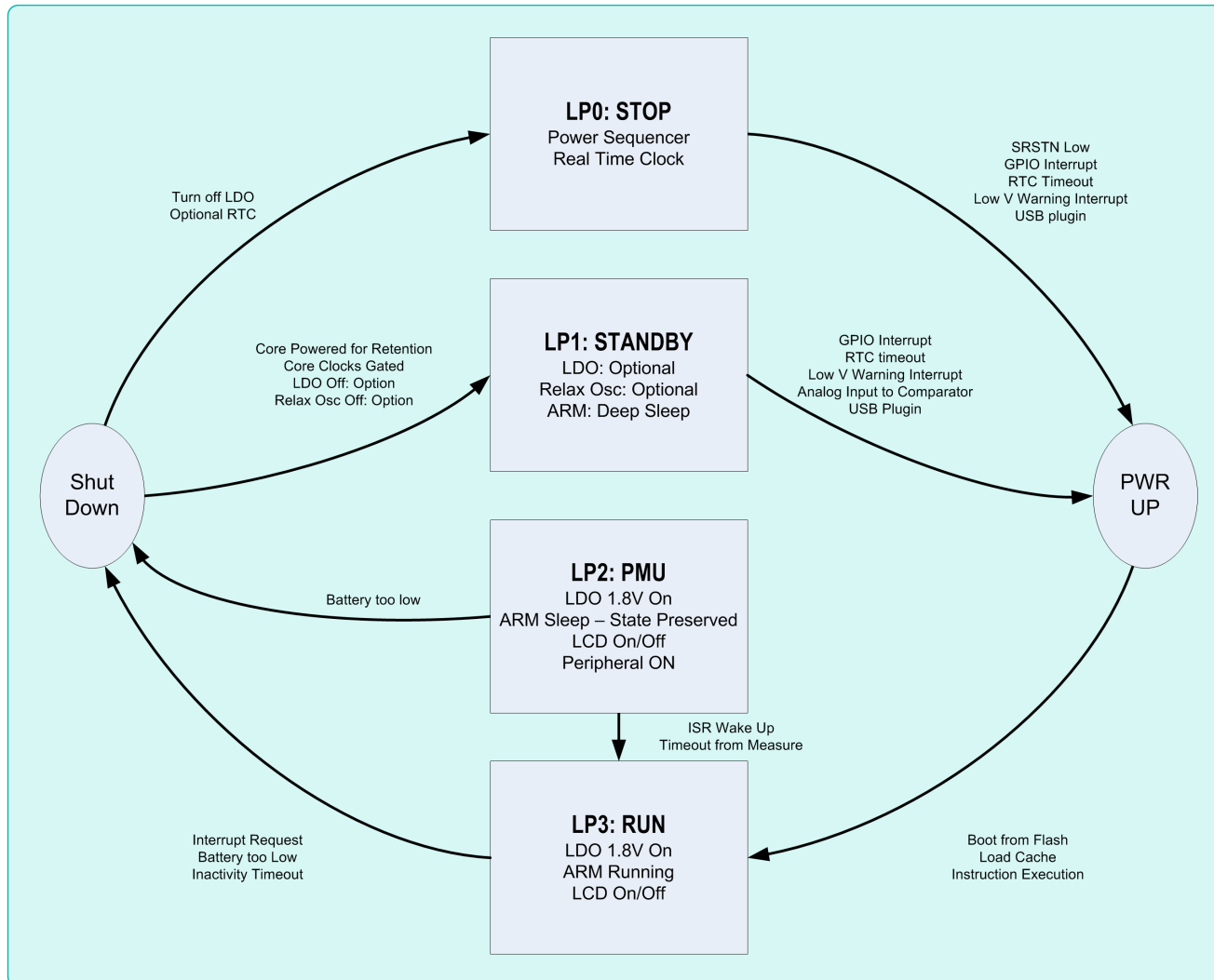


Figure 4.1: Power State Diagram

4.1.2 Low Power Modes (LP0: STOP and LP1: STANDBY)

Note Low Power Modes LP0 and LP1 can only transition to/from LP3: RUN; to enter LP2: PMU from one of these modes, the device must first enter LP3: Run.

4.1.2.1 Low Power Mode 0 (LP0: STOP)

LP0: STOP is the lowest power mode supported by the **MAX32600**: using as little as 850nA in normal operation and as little as 1.25 μ A with the RTC active. The core system registers and SRAM do not retain state and, upon exit from LP0: STOP, the system starts as if from a core reset. The sections of the **MAX32600** that maintain state during LP0: STOP are:

- Power Sequencer
- Real-Time Clock (RTC)
- Data retention registers
- POR/Failsafe

4.1.2.2 Low Power Mode 1 (LP1: STANDBY)

LP1: STANDBY is core data retention mode and supports fast wakeup time ($\sim 15\mu$ s typical) while maintaining ultra-low power. Typically, the **MAX32600** draws only 1.4 μ A in LP1: STANDBY without the RTC and only 1.8 μ A with the RTC enabled. All clocks are gated off, the core logic is in a static state, and the ARM is in deep sleep. SRAM and registers have full data retention. Many options for analog support circuitry are available in this mode and, to achieve the lowest possible power in LP1: STANDBY, it is recommended to turn off all unused analog circuitry. This includes the core 1.8V LDO because the Data Retention LDO is used to provide data retention.

To achieve the 15 μ s fast resume from LP1: STANDBY, a power-up sequence has been implemented. Instead of waiting for the 1.8V LDO voltage to become stable, a gear shift mechanism is used that allows the core to run with a divided clock while the 1.8V LDO ramps and stabilizes; when the LDO is at a safe voltage, the system shifts gears to the normal operating frequency.

Optional improvements to wakeup time can be achieved at the expense of burning more power. This may be appropriate for short periods of time in low power modes. In this situation, continuous power consumption of run state is avoided, but extremely fast wake up times from a low power mode may be needed. To accelerate the resume time, the 1.8V LDO and/or Relaxation Oscillator can be enabled during LP1: STANDBY. Power consumption during LP1: STANDBY will be higher, but immediate resume times can be achieved under these conditions.

If the Relaxation Oscillator is not enabled during LP1: STANDBY, a 2 μ s timeout is necessary due to waiting for the oscillator to stabilize prior to beginning code execution.

4.1.2.3 Entering LP0: STOP or LP1: STANDBY

The following illustrates the procedure to change the **MAX32600** operating state to either LP0: STOP or LP1: STANDBY. The examples used set up a GPIO wakeup on P0.0 that wakes up on an active high. Although P0.0 is used here, any available port.pin can be configured in this manner for a GPIO wakeup event.

Note The following steps to enter LP0: STOP and LP1: STANDBY should be used as guidelines; for best results, the user should always reference the appropriate API.

Entering LP0: STOP

In order to change the **MAX32600** operating state to LP0: STOP, the following steps should be followed:

- Configure P0.0 to be WUD Mode by setting [PWRMAN_WUD_CTRL.pad_select](#)
- Enable Active High WUD on P0.0 by setting [PWRMAN_WUD_CTRL.pad_mode](#)
- Activate WUD by setting bit 0 of [PWRMAN_WUD_PULSE0](#)
- Assert GPIO Freeze by setting [PWRSEQ_REG1.pwr_gpio_freeze](#) to 1
- Ensure pads are in the lowest power state by clearing the [PWRMAN_PWR_RST_CTRL.io_active](#) register
- Clear all flags in [PWRSEQ_FLAGS](#) register
- Set Run/Sleep mode of peripherals in [PWRSEQ_REG0](#)
- Set LP0 mode in [PWRSEQ_REG0](#)
- CM3_PWRMAN bit 2 to 1; Arm Command WFE – ARM command puts **MAX32600** in LP0 mode

Entering LP1: STANDBY

In order to change the **MAX32600** operating state to LP1: STANDBY, the following steps should be followed:

- Configure the Power Sequencer for quick resume by setting desired clocks in [PWRSEQ_REG3.pwr_rose1](#) to 64
- Configure P0.0 to be WUD Mode by setting [PWRMAN_WUD_CTRL.pad_select](#)
- Enable Active High WUD on P0.0 by setting [PWRMAN_WUD_CTRL.pad_mode](#)
- Activate WUD by setting bit 0 of [PWRMAN_WUD_PULSE0](#)

- Assert GPIO Freeze by setting `PWRSEQ_REG1.pwr_gpio_freeze` to 1
- Clear all flags in `PWRSEQ_FLAGS` register
- Set Run/Sleep mode of peripherals in `PWRSEQ_REG0`
- Set LP1 mode in `PWRSEQ_REG0`
- CM3_PWRMAN bit 2 to 1; Arm Command WFE – ARM command puts **MAX32600** in LP1 mode

Note CM3_PWRMAN bit 2 is the 'Deep Sleep' bit of the ARM Cortex-M3 System Control register. Setting this bit to 1 indicates to the system that the Cortex-M3 clock can be stopped; the 'Deep Sleep' port will be asserted when the processor can be stopped. Setting this bit to 0 prevents turning off the system clock.

4.1.2.4 Wakeup Events from LP0: STOP and LP1: STANDBY

The following events can wake up the **MAX32600** from the Low Power states:

- RTC timer interrupt
 - Timer has 244 μ s resolution
- GPIO sensed high/low (All GPIO are wakeup capable as programmed by firmware)
- Analog input to a comparator
- USB plugin/remove
- Supply Voltage Monitor (SVM) low voltage condition sensed via periodic or continuous monitoring

Each of these events is configurable and must be enabled by the firmware.

Note Certain wakeup events can be masked out by writing to the `PWRSEQ_MSK_FLAGS` register.

After Wakeup Events

After the **MAX32600** experiences a wakeup event from LP0: STOP or LP1: STANDBY, proceed with the following actions:

LP0 Wakeup

- Read `PWRSEQ_FLAGS` register to determine the source of the wakeup event

- If the wakeup event was a GPIO event, do one of the following to clear the GPIO WUD:
 1. If desired action is to clear *all* GPIO WUD latches:
 - Clear all GPIO flags by writing 1 to [PWRMAN_WUD_CTRL.clear_all](#) to clear all GPIO WUD setups
 - Take all pads out of the low power state by setting the [PWRMAN_PWR_RST_CTRL.io_active](#) register to 1
 - Deassert GPIO Freeze by setting [PWRSEQ_REG1.pwr_gpio_freeze](#) to 0
 - Clear all flags in [PWRSEQ_FLAGS](#) register
 2. If desired action is to clear *individual* GPIO WUD latches that initiated the wakeup event:
 - Set WUD Pad Select for the individual port in the [PWRMAN_WUD_CTRL.pad_select](#) register
 - Set WUD Pad Signal Mode by writing 2 to [PWRMAN_WUD_CTRL.pad_mode](#)
 - Set WUD Pulse 0 register, [PWRMAN_WUD_PULSE0](#), to 1. This register self-clears.
 - Take all pads out of the low power state by setting the [PWRMAN_PWR_RST_CTRL.io_active](#) register to 1
 - Deassert GPIO Freeze by setting [PWRSEQ_REG1.pwr_gpio_freeze](#) to 0
 - Clear all flags in [PWRSEQ_FLAGS](#) register

LP1 Wakeup

- Read [PWRSEQ_FLAGS](#) register to determine the source of the wakeup event
- If the wakeup event was a GPIO event, it is recommended that GPIO flags are cleared individually. Clearing of *all* GPIO WUD latches is not recommended when waking up from LP1.
 - Set WUD Pad Select for the individual port in the [PWRMAN_WUD_CTRL.pad_select](#) register
 - Set WUD Pad Signal Mode by writing 2 to [PWRMAN_WUD_CTRL.pad_mode](#)
 - Set WUD Pulse 0 register, [PWRMAN_WUD_PULSE0](#), to 1. This register self-clears.
 - Take all pads out of the low power state by setting the [PWRMAN_PWR_RST_CTRL.io_active](#) register to 1
 - Deassert GPIO Freeze by setting [PWRSEQ_REG1.pwr_gpio_freeze](#) to 0
 - Clear all flags in [PWRSEQ_FLAGS](#) register

4.1.3 Low Power Modes (LP2: PMU and LP3: RUN)

The [Peripheral Management Unit \(PMU\)](#) is in control of the system when using LP2: PMU. During LP3: RUN, the ARM core is in control of the system. A System Management unit controls entering and exiting both modes.

When operating in either mode, firmware is in control of the power used by the system. Firmware controls clock gating, peripheral enables, and the [Analog Front End \(AFE\)](#).

4.1.3.1 Low Power Mode 2 (LP2: Peripheral Management Unit)

The [Peripheral Management Unit \(PMU\)](#) is a DMA engine for the **MAX32600**. It enables the lowest noise floor for analog measurements and reduced operating power for peripherals. The PMU acts like an internal state machine that can orchestrate events via programmable [op codes](#) for:

- Peripheral to Memory
- Memory to Peripheral
- Analog to Memory
- Memory to Analog
 - Synchronization of analog measurements
 - Control and synchronization of pulse train signals and events

During this state, the ARM Cortex-M3 has relinquished control to the PMU. The ARM Cortex-M3 is in sleep mode, resulting in power reduction and noise minimization. To further reduce power consumption and noise, only the required clocks and data buses are active during LP2: PMU. Typically, the **MAX32600** draws 1.2mA of current (24MHz clock) with a single channel of the PMU active. Each additional channel draws approximately 100 μ A of current.

4.1.3.2 Low Power Mode 3 (LP3: RUN)

LP3: RUN mode is under full firmware control using the ARM core. During this state, the ARM Cortex-M3 and the digital core are fully powered and awake. The clocks to each peripheral are gated dynamically and only clock the circuitry in use. Firmware executes from the internal flash memory based on the instruction cache. Typical power of 175 μ A/MHz is achieved in LP3: RUN. This is the default power up state.

Reference [First Boot Power Up](#) for detailed information regarding entering LP3: RUN from an initial boot up.

4.1.4 Power State Matrix Control Options

The [Power State Diagram](#) depicts the four major power states and how control is handled in the **MAX32600**.

The [figure](#) illustrates:

- Hardware controlled powering of circuit blocks
- Firmware controllable power options
- Firmware controllable clock gating

Note The chip hardware will power the minimal amount of circuitry necessary to achieve functionality of each mode. To achieve the lowest optimized power solution, the user must fully analyze the usage case and choose the appropriate power and clock gating options.

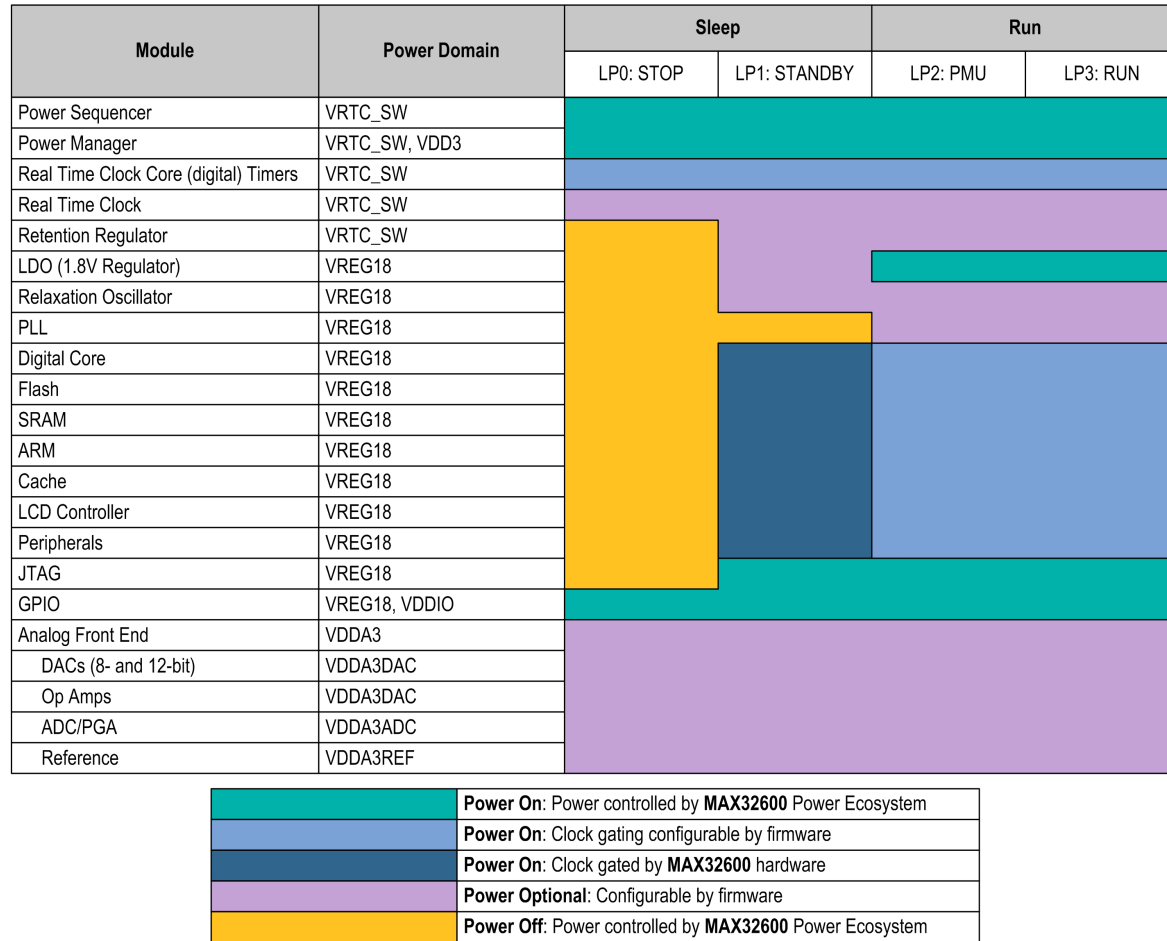


Figure 4.2: Power and Clock Gating Options

4.1.5 Power Domains

The **MAX32600** has multiple power domains that are controlled by the power management block. This includes the Power Sequencer, Power Manager, Trickle Charger, 1.8V LDO, 3.3V USB LDO, and Real Time Clock (RTC). The configuration registers for the Power Manager are within the battery backed V_{RTC} domain. The registers are configurable by firmware and dictate what analog and digital peripherals are enabled while in each of the Operating Modes, LP0: STOP; LP1: STANDBY; LP2: PMU; and LP3: RUN.

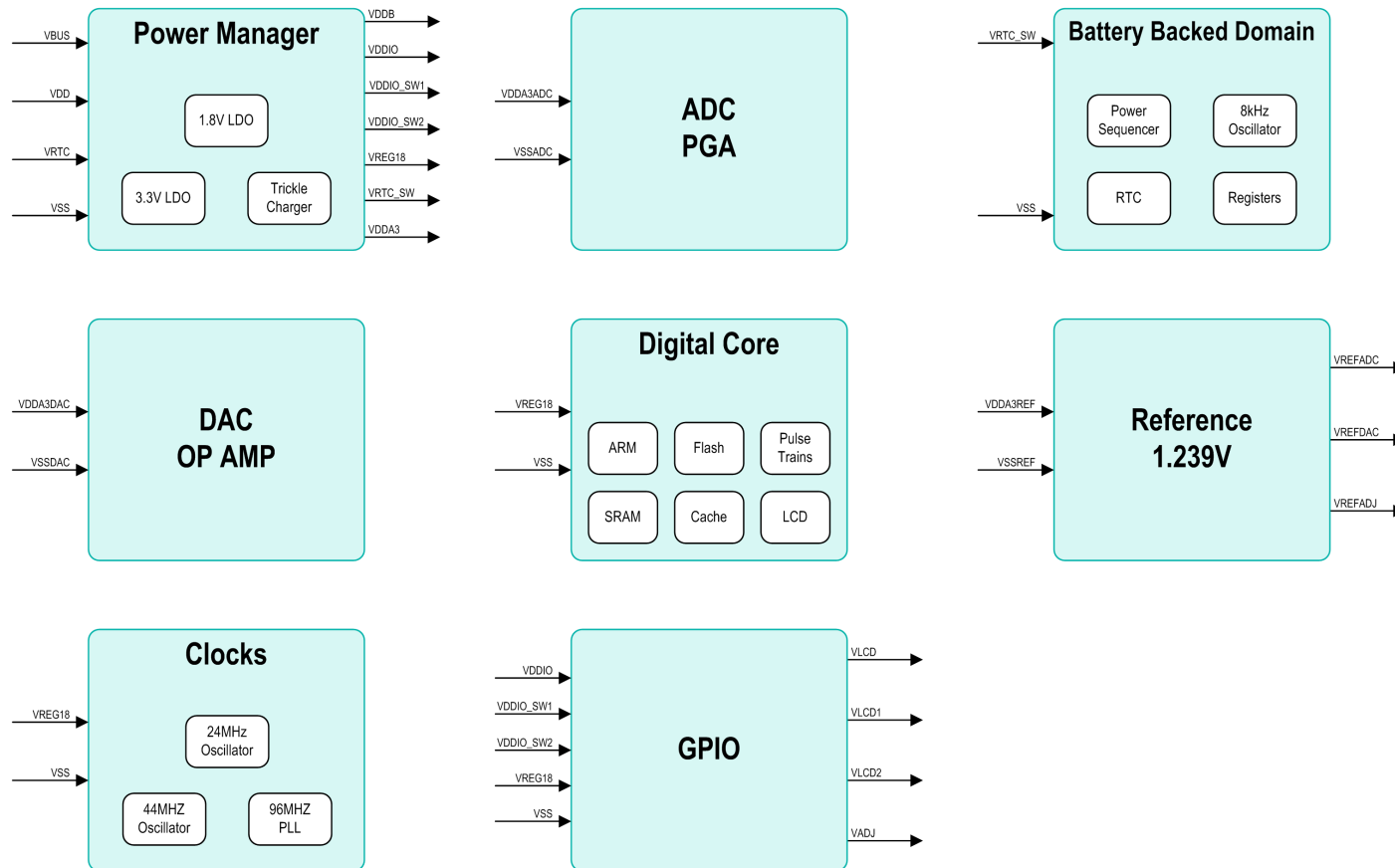


Figure 4.3: Power Domains

Reference [Power Pins](#) below for a more detailed description of the power I/O.

4.1.6 Power Manager

The Power Manager monitors and distributes the three main voltage rail inputs (V_{BUS} , V_{DD} , and V_{RTC}) on the **MAX32600**. The analog power domain inputs ($V_{DDA3ADC}$, $V_{DDA3DAC}$, and $V_{DDA3REF}$) are separate pins that allow the user to provide external isolation from each domain. The V_{DDA3} analog power output can be shorted to any/all of the analog power domain inputs or optionally connected through filtering networks for isolation. In addition to power distribution and management, the Power Manager monitors power levels using several Supply Voltage Monitors (SVMs) and Power-On Resets (PORs). SVM warning levels are programmable (see section [Supply Voltage Monitoring During LP0: STOP and LP1: STANDBY](#) for details). POR levels are not changeable and are set at the factory per the **MAX32600** data sheet.

4.1.7 Power Sequencer

The Power Sequencer controls the **MAX32600** during [Low Power Modes \(LP0: STOP and LP1: STANDBY\)](#). When LP0: STOP and LP1: STANDBY are exited, the Power Sequencer transfers control to the system manager and the part enters either LP2: PMU or LP3: RUN. One of the primary functions of the Power Sequencer is to ensure power, clocks, and resets are stable prior to transitioning the system to either LP2: PMU or LP3: RUN. During LP0: STOP and LP1: STANDBY, wakeup interrupts are continuously monitored while consuming very little power. Once an interrupt event occurs, the Power Sequencer automatically enables SVMs as well as active LP2: PMU and LP3: RUN peripherals, including the internal 1.8V regulator and the 24MHz Relaxation Oscillator.

When the Power Sequencer determines the power, clocks, and resets are valid, the clock gating circuitry is disabled and the **MAX32600** is allowed to enter LP2: PMU or LP3: RUN.

4.1.7.1 Power Mode Transitioning to Low Power Modes

To take full advantage of the low power modes of operation in the **MAX32600**, application firmware will need to spend as much time as possible in either LP0: STOP and LP1: STANDBY modes. Prior to entering these modes, it is extremely important to set up the wakeup interrupts and supply voltage monitor configuration. To enter LP0: STOP, reference section [Entering LP0: STOP or LP1: STANDBY](#) above.

4.1.7.2 Supply Voltage Monitoring During LP0: STOP and LP1: STANDBY

During LP0: STOP or LP1: STANDBY modes, the Power Sequencer can be set to monitor the Supply Voltage Monitors (SVM) periodically. Periodic monitoring enables the system to check to ensure power is adequate at a specific interval and reduces overall system power significantly. To enable periodic monitoring, set the field [PWRSEQ_REG3.pwr_svmssel](#) to a desired count value and enable one or more SVMs during LP0: STOP or LP1: STANDBY modes.

SVM Options for LP0 / LP1	Description	Enable Field
V_{DD}	3V Main Supply	PWRSEQ_REG0.pwr_svm3en_slp

SVM Options for LP0 / LP1	Description	Enable Field
V _{REG18}	1.8V Regulated Output	PWRSEQ_REG0.pwr_svm1en_slp
V _{RTC}	3V VRTC Supply	PWRSEQ_REG0.pwr_svmrtcen_slp

4.1.7.3 SVM Periodic Monitoring

For periodic SVM during STOP or STANDBY modes, a clock source for the periodic timer should be enabled. The 8kHz oscillator or the RTC clock can be chosen as the timer source. For the 8kHz nano ring, set the [PWRSEQ_REG0.pwr_nren_slp](#) bit; to use the RTC, set the [PWRSEQ_REG0.pwr_rtcent_slp](#) bit.

8kHz Oscillator

The 8kHz Oscillator is a low-power internal oscillator with a nominal frequency of 8kHz that allows periodic supply voltage monitoring during [Low Power Modes \(LP0: STOP and LP1: STANDBY\)](#).

4.1.7.4 First Boot Power Up

When initial power is applied, the [PWRSEQ_FLAGS.pwr_first_boot](#) flag will be set to indicate the **MAX32600** is powering up from a first boot condition. The [pwr_first_boot](#) register must be set to 0 by firmware prior to attempting to enter either [Low Power Modes \(LP0: STOP and LP1: STANDBY\)](#). All power lost on the chip or asserting RSTN will reset the [pwr_first_boot](#) flag.

First Boot Up (entering LP3: RUN)

- Ensure pads are in the lowest power state by clearing the [PWRMAN_PWR_RST_CTRL.io_active](#) register
- Clear I/O WUD on all pads by setting and clearing the [PWRMAN_PWR_RST_CTRL.wud_clear](#) register
- Take all pads out of low power state by setting [PWRMAN_PWR_RST_CTRL.io_active](#) to 1
- Deassert GPIO Freeze by setting [PWRSEQ_REG1.pwr_gpio_freeze](#) to 0
- Clear the first boot signal, [PWRSEQ_REG0.pwr_first_boot](#)
- Set the quick count bit in the the [PWRSEQ_REG3.pwr_rose1_quick](#) to 1
- The **MAX32600** is in LP3: RUN

After boot/resume for LP0: STOP or LP1: STANDBY, firmware should poll the `PWRSEQ_FLAGS` register to determine the source of the wakeup event. After determining the source, firmware should clear the specific `PWRSEQ_FLAGS` register prior to attempting to enter either LP0: STOP or LP1: STANDBY.

Reference [Wakeup Events from LP0: STOP and LP1: STANDBY](#) for a more detailed description of this process.

4.1.7.5 Brownout Detector

The Brownout Detector detects a failing power supply and sends a NMI to the ARM processor to shut down the **MAX32600** before losing and/or corrupting core data. The Brownout Detection monitors the power supply for a $\sim 700\text{mV}$ drop and, if detected, sends the NMI to the ARM. The amount of time the supply must be below the detection level is configurable on the **MAX32600** and can be set by firmware as shown in the following [table](#). This setting indicates the minimum window of detection that the Brownout Detector will see as a brownout.

Brownout Detection Window	<code>PWRSEQ_FLAGS.pwr_brownout_det</code> Setting
10uS	00b
50uS	01b
250uS	10b
750uS	11b (Default)

Note The brownout detect flag, `PWRSEQ_FLAGS.pwr_brownout_det`, is disabled by default and does not need to be cleared by firmware on power up; following any reset condition occurring while the brownout detect flag was enabled, it should be checked to determine if a brownout was detected.

4.1.8 Trickle Charger

When a super capacitor is connected to the V_{RTC} rail, the Trickle Charger can be enabled to charge the super capacitor from the main supply (V_{DD}) or from the USB supply (V_{USB}). Several charging options are available, including three different charging speeds as well as the optional addition of a series protection diode. Users can choose from 250 Ohm, 2K Ohm, or 4K Ohm series resistance between the power source voltage and V_{RTC} -connected super capacitor. In addition, a series diode can be enabled between power rails.

4.1.8.1 Trickle Charger Configuration

To enable and configure the trickle charger, series resistance, and protection diode, set the `PWRSEQ_REG1.pwr_trikl_chrg` field as shown in the [table](#) below.

Trickle Charge Setting	<code>pwr_trikl_chrg</code> Setting (8-bits)
No Diode, 250 Ohm series resistor	0xA5
No Diode, 2K Ohm series resistor	0xA6

Trickle Charge Setting	pwr_trikl_chrg Setting (8-bits)
No Diode, 4K Ohm series resistor	0xA7
Protection Diode, 250 Ohm series resistor	0xA9
Protection Diode, 2K Ohm series resistor	0xAA
Protection Diode, 4K Ohm series resistor	0xAB

If the Trickle Charger is enabled while USB power is available, the super capacitor will be charged using the USB power source. Otherwise, when enabled, the Trickle Charger will charge the super capacitor from the main V_{DD} supply.

Note When using a super capacitor, the **MAX32600** cannot go into LP1 until the the super capacitor has been been sufficiently charged. Poll the V_{RTC} Warning Level (user configurable) to set the appropriate level.

4.1.9 Low-Dropout Regulators (LDO)

4.1.9.1 1.8V LDO

The 1.8V LDO is the source of the V_{REG18} rail and powers the digital core when in LP2: PMU and LP3: RUN. It can source power from either the USB supply or the main V_{DD} supply. If the USB supply is available, the 1.8V regulator will automatically switch to USB as its source. The 1.8V regulator supports a 50mA maximum capacity (40mA maximum external capacity) over an input range of 2.2V to 3.6V and a 30mA maximum capacity (20mA maximum external capacity) over an input range of 2.0V to 2.2V. Once the LDO input voltage reaches 1.95V and below, the V_{REG18} output voltage will attempt to equal the input voltage. V_{REG18} can be supplied to external components, but the maximum external current should not be violated.

Retention Regulator

The Retention Regulator sources power to the V_{REG18} power rail when the **MAX32600** is in LP1: STANDBY. Optionally, the Data Retention Regulator can be enabled during LP2: PMU and LP3: RUN modes while the 1.8V LDO is enabled. No external loads should be placed on the Retention Regulator during LP1: STANDBY while the 1.8V LDO is disabled. The Retention Regulator is only capable of maintaining data, so any external load may cause a power fail event to occur.

4.1.9.2 3.3V USB LDO

The 3.3V USB LDO sources its power from the 5V USB power and outputs 3.3V via V_{DDB} . By default, the **MAX32600** will be powered from the USB supply when it is available. To disable this, refer to [PWRSEQ_REG4.pwr_usb_ido_off](#) and [PWRSEQ_REG4.pwr_usb_frc_vdd](#) register fields.

4.1.10 Reset Pins

The **MAX32600** contains two active low reset pins, RSTN and SRTSN. RSTN serves as the main chip reset input. Asserting the pin low will reset all registers on the chip except RTC related circuits and wakeup configuration. This allows a restart of all chip functions (analog and digital) while still maintaining the Real Time Clock.

Note Asserting RSTN will turn off the 1.8V LDO but external capacitance will keep the V_{REG18} rail and SRAM data intact during a momentary reset pulse. A continued RSTN assertion will keep the 1.8V LDO off, eventually causing the V_{REG18} rail to collapse and lose all SRAM data.

SRSTN is a bidirectional reset that will perform a reset to the digital core when asserted low and will subsequently drive a reset pulse to other components in the system.

To fully reset the entire chip including RTC related circuitry, all three main power inputs (V_{BUS} , V_{DD} , V_{RTC}) must be powered down.

4.1.11 Power Pins

Pin Name	Description
V_{DD}	Main chip power input. Connect to 3V nominal power supply or battery. This pin must be connected to V_{SS} through a 4.7 μ F capacitor.
V_{BUS}	USB power input. Connect a 5V nominal power supply, typically USB power. This pin must be connected to V_{SS} through a 4.7 μ F capacitor.
V_{DDB}	Output of USB 5V -> 3.3V LDO. This pin must be connected to V_{SS} through a 4.7 μ F capacitor
V_{RTC}	Backup rail or "Last Man Standing" rail. Connect to super capacitor or 3V nominal power supply or battery. This pin must be connected to V_{SS} through a 1.0 μ F capacitor if connected to a 3V nominal power supply or battery. Connect to V_{DD} if backup rail is not used.
V_{DDIO}	3V nominal GPIO power. This pin (connected to Ports 6, 7) must be connected to V_{SS} through a 4.7 μ F capacitor (if the 12mm x 12mm package is used). Up to 50mA may be sourced for external components. <i>Note: On the 12mm x 12mm package, all GPIOs are powered by V_{DDIO}.</i>
V_{DDIO_SW1}	3V / 1.8V nominal GPIO power. Bank of GPIOs (connected to Ports 0, 1 on the 12mm x 12mm package and Port 1 on the 7mm x 7mm package) using V_{DDIO} (3V nominal) or V_{REG18} (1.8V nominal) as a power source. This pin must be connected to V_{SS} through a 1.0 μ F capacitor.
V_{DDIO_SW2}	3V / 1.8V nominal GPIO power. Bank of GPIOs (connected to Ports 2, 3, 4, 5 on the 12mm x 12mm package, Port 2 on the 7mm x 7mm package, and JTAG) using V_{DDIO} (3V nominal) or V_{REG18} (1.8V nominal) as a power source. This pin must be connected to V_{SS} through a 1.0 μ F capacitor.
V_{DDA3}	Analog power source (3V nominal). Option to connect filter network between V_{DDA3} and $V_{DDA3ADC}$ / $V_{DDA3DAC}$ / $V_{DDA3REF}$. Short to V_{DDA3} pins if filter network not needed.
$V_{DDA3ADC}$	ADC power input (3V nominal). Connect to V_{DDA3} directly or via a filter network.

Pin Name	Description
V _{DDA3DAC}	DAC / op amp power input (3V nominal). Connect to V _{DDA3} directly or via a filter network.
V _{DDA3REF}	Reference power input (3V nominal). Connect to V _{DDA3} directly or via a filter network.
V _{REFADC}	ADC reference voltage output. Buffered output can be set to 1.024V, 1.5V, 2.048V, and 2.5V. This pin must be connected to V _{SS} through a 4.7μF capacitor.
V _{REFDAC}	DAC reference voltage output. Buffered output can be set to 1.024V, 1.5V, 2.048V, and 2.5V. This pin must be connected to V _{SS} through a 4.7μF capacitor.
V _{REFADJ}	Precision reference input that may be optionally used in place of internal reference by V _{REFADC} and V _{REFDAC} buffers. Ground if not used.
V _{SS}	Digital Ground. Tie all grounds together on circuit board.
V _{SSUB}	Substrate Ground.
V _{SSREF}	Reference Ground.
V _{SSADC}	ADC Ground.
V _{SSDAC}	DAC Ground.

Note All grounds are assumed to be tied together at the circuit board level.

4.1.12 Registers (PWRMAN)

4.1.12.1 Module PWRMAN Registers

Address	Register	32b Word Len	Description
0x40090800	PWRMAN_PWR_RST_CTRL	1	Power Reset Control and Status
0x40090804	PWRMAN_INTFL	1	Interrupt Flags
0x40090808	PWRMAN_INTEN	1	Interrupt Enable/Disable Controls
0x4009080C	PWRMAN_SVM_EVENTS	1	SVM Event Status Flags (read-only)
0x40090810	PWRMAN_WUD_CTRL	1	Wake-Up Detect Control
0x40090814	PWRMAN_WUD_PULSE0	1	WUD Pulse To Mode Bit 0
0x40090818	PWRMAN_WUD_PULSE1	1	WUD Pulse To Mode Bit 1
0x40090830	PWRMAN_WUD_SEEN0	1	Wake-up Detect Status for P0/P1/P2/P3
0x40090834	PWRMAN_WUD_SEEN1	1	Wake-up Detect Status for P4/P5/P6/P7

Address	Register	32b Word Len	Description
0x40090838	PWRMAN_DIE_TYPE	1	Die Type ID Register
0x4009083C	PWRMAN_BASE_PART_NUM	1	Base Part Number
0x40090840	PWRMAN_MASK_ID0	1	Mask ID Register 0
0x40090844	PWRMAN_MASK_ID1	1	Mask ID Register 1
0x40090848	PWRMAN_PERIPHERAL_RESET	1	Peripheral Reset Control Register

4.1.12.1.1 PWRMAN_PWR_RST_CTRL

PWRMAN_PWR_RST_CTRL.flash_active

Field	Bits	Default	Access	Description
flash_active	0	1	R/W	Flash Active (Not Used)

No effect

PWRMAN_PWR_RST_CTRL.sram_active

Field	Bits	Default	Access	Description
sram_active	1	0	R/W	SRAM Active (Not Used)

No effect

PWRMAN_PWR_RST_CTRL.afe_powered

Field	Bits	Default	Access	Description
afe_powered	2	0	R/W	AFE Powered

- 0:Entire AFE is powered off

- 1: AFE is powered on globally; individual AFE controls may be used to power sub-features of AFE on and off as needed.

PWRMAN_PWR_RST_CTRL.io_active

Field	Bits	Default	Access	Description
io_active	3	0	R/W	I/O Active

- 0: Puts all I/O (GPIO-only) pins in the lowest power state.
- 1: All I/O pins are powered on normally.

PWRMAN_PWR_RST_CTRL.usb_powered

Field	Bits	Default	Access	Description
usb_powered	4	0	R/W	USB Powered

1: Powers on the USB block.

PWRMAN_PWR_RST_CTRL.pullups_enabled

Field	Bits	Default	Access	Description
pullups_enabled	5	1	R/W	Static Pullups Enabled

1: Enables static pullups on dedicated I/O pins.

PWRMAN_PWR_RST_CTRL.firmware_reset

Field	Bits	Default	Access	Description
firmware_reset	8	0	R/W	Firmware Initiated Reset

Initiates a system reset when set to 1. This bit is self-clearing.

PWRMAN_PWR_RST_CTRL.arm_lockup_reset

Field	Bits	Default	Access	Description
arm_lockup_reset	9	0	R/W	ARM Lockup Reset

If this bit is set to 1, a system reset will be automatically triggered when the ARM core asserts its lockup state output signal.

PWRMAN_PWR_RST_CTRL.wud_clear

Field	Bits	Default	Access	Description
wud_clear	12	0	R/W	I/O WUD Clear

When I/O Active (bit 3 of PWRMAN_RST_CNTL) is deasserted, setting this bit will clear the WUD latch in all I/O pads. This bit self clears.

PWRMAN_PWR_RST_CTRL.tamper_detect

Field	Bits	Default	Access	Description
tamper_detect	16	special	R/O	Reset Caused By - Tamper Detect

PWRMAN_PWR_RST_CTRL.fw_command_sysman

Field	Bits	Default	Access	Description
fw_command_sysman	17	special	R/O	Reset Caused By - Firmware Commanded Reset (SysMan)

PWRMAN_PWR_RST_CTRL.watchdog_timeout

Field	Bits	Default	Access	Description
watchdog_timeout	18	special	R/O	Reset Caused By - Watchdog Reset

PWRMAN_PWR_RST_CTRL.fw_command_arm

Field	Bits	Default	Access	Description
fw_command_arm	19	special	R/O	Reset Caused By - Firmware Commanded Reset (ARM Core)

PWRMAN_PWR_RST_CTRL.arm_lockup

Field	Bits	Default	Access	Description
arm_lockup	20	special	R/O	Reset Caused By - ARM Lockup

PWRMAN_PWR_RST_CTRL.srstn_assertion

Field	Bits	Default	Access	Description
srstn_assertion	21	special	R/O	Reset Caused By - External System Reset

PWRMAN_PWR_RST_CTRL.por

Field	Bits	Default	Access	Description
por	22	special	R/O	Reset Caused By - Power On Reset (POR)

PWRMAN_PWR_RST_CTRL.low_power_mode

Field	Bits	Default	Access	Description
low_power_mode	31	0	R/W	Power Manager Dynamic Clock Gating Enable

1: Enables dynamic clock gating for pwrman functions.

4.1.12.1.2 PWRMAN_INTFL

PWRMAN_INTFL.v1_8_warning

Field	Bits	Default	Access	Description
v1_8_warning	0	0	W1C	1.8V Warning Monitor Int Flag

Write 1 to clear.

Set to 1 by hardware when the associated SVM event monitor detects the monitored condition.

PWRMAN_INTFL.v3_3_warning

Field	Bits	Default	Access	Description
v3_3_warning	1	0	W1C	3.3V Warning Monitor Int Flag

Write 1 to clear.

Set to 1 by hardware when the associated SVM event monitor detects the monitored condition.

PWRMAN_INTFL.rtc_warning

Field	Bits	Default	Access	Description
rtc_warning	2	0	W1C	RTC Warning Monitor Int Flag

Write 1 to clear.

Set to 1 by hardware when the associated SVM event monitor detects the monitored condition.

PWRMAN_INTFL.v3_3_reset

Field	Bits	Default	Access	Description
v3_3_reset	3	0	W1C	3.3V Reset Monitor Int Flag

Write 1 to clear.

Set to 1 by hardware when the associated SVM event monitor detects the monitored condition.

PWRMAN_INTFL.vdda_warning

Field	Bits	Default	Access	Description
vdda_warning	4	0	W1C	VddA Warning Monitor Int Flag

Write 1 to clear.

Set to 1 by hardware when the associated SVM event monitor detects the monitored condition.

4.1.12.1.3 PWRMAN_INTEN

PWRMAN_INTEN.v1_8_warning

Field	Bits	Default	Access	Description
v1_8_warning	0	0	R/W	1.8V Warning Monitor Int Enable

0:Int disabled; 1:Interrupt enabled for the associated SVM monitor event.

PWRMAN_INTEN.v3_3_warning

Field	Bits	Default	Access	Description
v3_3_warning	1	0	R/W	3.3V Warning Monitor Int Enable

0:Int disabled; 1:Interrupt enabled for the associated SVM monitor event.

PWRMAN_INTEN.rtc_warning

Field	Bits	Default	Access	Description
rtc_warning	2	0	R/W	RTC Warning Monitor Int Enable

0:Int disabled; 1:Interrupt enabled for the associated SVM monitor event.

PWRMAN_INTEN.v3_3_reset

Field	Bits	Default	Access	Description
v3_3_reset	3	0	R/W	3.3V Reset Monitor Int Enable

0:Int disabled; 1:Interrupt enabled for the associated SVM monitor event.

PWRMAN_INTEN.vdda_warning

Field	Bits	Default	Access	Description
vdda_warning	4	0	R/W	VddA Warning Monitor Int Enable

0:Int disabled; 1:Interrupt enabled for the associated SVM monitor event.

4.1.12.1.4 PWRMAN_SVM_EVENTS**PWRMAN_SVM_EVENTS.v1_8_warning**

Field	Bits	Default	Access	Description
v1_8_warning	0	n/a	R/O	1.8V Warning Monitor Event Input

Current state of the associated SVM event input.

PWRMAN_SVM_EVENTS.v3_3_warning

Field	Bits	Default	Access	Description
v3_3_warning	1	n/a	R/O	3.3V Warning Monitor Event Input

Current state of the associated SVM event input.

PWRMAN_SVM_EVENTS.rtc_warning

Field	Bits	Default	Access	Description
rtc_warning	2	n/a	R/O	RTC Warning Monitor Event Input

Current state of the associated SVM event input.

PWRMAN_SVM_EVENTS.v3_3_reset

Field	Bits	Default	Access	Description
v3_3_reset	3	n/a	R/O	3.3V Reset Monitor Event Input

Current state of the associated SVM event input.

PWRMAN_SVM_EVENTS.vdda_warning

Field	Bits	Default	Access	Description
vdda_warning	4	n/a	R/O	Vdda Warning Monitor Event Input

Current state of the associated SVM event input.

4.1.12.1.5 PWRMAN_WUD_CTRL**PWRMAN_WUD_CTRL.pad_select**

Field	Bits	Default	Access	Description
pad_select	5:0	000000b	R/W	Wake-Up Pad Select

Selects which pad to modify WUD/Weak latch states.

Pads are numbered from 0-63, where 0-7 corresponds to P0.0-P0.7, 8-15 corresponds to P1.0-P1.7, and so on.

PWRMAN_WUD_CTRL.pad_mode

Field	Bits	Default	Access	Description
pad_mode	9:8	00b	R/W	Wake-Up Pad Signal Mode

Defines WUD signal to be sent to selected pad.

- 0 = Clear/Activate WUD
- 1 = Set WUD Act Hi/Set WUD Act Lo
- 2 = Set Weak Hi/Set Weak Lo
- 3 = No pad state change

PWRMAN_WUD_CTRL.clear_all

Field	Bits	Default	Access	Description
clear_all	12	0	R/W	Clear All WUD Pad States

When set forces all pads into Clr WUD/Weak state until cleared.

4.1.12.1.6 PWRMAN_WUD_PULSE0

Default	Access	Description
n/a	W/O	WUD Pulse To Mode Bit 0

Writing to this register issues a pulse to the selected WUD pad mode[0] for one clock.

The effect on the pad behavior depends on the Wake-Up Pad Signal Mode as set in WUD_CTRL.

- 0 = Clr WUD/Weak
- 1 = Set WUD Act Hi
- 2 = Set Weak Hi
- 3 = No pad state change

4.1.12.1.7 PWRMAN_WUD_PULSE1

Default	Access	Description
n/a	W/O	WUD Pulse To Mode Bit 1

Writing to this register issues a pulse to the selected WUD pad mode[1] for one clock.

The effect on the pad behavior depends on the Wake-Up Pad Signal Mode as set in WUD_CTRL.

- 0 = WUD Activate
- 1 = Set WUD Act Lo
- 2 = Set Weak Lo
- 3 = No pad state change;

4.1.12.1.8 PWRMAN_WUD_SEEN0

PWRMAN_WUD_SEEN0.[gpio0, gpio1, gpio2, gpio3, gpio4, gpio5, gpio6, gpio7]

Field	Bits	Default	Access	Description
gpio0	0	0	R/O	Wake-Up Detect Status for P0.0
gpio1	1	0	R/O	Wake-Up Detect Status for P0.1
gpio2	2	0	R/O	Wake-Up Detect Status for P0.2
gpio3	3	0	R/O	Wake-Up Detect Status for P0.3
gpio4	4	0	R/O	Wake-Up Detect Status for P0.4
gpio5	5	0	R/O	Wake-Up Detect Status for P0.5
gpio6	6	0	R/O	Wake-Up Detect Status for P0.6
gpio7	7	0	R/O	Wake-Up Detect Status for P0.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

PWRMAN_WUD_SEEN0.[gpio8, gpio9, gpio10, gpio11, gpio12, gpio13, gpio14, gpio15]

Field	Bits	Default	Access	Description
gpio8	8	0	R/O	Wake-Up Detect Status for P1.0
gpio9	9	0	R/O	Wake-Up Detect Status for P1.1
gpio10	10	0	R/O	Wake-Up Detect Status for P1.2

Field	Bits	Default	Access	Description
gpio11	11	0	R/O	Wake-Up Detect Status for P1.3
gpio12	12	0	R/O	Wake-Up Detect Status for P1.4
gpio13	13	0	R/O	Wake-Up Detect Status for P1.5
gpio14	14	0	R/O	Wake-Up Detect Status for P1.6
gpio15	15	0	R/O	Wake-Up Detect Status for P1.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

PWRMAN_WUD_SEEN0.[gpio16, gpio17, gpio18, gpio19, gpio20, gpio21, gpio22, gpio23]

Field	Bits	Default	Access	Description
gpio16	16	0	R/O	Wake-Up Detect Status for P2.0
gpio17	17	0	R/O	Wake-Up Detect Status for P2.1
gpio18	18	0	R/O	Wake-Up Detect Status for P2.2
gpio19	19	0	R/O	Wake-Up Detect Status for P2.3
gpio20	20	0	R/O	Wake-Up Detect Status for P2.4
gpio21	21	0	R/O	Wake-Up Detect Status for P2.5

Field	Bits	Default	Access	Description
gpio22	22	0	R/O	Wake-Up Detect Status for P2.6
gpio23	23	0	R/O	Wake-Up Detect Status for P2.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

PWRMAN_WUD_SEEN0.[gpio24, gpio25, gpio26, gpio27, gpio28, gpio29, gpio30, gpio31]

Field	Bits	Default	Access	Description
gpio24	24	0	R/O	Wake-Up Detect Status for P3.0
gpio25	25	0	R/O	Wake-Up Detect Status for P3.1
gpio26	26	0	R/O	Wake-Up Detect Status for P3.2
gpio27	27	0	R/O	Wake-Up Detect Status for P3.3
gpio28	28	0	R/O	Wake-Up Detect Status for P3.4
gpio29	29	0	R/O	Wake-Up Detect Status for P3.5
gpio30	30	0	R/O	Wake-Up Detect Status for P3.6
gpio31	31	0	R/O	Wake-Up Detect Status for P3.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

4.1.12.1.9 PWRMAN_WUD_SEEN1

PWRMAN_WUD_SEEN1.[gpio32, gpio33, gpio34, gpio35, gpio36, gpio37, gpio38, gpio39]

Field	Bits	Default	Access	Description
gpio32	0	0	R/O	Wake-Up Detect Status for P4.0
gpio33	1	0	R/O	Wake-Up Detect Status for P4.1
gpio34	2	0	R/O	Wake-Up Detect Status for P4.2
gpio35	3	0	R/O	Wake-Up Detect Status for P4.3
gpio36	4	0	R/O	Wake-Up Detect Status for P4.4
gpio37	5	0	R/O	Wake-Up Detect Status for P4.5
gpio38	6	0	R/O	Wake-Up Detect Status for P4.6
gpio39	7	0	R/O	Wake-Up Detect Status for P4.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0

- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

PWRMAN_WUD_SEEN1.[gpio40, gpio41, gpio42, gpio43, gpio44, gpio45, gpio46, gpio47]

Field	Bits	Default	Access	Description
gpio40	8	0	R/O	Wake-Up Detect Status for P5.0
gpio41	9	0	R/O	Wake-Up Detect Status for P5.1
gpio42	10	0	R/O	Wake-Up Detect Status for P5.2
gpio43	11	0	R/O	Wake-Up Detect Status for P5.3
gpio44	12	0	R/O	Wake-Up Detect Status for P5.4
gpio45	13	0	R/O	Wake-Up Detect Status for P5.5
gpio46	14	0	R/O	Wake-Up Detect Status for P5.6
gpio47	15	0	R/O	Wake-Up Detect Status for P5.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3

- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

PWRMAN_WUD_SEEN1.[gpio48, gpio49, gpio50, gpio51, gpio52, gpio53, gpio54, gpio55]

Field	Bits	Default	Access	Description
gpio48	16	0	R/O	Wake-Up Detect Status for P6.0
gpio49	17	0	R/O	Wake-Up Detect Status for P6.1
gpio50	18	0	R/O	Wake-Up Detect Status for P6.2
gpio51	19	0	R/O	Wake-Up Detect Status for P6.3
gpio52	20	0	R/O	Wake-Up Detect Status for P6.4
gpio53	21	0	R/O	Wake-Up Detect Status for P6.5
gpio54	22	0	R/O	Wake-Up Detect Status for P6.6
gpio55	23	0	R/O	Wake-Up Detect Status for P6.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6

- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

PWRMAN_WUD_SEEN1.[gpio56, gpio57, gpio58, gpio59, gpio60, gpio61, gpio62, gpio63]

Field	Bits	Default	Access	Description
gpio56	24	0	R/O	Wake-Up Detect Status for P7.0
gpio57	25	0	R/O	Wake-Up Detect Status for P7.1
gpio58	26	0	R/O	Wake-Up Detect Status for P7.2
gpio59	27	0	R/O	Wake-Up Detect Status for P7.3
gpio60	28	0	R/O	Wake-Up Detect Status for P7.4
gpio61	29	0	R/O	Wake-Up Detect Status for P7.5
gpio62	30	0	R/O	Wake-Up Detect Status for P7.6
gpio63	31	0	R/O	Wake-Up Detect Status for P7.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

4.1.12.1.10 PWRMAN_DIE_TYPE

Default	Access	Description
n/a	R/O	Die Type ID Register

Read-only.

Always returns 4D513637h to identify as 'MQ67'.

4.1.12.1.11 PWRMAN_BASE_PART_NUM

PWRMAN_BASE_PART_NUM.base_part_number

Field	Bits	Default	Access	Description
base_part_number	15:0	n/a	R/O	Base Part Number

Always returns 3260h (base part number).

PWRMAN_BASE_PART_NUM.package_select

Field	Bits	Default	Access	Description
package_select	28	n/a	R/O	Package Select

Returns value of package select option setting.

4.1.12.1.12 PWRMAN_MASK_ID0

PWRMAN_MASK_ID0.revision_id

Field	Bits	Default	Access	Description
revision_id	3:0	n/a	R/O	Revision ID

Device revision information.

PWRMAN_MASK_ID0.mask_id

Field	Bits	Default	Access	Description
mask_id	31:4	n/a	R/O	Mask ID[27:0]

Mask identification information - low 28 bits.

4.1.12.1.13 PWRMAN_MASK_ID1**PWRMAN_MASK_ID1.mask_id**

Field	Bits	Default	Access	Description
mask_id	30:0	n/a	R/O	Mask ID[58:28]

Mask identification information - high 31 bits.

PWRMAN_MASK_ID1.mask_id_enable

Field	Bits	Default	Access	Description
mask_id_enable	31	0	R/W	Enable Mask ID

Must set to 1 in order for the Mask ID fields to be readable.

4.1.12.1.14 PWRMAN_PERIPHERAL_RESET**PWRMAN_PERIPHERAL_RESET.uart0**

Field	Bits	Default	Access	Description
uart0	0	0	R/W	Reset UART0

- 0: Peripheral is released to run normally.

- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.uart1

Field	Bits	Default	Access	Description
uart1	1	0	R/W	Reset UART1

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.timer0

Field	Bits	Default	Access	Description
timer0	2	0	R/W	Reset Timer0

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.timer1

Field	Bits	Default	Access	Description
timer1	3	0	R/W	Reset Timer1

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.timer2

Field	Bits	Default	Access	Description
timer2	4	0	R/W	Reset Timer2

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.timer3

Field	Bits	Default	Access	Description
timer3	5	0	R/W	Reset Timer3

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.watchdog0

Field	Bits	Default	Access	Description
watchdog0	6	0	R/W	Reset Watchdog Timer 0

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.usb

Field	Bits	Default	Access	Description
usb	7	0	R/W	Reset USB

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.adc

Field	Bits	Default	Access	Description
adc	8	0	R/W	Reset ADC

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.dac0

Field	Bits	Default	Access	Description
dac0	9	0	R/W	Reset 12-Bit DAC 0

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.dac1

Field	Bits	Default	Access	Description
dac1	10	0	R/W	Reset 12-Bit DAC 1

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.dac2

Field	Bits	Default	Access	Description
dac2	11	0	R/W	Reset 8-Bit DAC 0

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.dac3

Field	Bits	Default	Access	Description
dac3	12	0	R/W	Reset 8-Bit DAC 1

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.dma

Field	Bits	Default	Access	Description
dma	13	0	R/W	Reset DMA/PMU

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.lcd

Field	Bits	Default	Access	Description
lcd	14	0	R/W	Reset LCD

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.gpio

Field	Bits	Default	Access	Description
gpio	15	0	R/W	Reset GPIO Module

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.pulse_train

Field	Bits	Default	Access	Description
pulse_train	16	0	R/W	Reset All Pulse Trains

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.spi0

Field	Bits	Default	Access	Description
spi0	17	0	R/W	Reset SPI 0

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.spi1

Field	Bits	Default	Access	Description
spi1	18	0	R/W	Reset SPI 1

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.spi2

Field	Bits	Default	Access	Description
spi2	19	0	R/W	Reset SPI 2

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.i2cm0

Field	Bits	Default	Access	Description
i2cm0	20	0	R/W	Reset I2C Master 0

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.i2cm1

Field	Bits	Default	Access	Description
i2cm1	21	0	R/W	Reset I2C Master 1

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.i2cs

Field	Bits	Default	Access	Description
i2cs	22	0	R/W	Reset I2C Slave

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.crc

Field	Bits	Default	Access	Description
crc	23	0	R/W	Reset CRC Engine

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.tpu

Field	Bits	Default	Access	Description
tpu	24	0	R/W	Reset TPU

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.ssb

Field	Bits	Default	Access	Description
ssb	25	0	R/W	Reset RTC SSB

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

4.1.13 Registers (PWRSEQ)**4.1.13.1 Module PWRSEQ Registers**

Address	Register	32b Word Len	Description
0x40090A30	PWRSEQ_REG0	1	Power Sequencer Control Register 0
0x40090A34	PWRSEQ_REG1	1	Power Sequencer Control Register 1
0x40090A38	PWRSEQ_REG2	1	Power Sequencer Control Register 2
0x40090A3C	PWRSEQ_REG3	1	Power Sequencer Control Register 3
0x40090A40	PWRSEQ_REG4	1	Power Sequencer Control Register 4
0x40090A44	PWRSEQ_REG5	1	Power Sequencer Control Register 5 (Trim 0)
0x40090A48	PWRSEQ_REG6	1	Power Sequencer Control Register 6 (Trim 1)

Address	Register	32b Word Len	Description
0x40090A50	PWRSEQ_FLAGS	1	Power Sequencer Flags
0x40090A54	PWRSEQ_MSK_FLAGS	1	Power Sequencer Flags Mask Register

4.1.13.1.1 PWRSEQ_REG0

PWRSEQ_REG0.pwr_lp1

Field	Bits	Default	Access	Description
pwr_lp1	0	0 (PwrSeq RSTN, see note)	R/W	Shutdown Power Mode Select

- 0: Shutdown to LP0 (default)
- 1: Shutdown to LP1

Note This field is reset by any of the following conditions/ events:

- PwrSeq RSTN (power sequencer asynchronous reset)
- System Reboot event
- Whenever pwr_prv_pwr_fail_r == 1

PWRSEQ_REG0.pwr_first_boot

Field	Bits	Default	Access	Description
pwr_first_boot	1	1	R/W	Wake on First Boot

Wakeup on first power automatically. (default 1)

PWRSEQ_REG0.pwr_sys_reboot

Field	Bits	Default	Access	Description
pwr_sys_reboot	2	0	W/O	Firmware System Reboot Request

Writing a 1 to this bit triggers a system reboot.

PWRSEQ_REG0.pwr_ldoen_run

Field	Bits	Default	Access	Description
pwr_ldoen_run	3	1 (PwrSeq RSTN)	R/W	Enable Main 1.8V LDO Operation in Run Mode

3V to 1.8V LDO enable during run (default 1)

PWRSEQ_REG0.pwr_ldoen_slp

Field	Bits	Default	Access	Description
pwr_ldoen_slp	4	0 (PwrSeq RSTN)	R/W	Enable Main 1.8V LDO Operation in Sleep Mode

3V to 1.8V LDO enable during sleep (default 0)

PWRSEQ_REG0.pwr_chzyen_run

Field	Bits	Default	Access	Description
pwr_chzyen_run	5	0 (PwrSeq RSTN)	R/W	Enable Backup 1.8V LDO Operation in Run Mode

Chzy regulator enable during run (default 0)

PWRSEQ_REG0.pwr_chzyen_slp

Field	Bits	Default	Access	Description
pwr_chzyen_slp	6	1	R/W	Enable Backup 1.8V LDO Operation in Sleep Mode

Chzy regulator enable during sleep (default 1)

PWRSEQ_REG0.pwr_roen_run

Field	Bits	Default	Access	Description
pwr_roen_run	7	1	R/W	Enable System Relaxation Oscillator in Run Mode

Relaxation osc enable during run (default 1)

PWRSEQ_REG0.pwr_roen_slp

Field	Bits	Default	Access	Description
pwr_roen_slp	8	0 (PwrSeq RSTN)	R/W	Enable System Relaxation Oscillator in Sleep Mode

Relaxation osc enable during sleep (default 0)

PWRSEQ_REG0.pwr_nren_run

Field	Bits	Default	Access	Description
pwr_nren_run	9	0 (RTC POR)	R/W	Enable Nano Oscillator in Run Mode

Nano oscillator enable during run (default 0)

PWRSEQ_REG0.pwr_nren_slp

Field	Bits	Default	Access	Description
pwr_nren_slp	10	0 (RTC POR)	R/W	Enable Nano Oscillator in Sleep Mode

Nano oscillator enable during sleep (default 0)

PWRSEQ_REG0.pwr_rtcent_run

Field	Bits	Default	Access	Description
pwr_rtcent_run	11	0 (RTC POR)	R/W	Enable Real Time Clock operation in Run Mode

Real Time Clock enable during run (default 0)

PWRSEQ_REG0.pwr_rtcent_slp

Field	Bits	Default	Access	Description
pwr_rtcent_slp	12	0 (RTC POR)	R/W	Enable Real Time Clock operation in Sleep Mode

Real Time Clock enable during sleep (default 0)

PWRSEQ_REG0.pwr_svm3en_run

Field	Bits	Default	Access	Description
pwr_svm3en_run	13	1 (PwrSeq RSTN)	R/W	Enable VDD3 SVM operation in Run Mode

VDD3 SVM enable during run mode (default 1)

PWRSEQ_REG0.pwr_svm3en_slp

Field	Bits	Default	Access	Description
pwr_svm3en_slp	14	1 (PwrSeq RSTN)	R/W	Enable VDD3 SVM operation in Sleep Mode

VDD3 SVM enable during sleep mode (default 1)

PWRSEQ_REG0.pwr_svm1en_run

Field	Bits	Default	Access	Description
pwr_svm1en_run	15	1 (PwrSeq RSTN)	R/W	Enable VREG18 SVM operation in Run Mode

VREG18 SVM enable during run mode (default 1)

PWRSEQ_REG0.pwr_svm1en_slp

Field	Bits	Default	Access	Description
pwr_svm1en_slp	16	1 (PwrSeq RSTN)	R/W	Enable VREG18 SVM operation in Sleep Mode

VREG18 SVM enable during sleep mode (default 1)

PWRSEQ_REG0.pwr_svmrtcen_run

Field	Bits	Default	Access	Description
pwr_svmrtcen_run	17	0 (PwrSeq RSTN)	R/W	Enable VRTC SVM operation in Run Mode

VRTC SVM enable during run mode (default 0)

PWRSEQ_REG0.pwr_svmrtcen_slp

Field	Bits	Default	Access	Description
pwr_svmrtcen_slp	18	0 (PwrSeq RSTN)	R/W	Enable VRTC SVM operation in Sleep Mode

VRTC SVM enable during sleep mode (default 0)

PWRSEQ_REG0.pwr_svmvdda3en

Field	Bits	Default	Access	Description
pwr_svmvdda3en	19	0 (PwrSeq RSTN)	R/W	Enable VDDA3 SVM operation (in Run Mode only)

VDDA3 SVM enable (can only be enabled during RUN mode). default 0.

4.1.13.1.2 PWRSEQ_REG1

PWRSEQ_REG1.pwr_trikl_chrg

Field	Bits	Default	Access	Description
pwr_trikl_chrg	7:0	00h (PwrSeq RSTN)	R/W	Trickle Charge Control for VRTC External Capacitor

Trickle charger (for VRTC external supercap). To enable the trickle charger, bits [7:4] must equal 1010b (Ah); all other settings for [7:4] result in the function staying disabled.

- A5h - no diode + 250ohm
- A6h - no diode + 2kohm

- A7h - no diode + 4kohm
- A9h - diode + 250ohm
- AAh - diode + 2kohm
- ABh - diode + 4kohm

By default, the trickle charger is disabled.

PWRSEQ_REG1.pwr_pd_vdda3

Field	Bits	Default	Access	Description
pwr_pd_vdda3	8	0 (PwrSeq RSTN)	R/W	Power Down VDDA3 Supply Rail

- 0: VDDA3 supply rail is powered on (default).
- 1: VDDA3 supply rail is powered down.

PWRSEQ_REG1.pwr_temp_sensor_pd

Field	Bits	Default	Access	Description
pwr_temp_sensor_pd	9	1 (PwrSeq RSTN)	R/W	Power Down Internal Temperature Sensor

- 0: Internal temp sensor is powered on and can be used.
- 1: Internal temp sensor is powered off (default).

PWRSEQ_REG1.pwr_pd_vddio

Field	Bits	Default	Access	Description
pwr_pd_vddio	10	0 (PwrSeq RSTN)	R/W	Power Down VDDIO Supply Rail

- 0: VDDIO supply rail is powered on (default).
- 1: VDDIO supply rail is powered down.

PWRSEQ_REG1.pwr_man_vddio_sw

Field	Bits	Default	Access	Description
pwr_man_vddio_sw	11	0 (PwrSeq RSTN)	R/W	Manual Override Enable for VDDIO Switch 1/2

- 0: No effect (default).
- 1: Manual overrides are enabled for VDDIO_SW1 and VDDIO_SW2.

PWRSEQ_REG1.pwr_man_vddio_sw2

Field	Bits	Default	Access	Description
pwr_man_vddio_sw2	12	0 (PwrSeq RSTN)	R/W	Manual Override for VDDIO_SW2

This setting will only take effect when the Manual Override Enable for VDDIO Switch 1/2 has been set to 1.

- 0: VDDIO_SW2 is set to VREG18 mode (default).
- 1: VDDIO_SW2 is set to VDDIO mode.

PWRSEQ_REG1.pwr_man_vddio_sw1

Field	Bits	Default	Access	Description
pwr_man_vddio_sw1	13	0 (PwrSeq RSTN)	R/W	Manual Override for VDDIO_SW1

This setting will only take effect when the Manual Override Enable for VDDIO Switch 1/2 has been set to 1.

- 0: VDDIO_SW1 is set to VREG18 mode (default).
- 1: VDDIO_SW1 is set to VDDIO mode.

PWRSEQ_REG1.pwr_gpio_freeze

Field	Bits	Default	Access	Description
pwr_gpio_freeze	14	0 (PwrSeq RSTN)	R/W	Freeze GPIO WUD and Keeper Latches

- 0: GPIO WUD and keeper latches operate normally (default).
- 1: GPIO WUD and keeper latches are frozen.

4.1.13.1.3 PWRSEQ_REG2

PWRSEQ_REG2.pwr_rst3

Field	Bits	Default	Access	Description
pwr_rst3	4:0	4 (PwrSeq RSTN)	R/W	pwr_rst3_o - VDD3 Reset decode

approx 30-45mV step over applicable range.

PWRSEQ_REG2.pwr_w3

Field	Bits	Default	Access	Description
pwr_w3	9:5	16 (PwrSeq RSTN or VDD3 rail POR)	R/W	pwr_w3_o[4:0]

VDD3 Warning decode. approx 30-45mV step size over applicable range

PWRSEQ_REG2.pwr_w1

Field	Bits	Default	Access	Description
pwr_w1	14:10	15 (PwrSeq RSTN or VDD3 rail POR)	R/W	pwr_w1_o[4:0]

VREG18 Warning decode. approx 30mV step size over applicable range.

PWRSEQ_REG2.pwr_w1_low

Field	Bits	Default	Access	Description
pwr_w1_low	19:15	4 (PwrSeq RSTN)	R/W	pwr_w1_low_o[4:0]

VREG18 Warning LOW decode. approx 25mV step size over applicable range.

PWRSEQ_REG2.pwr_wrtc

Field	Bits	Default	Access	Description
pwr_wrtc	24:20	16 (PwrSeq RSTN)	R/W	pwr_wrtc_o[4:0]

VRTC decode. 32 steps. approx 30-45mV step size over applicable range.

PWRSEQ_REG2.pwr_wvdda3

Field	Bits	Default	Access	Description
pwr_wvdda3	30:25	3Fh (PwrSeq RSTN)	R/W	pwr_wvdda3_o[5:0]

VDDA3 decode. 3.6V...1.59V total, 3.48V...1.59V in 64 steps. 30mV step size.

4.1.13.1.4 PWRSEQ_REG3**PWRSEQ_REG3.pwr_rose1**

Field	Bits	Default	Access	Description
pwr_rose1	2:0	101b (PwrSeq RSTN)	R/W	pwr_rose1_o[2:0]

Relaxation Oscillator Stable timeout setting (in RO clocks)

- 000b - Bypass
- 001b - 64 clocks
- 010b - 128 clocks
- 011b - 256 clocks
- 100b - 512 clocks
- 101b - 1024 clocks (default setting)
- 110b - 2048 clocks
- 111b - 262144 clocks

PWRSEQ_REG3.pwr_rose1_quick

Field	Bits	Default	Access	Description
pwr_rose1_quick	4:3	00b (PwrSeq RSTN, see notes*)	R/W	pwr_rose1_quick_o[1:0]

Quick Relaxation Oscillator Stable timeout (in RO clocks)

- 00b: Bypass (default setting)
- 01b: 64 clocks

- 10b: 128 clocks
- 11b: 256 clocks

Notes*

- This field is reset by any of the following conditions/events:
 - PwrSeq RSTN (power sequencer asynchronous reset)
 - System Reboot event
 - Whenever pwr_prv_pwr_fail_r == 1
 - Whenever pwr_prv_boot_fail_r == 1

PWRSEQ_REG3.pwr_svmssel

Field	Bits	Default	Access	Description
pwr_svmssel	7:5	000b (PwrSeq RSTN)	R/W	pwr_svmssel_o[2:0]

SVM timeout count. (SVM clks)

- 000 - Bypass (Default)
- 001 - 30 sec
- 010 - 1 min
- 011 - 2 min
- 100 - 4 min
- 101 - 8 min
- 110 - 16 min
- 111 - 32 min.

PWRSEQ_REG3.pwr_pwrfltrsvmselo

Field	Bits	Default	Access	Description
pwr_pwrfltrsvmselo	9:8	11b (PwrSeq RSTN)	R/W	pwr_pwrfltrsvmselo_o[1:0]

Window of time power must be valid while checking SVMs (in SVM clocks)

- 00b: Bypass
- 01b: 2 clocks
- 10b: 4 clocks
- 11b: 8 clocks (default)

PWRSEQ_REG3.pwr_pwrfltrsel

Field	Bits	Default	Access	Description
pwr_pwrfltrsel	12:10	011b (PwrSeq RSTN)	R/W	pwr_pwrfltrsel_o[2:0]

Window of time power must be valid before entering Run mode (in RO clocks)

- 000b: Bypass
- 001b: 2 clocks
- 010b: 4 clocks
- 011b: 8 clocks (default)
- 100b: 16 clocks
- 101b: 32 clocks
- 110b: 64 clocks
- 111b: 128 clocks

PWRSEQ_REG3.pwr_svm_clk_mux

Field	Bits	Default	Access	Description
pwr_svm_clk_mux	14:13	00b (PwrSeq RSTN)	R/W	pwr_svm_clk_mux_o[1:0]

SVM clock mux

- 00b: Nano Oscillator (default)
- 01b: RTC clock
- 10b: Relaxation Oscillator
- 11b: External clock

PWRSEQ_REG3.pwr_ro_clk_mux

Field	Bits	Default	Access	Description
pwr_ro_clk_mux	15	0 (PwrSeq RSTN)	R/W	pwr_ro_clk_mux_o

Relaxation Clock mux

- 0: Relaxation Oscillator (default)
- 1: External Clock

PWRSEQ_REG3.pwr_quick_cnt

Field	Bits	Default	Access	Description
pwr_quick_cnt	16	0 (PwrSeq RSTN)	R/W	pwr_quick_cnt_o

500ms timeout during PowerFail/BootFail events

This bit should be set to 1 by firmware after initial boot has completed; this should be done at the same time the first_boot flag is cleared.

PWRSEQ_REG3.pwr_bo_tc

Field	Bits	Default	Access	Description
pwr_bo_tc	18:17	11b (PwrSeq RSTN)	R/W	Brownout Detection Time Constant

4.1.13.1.5 PWRSEQ_REG4**PWRSEQ_REG4.pwr_tm_ps_2_gpio**

Field	Bits	Default	Access	Description
pwr_tm_ps_2_gpio	0	0 (PwrSeq RSTN)	R/W	pwr_tm_ps_2_gpio_w

Enable power sequencer signals to GPIO pads (Test Mode)

PWRSEQ_REG4.pwr_tm_fast_timers

Field	Bits	Default	Access	Description
pwr_tm_fast_timers	1	0 (PwrSeq RSTN)	R/W	pwr_tm_fast_timers_o

Enable fast timers for simulation (Test Mode)

PWRSEQ_REG4.pwr_usb_prot_trim

Field	Bits	Default	Access	Description
pwr_usb_prot_trim	2	0 (PwrSeq RSTN)	R/W	pwr_usb_prot_trim_o

USB protection trim

PWRSEQ_REG4.pwr_usb_dis_comp

Field	Bits	Default	Access	Description
pwr_usb_dis_comp	3	0 (PwrSeq RSTN)	R/W	pwr_usb_dis_comp_o

Disable USB protection comparator

PWRSEQ_REG4.pwr_usb_to_vdd_fast

Field	Bits	Default	Access	Description
pwr_usb_to_vdd_fast	4	0 (PwrSeq RSTN)	R/W	pwr_usbToVddFast_w

Switch to VDD rail as soon as VDDBOK is deasserted

PWRSEQ_REG4.pwr_usb_ldo_off

Field	Bits	Default	Access	Description
pwr_usb_ldo_off	5	0 (PwrSeq RSTN)	R/W	pwr_usbLdoOff_w

Turn off 5v to 3.3v LDO while USB power is available

PWRSEQ_REG4.pwr_usb_frc_vdd

Field	Bits	Default	Access	Description
pwr_usb_frc_vdd	6	0 (PwrSeq RSTN)	R/W	pwr_usbFrcVdd_w

Force use of VDD rail instead of VDDB rail while USB power is available

4.1.13.1.6 PWRSEQ_REG5

PWRSEQ_REG5.pwr_trim_svm_bg

Field	Bits	Default	Access	Description
pwr_trim_svm_bg	5:0	000000b (PwrSeq RSTN)	R/W	pwr_trim_svm_bg_w[5:0]

Power manager bandgap trim

PWRSEQ_REG5.pwr_trim_reg1p8

Field	Bits	Default	Access	Description
pwr_trim_reg1p8	9:6	0000b (PwrSeq RSTN)	R/W	pwr_trim_reg1p8_w[3:0]

3 volt to 1.8 volt LDO trim

PWRSEQ_REG5.pwr_trim_reg3p3

Field	Bits	Default	Access	Description
pwr_trim_reg3p3	14:10	00000b (PwrSeq RSTN)	R/W	pwr_trim_reg3p3_o[4:0]

5 volt to 3.3v LDO trim

PWRSEQ_REG5.pwr_trim_osc_vref

Field	Bits	Default	Access	Description
pwr_trim_osc_vref	21:15	0000000b (PwrSeq RSTN)	R/W	pwr_trim_osc_vref_o[6:0]

Relaxation oscillator trim

4.1.13.1.7 PWRSEQ_REG6

PWRSEQ_REG6.pwr_trim_usb_bias

Field	Bits	Default	Access	Description
pwr_trim_usb_bias	2:0	000b (PwrSeq RSTN)	R/W	pwr_trim_usb_bias_o[2:0]

USB bias current trim

PWRSEQ_REG6.pwr_trim_usb_pm_res

Field	Bits	Default	Access	Description
pwr_trim_usb_pm_res	6:3	0000b (PwrSeq RSTN)	R/W	pwr_trim_usb_pm_res_o[3:0]

USB Data Plus slew rate trim

PWRSEQ_REG6.pwr_trim_usb_dm_res

Field	Bits	Default	Access	Description
pwr_trim_usb_dm_res	10:7	0000b (PwrSeq RSTN)	R/W	pwr_trim_usb_dm_res_o[3:0]

USB Data Minus slew rate trim

4.1.13.1.8 PWRSEQ_FLAGS

PWRSEQ_FLAGS.pwr_first_boot

Field	Bits	Default	Access	Description
pwr_first_boot	0	s	R/O	pwr_first_boot_o

Initial boot indication

PWRSEQ_FLAGS.pwr_sys_reboot

Field	Bits	Default	Access	Description
pwr_sys_reboot	1	s	R/O	pwr_sys_reboot_o

Firmware Reset event

PWRSEQ_FLAGS.pwr_prv_pwr_fail

Field	Bits	Default	Access	Description
pwr_prv_pwr_fail	2	s	W1C	pwr_prv_pwr_fail_r

Write 1 to clear power fail detect latch

Power Fail event detected

PWRSEQ_FLAGS.pwr_prv_boot_fail

Field	Bits	Default	Access	Description
pwr_prv_boot_fail	3	s	W1C	pwr_prv_boot_fail_r

Write 1 to clear boot fail detect latch

Boot Fail event detected

PWRSEQ_FLAGS.pwr_comp_wakeup

Field	Bits	Default	Access	Description
pwr_comp_wakeup	4	s	W1C	pwr_comp_wakeup_latch_r

Write 1 to clear analog comparator wakeup latch

Comparator wakeup event detected

PWRSEQ_FLAGS.pwr_io_wakeup

Field	Bits	Default	Access	Description
pwr_io_wakeup	5	s	W1C	pwr_iowakeup_latch_r

Write 1 to clear GPIO wakeup event latch

GPIO wakeup event detected

PWRSEQ_FLAGS.pwr_vdd3_rst

Field	Bits	Default	Access	Description
pwr_vdd3_rst	6	s	W1C	pwr_vdd3_rst_bad_r

Write 1 to clear VDD3 reset compare latch

VDD3 reset comparator tripped

PWRSEQ_FLAGS.pwr_vdd3_warn

Field	Bits	Default	Access	Description
pwr_vdd3_warn	7	s	W1C	pwr_vdd3_warn_bad_r

Write 1 to clear VDD3 warning compare latch

VDD3 warning comparator tripped

PWRSEQ_FLAGS.pwr_vdd1_rst

Field	Bits	Default	Access	Description
pwr_vdd1_rst	8	s	W1C	pwr_vdd1_rst_bad_r

Write 1 to clear VREG18 reset compare latch

VREG18 reset comparator tripped

PWRSEQ_FLAGS.pwr_vdd1_low_rst

Field	Bits	Default	Access	Description
pwr_vdd1_low_rst	9	s	W1C	pwr_vdd1_low_rst_bad_r

Write 1 to clear VREG18 reset LOW compare latch

VREG18 reset LOW comparator tripped

PWRSEQ_FLAGS.pwr_vdd1_warn

Field	Bits	Default	Access	Description
pwr_vdd1_warn	10	s	W1C	pwr_vdd1_warn_bad_r

Clear VREG18 warning compare latch (BIT 10)

VREG18 warning comparator tripped

PWRSEQ_FLAGS.pwr_vrtc_warn

Field	Bits	Default	Access	Description
pwr_vrtc_warn	11	s	W1C	pwr_vrtc_warn_bad_r

Write 1 to clear VRTC power compare latch

VRTC comparator tripped

PWRSEQ_FLAGS.pwr_por3z_fail

Field	Bits	Default	Access	Description
pwr_por3z_fail	12	s	W1C	pwr_por3z_fail_latch_r

Write 1 to clear por3z_fail

POR3 and POR3_lite have been tripped

PWRSEQ_FLAGS.rtc_cmpr0

Field	Bits	Default	Access	Description
rtc_cmpr0	13	s	W1C	rtc_cmpr0_flag

Write 1 to clear; causes 4kHz transaction in RTC

PWRSEQ_FLAGS.rtc_cmpr1

Field	Bits	Default	Access	Description
rtc_cmpr1	14	s	W1C	rtc_cmpr1_flag

Write 1 to clear; causes 4kHz transaction in RTC

PWRSEQ_FLAGS.rtc_prescale_cmp

Field	Bits	Default	Access	Description
rtc_prescale_cmp	15	s	W1C	rtc_prescale_cmp_flag

Write 1 to clear; causes 4kHz transaction in RTC

PWRSEQ_FLAGS.rtc_rollover

Field	Bits	Default	Access	Description
rtc_rollover	16	s	W1C	rtc_rollover_flag

Write 1 to clear; causes 4kHz transaction in RTC

PWRSEQ_FLAGS.pwr_brownout_det

Field	Bits	Default	Access	Description
pwr_brownout_det	17	s	W1C	pwr_brownout_detected

Write 1 to clear

PWRSEQ_FLAGS.pwr_usb_plug_wakeup

Field	Bits	Default	Access	Description
pwr_usb_plug_wakeup	18	s	W1C	pwr_usb_plug_wakeup_detected

Write 1 to clear

PWRSEQ_FLAGS.pwr_usb_remove_wakeup

Field	Bits	Default	Access	Description
pwr_usb_remove_wakeup	19	s	W1C	pwr_usb_remove_wakeup_– detected

Write 1 to clear

PWRSEQ_FLAGS.pwr_vdd22_rst

Field	Bits	Default	Access	Description
pwr_vdd22_rst	20	s	W1C	pwr_vdd22_rst

Write 1 to clear

PWRSEQ_FLAGS.pwr_vdd195_rst

Field	Bits	Default	Access	Description
pwr_vdd195_rst	21	s	W1C	pwr_vdd195_rst

Write 1 to clear

4.1.13.1.9 PWRSEQ_MSK_FLAGS

PWRSEQ_MSK_FLAGS.pwr_sys_reboot

Field	Bits	Default	Access	Description
pwr_sys_reboot	1	0	R/W	Mask for system reboot detect

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_prv_pwr_fail

Field	Bits	Default	Access	Description
pwr_prv_pwr_fail	2	0	R/W	Mask for previous power fail detect

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_priv_boot_fail

Field	Bits	Default	Access	Description
pwr_priv_boot_fail	3	0	R/W	Mask for previous boot fail detect

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_comp_wakeup

Field	Bits	Default	Access	Description
pwr_comp_wakeup	4	0	R/W	Mask for analog comparator wakeup detect

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_io_wakeup

Field	Bits	Default	Access	Description
pwr_io_wakeup	5	0	R/W	Mask for GPIO wakeup event detect

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_vdd3_rst

Field	Bits	Default	Access	Description
pwr_vdd3_rst	6	0	R/W	Mask for VDD3 reset comparator event detect

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_vdd3_warn

Field	Bits	Default	Access	Description
pwr_vdd3_warn	7	0	R/W	Mask for VDD3 voltage warning event detect

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_vdd1_rst

Field	Bits	Default	Access	Description
pwr_vdd1_rst	8	0	R/W	Mask for VREG18 reset compare event detect

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_vdd1_low_rst

Field	Bits	Default	Access	Description
pwr_vdd1_low_rst	9	0	R/W	Mask for VREG18 reset LOW compare event

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_vdd1_warn

Field	Bits	Default	Access	Description
pwr_vdd1_warn	10	0	R/W	Mask for VREG18 warning level event detect

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_vrtc_warn

Field	Bits	Default	Access	Description
pwr_vrtc_warn	11	0	R/W	Mask for VRTC power fail detect

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_por3z_fail

Field	Bits	Default	Access	Description
pwr_por3z_fail	12	0	R/W	Mask for POR3Z supply fail event detect

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.rtc_cmpr0

Field	Bits	Default	Access	Description
rtc_cmpr0	13	0	R/W	Mask for RTC compare 0 event

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.rtc_cmpr1

Field	Bits	Default	Access	Description
rtc_cmpr1	14	0	R/W	Mask for RTC compare 1 event

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.rtc_prescale_cmp

Field	Bits	Default	Access	Description
rtc_prescale_cmp	15	0	R/W	Mask for RTC prescale compare event

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.rtc_rollover

Field	Bits	Default	Access	Description
rtc_rollover	16	0	R/W	Mask for RTC rollover event

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_brownout_det

Field	Bits	Default	Access	Description
pwr_brownout_det	17	1	R/W	Mask for power brownout detect

Note Masked by default

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_usb_plug_wakeup

Field	Bits	Default	Access	Description
pwr_usb_plug_wakeup	18	0	R/W	Mask for USB power connected wakeup event

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_usb_remove_wakeup

Field	Bits	Default	Access	Description
pwr_usb_remove_wakeup	19	0	R/W	Mask for USB power disconnected wakeup event

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_vdd22_rst

Field	Bits	Default	Access	Description
pwr_vdd22_rst	20	0	R/W	Mask for VDD22 reset event

- 0: Event can be detected
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_vdd195_rst

Field	Bits	Default	Access	Description
pwr_vdd195_rst	21	0	R/W	Mask for VDD195 reset event

- 0: Event can be detected
- 1: Event is masked

4.2 Interrupt Vector Table

Interrupt Number	Vector	Interrupt Source
0	0x40	UART0
1	0x44	UART1
2	0x48	I ² C Master 0
3	0x4c	I ² C Slave
4	0x50	USB Device Controller
5	0x54	Peripheral Management Unit
6	0x58	Analog Front End
7	0x5c	Modular Arithmetic Accelerator
8	0x60	AES
9	0x64	SPI0
10	0x68	SPI1
11	0x6c	SPI2
12	0x70	Timer 0 (Lower timer in 16-bit mode or 32-bit timer)
13	0x74	Timer 1 (Lower timer in 16-bit mode or 32-bit timer)
14	0x78	Timer 2 (Lower timer in 16-bit mode or 32-bit timer)
15	0x7c	Timer 3 (Lower timer in 16-bit mode or 32-bit timer)
18	0x88	DAC0 (12-bit DAC)
19	0x8c	DAC1 (12-bit DAC)
20	0x90	DAC2 (8-bit DAC)
21	0x94	DAC3 (8-bit DAC)
22	0x98	ADC
23	0x9c	Flash Controller Exception
24	0xa0	Power Manager
25	0xa4	Clocks
26	0xa8	RTC Interrupt 0
27	0xac	RTC Interrupt 1
28	0xb0	RTC Interrupt 2

Interrupt Number	Vector	Interrupt Source
29	0xb4	RTC Interrupt 3
30	0xb8	Watchdog 0
31	0xbc	Watchdog 0 Pre-window
32	0xc0	Watchdog 1
33	0xc4	Watchdog 1 Pre-window
34	0xc8	P0 Port
35	0xcc	P1 Port
36	0xd0	P2 Port
37	0xd4	P3 Port
38	0xd8	P4 Port
39	0xdc	P5 Port
40	0xe0	P6 Port
41	0xe4	P7 Port
42	0xe8	Timer 0 (Upper timer in 16-bit mode)
43	0xec	Timer 1 (Upper timer in 16-bit mode)
44	0xf0	Timer 2 (Upper timer in 16-bit mode)
45	0xf4	Timer 3 (Upper timer in 16-bit mode)
46	0xf8	I ² C Master 1

4.3 Resets and Reset Sources

This section describes resets and the reset sources for the **MAX32600**.

4.3.1 System Reset

The system reset triggers a reset of the operational states and most digital and analog blocks on the **MAX32600**, including the ARM Cortex-M3 CPU.

Sources that can trigger a system reset include:

- Active low signal on the SRSTN external reset pin.
- Firmware requested reset.

- Reset generated by ARM core due to lockup condition.
- Reset generated by watchdog timer.

4.3.2 Power-On Reset

The Power-On Reset or POR on the **MAX32600** resets all functions and blocks that are reset by a System Reset, but also resets additional blocks and registers including certain system configuration registers and the ARM JTAG debugging engine on the Cortex-M3 core.

A POR reset is triggered when a power failure occurs on either the digital main supplies (V_{DD} and V_{BUS}) or the V_{REG18} supply that is used to maintain the state of the digital core.

The POR reset does not affect the state of certain registers and blocks, including the power sequencer, the dynamic tamper sensor, and the RTC.

4.3.3 RTC POR

When a standard POR occurs, the state of the RTC and certain other registers and blocks are still retained as long as the backup supply V_{RTC} is still available. A failure of the V_{RTC} supply in combination with a standard POR causes a full reset of the RTC core and all registers and internal state on the device (apart from information retained in non-volatile flash storage). This is referred to as an RTC POR.

4.4 Registers (IOMAN)

4.4.1 Module IOMAN Registers

Address	Register	32b Word Len	Description
0x40090C00	IOMAN_WUD_REQ0	1	Wakeup Detect Mode Request Register 0 (P0/P1/P2/P3)
0x40090C04	IOMAN_WUD_REQ1	1	Wakeup Detect Mode Request Register 1 (P4/P5/P6/P7)
0x40090C08	IOMAN_WUD_ACK0	1	Wakeup Detect Mode Acknowledge Register 0 (P0/P1/P2/P3)
0x40090C0C	IOMAN_WUD_ACK1	1	Wakeup Detect Mode Acknowledge Register 1 (P4/P5/P6/P7)
0x40090C10	IOMAN_ALI_REQ0	1	Analog Input Request Register 0 (P0/P1/P2/P3)
0x40090C14	IOMAN_ALI_REQ1	1	Analog Input Request Register 1 (P4/P5/P6/P7)
0x40090C18	IOMAN_ALI_ACK0	1	Analog Input Acknowledge Register 0 (P0/P1/P2/P3)
0x40090C1C	IOMAN_ALI_ACK1	1	Analog Input Acknowledge Register 1 (P4/P5/P6/P7)

Address	Register	32b Word Len	Description
0x40090C20	IOMAN_SPI0_REQ	1	SPI0 I/O Mode Request
0x40090C24	IOMAN_SPI0_ACK	1	SPI0 I/O Mode Acknowledge
0x40090C28	IOMAN_SPI1_REQ	1	SPI1 I/O Mode Request
0x40090C2C	IOMAN_SPI1_ACK	1	SPI1 I/O Mode Acknowledge
0x40090C30	IOMAN_SPI2_REQ	1	SPI2 I/O Mode Request
0x40090C34	IOMAN_SPI2_ACK	1	SPI2 I/O Mode Acknowledge
0x40090C38	IOMAN_UART0_REQ	1	UART0 I/O Mode Request
0x40090C3C	IOMAN_UART0_ACK	1	UART0 I/O Mode Acknowledge
0x40090C40	IOMAN_UART1_REQ	1	UART1 I/O Mode Request
0x40090C44	IOMAN_UART1_ACK	1	UART1 I/O Mode Acknowledge
0x40090C48	IOMAN_I2CM0_REQ	1	I2C Master 0 I/O Request
0x40090C4C	IOMAN_I2CM0_ACK	1	I2C Master 0 I/O Acknowledge
0x40090C50	IOMAN_I2CS0_REQ	1	I2C Slave 0 I/O Request
0x40090C54	IOMAN_I2CS0_ACK	1	I2C Slave 0 I/O Acknowledge
0x40090C58	IOMAN_LCD_COM_REQ	1	LCD COM Driver I/O Request
0x40090C5C	IOMAN_LCD_COM_ACK	1	LCD COM Driver I/O Acknowledge
0x40090C60	IOMAN_LCD_SEG_REQ0	1	LCD SEG Driver I/O Request Register 0
0x40090C64	IOMAN_LCD_SEG_REQ1	1	LCD SEG Driver I/O Request Register 1
0x40090C68	IOMAN_LCD_SEG_ACK0	1	LCD SEG Driver I/O Acknowledge Register 0
0x40090C6C	IOMAN_LCD_SEG_ACK1	1	LCD SEG Driver I/O Acknowledge Register 1
0x40090C70	IOMAN_CRNT_REQ	1	Current Drive I/O Request Register
0x40090C74	IOMAN_CRNT_ACK	1	Current Drive I/O Acknowledge Register
0x40090C78	IOMAN_CRNT_MODE	1	Current Drive I/O Mode Control
0x40090C7C	IOMAN_ALI_CONNECT0	1	Analog I/O Connection Control Register 0
0x40090C80	IOMAN_ALI_CONNECT1	1	Analog I/O Connection Control Register 1
0x40090C84	IOMAN_I2CM1_REQ	1	I2C Master 1 I/O Request
0x40090C88	IOMAN_I2CM1_ACK	1	I2C Master 1 I/O Acknowledge
0x40090C8C	IOMAN_PADX_CONTROL	1	Multichip Interface Pad Control Register

4.4.1.1 IOMAN_WUD_REQ0**IOMAN_WUD_REQ0.port0**

Field	Bits	Default	Access	Description
port0	7:0	00000000b	R/W	Wakeup Detect Request Mode: P0[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

IOMAN_WUD_REQ0.port1

Field	Bits	Default	Access	Description
port1	15:8	00000000b	R/W	Wakeup Detect Request Mode: P1[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

IOMAN_WUD_REQ0.port2

Field	Bits	Default	Access	Description
port2	23:16	00000000b	R/W	Wakeup Detect Request Mode: P2[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

IOMAN_WUD_REQ0.port3

Field	Bits	Default	Access	Description
port3	31:24	00000000b	R/W	Wakeup Detect Request Mode: P3[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

4.4.1.2 IOMAN_WUD_REQ1

IOMAN_WUD_REQ1.port4

Field	Bits	Default	Access	Description
port4	7:0	00000000b	R/W	Wakeup Detect Request Mode: P4[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

IOMAN_WUD_REQ1.port5

Field	Bits	Default	Access	Description
port5	15:8	00000000b	R/W	Wakeup Detect Request Mode: P5[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

IOMAN_WUD_REQ1.port6

Field	Bits	Default	Access	Description
port6	23:16	00000000b	R/W	Wakeup Detect Request Mode: P6[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

IOMAN_WUD_REQ1.port7

Field	Bits	Default	Access	Description
port7	31:24	00000000b	R/W	Wakeup Detect Request Mode: P7[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

4.4.1.3 IOMAN_WUD_ACK0

IOMAN_WUD_ACK0.port0

Field	Bits	Default	Access	Description
port0	7:0	00000000b	R/O	WUD Mode Acknowledge: P0[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

IOMAN_WUD_ACK0.port1

Field	Bits	Default	Access	Description
port1	15:8	00000000b	R/O	WUD Mode Acknowledge: P1[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

IOMAN_WUD_ACK0.port2

Field	Bits	Default	Access	Description
port2	23:16	00000000b	R/O	WUD Mode Acknowledge: P2[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

IOMAN_WUD_ACK0.port3

Field	Bits	Default	Access	Description
port3	31:24	00000000b	R/O	WUD Mode Acknowledge: P3[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

4.4.1.4 IOMAN_WUD_ACK1

IOMAN_WUD_ACK1.port4

Field	Bits	Default	Access	Description
port4	7:0	00000000b	R/O	WUD Mode Acknowledge: P4[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

IOMAN_WUD_ACK1.port5

Field	Bits	Default	Access	Description
port5	15:8	00000000b	R/O	WUD Mode Acknowledge: P5[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

IOMAN_WUD_ACK1.port6

Field	Bits	Default	Access	Description
port6	23:16	00000000b	R/O	WUD Mode Acknowledge: P6[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

IOMAN_WUD_ACK1.port7

Field	Bits	Default	Access	Description
port7	31:24	00000000b	R/O	WUD Mode Acknowledge: P7[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

4.4.1.5 IOMAN_ALI_REQ0

IOMAN_ALI_REQ0.port0

Field	Bits	Default	Access	Description
port0	7:0	00000000b	R/W	Analog Input Mode Request: P0[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

IOMAN_ALI_REQ0.port1

Field	Bits	Default	Access	Description
port1	15:8	00000000b	R/W	Analog Input Mode Request: P1[7:0]

- 0:No effect

- 1:Requests analog input mode on the associated pin.

IOMAN_ALI_REQ0.port2

Field	Bits	Default	Access	Description
port2	23:16	00000000b	R/W	Analog Input Mode Request: P2[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

IOMAN_ALI_REQ0.port3

Field	Bits	Default	Access	Description
port3	31:24	00000000b	R/W	Analog Input Mode Request: P3[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

4.4.1.6 IOMAN_ALI_REQ1

IOMAN_ALI_REQ1.port4

Field	Bits	Default	Access	Description
port4	7:0	00000000b	R/W	Analog Input Mode Request: P4[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

IOMAN_ALI_REQ1.port5

Field	Bits	Default	Access	Description
port5	15:8	00000000b	R/W	Analog Input Mode Request: P5[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

IOMAN_ALI_REQ1.port6

Field	Bits	Default	Access	Description
port6	23:16	00000000b	R/W	Analog Input Mode Request: P6[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

IOMAN_ALI_REQ1.port7

Field	Bits	Default	Access	Description
port7	31:24	00000000b	R/W	Analog Input Mode Request: P7[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

4.4.1.7 IOMAN_ALI_ACK0**IOMAN_ALI_ACK0.port0**

Field	Bits	Default	Access	Description
port0	7:0	00000000b	R/O	Analog In Mode Acknowledge: P0[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

IOMAN_ALI_ACK0.port1

Field	Bits	Default	Access	Description
port1	15:8	00000000b	R/O	Analog In Mode Acknowledge: P1[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

IOMAN_ALI_ACK0.port2

Field	Bits	Default	Access	Description
port2	23:16	00000000b	R/O	Analog In Mode Acknowledge: P2[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

IOMAN_ALI_ACK0.port3

Field	Bits	Default	Access	Description
port3	31:24	00000000b	R/O	Analog In Mode Acknowledge: P3[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

4.4.1.8 IOMAN_ALI_ACK1

IOMAN_ALI_ACK1.port4

Field	Bits	Default	Access	Description
port4	7:0	00000000b	R/O	Analog In Mode Acknowledge: P4[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

IOMAN_ALI_ACK1.port5

Field	Bits	Default	Access	Description
port5	15:8	00000000b	R/O	Analog In Mode Acknowledge: P5[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

IOMAN_ALI_ACK1.port6

Field	Bits	Default	Access	Description
port6	23:16	00000000b	R/O	Analog In Mode Acknowledge: P6[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

IOMAN_ALI_ACK1.port7

Field	Bits	Default	Access	Description
port7	31:24	00000000b	R/O	Analog In Mode Acknowledge: P7[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

4.4.1.9 IOMAN_SPI0_REQ

IOMAN_SPI0_REQ.mapping

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/W	SPI0 I/O Mapping Select

- 00b: Select pin mapping A for all enabled SPI pins
- 01b: Select pin mapping B (if supported)
- 10b: Select pin mapping C (if supported)
- 11b: Select pin mapping D (if supported)

IOMAN_SPI0_REQ.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/W	SPI0 Core I/O Request

1:Requests SPI mode for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPI0_REQ.ss0_io

Field	Bits	Default	Access	Description
ss0_io	8	0	R/W	SPI0 SS[0] I/O Request

1:Requests SPI mode for SS[0].

IOMAN_SPI0_REQ.ss1_io

Field	Bits	Default	Access	Description
ss1_io	9	0	R/W	SPI0 SS[1] I/O Request

1:Requests SPI mode for SS[1].

IOMAN_SPI0_REQ.ss2_io

Field	Bits	Default	Access	Description
ss2_io	10	0	R/W	SPI0 SS[2] I/O Request

1:Requests SPI mode for SS[2].

IOMAN_SPI0_REQ.ss3_io

Field	Bits	Default	Access	Description
ss3_io	11	0	R/W	SPI0 SS[3] I/O Request

1:Requests SPI mode for SS[3].

IOMAN_SPI0_REQ.ss4_io

Field	Bits	Default	Access	Description
ss4_io	12	0	R/W	SPI0 SS[4] I/O Request

1:Requests SPI mode for SS[4].

IOMAN_SPI0_REQ.sr0_io

Field	Bits	Default	Access	Description
sr0_io	16	0	R/W	SPI0 SR[0] I/O Request

1:Requests SPI mode for SR[0].

IOMAN_SPI0_REQ.sr1_io

Field	Bits	Default	Access	Description
sr1_io	17	0	R/W	SPI0 SR[1] I/O Request

1:Requests SPI mode for SR[1].

IOMAN_SPI0_REQ.quad_io

Field	Bits	Default	Access	Description
quad_io	20	0	R/W	SPI0 Quad I/O Request

1:Requests SPI mode for SDIO[2] and SDIO[3].

IOMAN_SPI0_REQ.fast_mode

Field	Bits	Default	Access	Description
fast_mode	24	0	R/W	SPI0 Fast Mode

1:Enables faster pad output transitions.

4.4.1.10 IOMAN_SPI0_ACK**IOMAN_SPI0_ACK.mapping**

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/O	SPI0 I/O Mapping Acknowledge

Mirror of I/O mapping select bits from REQ bits 1:0

IOMAN_SPI0_ACK.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/O	SPI0 Core I/O Acknowledge

1:Acknowledges SPI0 mode selected for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPI0_ACK.ss0_io

Field	Bits	Default	Access	Description
ss0_io	8	0	R/O	SPI0 SS[0] I/O Acknowledge

1:Acknowledges SPI0 mode selected for SS[0].

IOMAN_SPI0_ACK.ss1_io

Field	Bits	Default	Access	Description
ss1_io	9	0	R/O	SPI0 SS[1] I/O Acknowledge

1:Acknowledges SPI0 mode selected for SS[1].

IOMAN_SPI0_ACK.ss2_io

Field	Bits	Default	Access	Description
ss2_io	10	0	R/O	SPI0 SS[2] I/O Acknowledge

1:Acknowledges SPI0 mode selected for SS[2].

IOMAN_SPI0_ACK.ss3_io

Field	Bits	Default	Access	Description
ss3_io	11	0	R/O	SPI0 SS[3] I/O Acknowledge

1:Acknowledges SPI0 mode selected for SS[3].

IOMAN_SPI0_ACK.ss4_io

Field	Bits	Default	Access	Description
ss4_io	12	0	R/O	SPI0 SS[4] I/O Acknowledge

1:Acknowledges SPI0 mode selected for SS[4].

IOMAN_SPI0_ACK.sr0_io

Field	Bits	Default	Access	Description
sr0_io	16	0	R/O	SPI0 SR[0] I/O Acknowledge

1:Acknowledges SPI0 mode selected for SR[0].

IOMAN_SPI0_ACK.sr1_io

Field	Bits	Default	Access	Description
sr1_io	17	0	R/O	SPI0 SR[1] I/O Acknowledge

1:Acknowledges SPI0 mode selected for SR[1].

IOMAN_SPI0_ACK.quad_io

Field	Bits	Default	Access	Description
quad_io	20	0	R/O	SPI0 Quad I/O Acknowledge

1:Acknowledges SPI0 mode selected for SDIO[3:2].

IOMAN_SPI0_ACK.fast_mode

Field	Bits	Default	Access	Description
fast_mode	24	0	R/O	SPI0 Fast Mode Acknowledge

4.4.1.11 IOMAN_SPI1_REQ**IOMAN_SPI1_REQ.mapping**

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/W	SPI1 I/O Mapping Select

- 00b: Select pin mapping A for all enabled SPI pins
- 01b: Select pin mapping B (if supported)
- 10b: Select pin mapping C (if supported)
- 11b: Select pin mapping D (if supported)

IOMAN_SPI1_REQ.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/W	SPI1 Core I/O Request

1:Requests SPI mode for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPI1_REQ.ss0_io

Field	Bits	Default	Access	Description
ss0_io	8	0	R/W	SPI1 SS[0] I/O Request

1:Requests SPI mode for SS[0].

IOMAN_SPI1_REQ.ss1_io

Field	Bits	Default	Access	Description
ss1_io	9	0	R/W	SPI1 SS[1] I/O Request

1:Requests SPI mode for SS[1].

IOMAN_SPI1_REQ.ss2_io

Field	Bits	Default	Access	Description
ss2_io	10	0	R/W	SPI1 SS[2] I/O Request

1:Requests SPI mode for SS[2].

IOMAN_SPI1_REQ.ss3_io

Field	Bits	Default	Access	Description
ss3_io	11	0	R/W	SPI1 SS[3] I/O Request

1:Requests SPI mode for SS[3].

IOMAN_SPI1_REQ.ss4_io

Field	Bits	Default	Access	Description
ss4_io	12	0	R/W	SPI1 SS[4] I/O Request

1:Requests SPI mode for SS[4].

IOMAN_SPI1_REQ.sr0_io

Field	Bits	Default	Access	Description
sr0_io	16	0	R/W	SPI1 SR[0] I/O Request

1:Requests SPI mode for SR[0].

IOMAN_SPI1_REQ.sr1_io

Field	Bits	Default	Access	Description
sr1_io	17	0	R/W	SPI1 SR[1] I/O Request

1:Requests SPI mode for SR[1].

IOMAN_SPI1_REQ.quad_io

Field	Bits	Default	Access	Description
quad_io	20	0	R/W	SPI1 Quad I/O Request

1:Requests SPI mode for SDIO[2] and SDIO[3].

IOMAN_SPI1_REQ.fast_mode

Field	Bits	Default	Access	Description
fast_mode	24	0	R/W	SPI1 Fast Mode

1:Enables faster pad output transitions.

4.4.1.12 IOMAN_SPI1_ACK**IOMAN_SPI1_ACK.mapping**

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/O	SPI1 I/O Mapping Acknowledge

Mirror of I/O mapping select bits from REQ bits 1:0

IOMAN_SPI1_ACK.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/O	SPI1 Core I/O Acknowledge

1:Acknowledges SPI1 mode selected for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPI1_ACK.ss0_io

Field	Bits	Default	Access	Description
ss0_io	8	0	R/O	SPI1 SS[0] I/O Acknowledge

1:Acknowledges SPI1 mode selected for SS[0].

IOMAN_SPI1_ACK.ss1_io

Field	Bits	Default	Access	Description
ss1_io	9	0	R/O	SPI1 SS[1] I/O Acknowledge

1:Acknowledges SPI1 mode selected for SS[1].

IOMAN_SPI1_ACK.ss2_io

Field	Bits	Default	Access	Description
ss2_io	10	0	R/O	SPI1 SS[2] I/O Acknowledge

1:Acknowledges SPI1 mode selected for SS[2].

IOMAN_SPI1_ACK.ss3_io

Field	Bits	Default	Access	Description
ss3_io	11	0	R/O	SPI1 SS[3] I/O Acknowledge

1:Acknowledges SPI1 mode selected for SS[3].

IOMAN_SPI1_ACK.ss4_io

Field	Bits	Default	Access	Description
ss4_io	12	0	R/O	SPI1 SS[4] I/O Acknowledge

1:Acknowledges SPI1 mode selected for SS[4].

IOMAN_SPI1_ACK.sr0_io

Field	Bits	Default	Access	Description
sr0_io	16	0	R/O	SPI1 SR[0] I/O Acknowledge

1:Acknowledges SPI1 mode selected for SR[0].

IOMAN_SPI1_ACK.sr1_io

Field	Bits	Default	Access	Description
sr1_io	17	0	R/O	SPI1 SR[1] I/O Acknowledge

1:Acknowledges SPI1 mode selected for SR[1].

IOMAN_SPI1_ACK.quad_io

Field	Bits	Default	Access	Description
quad_io	20	0	R/O	SPI1 Quad I/O Acknowledge

1:Acknowledges SPI1 mode selected for SDIO[3:2].

IOMAN_SPI1_ACK.fast_mode

Field	Bits	Default	Access	Description
fast_mode	24	0	R/O	SPI1 Fast Mode Acknowledge

4.4.1.13 IOMAN_SPI2_REQ**IOMAN_SPI2_REQ.mapping**

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/W	SPI2 I/O Mapping Select

- 00b: Select pin mapping A for all enabled SPI pins
- 01b: Select pin mapping B (if supported)
- 10b: Select pin mapping C (if supported)
- 11b: Select pin mapping D (if supported)

IOMAN_SPI2_REQ.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/W	SPI2 Core I/O Request

1:Requests SPI mode for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPI2_REQ.ss0_io

Field	Bits	Default	Access	Description
ss0_io	8	0	R/W	SPI2 SS[0] I/O Request

1:Requests SPI mode for SS[0].

IOMAN_SPI2_REQ.ss1_io

Field	Bits	Default	Access	Description
ss1_io	9	0	R/W	SPI2 SS[1] I/O Request

1:Requests SPI mode for SS[1].

IOMAN_SPI2_REQ.ss2_io

Field	Bits	Default	Access	Description
ss2_io	10	0	R/W	SPI2 SS[2] I/O Request

1:Requests SPI mode for SS[2].

IOMAN_SPI2_REQ.ss3_io

Field	Bits	Default	Access	Description
ss3_io	11	0	R/W	SPI2 SS[3] I/O Request

1:Requests SPI mode for SS[3].

IOMAN_SPI2_REQ.ss4_io

Field	Bits	Default	Access	Description
ss4_io	12	0	R/W	SPI2 SS[4] I/O Request

1:Requests SPI mode for SS[4].

IOMAN_SPI2_REQ.sr0_io

Field	Bits	Default	Access	Description
sr0_io	16	0	R/W	SPI2 SR[0] I/O Request

1:Requests SPI mode for SR[0].

IOMAN_SPI2_REQ.sr1_io

Field	Bits	Default	Access	Description
sr1_io	17	0	R/W	SPI2 SR[1] I/O Request

1:Requests SPI mode for SR[1].

IOMAN_SPI2_REQ.quad_io

Field	Bits	Default	Access	Description
quad_io	20	0	R/W	SPI2 Quad I/O Request

1:Requests SPI mode for SDIO[2] and SDIO[3].

IOMAN_SPI2_REQ.fast_mode

Field	Bits	Default	Access	Description
fast_mode	24	0	R/W	SPI2 Fast Mode

1:Enables faster pad output transitions.

4.4.1.14 IOMAN_SPI2_ACK**IOMAN_SPI2_ACK.mapping**

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/O	SPI2 I/O Mapping Acknowledge

Mirror of I/O mapping select bits from REQ bits 1:0

IOMAN_SPI2_ACK.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/O	SPI2 Core I/O Acknowledge

1:Acknowledges SPI2 mode selected for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPI2_ACK.ss0_io

Field	Bits	Default	Access	Description
ss0_io	8	0	R/O	SPI2 SS[0] I/O Acknowledge

1:Acknowledges SPI2 mode selected for SS[0].

IOMAN_SPI2_ACK.ss1_io

Field	Bits	Default	Access	Description
ss1_io	9	0	R/O	SPI2 SS[1] I/O Acknowledge

1:Acknowledges SPI2 mode selected for SS[1].

IOMAN_SPI2_ACK.ss2_io

Field	Bits	Default	Access	Description
ss2_io	10	0	R/O	SPI2 SS[2] I/O Acknowledge

1:Acknowledges SPI2 mode selected for SS[2].

IOMAN_SPI2_ACK.ss3_io

Field	Bits	Default	Access	Description
ss3_io	11	0	R/O	SPI2 SS[3] I/O Acknowledge

1:Acknowledges SPI2 mode selected for SS[3].

IOMAN_SPI2_ACK.ss4_io

Field	Bits	Default	Access	Description
ss4_io	12	0	R/O	SPI2 SS[4] I/O Acknowledge

1:Acknowledges SPI2 mode selected for SS[4].

IOMAN_SPI2_ACK.sr0_io

Field	Bits	Default	Access	Description
sr0_io	16	0	R/O	SPI2 SR[0] I/O Acknowledge

1:Acknowledges SPI2 mode selected for SR[0].

IOMAN_SPI2_ACK.sr1_io

Field	Bits	Default	Access	Description
sr1_io	17	0	R/O	SPI2 SR[1] I/O Acknowledge

1:Acknowledges SPI2 mode selected for SR[1].

IOMAN_SPI2_ACK.quad_io

Field	Bits	Default	Access	Description
quad_io	20	0	R/O	SPI2 Quad I/O Acknowledge

1:Acknowledges SPI2 mode selected for SDIO[3:2].

IOMAN_SPI2_ACK.fast_mode

Field	Bits	Default	Access	Description
fast_mode	24	0	R/O	SPI2 Fast Mode Acknowledge

4.4.1.15 IOMAN_UART0_REQ**IOMAN_UART0_REQ.mapping**

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/W	UART0 I/O Mapping Select

- 00b: Select pin mapping A for all UART0 pins
- 01b: Select pin mapping B (if supported)
- 10b: Select pin mapping C (if supported)
- 11b: Select pin mapping D (if supported)

IOMAN_UART0_REQ.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/W	UART0 Core I/O (RX/TX) Request

1:Requests UART0 mode for RX and TX pins.

IOMAN_UART0_REQ.cts_io

Field	Bits	Default	Access	Description
cts_io	5	0	R/W	UART0 CTS I/O Request

1:Requests UART0 mode for CTS pin.

IOMAN_UART0_REQ.rts_io

Field	Bits	Default	Access	Description
rts_io	6	0	R/W	UART0 RTS I/O Request

1:Requests UART0 mode for RTS pin.

4.4.1.16 IOMAN_UART0_ACK

IOMAN_UART0_ACK.mapping

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/O	UART0 I/O Mapping Acknowledge

Mirror of I/O mapping select bits from REQ bits 1:0

IOMAN_UART0_ACK.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/O	UART0 Core I/O (RX/TX) Acknowledge

1:Acknowledges UART0 mode selected for RX and TX.

IOMAN_UART0_ACK.cts_io

Field	Bits	Default	Access	Description
cts_io	5	0	R/O	UART0 CTS I/O Acknowledge

1:Acknowledges UART0 mode selected for CTS.

IOMAN_UART0_ACK.rts_io

Field	Bits	Default	Access	Description
rts_io	6	0	R/O	UART0 RTS I/O Acknowledge

1:Acknowledges UART0 mode selected for RTS.

4.4.1.17 IOMAN_UART1_REQ

IOMAN_UART1_REQ.mapping

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/W	UART1 I/O Mapping Select

- 00b: Select pin mapping A for all UART1 pins
- 01b: Select pin mapping B (if supported)
- 10b: Select pin mapping C (if supported)
- 11b: Select pin mapping D (if supported)

IOMAN_UART1_REQ.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/W	UART1 Core I/O (RX/TX) Request

1:Requests UART1 mode for RX and TX pins.

IOMAN_UART1_REQ.cts_io

Field	Bits	Default	Access	Description
cts_io	5	0	R/W	UART1 CTS I/O Request

1:Requests UART1 mode for CTS pin.

IOMAN_UART1_REQ.rts_io

Field	Bits	Default	Access	Description
rts_io	6	0	R/W	UART1 RTS I/O Request

1:Requests UART1 mode for RTS pin.

4.4.1.18 IOMAN_UART1_ACK**IOMAN_UART1_ACK.mapping**

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/O	UART0 I/O Mapping Acknowledge

Mirror of I/O mapping select bits from REQ bits 1:0

IOMAN_UART1_ACK.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/O	UART1 Core I/O (RX/TX) Acknowledge

1:Acknowledges UART1 mode selected for RX and TX.

IOMAN_UART1_ACK.cts_io

Field	Bits	Default	Access	Description
cts_io	5	0	R/O	UART1 CTS I/O Acknowledge

1:Acknowledges UART1 mode selected for CTS.

IOMAN_UART1_ACK.rts_io

Field	Bits	Default	Access	Description
rts_io	6	0	R/O	UART1 RTS I/O Acknowledge

1:Acknowledges UART1 mode selected for RTS.

4.4.1.19 IOMAN_I2CM0_REQ**IOMAN_I2CM0_REQ.mapping**

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/W	I2C Master I/O Mapping Select

- 00b: Select pin mapping A for all I2C Master pins

- 01b: Select pin mapping B (if supported)
- 10b: Select pin mapping C (if supported)
- 11b: Select pin mapping D (if supported)

IOMAN_I2CM0_REQ.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/W	I2C Master I/O Request

1:Requests I2C Master mode for SCL and SDA pins.

4.4.1.20 IOMAN_I2CM0_ACK**IOMAN_I2CM0_ACK.mapping**

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/O	I2C Master I/O Mapping Acknowledge

Mirror of I/O mapping select bits from REQ bits 1:0

IOMAN_I2CM0_ACK.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/O	I2C Master I/O Acknowledge

1:Acknowledges I2C Master mode selected for SCL/SDA

4.4.1.21 IOMAN_I2CS0_REQ**IOMAN_I2CS0_REQ.mapping**

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/W	I2C Slave I/O Mapping Select

- 00b: Select pin mapping A for all I2C Slave pins
- 01b: Select pin mapping B (if supported)
- 10b: Select pin mapping C (if supported)
- 11b: Select pin mapping D (if supported)

IOMAN_I2CS0_REQ.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/W	I2C Slave I/O Request

1:Requests I2C Slave mode for SCL and SDA pins.

4.4.1.22 IOMAN_I2CS0_ACK**IOMAN_I2CS0_ACK.mapping**

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/O	I2C Slave I/O Mapping Acknowledge

Mirror of I/O mapping select bits from REQ bits 1:0

IOMAN_I2CS0_ACK.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/O	I2C Slave I/O Acknowledge

1:Acknowledges I2C Slave mode selected for SCL/SDA

4.4.1.23 IOMAN_LCD_COM_REQ**IOMAN_LCD_COM_REQ.com_io**

Field	Bits	Default	Access	Description
com_io	0	0	R/W	LCD COM I/O Request

1:Requests LCD COM mode for COM[3:0] (all four)

4.4.1.24 IOMAN_LCD_COM_ACK**IOMAN_LCD_COM_ACK.com_io**

Field	Bits	Default	Access	Description
com_io	0	0	R/O	LCD COM I/O Acknowledge

1:Acknowledges LCD COM mode selected for COM[3:0].

4.4.1.25 IOMAN_LCD_SEG_REQ0**IOMAN_LCD_SEG_REQ0.io_req_24**

Field	Bits	Default	Access	Description
io_req_24	0	0	R/W	LCD SEG I/O Request for GPIO 24

Field	Bits	Default	Access	Description
-------	------	---------	--------	-------------

1:Requests LCD SEG mode for this GPIO. (P3.0)

IOMAN_LCD_SEG_REQ0.io_req_25

Field	Bits	Default	Access	Description
io_req_25	1	0	R/W	LCD SEG I/O Request for GPIO 25

1:Requests LCD SEG mode for this GPIO. (P3.1)

IOMAN_LCD_SEG_REQ0.io_req_26

Field	Bits	Default	Access	Description
io_req_26	2	0	R/W	LCD SEG I/O Request for GPIO 26

1:Requests LCD SEG mode for this GPIO. (P3.2)

IOMAN_LCD_SEG_REQ0.io_req_27

Field	Bits	Default	Access	Description
io_req_27	3	0	R/W	LCD SEG I/O Request for GPIO 27

1:Requests LCD SEG mode for this GPIO. (P3.3)

IOMAN_LCD_SEG_REQ0.io_req_28

Field	Bits	Default	Access	Description
io_req_28	4	0	R/W	LCD SEG I/O Request for GPIO 28

1:Requests LCD SEG mode for this GPIO. (P3.4)

IOMAN_LCD_SEG_REQ0.io_req_29

Field	Bits	Default	Access	Description
io_req_29	5	0	R/W	LCD SEG I/O Request for GPIO 29

1:Requests LCD SEG mode for this GPIO. (P3.5)

IOMAN_LCD_SEG_REQ0.io_req_30

Field	Bits	Default	Access	Description
io_req_30	6	0	R/W	LCD SEG I/O Request for GPIO 30

1:Requests LCD SEG mode for this GPIO. (P3.6)

IOMAN_LCD_SEG_REQ0.io_req_31

Field	Bits	Default	Access	Description
io_req_31	7	0	R/W	LCD SEG I/O Request for GPIO 31

1:Requests LCD SEG mode for this GPIO. (P3.7)

IOMAN_LCD_SEG_REQ0.io_req_32

Field	Bits	Default	Access	Description
io_req_32	8	0	R/W	LCD SEG I/O Request for GPIO 32

1:Requests LCD SEG mode for this GPIO. (P4.0)

IOMAN_LCD_SEG_REQ0.io_req_33

Field	Bits	Default	Access	Description
io_req_33	9	0	R/W	LCD SEG I/O Request for GPIO 33

1:Requests LCD SEG mode for this GPIO. (P4.1)

IOMAN_LCD_SEG_REQ0.io_req_34

Field	Bits	Default	Access	Description
io_req_34	10	0	R/W	LCD SEG I/O Request for GPIO 34

1:Requests LCD SEG mode for this GPIO. (P4.2)

IOMAN_LCD_SEG_REQ0.io_req_35

Field	Bits	Default	Access	Description
io_req_35	11	0	R/W	LCD SEG I/O Request for GPIO 35

1:Requests LCD SEG mode for this GPIO. (P4.3)

IOMAN_LCD_SEG_REQ0.io_req_36

Field	Bits	Default	Access	Description
io_req_36	12	0	R/W	LCD SEG I/O Request for GPIO 36

1:Requests LCD SEG mode for this GPIO. (P4.4)

IOMAN_LCD_SEG_REQ0.io_req_37

Field	Bits	Default	Access	Description
io_req_37	13	0	R/W	LCD SEG I/O Request for GPIO 37

1:Requests LCD SEG mode for this GPIO. (P4.5)

IOMAN_LCD_SEG_REQ0.io_req_38

Field	Bits	Default	Access	Description
io_req_38	14	0	R/W	LCD SEG I/O Request for GPIO 38

1:Requests LCD SEG mode for this GPIO. (P4.6)

IOMAN_LCD_SEG_REQ0.io_req_39

Field	Bits	Default	Access	Description
io_req_39	15	0	R/W	LCD SEG I/O Request for GPIO 39

1:Requests LCD SEG mode for this GPIO. (P4.7)

IOMAN_LCD_SEG_REQ0.io_req_40

Field	Bits	Default	Access	Description
io_req_40	16	0	R/W	LCD SEG I/O Request for GPIO 40

1:Requests LCD SEG mode for this GPIO. (P5.0)

IOMAN_LCD_SEG_REQ0.io_req_41

Field	Bits	Default	Access	Description
io_req_41	17	0	R/W	LCD SEG I/O Request for GPIO 41

1:Requests LCD SEG mode for this GPIO. (P5.1)

IOMAN_LCD_SEG_REQ0.io_req_42

Field	Bits	Default	Access	Description
io_req_42	18	0	R/W	LCD SEG I/O Request for GPIO 42

1:Requests LCD SEG mode for this GPIO. (P5.2)

IOMAN_LCD_SEG_REQ0.io_req_43

Field	Bits	Default	Access	Description
io_req_43	19	0	R/W	LCD SEG I/O Request for GPIO 43

1:Requests LCD SEG mode for this GPIO. (P5.3)

IOMAN_LCD_SEG_REQ0.io_req_44

Field	Bits	Default	Access	Description
io_req_44	20	0	R/W	LCD SEG I/O Request for GPIO 44

1:Requests LCD SEG mode for this GPIO. (P5.4)

IOMAN_LCD_SEG_REQ0.io_req_45

Field	Bits	Default	Access	Description
io_req_45	21	0	R/W	LCD SEG I/O Request for GPIO 45

1:Requests LCD SEG mode for this GPIO. (P5.5)

IOMAN_LCD_SEG_REQ0.io_req_46

Field	Bits	Default	Access	Description
io_req_46	22	0	R/W	LCD SEG I/O Request for GPIO 46

1:Requests LCD SEG mode for this GPIO. (P5.6)

IOMAN_LCD_SEG_REQ0.io_req_47

Field	Bits	Default	Access	Description
io_req_47	23	0	R/W	LCD SEG I/O Request for GPIO 47

1:Requests LCD SEG mode for this GPIO. (P5.7)

IOMAN_LCD_SEG_REQ0.io_req_48

Field	Bits	Default	Access	Description
io_req_48	24	0	R/W	LCD SEG I/O Request for GPIO 48

1:Requests LCD SEG mode for this GPIO. (P6.0)

IOMAN_LCD_SEG_REQ0.io_req_49

Field	Bits	Default	Access	Description
io_req_49	25	0	R/W	LCD SEG I/O Request for GPIO 49

1:Requests LCD SEG mode for this GPIO. (P6.1)

IOMAN_LCD_SEG_REQ0.io_req_50

Field	Bits	Default	Access	Description
io_req_50	26	0	R/W	LCD SEG I/O Request for GPIO 50

1:Requests LCD SEG mode for this GPIO. (P6.2)

IOMAN_LCD_SEG_REQ0.io_req_51

Field	Bits	Default	Access	Description
io_req_51	27	0	R/W	LCD SEG I/O Request for GPIO 51

1:Requests LCD SEG mode for this GPIO. (P6.3)

IOMAN_LCD_SEG_REQ0.io_req_52

Field	Bits	Default	Access	Description
io_req_52	28	0	R/W	LCD SEG I/O Request for GPIO 52

1:Requests LCD SEG mode for this GPIO. (P6.4)

IOMAN_LCD_SEG_REQ0.io_req_53

Field	Bits	Default	Access	Description
io_req_53	29	0	R/W	LCD SEG I/O Request for GPIO 53

1:Requests LCD SEG mode for this GPIO. (P6.5)

IOMAN_LCD_SEG_REQ0.io_req_54

Field	Bits	Default	Access	Description
io_req_54	30	0	R/W	LCD SEG I/O Request for GPIO 54

1:Requests LCD SEG mode for this GPIO. (P6.6)

IOMAN_LCD_SEG_REQ0.io_req_55

Field	Bits	Default	Access	Description
io_req_55	31	0	R/W	LCD SEG I/O Request for GPIO 55

1:Requests LCD SEG mode for this GPIO. (P6.7)

4.4.1.26 IOMAN_LCD_SEG_REQ1**IOMAN_LCD_SEG_REQ1.io_req_56**

Field	Bits	Default	Access	Description
io_req_56	0	0	R/W	LCD SEG I/O Request for GPIO 56

1:Requests LCD SEG mode for this GPIO. (P7.0)

IOMAN_LCD_SEG_REQ1.io_req_57

Field	Bits	Default	Access	Description
io_req_57	1	0	R/W	LCD SEG I/O Request for GPIO 57

1:Requests LCD SEG mode for this GPIO. (P7.1)

IOMAN_LCD_SEG_REQ1.io_req_58

Field	Bits	Default	Access	Description
io_req_58	2	0	R/W	LCD SEG I/O Request for GPIO 58

1:Requests LCD SEG mode for this GPIO. (P7.2)

IOMAN_LCD_SEG_REQ1.io_req_59

Field	Bits	Default	Access	Description
io_req_59	3	0	R/W	LCD SEG I/O Request for GPIO 59

1:Requests LCD SEG mode for this GPIO. (P7.3)

IOMAN_LCD_SEG_REQ1.io_req_60

Field	Bits	Default	Access	Description
io_req_60	4	0	R/W	LCD SEG I/O Request for GPIO 60

1:Requests LCD SEG mode for this GPIO. (P7.4)

IOMAN_LCD_SEG_REQ1.io_req_61

Field	Bits	Default	Access	Description
io_req_61	5	0	R/W	LCD SEG I/O Request for GPIO 61

1:Requests LCD SEG mode for this GPIO. (P7.5)

IOMAN_LCD_SEG_REQ1.io_req_62

Field	Bits	Default	Access	Description
io_req_62	6	0	R/W	LCD SEG I/O Request for GPIO 62

1:Requests LCD SEG mode for this GPIO. (P7.6)

IOMAN_LCD_SEG_REQ1.io_req_63

Field	Bits	Default	Access	Description
io_req_63	7	0	R/W	LCD SEG I/O Request for GPIO 63

1:Requests LCD SEG mode for this GPIO. (P7.7)

4.4.1.27 IOMAN_LCD_SEG_ACK0**IOMAN_LCD_SEG_ACK0.io_ack_24**

Field	Bits	Default	Access	Description
io_ack_24	0	0	R/O	LCD SEG I/O Acknowledge for GPIO 24

1:Acknowledges SEG mode selected. (P3.0)

IOMAN_LCD_SEG_ACK0.io_ack_25

Field	Bits	Default	Access	Description
io_ack_25	1	0	R/O	LCD SEG I/O Acknowledge for GPIO 25

1:Acknowledges SEG mode selected. (P3.1)

IOMAN_LCD_SEG_ACK0.io_ack_26

Field	Bits	Default	Access	Description
io_ack_26	2	0	R/O	LCD SEG I/O Acknowledge for GPIO 26

1:Acknowledges SEG mode selected. (P3.2)

IOMAN_LCD_SEG_ACK0.io_ack_27

Field	Bits	Default	Access	Description
io_ack_27	3	0	R/O	LCD SEG I/O Acknowledge for GPIO 27

1:Acknowledges SEG mode selected. (P3.3)

IOMAN_LCD_SEG_ACK0.io_ack_28

Field	Bits	Default	Access	Description
io_ack_28	4	0	R/O	LCD SEG I/O Acknowledge for GPIO 28

1:Acknowledges SEG mode selected. (P3.4)

IOMAN_LCD_SEG_ACK0.io_ack_29

Field	Bits	Default	Access	Description
io_ack_29	5	0	R/O	LCD SEG I/O Acknowledge for GPIO 29

1:Acknowledges SEG mode selected. (P3.5)

IOMAN_LCD_SEG_ACK0.io_ack_30

Field	Bits	Default	Access	Description
io_ack_30	6	0	R/O	LCD SEG I/O Acknowledge for GPIO 30

1:Acknowledges SEG mode selected. (P3.6)

IOMAN_LCD_SEG_ACK0.io_ack_31

Field	Bits	Default	Access	Description
io_ack_31	7	0	R/O	LCD SEG I/O Acknowledge for GPIO 31

1:Acknowledges SEG mode selected. (P3.7)

IOMAN_LCD_SEG_ACK0.io_ack_32

Field	Bits	Default	Access	Description
io_ack_32	8	0	R/O	LCD SEG I/O Acknowledge for GPIO 32

1:Acknowledges SEG mode selected. (P4.0)

IOMAN_LCD_SEG_ACK0.io_ack_33

Field	Bits	Default	Access	Description
io_ack_33	9	0	R/O	LCD SEG I/O Acknowledge for GPIO 33

1:Acknowledges SEG mode selected. (P4.1)

IOMAN_LCD_SEG_ACK0.io_ack_34

Field	Bits	Default	Access	Description
io_ack_34	10	0	R/O	LCD SEG I/O Acknowledge for GPIO 34

1:Acknowledges SEG mode selected. (P4.2)

IOMAN_LCD_SEG_ACK0.io_ack_35

Field	Bits	Default	Access	Description
io_ack_35	11	0	R/O	LCD SEG I/O Acknowledge for GPIO 35

1:Acknowledges SEG mode selected. (P4.3)

IOMAN_LCD_SEG_ACK0.io_ack_36

Field	Bits	Default	Access	Description
io_ack_36	12	0	R/O	LCD SEG I/O Acknowledge for GPIO 36

1:Acknowledges SEG mode selected. (P4.4)

IOMAN_LCD_SEG_ACK0.io_ack_37

Field	Bits	Default	Access	Description
io_ack_37	13	0	R/O	LCD SEG I/O Acknowledge for GPIO 37

1:Acknowledges SEG mode selected. (P4.5)

IOMAN_LCD_SEG_ACK0.io_ack_38

Field	Bits	Default	Access	Description
io_ack_38	14	0	R/O	LCD SEG I/O Acknowledge for GPIO 38

1:Acknowledges SEG mode selected. (P4.6)

IOMAN_LCD_SEG_ACK0.io_ack_39

Field	Bits	Default	Access	Description
io_ack_39	15	0	R/O	LCD SEG I/O Acknowledge for GPIO 39

1:Acknowledges SEG mode selected. (P4.7)

IOMAN_LCD_SEG_ACK0.io_ack_40

Field	Bits	Default	Access	Description
io_ack_40	16	0	R/O	LCD SEG I/O Acknowledge for GPIO 40

1:Acknowledges SEG mode selected. (P5.0)

IOMAN_LCD_SEG_ACK0.io_ack_41

Field	Bits	Default	Access	Description
io_ack_41	17	0	R/O	LCD SEG I/O Acknowledge for GPIO 41

1:Acknowledges SEG mode selected. (P5.1)

IOMAN_LCD_SEG_ACK0.io_ack_42

Field	Bits	Default	Access	Description
io_ack_42	18	0	R/O	LCD SEG I/O Acknowledge for GPIO 42

1:Acknowledges SEG mode selected. (P5.2)

IOMAN_LCD_SEG_ACK0.io_ack_43

Field	Bits	Default	Access	Description
io_ack_43	19	0	R/O	LCD SEG I/O Acknowledge for GPIO 43

1:Acknowledges SEG mode selected. (P5.3)

IOMAN_LCD_SEG_ACK0.io_ack_44

Field	Bits	Default	Access	Description
io_ack_44	20	0	R/O	LCD SEG I/O Acknowledge for GPIO 44

1:Acknowledges SEG mode selected. (P5.4)

IOMAN_LCD_SEG_ACK0.io_ack_45

Field	Bits	Default	Access	Description
io_ack_45	21	0	R/O	LCD SEG I/O Acknowledge for GPIO 45

1:Acknowledges SEG mode selected. (P5.5)

IOMAN_LCD_SEG_ACK0.io_ack_46

Field	Bits	Default	Access	Description
io_ack_46	22	0	R/O	LCD SEG I/O Acknowledge for GPIO 46

1:Acknowledges SEG mode selected. (P5.6)

IOMAN_LCD_SEG_ACK0.io_ack_47

Field	Bits	Default	Access	Description
io_ack_47	23	0	R/O	LCD SEG I/O Acknowledge for GPIO 47

1:Acknowledges SEG mode selected. (P5.7)

IOMAN_LCD_SEG_ACK0.io_ack_48

Field	Bits	Default	Access	Description
io_ack_48	24	0	R/O	LCD SEG I/O Acknowledge for GPIO 48

1:Acknowledges SEG mode selected. (P6.0)

IOMAN_LCD_SEG_ACK0.io_ack_49

Field	Bits	Default	Access	Description
io_ack_49	25	0	R/O	LCD SEG I/O Acknowledge for GPIO 49

1:Acknowledges SEG mode selected. (P6.1)

IOMAN_LCD_SEG_ACK0.io_ack_50

Field	Bits	Default	Access	Description
io_ack_50	26	0	R/O	LCD SEG I/O Acknowledge for GPIO 50

1:Acknowledges SEG mode selected. (P6.2)

IOMAN_LCD_SEG_ACK0.io_ack_51

Field	Bits	Default	Access	Description
io_ack_51	27	0	R/O	LCD SEG I/O Acknowledge for GPIO 51

1:Acknowledges SEG mode selected. (P6.3)

IOMAN_LCD_SEG_ACK0.io_ack_52

Field	Bits	Default	Access	Description
io_ack_52	28	0	R/O	LCD SEG I/O Acknowledge for GPIO 52

1:Acknowledges SEG mode selected. (P6.4)

IOMAN_LCD_SEG_ACK0.io_ack_53

Field	Bits	Default	Access	Description
io_ack_53	29	0	R/O	LCD SEG I/O Acknowledge for GPIO 53

1:Acknowledges SEG mode selected. (P6.5)

IOMAN_LCD_SEG_ACK0.io_ack_54

Field	Bits	Default	Access	Description
io_ack_54	30	0	R/O	LCD SEG I/O Acknowledge for GPIO 54

1:Acknowledges SEG mode selected. (P6.6)

IOMAN_LCD_SEG_ACK0.io_ack_55

Field	Bits	Default	Access	Description
io_ack_55	31	0	R/O	LCD SEG I/O Acknowledge for GPIO 55

1:Acknowledges SEG mode selected. (P6.7)

4.4.1.28 IOMAN_LCD_SEG_ACK1**IOMAN_LCD_SEG_ACK1.io_ack_56**

Field	Bits	Default	Access	Description
io_ack_56	0	0	R/O	LCD SEG I/O Acknowledge for GPIO 56

1:Acknowledges SEG mode selected. (P7.0)

IOMAN_LCD_SEG_ACK1.io_ack_57

Field	Bits	Default	Access	Description
io_ack_57	1	0	R/O	LCD SEG I/O Acknowledge for GPIO 57

1:Acknowledges SEG mode selected. (P7.1)

IOMAN_LCD_SEG_ACK1.io_ack_58

Field	Bits	Default	Access	Description
io_ack_58	2	0	R/O	LCD SEG I/O Acknowledge for GPIO 58

1:Acknowledges SEG mode selected. (P7.2)

IOMAN_LCD_SEG_ACK1.io_ack_59

Field	Bits	Default	Access	Description
io_ack_59	3	0	R/O	LCD SEG I/O Acknowledge for GPIO 59

1:Acknowledges SEG mode selected. (P7.3)

IOMAN_LCD_SEG_ACK1.io_ack_60

Field	Bits	Default	Access	Description
io_ack_60	4	0	R/O	LCD SEG I/O Acknowledge for GPIO 60

1:Acknowledges SEG mode selected. (P7.4)

IOMAN_LCD_SEG_ACK1.io_ack_61

Field	Bits	Default	Access	Description
io_ack_61	5	0	R/O	LCD SEG I/O Acknowledge for GPIO 61

1:Acknowledges SEG mode selected. (P7.5)

IOMAN_LCD_SEG_ACK1.io_ack_62

Field	Bits	Default	Access	Description
io_ack_62	6	0	R/O	LCD SEG I/O Acknowledge for GPIO 62

1:Acknowledges SEG mode selected. (P7.6)

IOMAN_LCD_SEG_ACK1.io_ack_63

Field	Bits	Default	Access	Description
io_ack_63	7	0	R/O	LCD SEG I/O Acknowledge for GPIO 63

1:Acknowledges SEG mode selected. (P7.7)

4.4.1.29 IOMAN_CRNT_REQ

IOMAN_CRNT_REQ.[io_req_crnt0, io_req_crnt1, io_req_crnt2, io_req_crnt3, io_req_crnt4, io_req_crnt5, io_req_crnt6, io_req_crnt7]

Field	Bits	Default	Access	Description
io_req_crnt0	0	0	R/W	Request for pair CRNT0
io_req_crnt1	1	0	R/W	Request for pair CRNT1
io_req_crnt2	2	0	R/W	Request for pair CRNT2

Field	Bits	Default	Access	Description
io_req_crnt3	3	0	R/W	Request for pair CRNT3
io_req_crnt4	4	0	R/W	Request for pair CRNT4
io_req_crnt5	5	0	R/W	Request for pair CRNT5
io_req_crnt6	6	0	R/W	Request for pair CRNT6
io_req_crnt7	7	0	R/W	Request for pair CRNT7

1:Requests Current Drive I/O mode for drain/source pair.

4.4.1.30 IOMAN_CRNT_ACK

IOMAN_CRNT_ACK.io_ack_crnt0

Field	Bits	Default	Access	Description
io_ack_crnt0	0	0	R/O	Acknowledge for pair CRNT0

1:Acknowledges Current Drive I/O mode enabled for this pair.

IOMAN_CRNT_ACK.io_ack_crnt1

Field	Bits	Default	Access	Description
io_ack_crnt1	1	0	R/O	Acknowledge for pair CRNT1

1:Acknowledges Current Drive I/O mode enabled for this pair.

IOMAN_CRNT_ACK.io_ack_crnt2

Field	Bits	Default	Access	Description
io_ack_crnt2	2	0	R/O	Acknowledge for pair CRNT2

1:Acknowledges Current Drive I/O mode enabled for this pair.

IOMAN_CRNT_ACK.io_ack_crnt3

Field	Bits	Default	Access	Description
io_ack_crnt3	3	0	R/O	Acknowledge for pair CRNT3

1:Acknowledges Current Drive I/O mode enabled for this pair.

IOMAN_CRNT_ACK.io_ack_crnt4

Field	Bits	Default	Access	Description
io_ack_crnt4	4	0	R/O	Acknowledge for pair CRNT4

1:Acknowledges Current Drive I/O mode enabled for this pair.

IOMAN_CRNT_ACK.io_ack_crnt5

Field	Bits	Default	Access	Description
io_ack_crnt5	5	0	R/O	Acknowledge for pair CRNT5

1:Acknowledges Current Drive I/O mode enabled for this pair.

IOMAN_CRNT_ACK.io_ack_crnt6

Field	Bits	Default	Access	Description
io_ack_crnt6	6	0	R/O	Acknowledge for pair CRNT6

1:Acknowledges Current Drive I/O mode enabled for this pair.

IOMAN_CRNT_ACK.io_ack_crnt7

Field	Bits	Default	Access	Description
io_ack_crnt7	7	0	R/O	Acknowledge for pair CRNT7

1:Acknowledges Current Drive I/O mode enabled for this pair.

4.4.1.31 IOMAN_CRNT_MODE**IOMAN_CRNT_MODE.[io_crnt0, io_crnt1, io_crnt2, io_crnt3, io_crnt4, io_crnt5, io_crnt6, io_crnt7]**

Field	Bits	Default	Access	Description
io_crnt0	3:0	0000b	R/W	Current Drive Mode Select for pair CRNT0
io_crnt1	7:4	0000b	R/W	Current Drive Mode Select for pair CRNT1
io_crnt2	11:8	0000b	R/W	Current Drive Mode Select for pair CRNT2
io_crnt3	15:12	0000b	R/W	Current Drive Mode Select for pair CRNT3
io_crnt4	19:16	0000b	R/W	Current Drive Mode Select for pair CRNT4
io_crnt5	23:20	0000b	R/W	Current Drive Mode Select for pair CRNT5
io_crnt6	27:24	0000b	R/W	Current Drive Mode Select for pair CRNT6
io_crnt7	31:28	0000b	R/W	Current Drive Mode Select for pair CRNT7

Mode control for CRNT output pair.

- 0 = Current Sink A
- 1 = Current Sink B
- 2 = Current Sink A w/Observe
- 3 = Current Sink B w/Observe
- 4 = H-Bridge A
- 5 = H-Bridge B

- 6 = H-Bridge A w/Observe
- 7 = H-Bridge B w/Observe

4.4.1.32 IOMAN_ALI_CONNECT0

Default	Access	Description
00000000h	R/W	Analog I/O Connection Control Register 0

Selects analog connection input for first 32 GPIO.

4.4.1.33 IOMAN_ALI_CONNECT1

Default	Access	Description
00000000h	R/W	Analog I/O Connection Control Register 1

Selects analog connection input for last 32 GPIO.

4.4.1.34 IOMAN_I2CM1_REQ

IOMAN_I2CM1_REQ.mapping

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/W	I2C Master I/O Mapping Select

- 00b: Select pin mapping A for all I2C Master pins
- 01b: Select pin mapping B (if supported)
- 10b: Select pin mapping C (if supported)
- 11b: Select pin mapping D (if supported)

IOMAN_I2CM1_REQ.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/W	I2C Master I/O Request

1:Requests I2C Master mode for SCL and SDA pins.

4.4.1.35 IOMAN_I2CM1_ACK**IOMAN_I2CM1_ACK.mapping**

Field	Bits	Default	Access	Description
mapping	1:0	00b	R/O	I2C Master I/O Mapping Acknowledge

Mirror of I/O mapping select bits from REQ bits 1:0

IOMAN_I2CM1_ACK.core_io

Field	Bits	Default	Access	Description
core_io	4	0	R/O	I2C Master I/O Acknowledge

1:Acknowledges I2C Master mode selected for SCL/SDA

4.4.1.36 IOMAN_PADX_CONTROL**IOMAN_PADX_CONTROL.padx_power_control**

Field	Bits	Default	Access	Description
padx_power_control	0	0	R/W	PADX Power Control

1: Enables I/O on PADX GPIO0 and GPIO1

IOMAN_PADX_CONTROL.padx_gpio0_out_mode

Field	Bits	Default	Access	Description
padx_gpio0_out_mode	5:4	00b	R/W	PADX GPIO0 Out Mode

Sets output drive mode for GPIO PADX pad.

- 0: High impedance
- 1: Pullup
- 2: Drive 0
- 3: Drive 1

IOMAN_PADX_CONTROL.padx_gpio0_input_state

Field	Bits	Default	Access	Description
padx_gpio0_input_state	6	0	R/O	PADX GPIO0 Input Value

Returns input value on GPIO PADX pad.

IOMAN_PADX_CONTROL.padx_gpio1_out_mode

Field	Bits	Default	Access	Description
padx_gpio1_out_mode	9:8	00b	R/W	PADX GPIO1 Out Mode

Sets output drive mode for GPIO PADX pad.

- 0: High impedance
- 1: Pullup
- 2: Drive 0
- 3: Drive 1

IOMAN_PADX_CONTROL.padx_gpio1_input_state

Field	Bits	Default	Access	Description
padx_gpio1_input_state	10	0	R/O	PADX GPIO1 Input Value

Returns input value on GPIO PADX pad.

5 Pin Configurations, Packages, and Special Function Multiplexing

5.1 Pin Layout

There are two logical pin layouts available on the **MAX32600** which determine the location of certain multiplexed functions. The pin layout in effect is determined by the state of an internal pad in the **MAX32600** package which can be bonded either to a high or low setting. The pin layout remapping is performed internal to the **MAX32600** and is independent of the actual pad-to-pin/ball packaging layout that occurs as part of the device assembly process.

The two pin layouts are known as Standard and Compact, with pin functions available as follows (assuming all enabled pins are bonded out on the package used):

Standard Pin Layout

The Standard pin layout is featured on the 12mm x 12mm package and contains all GPIO ports available on the **MAX32600** (P0 through P7). It also includes LCD functionality.

Compact Pin Layout

The Compact pin layout is featured on the 7mm x 7mm package and wafer-level packaging (WLP). It includes a reduced subset of the GPIO port pins, with only P0, P1, and P2 available for use. The LCD function is not available on the Compact pin layout.

Note On the WLP package, the package mux pin is brought out and must be strapped by the user circuit board.

Pin Function Mapping Between Layouts

Certain GPIO-multiplexed pin functions are mapped differently depending on the enabled pin layout.

- P0[7:0] - Mapping is the same, except that current drive options are also included for the Compact layout.
- P1[3:0] - No differences.
- P1[7:4] - Different mapping on Standard vs. Compact layout.
- P2[3:0] - No differences.
- P2[7:4] - Different mapping on Standard vs. Compact layout.
- P3[7:0] - These pins exist on Standard layout only.

- P4[7:0] - These pins exist on Standard layout only.
- P5[7:0] - These pins exist on Standard layout only.
- P6[7:0] - These pins exist on Standard layout only.
- P7[7:0] - These pins exist on Standard layout only.

5.2 Pin Function Mapping

The Port Function Muxing tables below show the peripheral function options available on each port. The functions are shown in order of priority (highest to lowest from left-to-right) for the specific port. In the event that more than one function is enabled on a given port/pin, the highest priority level takes precedence.

The lowest priority function for each [General-Purpose I/O](#) (GPIO) pin is the GPIO function itself, which consists of either direct firmware control over the state of the port pin or linking the pin to a GPIO function such as a pulse train output or a 32-bit timer input or output. GPIO functionality is enabled by default for any GPIO pin that does not have any other function selected.

All GPIO pins can be enabled for external interrupt mode and/or wakeup detection mode.

Note Each peripheral function is shown in different shades of the same base color (e.g., blue represents SPI peripherals). For each peripheral function, multiple mappings to port pins may be available. These are indicated in the function muxing tables with alpha characters in parentheses. For example, SPI0 has four possible mappings; these are shown as SPI0 (A), SPI0 (B), SPI0 (C), and SPI0 (D). In addition, some peripherals support multiple pin options for specific functionality. These are shown in the tables using [#]. For example, SPI supports up to four slave select (SS) signals, which are labeled SS[0], SS[1], SS[2], and SS[3]. These should be used in order as numbered. For one slave select line, SS[0] would be used. For two slave select lines, use SS[0] and SS[1].

5.2.1 Compact Package GPIO Mapping

GPIO port and pin mapping for the Compact Package (7mm x 7mm and WLP).

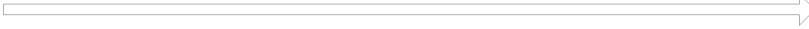
	Highest Priority 						Lowest Priority	
P0 . 0	SPI0 (A) SCK	SPI1 (A) SS[1]	SPI1 (A) SR[0]			Current Drive 0 Drain	UART0 (D) RX	GPIO, TMR0-3 PT0, PT4
P0 . 1	SPI0 (A) SDIO[0] MOSI	SPI1 (A) SS[2]	SPI1 (A) SR[1]			Current Drive 0 Source	UART0 (D) TX	GPIO, TMR0-3 PT0, PT1, PT4
P0 . 2	SPI0 (A) SDIO[1] MISO	SPI1 (A) SS[3]	SPI1 (A) SDIO[2]			Current Drive 1 Drain	UART0 (D) CTS	GPIO, TMR0-3 PT1, PT2, PT5
P0 . 3	SPI0 (A) SS[0]	SPI1 (A) SS[4]	SPI1 (A) SDIO[3]			Current Drive 1 Source	UART0 (D) RTS	GPIO, TMR0-3 PT1, PT3, PT5
P0 . 4	SPI1 (A) SCK	SPI0 (A) SS[1]	SPI0 (A) SR[0]			Current Drive 2 Drain	I2CM0 (D) SDA I2CS0 (D) SDA	GPIO, TMR0-3 PT2, PT4, PT6
P0 . 5	SPI1 (A) SDIO[0] MOSI	SPI0 (A) SS[2]	SPI0 (A) SR[1]			Current Drive 2 Source	I2CM0 (D) SCL I2CS0 (D) SCL	GPIO, TMR0-3 PT2, PT5, PT6
P0 . 6	SPI1 (A) SDIO[1] MISO	SPI0 (A) SS[3]	SPI0 (A) SDIO[2]			Current Drive 3 Drain	I2CM1 (D) SDA I2CS0 (H) SDA	GPIO, TMR0-3 PT3, PT6, PT7
P0 . 7	SPI1 (A) SS[0]	SPI0 (A) SS[4]	SPI0 (A) SDIO[3]			Current Drive 3 Source	I2CM1 (D) SCL I2CS0 (H) SCL	GPIO, TMR0-3 PT3, PT7

Figure 5.1: Compact Port 0 IO Function Muxing

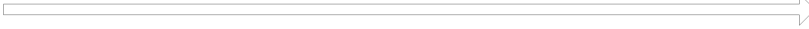
	Highest Priority 						Lowest Priority	
P1 . 0	UART0 (A) RX			SPI0 (B) SCK	SPI1 (B) SS[1]	SPI1 (B) SR[0]		GPIO, TMR0-3 PT0, PT4
P1 . 1	UART0 (A) TX			SPI0 (B) SDIO[0] MOSI	SPI1 (B) SS[2]	SPI1 (B) SR[1]		GPIO, TMR0-3 PT0, PT1, PT4
P1 . 2	UART1 (A) RX	UART0 (A) CTS		SPI0 (B) SDIO[1] MISO	SPI1 (B) SS[3]	SPI1 (B) SDIO[2]		GPIO, TMR0-3 PT1, PT2, PT5
P1 . 3	UART1 (A) TX	UART0 (A) RTS		SPI0 (B) SS[0]	SPI1 (B) SS[4]	SPI1 (B) SDIO[3]		GPIO, TMR0-3 PT1, PT3, PT5
P1 . 4	I2CM0 (A) SDA I2CS0 (A) SDA			SPI1 (B) SCK	SPI0 (B) SS[1]	SPI0 (B) SR[0]		GPIO, TMR0-3 PT2, PT4, PT6
P1 . 5	I2CM0 (A) SCL I2CS0 (A) SCL			SPI1 (B) SDIO[0] MOSI	SPI0 (B) SS[2]	SPI0 (B) SR[1]		GPIO, TMR0-3 PT2, PT5, PT6
P1 . 6	I2CM1 (A) SDA I2CS0 (E) SDA	UART1 (A) CTS		SPI1 (B) SDIO[1] MISO	SPI0 (B) SS[3]	SPI0 (B) SDIO[2]	UART1 (D) RX	GPIO, TMR0-3 PT3, PT6, PT7
P1 . 7	I2CM1 (A) SCL I2CS0 (E) SCL	UART1 (A) RTS	SPI2 (A) SR[0]	SPI1 (B) SS[0]	SPI0 (B) SS[4]	SPI0 (B) SDIO[3]	UART1 (D) TX	GPIO, TMR0-3 PT3, PT7

Figure 5.2: Compact Port 1 IO Function Muxing

	Highest Priority → Lowest Priority						
P2 . 0	SPI2 (A/B) SCK	UART0 (B) RX					GPIO, TMR0-3 PT0, PT4
P2 . 1	SPI2 (A/B) SDIO[0] MOSI	UART0 (B) TX					GPIO, TMR0-3 PT0, PT1, PT4
P2 . 2	SPI2 (A/B) SDIO[1] MISO	I2CM0 (B) SDA	I2CS0 (B) SDA				GPIO, TMR0-3 PT1, PT2, PT5
P2 . 3	SPI2 (A/B) SS[0]	I2CM0 (B) SCL	I2CS0 (B) SCL				GPIO, TMR0-3 PT1, PT3, PT5
P2 . 4		UART1 (B) RX	UART0 (B) CTS	SPI2 (A/B) SS[1]	SPI2 (B) SR[0]		GPIO, TMR0-3 PT2, PT4, PT6
P2 . 5		UART1 (B) TX	UART0 (B) RTS	SPI2 (A/B) SS[2]	SPI2 (A/B) SR[1]		GPIO, TMR0-3 PT2, PT5, PT6
P2 . 6		I2CM1 (B) SDA	I2CS0 (F) SDA	UART1 (B) CTS	SPI2 (A/B) SS[3]	SPI2 (A/B) SDIO[2]	GPIO, TMR0-3 PT3, PT6, PT7
P2 . 7		I2CM1 (B) SCL	I2CS0 (F) SCL	UART1 (B) RTS	SPI2 (A/B) SS[4]	SPI2 (A/B) SDIO[3]	GPIO, TMR0-3 PT3, PT7

Figure 5.3: Compact Port 2 IO Function Muxing

5.2.2 Standard Package GPIO Mapping

GPIO port and pin mapping for the Standard Package (12mm x 12mm).


	Highest Priority 						Lowest Priority	
P0 . 0	SPI0 (A) SCK	SPI1 (A) SS[1]	SPI1 (A) SR[0]				UART0 (D) RX	GPIO, TMR0-3 PT0, PT4
P0 . 1	SPI0 (A) SDIO[0] MOSI	SPI1 (A) SS[2]	SPI1 (A) SR[1]				UART0 (D) TX	GPIO, TMR0-3 PT0, PT1, PT4
P0 . 2	SPI0 (A) SDIO[1] MISO	SPI1 (A) SS[3]	SPI1 (A) SDIO[2]				UART0 (D) CTS	GPIO, TMR0-3 PT1, PT2, PT5
P0 . 3	SPI0 (A) SS[0]	SPI1 (A) SS[4]	SPI1 (A) SDIO[3]				UART0 (D) RTS	GPIO, TMR0-3 PT1, PT3, PT5
P0 . 4	SPI1 (A) SCK	SPI0 (A) SS[1]	SPI0 (A) SR[0]				I2CM0 (D) SDA I2CS0 (D) SDA	GPIO, TMR0-3 PT2, PT4, PT6
P0 . 5	SPI1 (A) SDIO[0] MOSI	SPI0 (A) SS[2]	SPI0 (A) SR[1]				I2CM0 (D) SCL I2CS0 (D) SCL	GPIO, TMR0-3 PT2, PT5, PT6
P0 . 6	SPI1 (A) SDIO[1] MISO	SPI0 (A) SS[3]	SPI0 (A) SDIO[2]				I2CM1 (D) SDA I2CS0 (H) SDA	GPIO, TMR0-3 PT3, PT6, PT7
P0 . 7	SPI1 (A) SS[0]	SPI0 (A) SS[4]	SPI0 (A) SDIO[3]				I2CM1 (D) SCL I2CS0 (H) SCL	GPIO, TMR0-3 PT3, PT7

Figure 5.4: Standard Port 0 IO Function Muxing


	Highest Priority 						Lowest Priority	
P1 . 0	UART0 (A) RX			SPI0 (B) SCK	SPI1 (B) SS[1]	SPI1 (B) SR[0]		GPIO, TMR0-3 PT0, PT4
P1 . 1	UART0 (A) TX			SPI0 (B) SDIO[0] MOSI	SPI1 (B) SS[2]	SPI1 (B) SR[1]		GPIO, TMR0-3 PT0, PT1, PT4
P1 . 2	UART1 (A) RX	UART0 (A) CTS		SPI0 (B) SDIO[1] MISO	SPI1 (B) SS[3]	SPI1 (B) SDIO[2]		GPIO, TMR0-3 PT1, PT2, PT5
P1 . 3	UART1 (A) TX	UART0 (A) RTS		SPI0 (B) SS[0]	SPI1 (B) SS[4]	SPI1 (B) SDIO[3]		GPIO, TMR0-3 PT1, PT3, PT5
P1 . 4	LCD COM0	UART1 (B) RX	UART0 (B) CTS	SPI2 (A/B) SS[1]	SPI2 (B) SR[0]			GPIO, TMR0-3 PT2, PT4, PT6
P1 . 5	LCD COM1	UART1 (B) TX	UART0 (B) RTS	SPI2 (A/B) SS[2]	SPI2 (A/B) SR[1]			GPIO, TMR0-3 PT2, PT5, PT6
P1 . 6	LCD COM2	I2CM1 (B) SDA I2CS0 (F) SDA	UART1 (B) CTS	SPI2 (A/B) SS[3]	SPI2 (A/B) SDIO[2]			GPIO, TMR0-3 PT3, PT6, PT7
P1 . 7	LCD COM3	I2CM1 (B) SCL I2CS0 (F) SCL	UART1 (B) RTS	SPI2 (A/B) SS[4]	SPI2 (A/B) SDIO[3]			GPIO, TMR0-3 PT3, PT7

Figure 5.5: Standard Port 1 IO Function Muxing

	Highest Priority ←							Lowest Priority
P2 . 0	SPI2 (A/B) SCK	UART0 (B) RX						GPIO, TMR0-3 PT0, PT4
P2 . 1	SPI2 (A/B) SDIO[0] MOSI	UART0 (B) TX						GPIO, TMR0-3 PT0, PT1, PT4
P2 . 2	SPI2 (A/B) SDIO[1] MISO	I2CM0 (B) SDA I2CS0 (B) SDA						GPIO, TMR0-3 PT1, PT2, PT5
P2 . 3	SPI2 (A/B) SS[0]	I2CM0 (B) SCL I2CS0 (B) SCL						GPIO, TMR0-3 PT1, PT3, PT5
P2 . 4	I2CM0 (A) SDA I2CS0 (A) SDA			SPI1 (B) SCK	SPI0 (B) SS[1]	SPI0 (B) SR[0]		GPIO, TMR0-3 PT2, PT4, PT6
P2 . 5	I2CM0 (A) SCL I2CS0 (A) SCL			SPI1 (B) SDIO[0] MOSI	SPI0 (B) SS[2]	SPI0 (B) SR[1]		GPIO, TMR0-3 PT2, PT5, PT6
P2 . 6	I2CM1 (A) SDA I2CS0 (E) SDA	UART1 (A) CTS		SPI1 (B) SDIO[1] MISO	SPI0 (B) SS[3]	SPI0 (B) SDIO[2]	UART1 (D) RX	GPIO, TMR0-3 PT3, PT6, PT7
P2 . 7	I2CM1 (A) SCL I2CS0 (E) SCL	UART1 (A) RTS	SPI2 (A) SR[0]	SPI1 (B) SS[0]	SPI0 (B) SS[4]	SPI0 (B) SDIO[3]	UART1 (D) TX	GPIO, TMR0-3 PT3, PT7

Figure 5.6: Standard Port 2 IO Function Muxing

	Highest Priority ←							Lowest Priority
P3 . 0	LCD SEG[0]							GPIO
P3 . 1	LCD SEG[1]							GPIO
P3 . 2	LCD SEG[2]							GPIO
P3 . 3	LCD SEG[3]							GPIO
P3 . 4	LCD SEG[4]							GPIO
P3 . 5	LCD SEG[5]							GPIO
P3 . 6	LCD SEG[6]							GPIO
P3 . 7	LCD SEG[7]							GPIO

Figure 5.7: Standard Port 3 IO Function Muxing

	Highest Priority	→						Lowest Priority
P4 . 0	LCD SEG8							GPIO
P4 . 1	LCD SEG9							GPIO
P4 . 2	LCD SEG10							GPIO
P4 . 3	LCD SEG11							GPIO
P4 . 4	LCD SEG12							GPIO
P4 . 5	LCD SEG13							GPIO
P4 . 6	LCD SEG14							GPIO
P4 . 7	LCD SEG15							GPIO

Figure 5.8: Standard Port 4 IO Function Muxing

	Highest Priority	→						Lowest Priority
P5 . 0	LCD SEG16							GPIO
P5 . 1	LCD SEG17							GPIO
P5 . 2	LCD SEG18							GPIO
P5 . 3	LCD SEG19							GPIO
P5 . 4	LCD SEG20							GPIO
P5 . 5	LCD SEG21							GPIO
P5 . 6	LCD SEG22							GPIO
P5 . 7	LCD SEG23							GPIO

Figure 5.9: Standard Port 5 IO Function Muxing

	Highest Priority				Lowest Priority		
P6 . 0	LCD SEG24	SPI0 (C) SCK	SPI1 (C) SS[1]	SPI1 (C) SR[0]		Current Drive 0 Drain	GPIO, TMR0-3 PT0, PT4
P6 . 1	Analog In LED op0 LCD SEG25	SPI0 (C) SDIO[0] MOSI	SPI1 (C) SS[2]	SPI1 (C) SR[1]		Current Drive 0 Source	GPIO, TMR0-3 PT0, PT1, PT4
P6 . 2	LCD SEG26	SPI0 (C) SDIO[1] MISO	SPI1 (C) SS[3]	SPI1 (C) SDIO[2]		Current Drive 1 Drain	GPIO, TMR0-3 PT1, PT2, PT5
P6 . 3	Analog In LED op0 LCD SEG27	SPI0 (C) SS[0]	SPI1 (C) SS[4]	SPI1 (C) SDIO[3]		Current Drive 1 Source	GPIO, TMR0-3 PT1, PT3, PT5
P6 . 4	LCD SEG28	SPI1 (C) SCK	SPI0 (C) SS[1]	SPI0 (C) SR[0]		Current Drive 2 Drain	GPIO, TMR0-3 PT2, PT4, PT6
P6 . 5	Analog In LED op1 LCD SEG29	SPI1 (C) SDIO[0] MOSI	SPI0 (C) SS[2]	SPI0 (C) SR[1]		Current Drive 2 Source	GPIO, TMR0-3 PT2, PT5, PT6
P6 . 6	LCD SEG30	SPI1 (C) SDIO[1] MISO	SPI0 (C) SS[3]	SPI0 (C) SDIO[2]		Current Drive 3 Drain	GPIO, TMR0-3 PT3, PT6, PT7
P6 . 7	Analog In LED op1 LCD SEG31	SPI1 (C) SS[0]	SPI0 (C) SS[4]	SPI0 (C) SDIO[3]		Current Drive 3 Source	GPIO, TMR0-3 PT3, PT7

Figure 5.10: Standard Port 6 IO Function Muxing

	Highest Priority				Lowest Priority		
P7 . 0	LCD SEG32	UART0 (C) RX				Current Drive 4 Drain	GPIO, TMR0-3 PT0, PT4
P7 . 1	Analog In LED op0 LCD SEG33	UART0 (C) TX				Current Drive 4 Source	GPIO, TMR0-3 PT0, PT1, PT4
P7 . 2	LCD SEG34	UART1 (C) RX	UART0 (C) CTS			Current Drive 5 Drain	GPIO, TMR0-3 PT1, PT2, PT5
P7 . 3	Analog In LED op0 LCD SEG35	UART1 (C) TX	UART0 (C) RTS			Current Drive 5 Source	GPIO, TMR0-3 PT1, PT3, PT5
P7 . 4	LCD SEG36	I2CM0 (C) SDA I2CS0 (C) SDA				Current Drive 6 Drain	GPIO, TMR0-3 PT2, PT4, PT6
P7 . 5	Analog In LED op0 LCD SEG37	I2CM0 (C) SCL I2CS0 (C) SCL				Current Drive 6 Source	GPIO, TMR0-3 PT2, PT5, PT6
P7 . 6	LCD SEG38	I2CM1 (C) SDA I2CS0 (G) SDA	UART1 (C) CTS			Current Drive 7 Drain	GPIO, TMR0-3 PT3, PT6, PT7
P7 . 7	Analog In LED op0 LCD SEG39	I2CM1 (C) SCL I2CS0 (G) SCL	UART1 (C) RTS			Current Drive 7 Source	GPIO, TMR0-3 PT3, PT7

Figure 5.11: Standard Port 7 IO Function Muxing

5.3 General-Purpose I/O

The **MAX32600** includes 64 general-purpose I/O (GPIO) pins which can be controlled directly by firmware or indirectly by other hardware functions (such as output peripherals). GPIOs can be powered by either V_{DDIO} or V_{REG18} (reference [Power Pins](#) for detailed information).

These GPIO pins are logically divided into eight GPIO ports, each port consisting of eight pins. The ports are named as follows: P0, P1, P2, P3, P4, P5, P6, and P7. A single GPIO pin within a port is typically referred to by the notation (port).(pin), where the pin within each port is numbered from 0 to 7. For example, the third pin in Port 2 is referred to as P2.2.

Features supported by the GPIO module include the following:

- Functions are selectable on a per-pin basis, with GPIO operation (meaning that the firmware controls the state of the pin) being the lowest-priority function type.
- When a pin is using GPIO functionality, firmware can directly control the drive strength and output type of the pin (e.g., normal drive vs. open-drain or high-impedance).
- Regardless of the selected function for a given pin, firmware can always monitor the current logic level of the pin using the appropriate registers.
 - All GPIO pins support interrupt detection based on input signals external to the device.
 - Interrupts may be detected on rising or falling edges or high/low levels.
 - Interrupts to the CPU are generated on a per-port basis (i.e., any interrupts originating from Port 0 are assigned to a single interrupt vector, interrupts from Port 1 are assigned to a different vector, and so on).
- When in GPIO operation, simple output-only functions can be assigned to a GPIO pin:
 - Output from Pulse Trains (0 through 7)
 - Output from Timers running in 32-bit mode
 - In this case, output drive strength and type will still be determined by firmware

Memory Mapping

The GPIO module is controlled and configured by a set of registers mapped into data memory space as follows.

- Access through AHB System Bus to peripheral space (0x4000_0000..0x5FFF_FFFF)
 - AHB-to-APB Bridge (0x4000_0000..0x400F_FFFF)
 - * GPIO APB Bus Slave (0x4000_0000..0x4000_0FFF)

The GPIO module does not include any registers or data areas mapped into AHB peripheral space (from 0x4010_0000..0x401F_FFFF).

Device Pins

When not overridden by a higher-priority function, the GPIO module can be used to control the I/O state of any of the 64 port pins:

- P0.0 through P0.7
- P1.0 through P1.7
- P2.0 through P2.7
- P3.0 through P3.7
- P4.0 through P4.7
- P5.0 through P5.7
- P6.0 through P6.7
- P7.0 through P7.7

Interrupts

The GPIO module reports interrupts to the CPU core using the following interrupt vector channels.

Interrupt Number	Vector	Description
34	0xc8	External interrupt triggered on one or more pins of GPIO Port 0
35	0xcc	External interrupt triggered on one or more pins of GPIO Port 1
36	0xd0	External interrupt triggered on one or more pins of GPIO Port 2
37	0xd4	External interrupt triggered on one or more pins of GPIO Port 3
38	0xd8	External interrupt triggered on one or more pins of GPIO Port 4
39	0xdc	External interrupt triggered on one or more pins of GPIO Port 5
40	0xe0	External interrupt triggered on one or more pins of GPIO Port 6
41	0xe4	External interrupt triggered on one or more pins of GPIO Port 7

Detailed Description

The GPIO function provides firmware with basic direct control and monitoring capabilities for digital I/O pins. The GPIO module only provides an APB slave interface, as no burst capability is required to implement this function.

Firmware control of the output state is handled through two control registers: `GPIO_OUT_MODE_Pn` which defines one of four modes of operation, while the `GPIO_OUT_VAL_Pn` defines which of the two output states is enabled for that mode.

The following table defines all the combinations of mode and value with their defined drive state and pad control states.

MODE	Value	Drive State	gpio_out	gpio_en
three-state	0	High Z	0	0
three-state	1	Pullup	1	0
Normal	0	Drive 0	0	1
Normal	1	Drive 1	1	1
Open-Drain	0	Drive 0	0	1
Open-Drain	1	High Z	0	0
Open-Drain w/ PU	0	Drive 0	0	1
Open-Drain w/ PU	1	Pullup	1	0

In addition to firmware monitoring of I/O input state, logic is provided to monitor inputs for certain events and to generate interrupts to the processor on detection of these events. Input monitoring functions as expected regardless of pad ownership.

The user can enable interrupt generation on rising edge detection, falling edge detection, or any edge detection. The user can also generate an interrupt on detection of a high or low logic level. Each interrupt status event has a status bit and an enable register. The enable can mask the interrupt from being asserted to the processor. This mask does not affect the setting of the interrupt status bit. Status bits and enable registers are organized along port boundaries; each port that supports interrupt generation then provides an interrupt signal back to the processor.

Inputs can also be monitored for wakeup event generation to wake the system up from a low power state. This logic does not require a clock to be running to generate the event, as a clock is typically not running while asleep.

5.4 GPIO Pins and Peripheral Mode Functions

For all GPIO pins, the GPIO operation is the lowest priority functionality. This mode will be enabled by default for any GPIO pins that have not been requested for use as part of a higher-priority peripheral function.

GPIO Function

All eight GPIO pins in each GPIO port have access to a common set of low-level peripheral functions that can use GPIO pins as inputs or outputs:

- Output signal streams from Pulse Trains 0 through 7 (three options for each pin)
- Input/output signals (depending on configured timer mode) for the 32-bit Timer (0, 1, 2, and 3) modules

Note The following tables apply to both Compact and Standard Pin Layouts; however, only P0, P1, and P2 are available for the Compact Pin Layout.

Px.0 GPIO Pins and Functions

Function Support by Port (Px.0, x=0 to 7)

Peripheral	Pin Function	Function Description	Direction	P0.0	P1.0	P2.0	P3.0	P4.0	P5.0	P6.0	P7.0
GPIO Port Pin	Px.0	Port Pin (Firmware Controlled)	In/Out	X	X	X	X	X	X	X	X
Pulse Train 0	PT0	Pulse Train 0 Output	Out	X	X	X				X	X
Pulse Train 0	PT0	Pulse Train 0 Output	Out	X	X	X				X	X
Pulse Train 4	PT4	Pulse Train 4 Output	Out	X	X	X				X	X
32-bit Timer 0	TMR0	32-bit Timer/Counter 0 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 1	TMR1	32-bit Timer/Counter 1 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 2	TMR2	32-bit Timer/Counter 2 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 3	TMR3	32-bit Timer/Counter 3 Input or Output	In/Out	X	X	X				X	X

Px.1 GPIO Pins and Functions

Function Support by Port (Px.1, x=0 to 7)

Peripheral	Pin Function	Function Description	Direction	P0.1	P1.1	P2.1	P3.1	P4.1	P5.1	P6.1	P7.1
GPIO Port Pin	Px.1	Port Pin (Firmware Controlled)	In/Out	X	X	X	X	X	X	X	X

Peripheral	Pin Function	Function Description	Direction	P0.1	P1.1	P2.1	P3.1	P4.1	P5.1	P6.1	P7.1
Pulse Train 1	PT1	Pulse Train 1 Output	Out	X	X	X				X	X
Pulse Train 4	PT4	Pulse Train 4 Output	Out	X	X	X				X	X
Pulse Train 0	PT0	Pulse Train 0 Output	Out	X	X	X				X	X
32-bit Timer 1	TMR1	32-bit Timer/Counter 1 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 2	TMR2	32-bit Timer/Counter 2 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 3	TMR3	32-bit Timer/Counter 3 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 0	TMR0	32-bit Timer/Counter 0 Input or Output	In/Out	X	X	X				X	X

Px.2 GPIO Pins and Functions

Function Support by Port (Px.2, x=0 to 7)

Peripheral	Pin Function	Function Description	Direction	P0.2	P1.2	P2.2	P3.2	P4.2	P5.2	P6.2	P7.2
GPIO Port Pin	Px.2	Port Pin (Firmware Controlled)	In/Out	X	X	X	X	X	X	X	X
Pulse Train 2	PT2	Pulse Train 2 Output	Out	X	X	X				X	X
Pulse Train 1	PT1	Pulse Train 1 Output	Out	X	X	X				X	X
Pulse Train 5	PT5	Pulse Train 5 Output	Out	X	X	X				X	X
32-bit Timer 2	TMR2	32-bit Timer/Counter 2 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 3	TMR3	32-bit Timer/Counter 3 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 0	TMR0	32-bit Timer/Counter 0 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 1	TMR1	32-bit Timer/Counter 1 Input or Output	In/Out	X	X	X				X	X

Px.3 GPIO Pins and Functions*Function Support by Port (Px.3, x=0 to 7)*

Peripheral	Pin Function	Function Description	Direction	P0.3	P1.3	P2.3	P3.3	P4.3	P5.3	P6.3	P7.3
GPIO Port Pin	Px.3	Port Pin (Firmware Controlled)	In/Out	X	X	X	X	X	X	X	X
Pulse Train 3	PT3	Pulse Train 3 Output	Out	X	X	X				X	X
Pulse Train 5	PT5	Pulse Train 5 Output	Out	X	X	X				X	X
Pulse Train 1	PT1	Pulse Train 1 Output	Out	X	X	X				X	X
32-bit Timer 3	TMR3	32-bit Timer/Counter 3 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 0	TMR0	32-bit Timer/Counter 0 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 1	TMR1	32-bit Timer/Counter 1 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 2	TMR2	32-bit Timer/Counter 2 Input or Output	In/Out	X	X	X				X	X

Px.4 GPIO Pins and Functions*Function Support by Port (Px.4, x=0 to 7)*

Peripheral	Pin Function	Function Description	Direction	P0.4	P1.4	P2.4	P3.4	P4.4	P5.4	P6.4	P7.4
GPIO Port Pin	Px.4	Port Pin (Firmware Controlled)	In/Out	X	X	X	X	X	X	X	X
Pulse Train 4	PT4	Pulse Train 4 Output	Out	X	X	X				X	X
Pulse Train 2	PT2	Pulse Train 2 Output	Out	X	X	X				X	X
Pulse Train 6	PT6	Pulse Train 6 Output	Out	X	X	X				X	X
32-bit Timer 0	TMR0	32-bit Timer/Counter 0 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 1	TMR1	32-bit Timer/Counter 1 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 2	TMR2	32-bit Timer/Counter 2 Input or Output	In/Out	X	X	X				X	X

Peripheral	Pin Function	Function Description	Direction	P0.4	P1.4	P2.4	P3.4	P4.4	P5.4	P6.4	P7.4
32-bit Timer 3	TMR3	32-bit Timer/Counter 3 Input or Output	In/Out	X	X	X				X	X

Px.5 GPIO Pins and Functions

Function Support by Port (Px.5, x=0 to 7)

Peripheral	Pin Function	Function Description	Direction	P0.5	P1.5	P2.5	P3.5	P4.5	P5.5	P6.5	P7.5
GPIO Port Pin	Px.5	Port Pin (Firmware Controlled)	In/Out	X	X	X	X	X	X	X	X
Pulse Train 5	PT5	Pulse Train 5 Output	Out	X	X	X				X	X
Pulse Train 6	PT6	Pulse Train 6 Output	Out	X	X	X				X	X
Pulse Train 2	PT2	Pulse Train 2 Output	Out	X	X	X				X	X
32-bit Timer 1	TMR1	32-bit Timer/Counter 1 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 2	TMR2	32-bit Timer/Counter 2 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 3	TMR3	32-bit Timer/Counter 3 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 0	TMR0	32-bit Timer/Counter 0 Input or Output	In/Out	X	X	X				X	X

Px.6 GPIO Pins and Functions

Function Support by Port (Px.6, x=0 to 7)

Peripheral	Pin Function	Function Description	Direction	P0.6	P1.6	P2.6	P3.6	P4.6	P5.6	P6.6	P7.6
GPIO Port Pin	Px.6	Port Pin (Firmware Controlled)	In/Out	X	X	X	X	X	X	X	X
Pulse Train 6	PT6	Pulse Train 6 Output	Out	X	X	X				X	X
Pulse Train 3	PT3	Pulse Train 3 Output	Out	X	X	X				X	X
Pulse Train 7	PT7	Pulse Train 7 Output	Out	X	X	X				X	X
32-bit Timer 2	TMR2	32-bit Timer/Counter 2 Input or Output	In/Out	X	X	X				X	X

Peripheral	Pin Function	Function Description	Direction	P0.6	P1.6	P2.6	P3.6	P4.6	P5.6	P6.6	P7.6
32-bit Timer 3	TMR3	32-bit Timer/Counter 3 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 0	TMR0	32-bit Timer/Counter 0 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 1	TMR1	32-bit Timer/Counter 1 Input or Output	In/Out	X	X	X				X	X

Px.7 GPIO Pins and Functions

Function Support by Port (Px.7, x=0 to 7)

Peripheral	Pin Function	Function Description	Direction	P0.7	P1.7	P2.7	P3.7	P4.7	P5.7	P6.7	P7.7
GPIO Port Pin	Px.7	Port Pin (Firmware Controlled)	In/Out	X	X	X	X	X	X	X	X
Pulse Train 7	PT7	Pulse Train 7 Output	Out	X	X	X				X	X
Pulse Train 7	PT7	Pulse Train 7 Output	Out	X	X	X				X	X
Pulse Train 3	PT3	Pulse Train 3 Output	Out	X	X	X				X	X
32-bit Timer 3	TMR3	32-bit Timer/Counter 3 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 0	TMR0	32-bit Timer/Counter 0 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 1	TMR1	32-bit Timer/Counter 1 Input or Output	In/Out	X	X	X				X	X
32-bit Timer 2	TMR2	32-bit Timer/Counter 2 Input or Output	In/Out	X	X	X				X	X

5.5 Registers (GPIO)

5.5.1 Module GPIO Registers

Address	Register	32b Word Len	Description
0x40000040	GPIO_FREE_P0	1	Port P0 Free for GPIO Operation Flags
0x40000044	GPIO_FREE_P1	1	Port P1 Free for GPIO Operation Flags
0x40000048	GPIO_FREE_P2	1	Port P2 Free for GPIO Operation Flags
0x4000004C	GPIO_FREE_P3	1	Port P3 Free for GPIO Operation Flags
0x40000050	GPIO_FREE_P4	1	Port P4 Free for GPIO Operation Flags
0x40000054	GPIO_FREE_P5	1	Port P5 Free for GPIO Operation Flags
0x40000058	GPIO_FREE_P6	1	Port P6 Free for GPIO Operation Flags
0x4000005C	GPIO_FREE_P7	1	Port P7 Free for GPIO Operation Flags
0x40000080	GPIO_OUT_MODE_P0	1	Port P0 GPIO Output Drive Mode
0x40000084	GPIO_OUT_MODE_P1	1	Port P1 GPIO Output Drive Mode
0x40000088	GPIO_OUT_MODE_P2	1	Port P2 GPIO Output Drive Mode
0x4000008C	GPIO_OUT_MODE_P3	1	Port P3 GPIO Output Drive Mode
0x40000090	GPIO_OUT_MODE_P4	1	Port P4 GPIO Output Drive Mode
0x40000094	GPIO_OUT_MODE_P5	1	Port P5 GPIO Output Drive Mode
0x40000098	GPIO_OUT_MODE_P6	1	Port P6 GPIO Output Drive Mode
0x4000009C	GPIO_OUT_MODE_P7	1	Port P7 GPIO Output Drive Mode
0x400000C0	GPIO_OUT_VAL_P0	1	Port P0 GPIO Output Value
0x400000C4	GPIO_OUT_VAL_P1	1	Port P1 GPIO Output Value
0x400000C8	GPIO_OUT_VAL_P2	1	Port P2 GPIO Output Value
0x400000CC	GPIO_OUT_VAL_P3	1	Port P3 GPIO Output Value
0x400000D0	GPIO_OUT_VAL_P4	1	Port P4 GPIO Output Value
0x400000D4	GPIO_OUT_VAL_P5	1	Port P5 GPIO Output Value
0x400000D8	GPIO_OUT_VAL_P6	1	Port P6 GPIO Output Value
0x400000DC	GPIO_OUT_VAL_P7	1	Port P7 GPIO Output Value
0x40000100	GPIO_FUNC_SEL_P0	1	Port P0 GPIO Function Select
0x40000104	GPIO_FUNC_SEL_P1	1	Port P1 GPIO Function Select
0x40000108	GPIO_FUNC_SEL_P2	1	Port P2 GPIO Function Select
0x40000118	GPIO_FUNC_SEL_P6	1	Port P6 GPIO Function Select
0x4000011C	GPIO_FUNC_SEL_P7	1	Port P7 GPIO Function Select

Address	Register	32b Word Len	Description
0x40000140	GPIO_IN_MODE_P0	1	Port P0 GPIO Input Monitoring Mode
0x40000144	GPIO_IN_MODE_P1	1	Port P1 GPIO Input Monitoring Mode
0x40000148	GPIO_IN_MODE_P2	1	Port P2 GPIO Input Monitoring Mode
0x4000014C	GPIO_IN_MODE_P3	1	Port P3 GPIO Input Monitoring Mode
0x40000150	GPIO_IN_MODE_P4	1	Port P4 GPIO Input Monitoring Mode
0x40000154	GPIO_IN_MODE_P5	1	Port P5 GPIO Input Monitoring Mode
0x40000158	GPIO_IN_MODE_P6	1	Port P6 GPIO Input Monitoring Mode
0x4000015C	GPIO_IN_MODE_P7	1	Port P7 GPIO Input Monitoring Mode
0x40000180	GPIO_IN_VAL_P0	1	Port P0 GPIO Input Value
0x40000184	GPIO_IN_VAL_P1	1	Port P1 GPIO Input Value
0x40000188	GPIO_IN_VAL_P2	1	Port P2 GPIO Input Value
0x4000018C	GPIO_IN_VAL_P3	1	Port P3 GPIO Input Value
0x40000190	GPIO_IN_VAL_P4	1	Port P4 GPIO Input Value
0x40000194	GPIO_IN_VAL_P5	1	Port P5 GPIO Input Value
0x40000198	GPIO_IN_VAL_P6	1	Port P6 GPIO Input Value
0x4000019C	GPIO_IN_VAL_P7	1	Port P7 GPIO Input Value
0x400001C0	GPIO_INT_MODE_P0	1	Port P0 Interrupt Detection Mode
0x400001C4	GPIO_INT_MODE_P1	1	Port P1 Interrupt Detection Mode
0x400001C8	GPIO_INT_MODE_P2	1	Port P2 Interrupt Detection Mode
0x400001CC	GPIO_INT_MODE_P3	1	Port P3 Interrupt Detection Mode
0x400001D0	GPIO_INT_MODE_P4	1	Port P4 Interrupt Detection Mode
0x400001D4	GPIO_INT_MODE_P5	1	Port P5 Interrupt Detection Mode
0x400001D8	GPIO_INT_MODE_P6	1	Port P6 Interrupt Detection Mode
0x400001DC	GPIO_INT_MODE_P7	1	Port P7 Interrupt Detection Mode
0x40000200	GPIO_INTFL_P0	1	Port P0 Interrupt Flags
0x40000204	GPIO_INTFL_P1	1	Port P1 Interrupt Flags
0x40000208	GPIO_INTFL_P2	1	Port P2 Interrupt Flags
0x4000020C	GPIO_INTFL_P3	1	Port P3 Interrupt Flags
0x40000210	GPIO_INTFL_P4	1	Port P4 Interrupt Flags

Address	Register	32b Word Len	Description
0x40000214	GPIO_INTFL_P5	1	Port P5 Interrupt Flags
0x40000218	GPIO_INTFL_P6	1	Port P6 Interrupt Flags
0x4000021C	GPIO_INTFL_P7	1	Port P7 Interrupt Flags
0x40000240	GPIO_INTEN_P0	1	Port P0 Interrupt Enables
0x40000244	GPIO_INTEN_P1	1	Port P1 Interrupt Enables
0x40000248	GPIO_INTEN_P2	1	Port P2 Interrupt Enables
0x4000024C	GPIO_INTEN_P3	1	Port P3 Interrupt Enables
0x40000250	GPIO_INTEN_P4	1	Port P4 Interrupt Enables
0x40000254	GPIO_INTEN_P5	1	Port P5 Interrupt Enables
0x40000258	GPIO_INTEN_P6	1	Port P6 Interrupt Enables
0x4000025C	GPIO_INTEN_P7	1	Port P7 Interrupt Enables

5.5.1.1 GPIO_FREE_Pn

GPIO_FREE_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Default	Access	Description
pin0	0	s	R/O	Pn.0 GPIO Mode Acknowledge
pin1	1	s	R/O	Pn.1 GPIO Mode Acknowledge
pin2	2	s	R/O	Pn.2 GPIO Mode Acknowledge
pin3	3	s	R/O	Pn.3 GPIO Mode Acknowledge
pin4	4	s	R/O	Pn.4 GPIO Mode Acknowledge
pin5	5	s	R/O	Pn.5 GPIO Mode Acknowledge
pin6	6	s	R/O	Pn.6 GPIO Mode Acknowledge
pin7	7	s	R/O	Pn.7 GPIO Mode Acknowledge

Each bit determines the availability of the associated port pin for GPIO use. If a pin is not available for GPIO use, this means that it has been requested for use by a higher-priority function. Another description would be that these bits act as acknowledge bits for GPIO mode requests (similar to the ACK bits in the I/O Manager module), with the difference that the GPIO mode 'request' is always active; this is not a problem since GPIO is the lowest priority mode that can be requested for pins.

- 0: Port pin is not available for GPIO use.
- 1: Port pin is available for GPIO use.

Note All GPIO registers of this type (GPIO_FREE_Px) follow this same format.

5.5.1.2 GPIO_OUT_MODE_Pn

GPIO_OUT_MODE_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Default	Access	Description
pin0	3:0	0000b	R/W	Pn.0 Output Drive Mode
pin1	7:4	0000b	R/W	Pn.1 Output Drive Mode
pin2	11:8	0000b	R/W	Pn.2 Output Drive Mode
pin3	15:12	0000b	R/W	Pn.3 Output Drive Mode
pin4	19:16	0000b	R/W	Pn.4 Output Drive Mode
pin5	23:20	0000b	R/W	Pn.5 Output Drive Mode
pin6	27:24	0000b	R/W	Pn.6 Output Drive Mode
pin7	31:28	0000b	R/W	Pn.7 Output Drive Mode

- 0: High impedance w/optional weak pullup
- 1: Open drain
- 2: Open drain w/weak pullup
- 3: High impedance
- 4: Normal drive, High impedance
- 5: Normal drive, Drive high/low
- 6: Slow drive, High impedance
- 7: Slow drive, Drive high/low
- 8: Fast drive, High impedance
- 9: Fast drive, Drive high/low

5.5.1.3 GPIO_OUT_VAL_Pn

GPIO_OUT_VAL_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Default	Access	Description
pin0	0	1	R/W	Pn.0 GPIO Output Drive Value
pin1	1	1	R/W	Pn.1 GPIO Output Drive Value
pin2	2	1	R/W	Pn.2 GPIO Output Drive Value
pin3	3	1	R/W	Pn.3 GPIO Output Drive Value
pin4	4	1	R/W	Pn.4 GPIO Output Drive Value
pin5	5	1	R/W	Pn.5 GPIO Output Drive Value
pin6	6	1	R/W	Pn.6 GPIO Output Drive Value
pin7	7	1	R/W	Pn.7 GPIO Output Drive Value

In GPIO mode, selects output drive state for pin.

5.5.1.4 GPIO_FUNC_SEL_Pn

GPIO_FUNC_SEL_Pn.pin0

Field	Bits	Default	Access	Description
pin0	3:0	0000b	R/W	Pn.0 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 0
- 2: Pulse train 0
- 3: Pulse train 4
- 4: 32-bit Timer 0 I/O
- 5: 32-bit Timer 1 I/O

- 6: 32-bit Timer 2 I/O
- 7: 32-bit Timer 3 I/O

GPIO_FUNC_SEL_Pn.pin1

Field	Bits	Default	Access	Description
pin1	7:4	0000b	R/W	Pn.1 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 1
- 2: Pulse train 4
- 3: Pulse train 0
- 4: 32-bit Timer 1 I/O
- 5: 32-bit Timer 2 I/O
- 6: 32-bit Timer 3 I/O
- 7: 32-bit Timer 0 I/O

GPIO_FUNC_SEL_Pn.pin2

Field	Bits	Default	Access	Description
pin2	11:8	0000b	R/W	Pn.2 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 2
- 2: Pulse train 1
- 3: Pulse train 5

- 4: 32-bit Timer 2 I/O
- 5: 32-bit Timer 3 I/O
- 6: 32-bit Timer 0 I/O
- 7: 32-bit Timer 1 I/O

GPIO_FUNC_SEL_Pn.pin3

Field	Bits	Default	Access	Description
pin3	15:12	0000b	R/W	Pn.3 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 3
- 2: Pulse train 5
- 3: Pulse train 1
- 4: 32-bit Timer 3 I/O
- 5: 32-bit Timer 0 I/O
- 6: 32-bit Timer 1 I/O
- 7: 32-bit Timer 2 I/O

GPIO_FUNC_SEL_Pn.pin4

Field	Bits	Default	Access	Description
pin4	19:16	0000b	R/W	Pn.4 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 4

- 2: Pulse train 2
- 3: Pulse train 6
- 4: 32-bit Timer 0 I/O
- 5: 32-bit Timer 1 I/O
- 6: 32-bit Timer 2 I/O
- 7: 32-bit Timer 3 I/O

GPIO_FUNC_SEL_Pn.pin5

Field	Bits	Default	Access	Description
pin5	23:20	0000b	R/W	Pn.5 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 5
- 2: Pulse train 6
- 3: Pulse train 2
- 4: 32-bit Timer 1 I/O
- 5: 32-bit Timer 2 I/O
- 6: 32-bit Timer 3 I/O
- 7: 32-bit Timer 0 I/O

GPIO_FUNC_SEL_Pn.pin6

Field	Bits	Default	Access	Description
pin6	27:24	0000b	R/W	Pn.6 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 6
- 2: Pulse train 3
- 3: Pulse train 0
- 4: 32-bit Timer 2 I/O
- 5: 32-bit Timer 3 I/O
- 6: 32-bit Timer 0 I/O
- 7: 32-bit Timer 1 I/O

GPIO_FUNC_SEL_Pn.pin7

Field	Bits	Default	Access	Description
pin7	31:28	0000b	R/W	Pn.7 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 7
- 2: Pulse train 7
- 3: Pulse train 3
- 4: 32-bit Timer 3 I/O
- 5: 32-bit Timer 0 I/O
- 6: 32-bit Timer 1 I/O
- 7: 32-bit Timer 2 I/O

5.5.1.5 GPIO_IN_MODE_Pn

GPIO_IN_MODE_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Default	Access	Description
pin0	1:0	10b	R/W	Pn.0 Input Monitoring Mode
pin1	5:4	10b	R/W	Pn.1 Input Monitoring Mode
pin2	9:8	10b	R/W	Pn.2 Input Monitoring Mode
pin3	13:12	10b	R/W	Pn.3 Input Monitoring Mode
pin4	17:16	10b	R/W	Pn.4 Input Monitoring Mode
pin5	21:20	10b	R/W	Pn.5 Input Monitoring Mode
pin6	25:24	10b	R/W	Pn.6 Input Monitoring Mode
pin7	29:28	10b	R/W	Pn.7 Input Monitoring Mode

Determines how corresponding GPIO Input Value bit is calculated; also affects input signal for interrupt detection on this pin (if enabled).

- 00b: Normal input
- 01b: Inverted input
- 10b: Always returns 0 regardless of pin logic level (default)
- 11b: Always returns 1 regardless of logic level at pin

5.5.1.6 GPIO_IN_VAL_Pn

GPIO_IN_VAL_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Default	Access	Description
pin0	0	s	R/O	Pn.0 Input Value
pin1	1	s	R/O	Pn.1 Input Value
pin2	2	s	R/O	Pn.2 Input Value
pin3	3	s	R/O	Pn.3 Input Value
pin4	4	s	R/O	Pn.4 Input Value
pin5	5	s	R/O	Pn.5 Input Value
pin6	6	s	R/O	Pn.6 Input Value
pin7	7	s	R/O	Pn.7 Input Value

Returns current input value on this pin, as modified by the corresponding Input Monitoring Mode setting.

5.5.1.7 GPIO_INT_MODE_Pn

GPIO_INT_MODE_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Default	Access	Description
pin0	2:0	000b	R/W	Pn.0 GPIO Interrupt Detection Mode
pin1	6:4	000b	R/W	Pn.1 GPIO Interrupt Detection Mode
pin2	10:8	000b	R/W	Pn.2 GPIO Interrupt Detection Mode
pin3	14:12	000b	R/W	Pn.3 GPIO Interrupt Detection Mode
pin4	18:16	000b	R/W	Pn.4 GPIO Interrupt Detection Mode
pin5	22:20	000b	R/W	Pn.5 GPIO Interrupt Detection Mode
pin6	26:24	000b	R/W	Pn.6 GPIO Interrupt Detection Mode
pin7	30:28	000b	R/W	Pn.7 GPIO Interrupt Detection Mode

Sets interrupt detection condition for this pin.

- 000b: Disabled
- 001b: Detect interrupt on falling edge
- 010b: Detect interrupt on rising edge
- 011b: Detect interrupt on rising or falling edge
- 100b: Active low state interrupt detection
- 101b: Active high state interrupt detection
- 11xb: Disabled

5.5.1.8 GPIO_INTFL_Pn

GPIO_INTFL_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Default	Access	Description
pin0	0	0	W1C	Pn.0 External Interrupt Flags
pin1	1	0	W1C	Pn.1 External Interrupt Flags
pin2	2	0	W1C	Pn.2 External Interrupt Flags
pin3	3	0	W1C	Pn.3 External Interrupt Flags
pin4	4	0	W1C	Pn.4 External Interrupt Flags
pin5	5	0	W1C	Pn.5 External Interrupt Flags
pin6	6	0	W1C	Pn.6 External Interrupt Flags
pin7	7	0	W1C	Pn.7 External Interrupt Flags

Write a 1 value to clear this bit; writes to 0 have no effect.

Set to 1 by hardware when an interrupt has been detected on this pin.

5.5.1.9 GPIO_INTEN_Pn

GPIO_INTEN_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Default	Access	Description
pin0	0	0	R/W	Pn.0 External Interrupt Enable
pin1	1	0	R/W	Pn.1 External Interrupt Enable
pin2	2	0	R/W	Pn.2 External Interrupt Enable
pin3	3	0	R/W	Pn.3 External Interrupt Enable
pin4	4	0	R/W	Pn.4 External Interrupt Enable
pin5	5	0	R/W	Pn.5 External Interrupt Enable
pin6	6	0	R/W	Pn.6 External Interrupt Enable
pin7	7	0	R/W	Pn.7 External Interrupt Enable

Enables interrupt to be triggered for this pin when the corresponding Interrupt Flag bit is set.

6 Peripheral Management Unit (PMU)

6.1 Overview

The Peripheral Management Unit (PMU) on the **MAX32600** is a DMA-based linked list processing engine. The PMU can perform operations and data transfers involving memory and/or peripherals in the Advanced Peripheral Bus (APB) and Advanced High-performance Bus (AHB) peripheral memory space while the main CPU is in a sleep state. This allows low-overhead peripheral operations — for which intensive CPU resources are not required — to be performed without the CPU, significantly reducing overall power consumption. Additionally, for certain analog and digital operations, switching the CPU off and handling the operations using the PMU provides a lower-noise environment that is critical for obtaining optimum analog-to-digital converter (ADC) and digital-to-analog converter (DAC) performance.

Key features of the PMU engine include:

- Six independent channels with round-robin scheduling to allow multiple parallel operations without requiring use of the CPU
- Suited for moving data to/from memory and peripherals
- Co-processor-like state machine within the **MAX32600**
- Integrated AHB bus master
- Programmed using SRAM-based PMU op codes
- Interrupt conditions from peripherals to initiate PMU actions without requiring actual CPU-based interrupt handling
- Trigger interrupts to the CPU if needed
- Operation with Cortex-M3 core in sleep state reduces power consumption

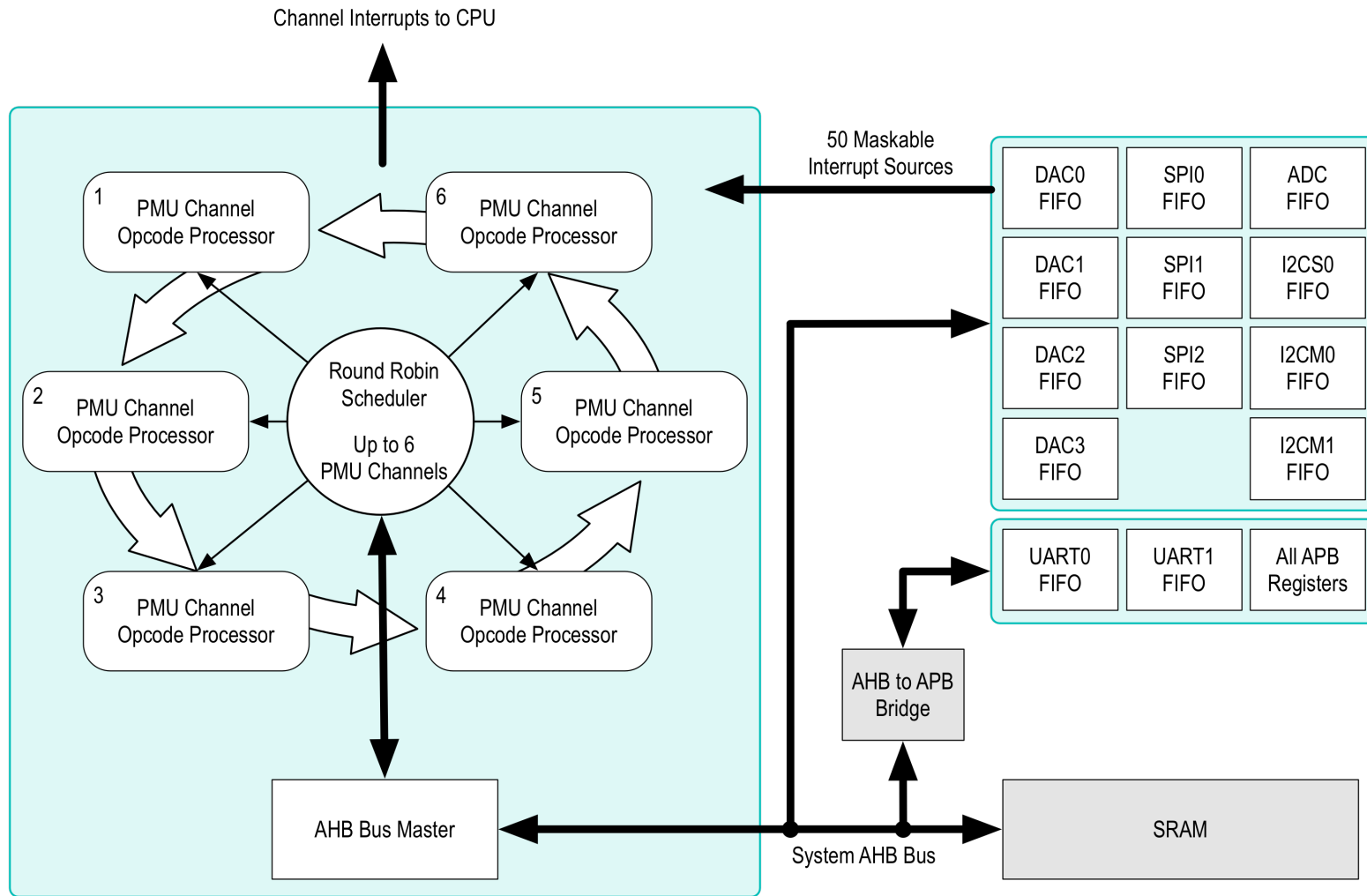


Figure 6.1: PMU Interface

6.2 PMU Operation

The PMU is a six-channel, programmable state machine that executes user-created op codes and operands that reside in the **MAX32600** memory space. Acting as an AHB master, the PMU can access any memory address in either the AHB or APB memory space. Each channel runs independently and shares accesses to the AHB/APB address space using an internal round-robin arbiter, on an op code-by-op code basis, with priority encoding. Priority ranges from channel one with the highest priority and channel six with the lowest priority. The PMU runs from a fixed-source clock and is clocked at the same frequency as the main **MAX32600** system clock (scaling configured by `CLKMAN_CLK_CTRL_0_SYSTEM`), which has a maximum frequency of 24MHz.

PMU op codes and operands are executed sequentially, with the user specifying the address of the first op code for each enabled channel. Op code execution for a given PMU channel is terminated when an op code has been executed with the STOP bit set or an error condition has been encountered. There is no limit to the number of op codes that can be executed.

Each PMU channel has two control registers that control the operation of the channel. The op code address register (`PMUn_DSCADR`) is used to set the 32-bit starting address of the first op code. The channel configuration register (`PMUn_CFG`) is used to start and stop the execution of op codes and operands for that channel. This register also reports status and error information.

6.2.1 PMU Channel Setup

To begin execution on a PMU channel, the user must specify an area of memory, which may be RAM or Flash memory, to hold the PMU op codes and operands. Once the PMU op codes and operands are written to memory, the user must set the starting address of the first op code using the `PMUn_DSCADR` register, followed by setting the START bit in the `PMUn_CFG` register. The PMU op code engine will then fetch the first op code and associated operands and begin processing. Upon completion of the first PMU op code, subsequent op codes are fetched sequentially from memory unless otherwise specified by a `JUMP`, `LOOP`, or `BRANCH` op code or an op code with the STOP bit set.

Completion of the entire series of PMU op codes will be signified by clearing of the START bit in the `PMUn_CFG` register, and if programmed to do so, an interrupt to the CPU may be asserted. Each PMU channel operates independently in this manner.

6.2.2 PMU Channel Arbitration

The PMU is an AHB bus master, and allows each of the channels access to the AHB/APB memory space. Arbitration between PMU channels for access to the bus master is performed by a round-robin scheduler, on an op code-by-op code basis, with priority encoding.

For example: If three PMU channels are active, execution would start with channel 1, op code 1 then proceed to channel 2, op code 1; channel 3, op code 1; channel 1, op code 2; etc. If a channel's op code is blocked due to flow control, execution of the next active channel's op code will subsequently occur. The `WAIT`, `POLL`, and `TRANSFER` op codes do not block the round-robin arbitration.

6.3 PMU Programming Details

There are eight op codes available for programming the PMU engine. These op codes are: **MOVE**, **WRITE**, **WAIT**, **JUMP**, **LOOP**, **BRANCH**, **POLL**, and **TRANSFER**. Each op code has one or more operands that are specific to the op code. All op codes and operands are 32-bits wide. A description of each op code is given below.

6.3.1 PMU Op Code: MOVE (0x00)

The MOVE op code is used to move a user specified block of data from the read address location to the write address location. Read and write addresses may be in either the AHB or APB memory space. APB accesses are restricted to 32-bit accesses. The op code bit settings and operands are shown below.

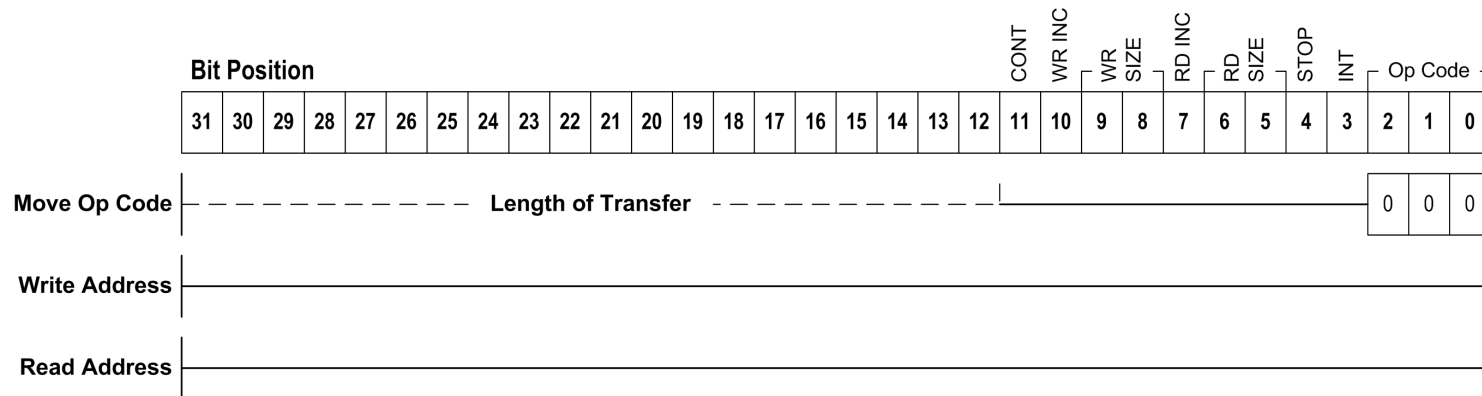


Figure 6.2: PMU MOVE Op Code Details

INT

- Set to 1 to generate an interrupt to the CPU upon completion of this op code.

STOP

- Set to 1 if this op code is to terminate op code processing after execution.
- This also clears the START bit field in the **PMUn_CFG** register.

RD SIZE	WR SIZE	Operation
00b	00b	Perform 8-bit reads and 8-bit writes
00b	01b	Perform 8-bit reads and pack data into 16-bit writes
00b	10b	Perform 8-bit reads and pack data into 32-bit writes
01b	00b	Perform 16-bit reads and unpack into 8-bit writes
01b	01b	Perform 16-bit reads and writes
01b	10b	Perform 16-bit reads and pack data into 32-bit writes
10b	00b	Perform 32-bit reads and unpack data into 8-bit writes
10b	01b	Perform 32-bit reads and unpack data into 16-bit writes
10b	10b	Perform 32-bit reads and writes
XX	11	(Reserved)
11b	XX	(Reserved)

RD INC

- Setting this bit to 0 disables auto incrementing of the read address. This may be useful when reading from FIFO storage elements.

WR INC

- Setting this bit to 1 enables auto incrementing of the write address. This may be useful when writing to SRAM memory.

CONT

- Setting this bit to 1 allows a subsequent MOVE op code to continue operation using the read and write addresses from the previous MOVE op code.

LENGTH

- Length of transfer (in bytes) from read address to write address.

6.3.2 PMU Op Code: WRITE (0x01)

The WRITE op code will cause the write value to be written to the write address location. Only bits set in the write mask field will be modified. This allows the user to set or clear individual bits in a 32-bit field. Write accesses may be in either the AHB or APB memory space. All accesses are restricted to 32-bit accesses. The op code bit settings and operands are shown below.

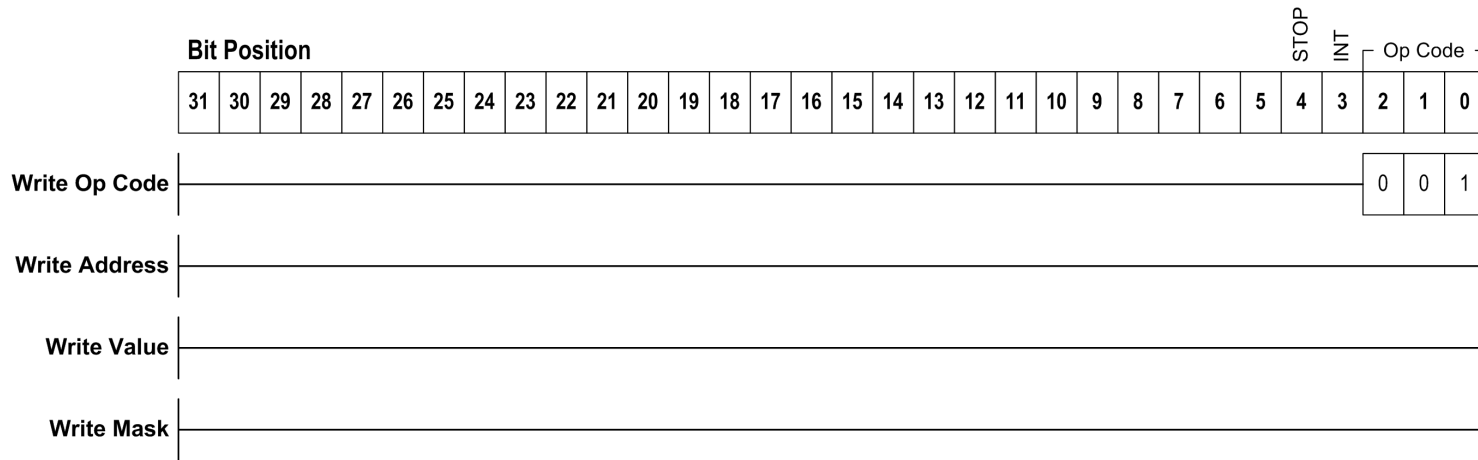


Figure 6.3: PMU WRITE Op Code Details

INT

- Set to 1 to generate an interrupt to the CPU upon completion of this op code.

STOP

- Set to 1 if this op code is to terminate op code processing after execution.
- This also clears the START bit field in the `PMUn_CFG` register.

6.3.3 PMU Op Code: WAIT (0x02)

The WAIT op code will suspend the execution of subsequent op codes until any of the corresponding bit(s) in the interrupt mask are set. An interrupt mask of all zeros will terminate execution of this op code. The op code bit settings and operands are shown below.

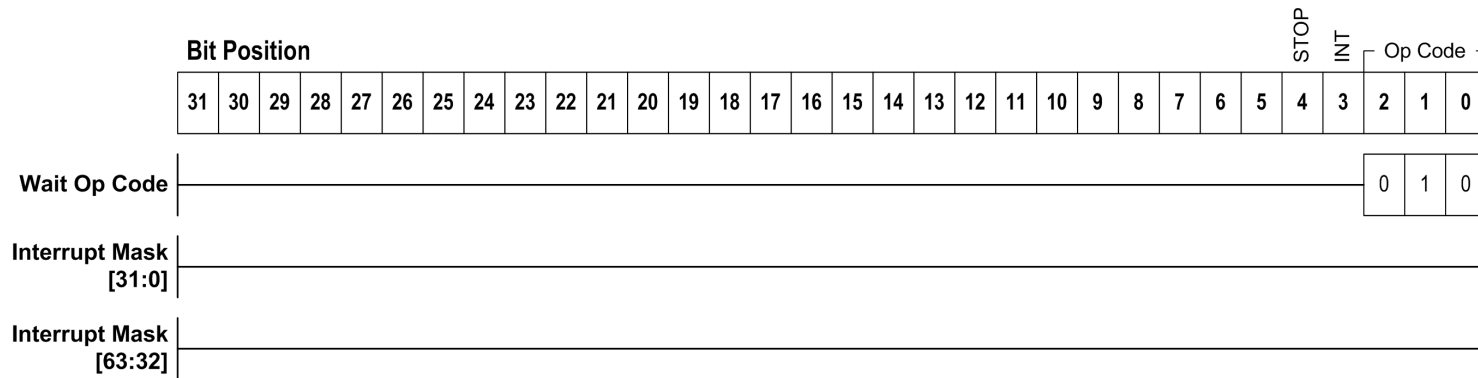


Figure 6.4: PMU WAIT Op Code Details

INT

- Set to 1 to generate an interrupt to the CPU upon completion of this op code.

STOP

- Set to 1 if this op code is to terminate op code processing after execution.
- This also clears the START bit field in the [PMUn_CFG](#) register.

The interrupt mask field corresponds to the following interrupt sources available to the PMU from the **MAX32600**.

The following interrupts are generated by the Peripheral FIFOs automatically as listed below based on FIFO configuration registers. These interrupts *do not* require the user to clear or enable the interrupt source, they are self clearing and enabled by the appropriate FIFO configuration registers as shown in the table below.

PMU FIFO Interrupt Sources - Peripheral FIFO Generated

Interrupt Bit	Source	Cause
0	DAC0 almost empty	Interrupt is set when FIFO level falls below user defined threshold in the DAC0_CTRL0.fifo_ae_cnt register field, interrupt self-clears when the FIFO level is above the user defined threshold in DAC0_CTRL0.fifo_ae_cnt .
1	DAC1 almost empty	Interrupt is set when FIFO level falls below user defined threshold in the DAC1_CTRL0.fifo_ae_cnt register field, interrupt self-clears when the FIFO level is above the user defined threshold in DAC1_CTRL0.fifo_ae_cnt .

Interrupt Bit	Source	Cause
2	DAC2 almost empty	Interrupt is set when FIFO level falls below user defined threshold in the DAC2_CTRL0.fifo_ae_cnt register field, interrupt self-clears when the FIFO level is above the user defined threshold in DAC2_CTRL0.fifo_ae_cnt .
3	DAC3 almost empty	Interrupt is set when FIFO level falls below user defined threshold in the DAC3_CTRL0.fifo_ae_cnt register field, interrupt self-clears when the FIFO level is above the user defined threshold in DAC3_CTRL0.fifo_ae_cnt .
8	ADC almost full	Interrupt is set when FIFO level is above user defined threshold in the ADC_TG_CTRL1.fifo_af_cnt register field, interrupt self-clears when the FIFO level falls below the user defined threshold in ADC_TG_CTRL1.fifo_af_cnt .
16	SPI0 trans_fifo_empty	Interrupt is set when FIFO level falls below the user defined threshold in the SPI0_FIFO_CTRL.tx_fifo_ae_lvl register field, the interrupt self-clears when the FIFO level is above the user defined threshold in SPI0_FIFO_CTRL.tx_fifo_ae_lvl .
17	SPI0 rsrts_fifo_almost_full	Interrupt is set when the FIFO level is above the user defined threshold in the SPI0_FIFO_CTRL.rx_fifo_af_lvl , the interrupt self-clears when the FIFO level falls below the user defined threshold in SPI0_FIFO_CTRL.rx_fifo_af_lvl .
18	SPI1 trans_fifo_empty	Interrupt is set when FIFO level falls below the user defined threshold in the SPI1_FIFO_CTRL.tx_fifo_ae_lvl register field, the interrupt self-clears when the FIFO level is above the user defined threshold in SPI1_FIFO_CTRL.tx_fifo_ae_lvl .
19	SPI1 rsrts_fifo_almost_full	Interrupt is set when the FIFO level is above the user defined threshold in the SPI1_FIFO_CTRL.rx_fifo_af_lvl , the interrupt self-clears when the FIFO level falls below the user defined threshold in SPI1_FIFO_CTRL.rx_fifo_af_lvl .
20	SPI2 trans_fifo_empty	Interrupt is set when FIFO level falls below the user defined threshold in the SPI2_FIFO_CTRL.tx_fifo_ae_lvl register field, the interrupt self-clears when the FIFO level is above the user defined threshold in SPI2_FIFO_CTRL.tx_fifo_ae_lvl .
21	SPI2 rsrts_fifo_almost_full	Interrupt is set when the FIFO level is above the user defined threshold in the SPI2_FIFO_CTRL.rx_fifo_af_lvl , the interrupt self-clears when the FIFO level falls below the user defined threshold in SPI2_FIFO_CTRL.rx_fifo_af_lvl .
22	I2CM0 tx_fifo_empty	Interrupt is set when no data is in FIFO. Interrupt is cleared when FIFO has at least one byte of data.
23	I2CM0 rx_fifo_not_empty	Interrupt is set when FIFO has at least one byte of data. Interrupt is cleared when no data is in FIFO.
24	I2CM1 tx_fifo_empty	Interrupt is set when no data is in FIFO. Interrupt is cleared when FIFO has at least one byte of data.
25	I2CM1 rx_fifo_not_empty	Interrupt is set when FIFO has at least one byte of data. Interrupt is cleared when no data is in FIFO.
26	I2CS0 tx_fifo_empty	Interrupt is set when no data is in FIFO. Interrupt is cleared when FIFO has at least one byte of data.
27	I2CS0 rx_fifo_not_empty	Interrupt is set when FIFO has at least one byte of data. Interrupt is cleared when no data is in FIFO.
28	UART0 tx_fifo_empty	Interrupt is set when no data is in FIFO. Interrupt is cleared when FIFO has at least one byte of data.
29	UART0 rx_fifo_almost_full	Interrupt is set when FIFO level is above the user defined threshold in the UART0_CTRL.rx_threshold , interrupt self-clears when FIFO level falls below the user defined threshold in UART0_CTRL.rx_threshold
30	UART1 tx_fifo_empty	Interrupt is set when no data is in FIFO. Interrupt is cleared when FIFO has at least one byte of data.
31	UART1 rx_fifo_almost_full	Interrupt is set when FIFO level is above the user defined threshold in the UART1_CTRL.rx_threshold , interrupt self-clears when FIFO level falls below the user defined threshold in UART1_CTRL.rx_threshold

Interrupt Bit	Source	Cause
---------------	--------	-------

The remaining interrupts, shown below, *must be enabled* through the appropriate peripheral register and also *cleared* after the interrupt becomes set.

PMU Interrupt Event Table

Interrupt Bit	Source	Enable and Clear
4	DAC0 done	Interrupt does not need to be enabled but must be cleared - DAC0_CTRL1_INT.out_done_if (W1C)
5	DAC1 done	Interrupt does not need to be enabled but must be cleared - DAC1_CTRL1_INT.out_done_if (W1C)
6	DAC2 done	Interrupt does not need to be enabled but must be cleared - DAC2_CTRL1_INT.out_done_if (W1C)
7	DAC3 done	Interrupt does not need to be enabled but must be cleared - DAC3_CTRL1_INT.out_done_if (W1C)
9	ADC finished	Interrupt does not need to be enabled but must be cleared - ADC_INTR.done to clear (W1C)
10	I2CM0 finished	I2CM0_INTEN.tx_done to enable, I2CM0_INTFL.tx_done to clear (W1C)
11	I2CM1 finished	I2CM1_INTEN.tx_done to enable, I2CM1_INTFL.tx_done to clear (W1C)
12	SPI0 Receive Done	SPI0_INTEN.rx_done to enable, SPI0_INTFL.rx_done to clear (W1C)
13	SPI1 Receive Done	SPI1_INTEN.rx_done to enable, SPI1_INTFL.rx_done to clear (W1C)
14	SPI2 Receive Done	SPI2_INTEN.rx_done to enable, SPI2_INTFL.rx_done to clear (W1C)
15	MAA Done	MAA_CTRL.inten to enable, MAA_CTRL.if_done to clear (W0C)
32	SPI0 rx_stalled OR tx_ready OR tx_stalled	SPI0_INTEN.rx_stalled , SPI0_INTEN.tx_stalled or SPI0_INTEN.tx_ready to enable; SPI0_INTFL.rx_stalled , SPI0_INTFL.tx_stalled or SPI0_INTFL.tx_ready to clear (W1C)
33	SPI1 rx_stalled OR tx_ready OR tx_stalled	SPI1_INTEN.rx_stalled , SPI1_INTEN.tx_stalled or SPI1_INTEN.tx_ready to enable; SPI1_INTFL.rx_stalled , SPI1_INTFL.tx_stalled or SPI1_INTFL.tx_ready to clear (W1C)
34	SPI2 rx_stalled OR tx_ready OR tx_stalled	SPI2_INTEN.rx_stalled , SPI2_INTEN.tx_stalled or SPI2_INTEN.tx_ready to enable; SPI2_INTFL.rx_stalled , SPI2_INTFL.tx_stalled or SPI2_INTFL.tx_ready to clear (W1C)
36	I2CM0 tx_timeout OR tx_lost_arbitr OR tx_nacked	I2CM0_INTEN.tx_timeout , I2CM0_INTEN.tx_lost_arbitr or I2CM0_INTEN.tx_nacked to enable, I2CM0_INTFL.tx_timeout , I2CM0_INTFL.tx_lost_arbitr or I2CM0_INTFL.tx_nacked to clear (W1C)
37	I2CM1 tx_timeout OR tx_lost_arbitr OR tx_nacked	I2CM1_INTEN.tx_timeout , I2CM1_INTEN.tx_lost_arbitr or I2CM1_INTEN.tx_nacked to enable, I2CM1_INTFL.tx_timeout , I2CM1_INTFL.tx_lost_arbitr or I2CM1_INTFL.tx_nacked to clear (W1C)
38	I2CS0 clk_stretch_to, rx_clk_stretch, tx_clk_stretch_id0, tx_clk_stretch_id1, restart_id0 OR restart_id1	See I2CS0_INTEN to enable these specific interrupts, and I2CS0_INTFL to clear (W1C)
40	Interrupts on Port 0 GPIO	GPIO_INT_MODE_P0.pin[7:0] to enable, GPIO_INTFL_P0.pin[7:0] to clear (W1C)

Interrupt Bit	Source	Enable and Clear
41	Interrupts on Port 1 GPIO	GPIO_INT_MODE_P1.pin[7:0] to enable, GPIO_INTFL_P1.pin[7:0] to clear (W1C)
42	Interrupts on Port 2 GPIO	GPIO_INT_MODE_P2.pin[7:0] to enable, GPIO_INTFL_P2.pin[7:0] to clear (W1C)
43	Interrupts on Port 3 GPIO	GPIO_INT_MODE_P3.pin[7:0] to enable, GPIO_INTFL_P3.pin[7:0] to clear (W1C)
44	Interrupts on Port 4 GPIO	GPIO_INT_MODE_P4.pin[7:0] to enable, GPIO_INTFL_P4.pin[7:0] to clear (W1C)
45	Interrupts on Port 5 GPIO	GPIO_INT_MODE_P5.pin[7:0] to enable, GPIO_INTFL_P5.pin[7:0] to clear (W1C)
46	Interrupts on Port 6 GPIO	GPIO_INT_MODE_P6.pin[7:0] to enable, GPIO_INTFL_P6.pin[7:0] to clear (W1C)
47	Interrupts on Port 7 GPIO	GPIO_INT_MODE_P7.pin[7:0] to enable, GPIO_INTFL_P7.pin[7:0] to clear (W1C)
48	Analog Front End	AFE_INTR bits 15:8 (Low Power Comparator A/B/C/D, OpAmp/Comp A/B/C/D) to enable, AFE_INTR bits 7:0 (Low Power Comparator A/B/C/D, OpAmp/Comp A/B/C/D) to clear (W1C)
49	AES	AES_CTRL.inten to enable, AES_CTRL.intfl to clear (W1C)

6.3.4 PMU Op Code: JUMP (0x03)

The JUMP op code will allow the PMU engine to fetch the next op code at the specified location in the operand field rather than sequentially in memory.

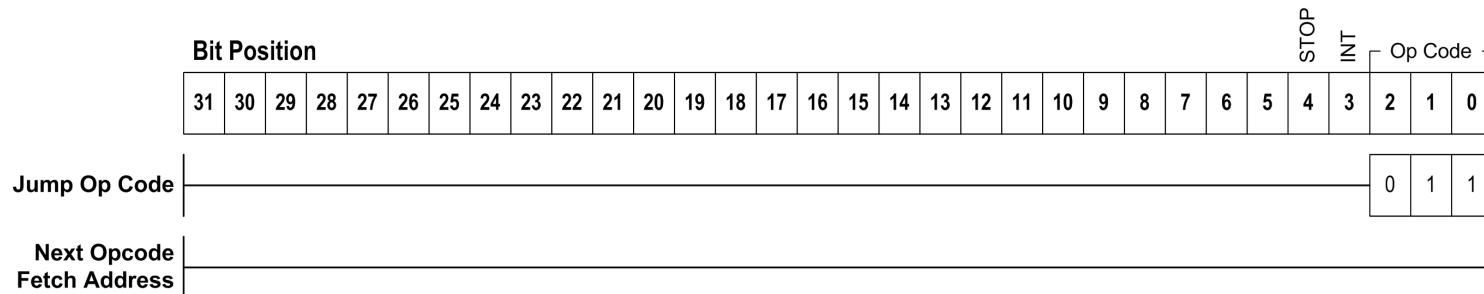


Figure 6.5: PMU JUMP Op Code Details

INT

- Set to 1 to generate an interrupt to the CPU upon completion of this op code.

STOP

- Set to 1 if this op code is to terminate op code processing after execution.
- This also clears the START bit field in the `PMUn_CFG` register.

6.3.5 PMU Op Code: LOOP (0x04)

The LOOP op code will cause the PMU engine to fetch the next op code at the specified location rather than sequentially in memory until the specified loop counter (counter0 or counter1, in `PMUn_LOOP`) decrements to zero. The specified loop counter will decrement by one each time this op code is executed by the PMU engine. When the specified loop counter reaches zero, the next sequential op code will be fetched. The zero check is performed after the specified counter is decremented.

The INT and STOP fields in this op code are not checked until the specified counter reaches zero.

The loop counters are sticky counters and do not need to be reloaded when a new LOOP op code is fetched unless a new counter value is needed.

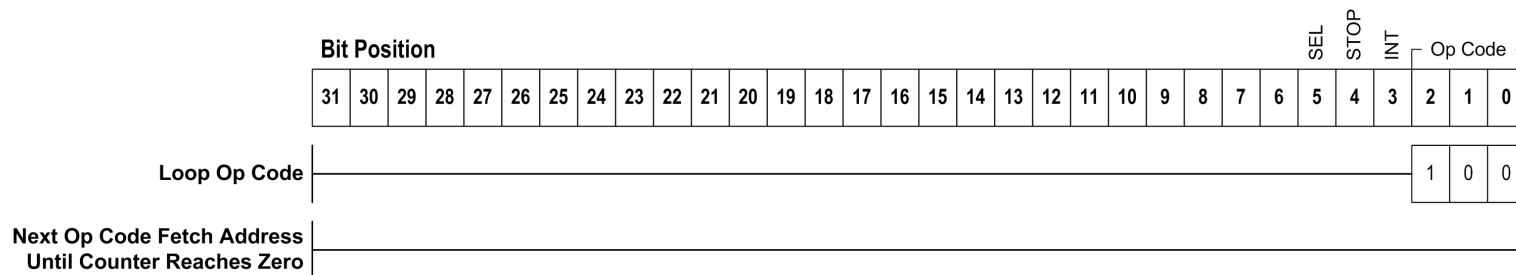


Figure 6.6: PMU LOOP Op Code Details

INT

- Set to 1 to generate an interrupt to the CPU upon completion of this op code.

STOP

- Set to 1 if this op code is to terminate op code processing after execution.
- This also clears the START bit field in the `PMUn_CFG` register.

6.3.6 PMU Op Code: POLL (0x05)

The POLL op code will cause the PMU engine to pause, wait for the specified polling interval to occur, and then read the specified address location. Read accesses may be in either the AHB or APB memory space. APB accesses are restricted to 32-bit accesses. When the bit(s) set in the data mask match those in the expected data field, the execution of this op code will terminate. The polling interval is specified in system clocks.

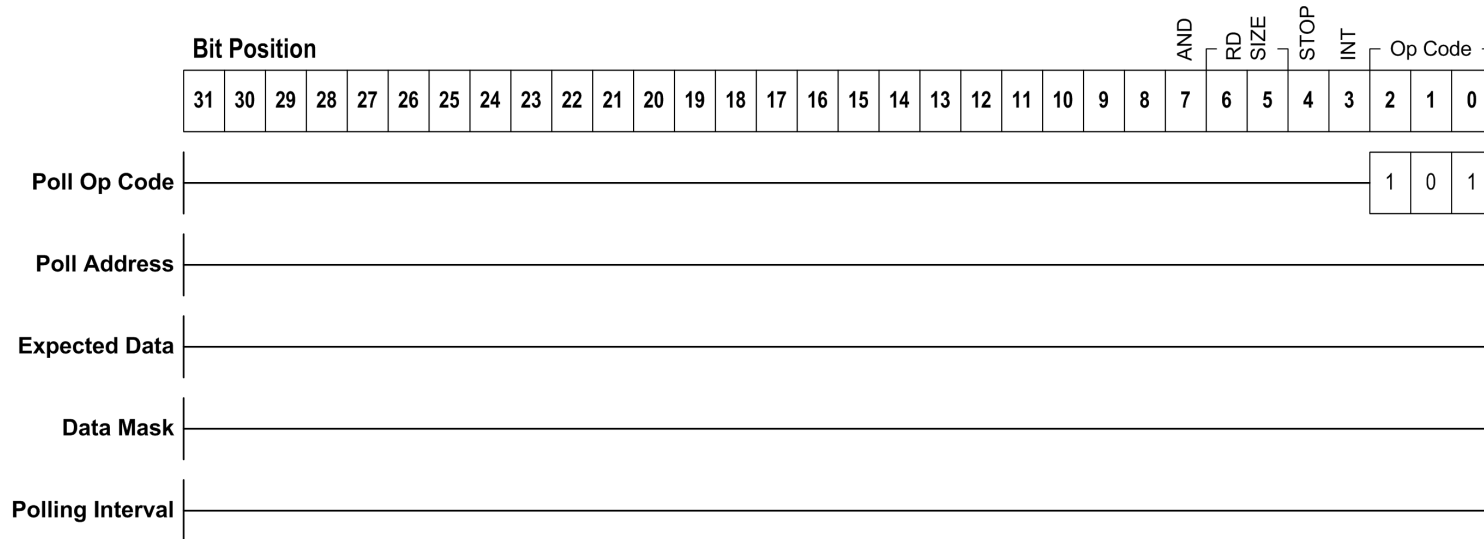


Figure 6.7: PMU POLL Op Code Details

INT

- Set to 1 to generate an interrupt to the CPU upon completion of this op code.

STOP

- Set to 1 if this op code is to terminate op code processing after execution.
- This also clears the START bit field in the [PMUn_CFG](#) register.

RD SIZE

- This field determines the size of the transfer: 00 = 8-bit, 01 = 16-bit and 10 = 32-bit.

AND

- This bit, when set to 1, will require that all bits set in the data mask be set in the data read from the poll address. Otherwise, polling will complete when any of the bits in the data mask are set in the data read from the poll address.

6.3.7 PMU Op Code: BRANCH (0x06)

The BRANCH op code will cause the PMU engine to read the specified address location and fetch the next op code from the specified location when the bit(s) set in the data mask do not match those in the expected data. If a match occurs, the next op code is fetched sequentially. Read accesses may be in either the AHB or APB memory space. APB accesses are restricted to 32-bit accesses.

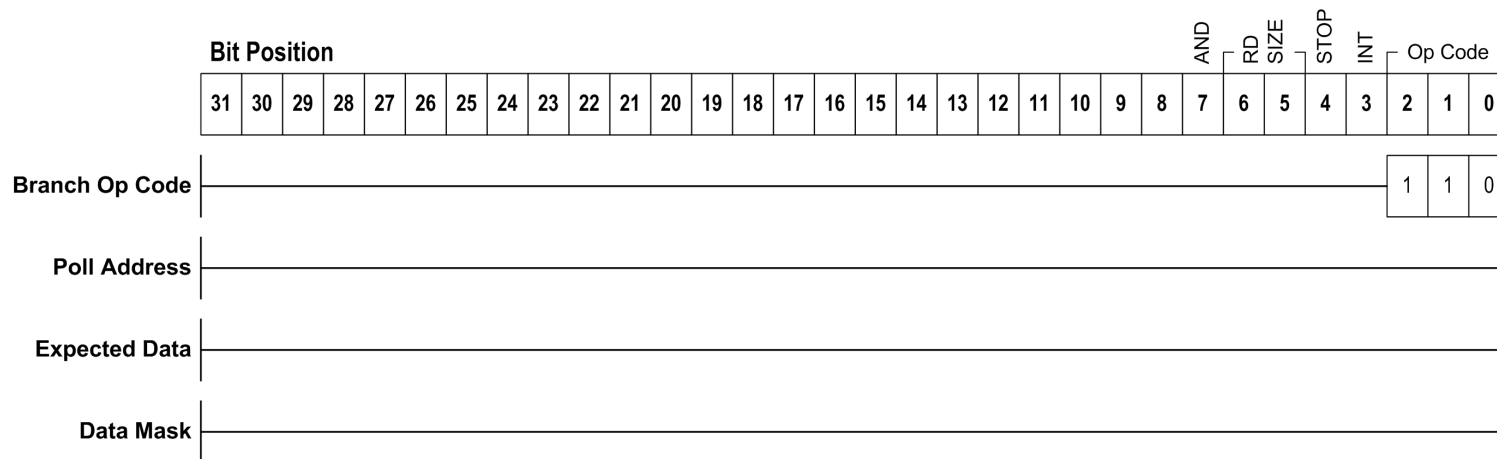


Figure 6.8: PMU BRANCH Op Code Details

INT

- Set to 1 to generate an interrupt to the CPU upon completion of this op code.

STOP

- Set to 1 if this op code is to terminate op code processing after execution.
- This also clears the START bit field in the [PMUn_CFG](#) register.

RD SIZE

- This field determines the size of the transfer: 00 = 8-bit, 01 = 16-bit and 10 = 32-bit.

AND

- This bit, when set to 1, will require that all bits set in the data mask match the expected value in the data read from the poll address in order to avoid a branch. Otherwise, a branch will be avoided when any of the bits set in the data mask match the expected value.

6.3.8 PMU Op Code: TRANSFER (0x07)

The TRANSFER op code is used to move a user specified total block of data (in bytes) from the read address location to the write address location in burst size blocks (in bytes) as specified by the user. Transfers are only performed when the specified bit(s) in the interrupt mask [31:0] are set.

The TRANSFER op code will continue to execute until the total number of bytes has been transferred, as specified in bit fields [31:12]. Interrupt mask bits shown in the [PMU FIFO Interrupt Table](#) are associated with peripheral FIFOs and are self-clearing. Interrupt mask bits shown in the [PMU Interrupt Event Table](#) are exception interrupts and will cause the termination of this op code. This op code combines the functionality of the WAIT, MOVE, and JUMP op codes into a single op code and is particularly useful for servicing the FIFOs on various peripherals. Read and write accesses may be in either the AHB or APB memory space. APB accesses are restricted to 32-bit accesses. The op code bit settings and operands are shown below.

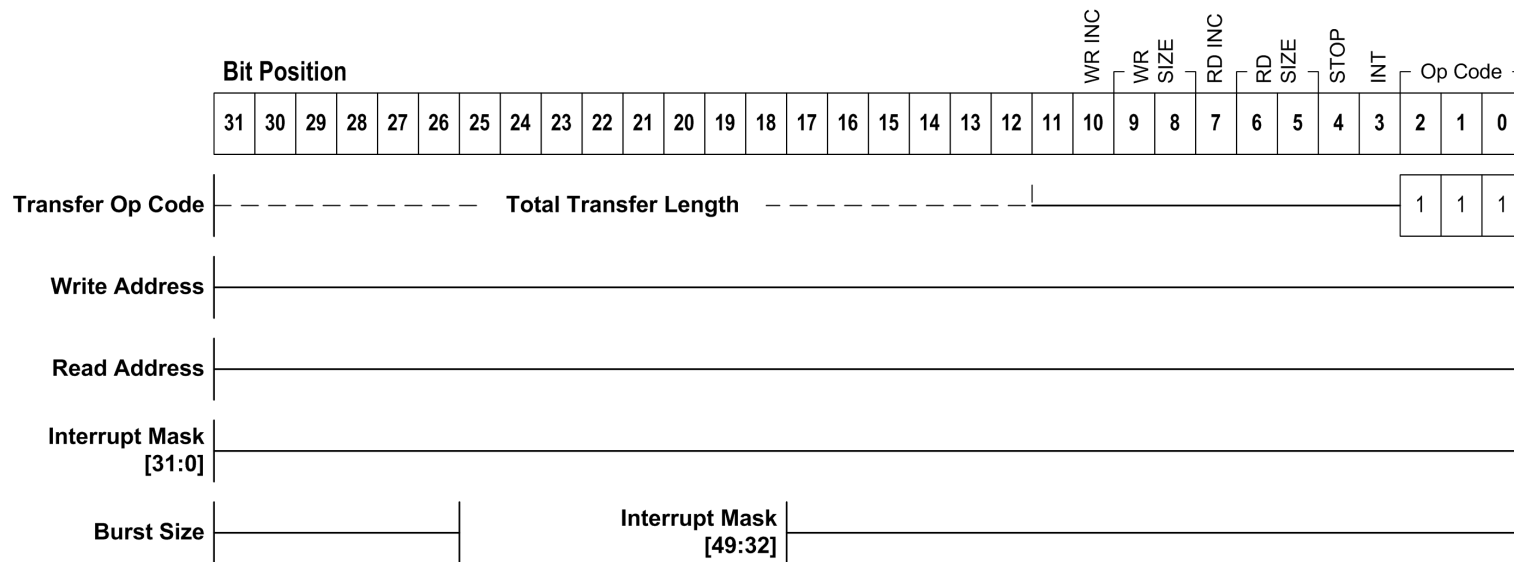


Figure 6.9: PMU TRANSFER Op Code Details

INT

- Set to 1 to generate an interrupt to the CPU upon completion of this op code.

STOP

- Set to 1 if this op code is to terminate op code processing after execution.
- This also clears the START bit field in the [PMUn_CFG](#) register.

RD SIZE	WR SIZE	Operation
00b	00b	Perform 8-bit reads and 8-bit writes
00b	01b	Perform 8-bit reads and pack data into 16-bit writes

RD SIZE	WR SIZE	Operation
00b	10b	Perform 8-bit reads and pack data into 32-bit writes
01b	00b	Perform 16-bit reads and unpack into 8-bit writes
01b	01b	Perform 16-bit reads and writes
01b	10b	Perform 16-bit reads and pack data into 32-bit writes
10b	00b	Perform 32-bit reads and unpack data into 8-bit writes
10b	01b	Perform 32-bit reads and unpack data into 16-bit writes
10b	10b	Perform 32-bit reads and writes
XX	11	(Reserved)
11b	XX	(Reserved)

RD INC

- Setting this bit to 0 disables auto-incrementing of the read address. This may be useful when reading from FIFO storage elements.

WR INC

- Setting this bit to 1 enables auto-incrementing of the write address. This may be useful when writing to SRAM memory.

LENGTH

- Length of total transfer (in bytes) from read address to write address.

BURST

- Maximum burst length of transfer (in bytes) from read address to write address when specified interrupt source is detected.

6.4 Registers (PMU)

6.4.1 Module PMU Registers

Address	Register	32b Word Len	Description
0x40070000	PMU0_DSCADR	1	PMU Channel 0 Next Descriptor Address
0x40070004	PMU0_CFG	1	PMU Channel 0 Configuration
0x40070008	PMU0_LOOP	1	PMU Channel 0 Loop Counters
0x4007000C	PMU0_OP	1	PMU Channel 0 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]
0x40070010	PMU0_DSC1	1	PMU Channel 0 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]
0x40070014	PMU0_DSC2	1	PMU Channel 0 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]
0x40070018	PMU0_DSC3	1	PMU Channel 0 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]
0x4007001C	PMU0_DSC4	1	PMU Channel 0 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]
0x40070020	PMU1_DSCADR	1	PMU Channel 1 Next Descriptor Address
0x40070024	PMU1_CFG	1	PMU Channel 1 Configuration
0x40070028	PMU1_LOOP	1	PMU Channel 1 Loop Counters
0x4007002C	PMU1_OP	1	PMU Channel 1 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]
0x40070030	PMU1_DSC1	1	PMU Channel 1 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]
0x40070034	PMU1_DSC2	1	PMU Channel 1 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]
0x40070038	PMU1_DSC3	1	PMU Channel 1 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]
0x4007003C	PMU1_DSC4	1	PMU Channel 1 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]
0x40070040	PMU2_DSCADR	1	PMU Channel 2 Next Descriptor Address
0x40070044	PMU2_CFG	1	PMU Channel 2 Configuration
0x40070048	PMU2_LOOP	1	PMU Channel 2 Loop Counters
0x4007004C	PMU2_OP	1	PMU Channel 2 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]
0x40070050	PMU2_DSC1	1	PMU Channel 2 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]
0x40070054	PMU2_DSC2	1	PMU Channel 2 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]
0x40070058	PMU2_DSC3	1	PMU Channel 2 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]
0x4007005C	PMU2_DSC4	1	PMU Channel 2 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]
0x40070060	PMU3_DSCADR	1	PMU Channel 3 Next Descriptor Address
0x40070064	PMU3_CFG	1	PMU Channel 3 Channel Configuration
0x40070068	PMU3_LOOP	1	PMU Channel 3 Loop Counters
0x4007006C	PMU3_OP	1	PMU Channel 3 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]
0x40070070	PMU3_DSC1	1	PMU Channel 3 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]

Address	Register	32b Word Len	Description
0x40070074	PMU3_DSC2	1	PMU Channel 3 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]
0x40070078	PMU3_DSC3	1	PMU Channel 3 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]
0x4007007C	PMU3_DSC4	1	PMU Channel 3 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]
0x40070080	PMU4_DSCADR	1	PMU Channel 4 Next Descriptor Address
0x40070084	PMU4_CFG	1	PMU Channel 4 Configuration
0x40070088	PMU4_LOOP	1	PMU Channel 4 Loop Counters
0x4007008C	PMU4_OP	1	PMU Channel 4 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]
0x40070090	PMU4_DSC1	1	PMU Channel 4 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]
0x40070094	PMU4_DSC2	1	PMU Channel 4 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]
0x40070098	PMU4_DSC3	1	PMU Channel 4 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]
0x4007009C	PMU4_DSC4	1	PMU Channel 4 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]
0x400700A0	PMU5_DSCADR	1	PMU Channel 5 Next Descriptor Address
0x400700A4	PMU5_CFG	1	PMU Channel 5 Configuration
0x400700A8	PMU5_LOOP	1	PMU Channel 5 Loop Counters
0x400700AC	PMU5_OP	1	PMU Channel 5 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]
0x400700B0	PMU5_DSC1	1	PMU Channel 5 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]
0x400700B4	PMU5_DSC2	1	PMU Channel 5 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]
0x400700B8	PMU5_DSC3	1	PMU Channel 5 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]
0x400700BC	PMU5_DSC4	1	PMU Channel 5 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]

6.4.1.1 PMUn_DSCADR

Default	Access	Description
00000000h	R/W	PMU Channel 0 Next Descriptor Address

This register contains a byte address which points to the first 32-bit word of the next PMU descriptor which will be fetched by this PMU channel. Before the PMU channel is started, this register must be set to the address of the first PMU descriptor to be fetched.

As each PMU descriptor is fetched and executed, this register is automatically updated to point to the first 32-bit word of the next PMU descriptor to be executed by the PMU channel controller. In order for a PMU descriptor sequence to be rerun (after the PMU channel has stopped), this register must be reloaded with the

address of the first PMU descriptor in the sequence.

6.4.1.2 PMUn_CFG

PMUn_CFG.enable

Field	Bits	Default	Access	Description
enable	0	0	R/W	PMU Channel Enable

- 0: Disabled/stopped (this channel only)
- 1: Enabled/running (this channel only)

To begin a PMU descriptor sequence, firmware must set this bit to 1. It is possible to halt the PMU channel before the end of the descriptor sequence has been reached by manually clearing this bit to 0.

This bit will be automatically cleared to 0 by hardware when the PMU channel stops running. This will occur whenever the PMU channel controller executes a descriptor with the STOP bit set (which will also cause `ll_stopped` to be set to 1), or whenever either the Bus Error Flag or the Bus Timeout Flag is set to 1.

PMUn_CFG.ll_stopped

Field	Bits	Default	Access	Description
ll_stopped	2	0	W1C	PMU Channel Stopped Flag

This bit is set to 1 by hardware when the PMU channel completes execution of a descriptor which has a STOP bit set to 1. This bit is not set if the PMU channel halts due to a bus error or timeout, or if the PMU channel is halted manually by clearing the enable bit.

In order to clear this bit, firmware must write this bit to 1. This bit will not clear automatically when the PMU channel is started. Firmware should always clear this bit before starting PMU channel execution, so that the (STOP == 1) channel halt condition can be properly detected.

PMUn_CFG.manual

Field	Bits	Default	Access	Description
manual	3	0	R/W	Manual Mode Enable [INTERNAL USE ONLY]

This bit is intended for Maxim internal test use only.

This bit should not be used by application firmware. This bit must remain set to the default value (0) for proper PMU operation.

PMUn_CFG.bus_error

Field	Bits	Default	Access	Description
bus_error	4	0	W1C	AHB Bus Error Flag

This bit is set to 1 by hardware when an AHB bus error occurs during channel descriptor execution. When this occurs, the PMU channel will also be halted, and the enable bit will be cleared to zero by hardware automatically. Other active PMU channels will continue execution normally, however.

In order to clear this bit, firmware must write this bit to 1. This bit will not clear automatically when the PMU channel is started. Firmware should always clear this bit before starting PMU channel execution, so that the AHB bus error channel halt condition can be properly detected.

PMUn_CFG.to_stat

Field	Bits	Default	Access	Description
to_stat	6	0	W1C	AHB Bus Timeout Flag

This bit is set to 1 by hardware when a bus timeout occurs during channel descriptor execution. When this occurs, the PMU channel will also be halted, and the enable bit will be cleared to zero by hardware automatically. Other active PMU channels will continue execution normally, however.

Conditions for the bus timeout are determined by the time out interval select and prescale select fields in this register.

In order to clear this bit, firmware must write this bit to 1. This bit will not clear automatically when the PMU channel is started. Firmware should always clear this bit before starting PMU channel execution, so that the bus timeout channel halt condition can be properly detected.

PMUn_CFG.to_sel

Field	Bits	Default	Access	Description
to_sel	13:11	000b	R/W	Time Out Interval Select

This field selects the bus timeout period that is used for this PMU channel. If a single AHB bus operation exceeds this period, a bus timeout condition will occur, causing the PMU channel to halt and causing the bus timeout flag to be set to 1.

Note that the setting of this field has no effect if the `ps_sel` field is set to 0 (which disables the AHB bus timeout counter).

The timeout interval selected by this field is derived from the base period selected by the `ps_sel` field (when `ps_sel` != 0).

- 0: 4 x (`ps_sel` period)
- 1: 8 x (`ps_sel` period)
- 2: 16 x (`ps_sel` period)
- 3: 32 x (`ps_sel` period)
- 4: 64 x (`ps_sel` period)
- 5: 128 x (`ps_sel` period)
- 6: 256 x (`ps_sel` period)
- 7: 512 x (`ps_sel` period)

PMUn_CFG.ps_sel

Field	Bits	Default	Access	Description
<code>ps_sel</code>	15:14	00b	R/W	Time Out Interval Prescale Select

This field selects the base time period used for the AHB bus timeout interval, based on the PMU module clock. The value of this field is combined with the `to_sel` field to determine the total AHB bus timeout interval.

- 0: Disabled (default)
- 1: (PMU module clock) divided by 256
- 2: (PMU module clock) divided by 65,536 (2^{16})
- 3: (PMU module clock) divided by 16,777,216 (2^{24})

PMUn_CFG.interrupt

Field	Bits	Default	Access	Description
interrupt	16	0	W1C	Descriptor Interrupt Flag

This interrupt flag is set to 1 by hardware when the PMU channel completes execution of a descriptor which has (INT == 1).

When this interrupt flag is set to 1, an interrupt will be triggered by the PMU module if int_en is also set to 1.

This interrupt flag must be cleared by firmware when the PMU interrupt is serviced. To clear this flag, write the flag to 1.

PMUn_CFG.int_en

Field	Bits	Default	Access	Description
int_en	17	0	R/W	PMU Channel Interrupt Enable

CPU interrupt enable/disable for this PMU channel.

- 0: No interrupts will be generated for this channel.
- 1: The PMU will generate an interrupt when the Descriptor Interrupt flag is set. AHB bus error or bus timeout conditions will cause a halt to PMU operation.

PMUn_CFG.burst_size

Field	Bits	Default	Access	Description
burst_size	28:24	0	R/W	AHB Maximum Burst Size

The maximum size of the PMU transfer bursts (in bytes) is set to (field value + 1).

6.4.1.3 PMUn_LOOP

PMUn_LOOP.counter_0

Field	Bits	Default	Access	Description
counter_0	15:0	0000h	R/W	PMU Channel Loop Counter 0

This field contains the initial value that will be loaded into the PMU channel internal loop counter 0 when the PMU executes a LOOP descriptor and the internal loop counter 0 is at zero.

Note that the internal loop counter (which is the counter that is decremented each time the LOOP descriptor is executed) is not accessed through this field. This field contains a 'sticky' loop counter and its contents will not change as the internal loop counter 0 is decremented.

This means that it is possible to have more than one LOOP descriptor in a PMU descriptor sequence that uses loop counter 0.

PMUn_LOOP.counter_1

Field	Bits	Default	Access	Description
counter_1	31:16	0000h	R/W	PMU Channel Loop Counter 1

This field contains the initial value that will be loaded into the PMU channel internal loop counter 1 when the PMU executes a LOOP descriptor and the internal loop counter 1 is at zero.

Note that the internal loop counter (which is the counter that is decremented each time the LOOP descriptor is executed) is not accessed through this field. This field contains a 'sticky' loop counter and its contents will not change as the internal loop counter 1 is decremented.

This means that it is possible to have more than one LOOP descriptor in a PMU descriptor sequence that uses loop counter 1.

6.4.1.4 PMUn_OP

Default	Access	Description
00000000h	R/W	PMU Channel 0 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]

This register is intended for Maxim internal test use only.

Application software should not use this register. This register must remain set to its default value for proper PMU operation.

6.4.1.5 PMUn_DSC1

Default	Access	Description
00000000h	R/W	PMU Channel 0 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]

This register is intended for Maxim internal test use only.

Application software should not use this register. This register must remain set to its default value for proper PMU operation.

6.4.1.6 PMUn_DSC2

Default	Access	Description
00000000h	R/W	PMU Channel 0 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]

This register is intended for Maxim internal test use only.

Application software should not use this register. This register must remain set to its default value for proper PMU operation.

6.4.1.7 PMUn_DSC3

Default	Access	Description
00000000h	R/W	PMU Channel 0 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]

This register is intended for Maxim internal test use only.

Application software should not use this register. This register must remain set to its default value for proper PMU operation.

6.4.1.8 PMUn_DSC4

Default	Access	Description
00000000h	R/W	PMU Channel 0 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]

This register is intended for Maxim internal test use only.

Application software should not use this register. This register must remain set to its default value for proper PMU operation.

7 Communication Peripherals

7.1 I²C

7.1.1 I²C Overview

The **MAX32600** integrates two I²C bus masters for communication with a wide variety of I²C enabled slaves. The I²C bus is a two-wire, bidirectional bus (i.e., can operate as a master-transmitter or master-receiver) using a ground line and two bus lines: the serial data access line (SDA) and the serial clock line (SCL). Both the SDA and SCL lines must be driven as open-collector/drain outputs. External resistors (R_P) are recommended to pull the lines to a logic-high state; internal pullups can also be used to start I²C buses with low capacitance.

The device supports both the master and slave protocols. The master I²C peripherals have ownership of the I²C bus, drive the clock via the SCL pin, and generate the START and STOP signals. This enables the **MAX32600** to send and receive data from a slave as required by the user's application. In slave mode, the device relies on an externally generated clock to drive SCL and responds to data and commands only when requested by the I²C master device.

7.1.2 I²C Features

The I²C host port is compliant with the I²C Bus Specification with features:

- I²C bus specification version 2.1 compliant (100kHz and 400kHz)
- Programmable for both normal (100 kHz) and fast bus data rates (400kHz)
- Tagged-byte (16-byte depth) FIFOs:
 - Transaction FIFO
 - Results FIFO
- Support for 10-bit device addressing
- Clock synchronization and bus arbitration
- Supports arbitration in a multi-master environment
- Supports I²C bus hold for slow host service
- Transfer status interrupts and flags
- Support DMA data transfer via the [Peripheral Management Unit \(PMU\)](#)

The **MAX32600** I²C bus supports the following FIFO slave features:

- Tagged-byte Rx FIFO with parameterized depth
- Tx Data FIFO with parameterized depth
- Support for 10-bit device addressing
- Clock stretching support
- I²C timing parameters fully controllable via firmware (Resolution and range defined by system clock)
 - Full-speed filter rates
 - Spreadsheet available to compute programming values
- Supports two device addresses for SMBus support
- Appropriate control, status, and interrupt events are available for maximum flexibility

Speed Categories

The I²C master ports support two operating speed categories:

- Standard-mode with a bit rate up to 100Kbps
- Fast-mode with a bit rate up to 400Kbps

Note All interfaces are downward compatible and will operate at a lower bus speed as necessary.

7.1.3 I²C Port and Pin Configurations

Note See [Pin Layout](#) for a detailed mapping of **MAX32600** multiplexed function locations. Functional priority distinction is included in the mapping.

7.1.3.1 Compact Layout (7mm x 7mm) Configuration

Available SDA and SCL port and pin configurations for each of the I²C Master ports (I2CM0 and I2CM1) and the I²C Slave port (I2CS):

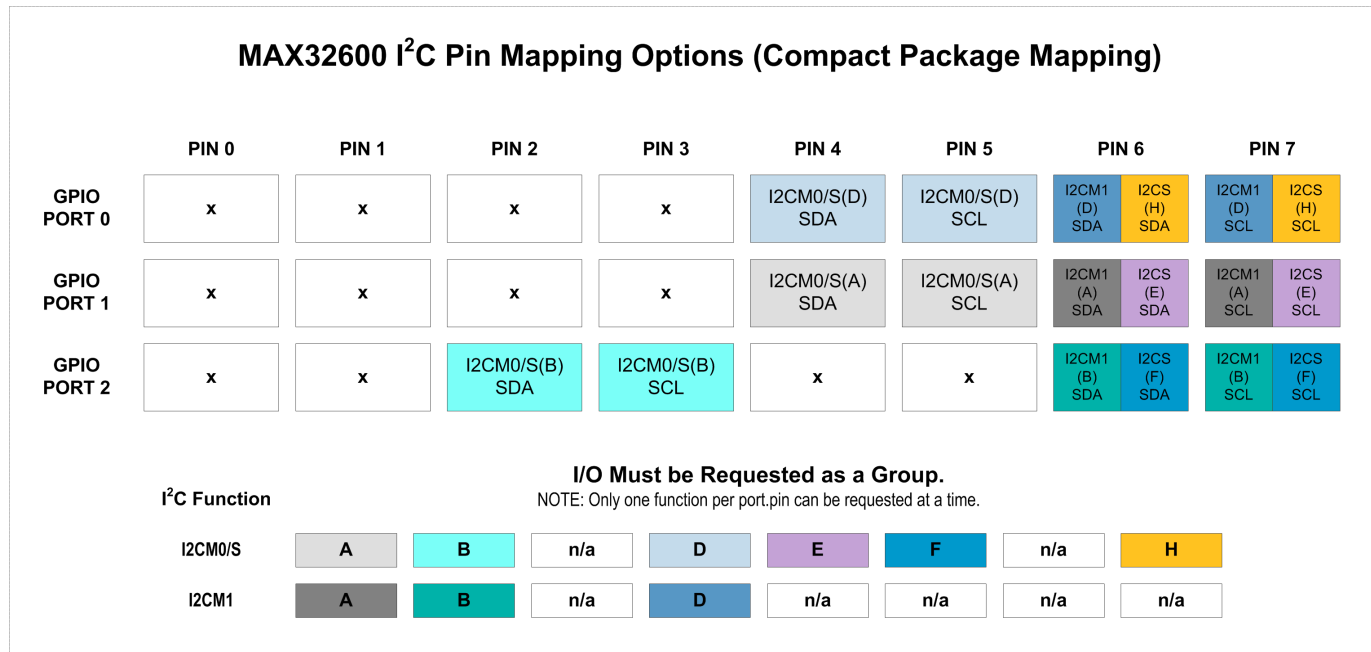


Figure 7.1: Compact Package Mapping Options

I²C Master Configurations (Compact)

I2CM0

Logic Signal	Port and Pin
SDA	A) P1.4 B) P2.2 D) P0.4
SCL	A) P1.5 B) P2.3 D) P0.5

I2CM1

Logic Signal	Port and Pin
SDA	A) P1.6 B) P2.6 D) P0.6
SCL	A) P1.7 B) P2.7 D) P0.7

I²C Slave Configurations (Compact)**I2CS**

Logic Signal	Port and Pin
SDA	A) P1.4 B) P2.2 D) P0.4 E) P1.6 F) P2.6 H) P0.6
SCL	A) P1.5 B) P2.3 D) P0.5 E) P1.7 F) P2.7 H) P0.7

7.1.3.2 Standard Layout (12mm x 12mm) Configuration

Available SDA and SCL port and pin configurations for each of the I²C Master ports (I2CM0 and I2CM1) and the I²C Slave port (I2CS):

MAX32600 I²C Pin Mapping Options (Standard Package Mapping)

	PIN 0	PIN 1	PIN 2	PIN 3	PIN 4	PIN 5	PIN 6		PIN 7	
GPIO PORT 0	x	x	x	x	I2CM0/S(D) SDA	I2CM0/S(D) SCL	I2CM1 (D) SDA	I2CS (H) SDA	I2CM1 (D) SCL	I2CS (H) SCL
GPIO PORT 1	x	x	x	x	x	x	I2CM1 (B) SDA	I2CS (F) SDA	I2CM1 (B) SCL	I2CS (F) SCL
GPIO PORT 2	x	x	I2CM0/S(B) SDA	I2CM0/S(B) SCL	I2CM0/S(A) SDA	I2CM0/S(A) SCL	I2CM1 (A) SDA	I2CS (E) SDA	I2CM1 (A) SCL	I2CS (E) SCL
GPIO PORT 3	x	x	x	x	x	x	x	x	x	x
GPIO PORT 4	x	x	x	x	x	x	x	x	x	x
GPIO PORT 5	x	x	x	x	x	x	x	x	x	x
GPIO PORT 6	x	x	x	x	x	x	x	x	x	x
GPIO PORT 7	x	x	x	x	I2CM0/S(C) SDA	I2CM0/S(C) SCL	I2CM1 (C) SDA	I2CS (G) SDA	I2CM1 (C) SCL	I2CS (G) SCL

I/O Must be Requested as a Group.
NOTE: Only one function per port.pin can be requested at a time.

I ² C Function	A	B	C	D	E	F	G	H
I2CM0/S	A	B	C	D	E	F	G	H
I2CM1	A	B	C	D	n/a	n/a	n/a	n/a

Figure 7.2: Standard Package Mapping Options

I²C Master Configurations (Standard)**I2CM0**

Logic Signal	Port and Pin
SDA	A) P2.4 B) P2.2 C) P7.4 D) P0.4
SCL	A) P2.5 B) P2.3 C) P7.5 D) P0.5

I2CM1

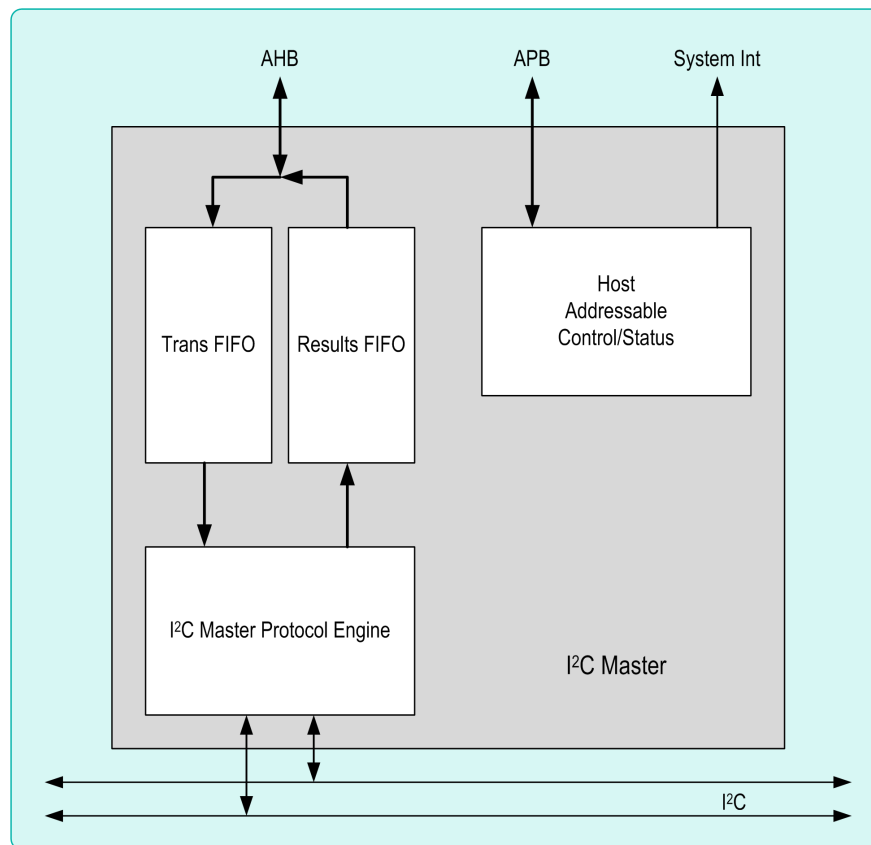
Logic Signal	Port and Pin
SDA	A) P2.6 B) P1.6 C) P7.6 D) P0.6
SCL	A) P2.7 B) P1.7 C) P7.7 D) P0.7

I²C Slave Configurations (Standard)**I2CS**

Logic Signal	Port and Pin
SDA	A) P2.4 B) P2.2 C) P7.4 D) P0.4 E) P2.6 F) P1.6 G) P7.6 H) P0.6
SCL	A) P2.5 B) P2.3 C) P7.5 D) P0.5 E) P2.7 F) P1.7 G) P7.7 H) P0.7

7.1.4 I²C Master Operation

Firmware defines I²C read and write operations via a transaction packet pushed onto the Master Transaction FIFO. I²C operation results are then read from Master Results FIFO. An unexpected NACK on write data results in system interrupt. In this case, hardware can be opted to automatically issue a Stop under this condition to free the bus. Results FIFO records the ACK/NACK status for each read data value received. Further, a full Results FIFO or empty Transaction FIFO will result in clock stretching by master. There is a timeout feature which will issue a system interrupt if this delay exceeds a firmware controllable value (in ms). Loss of arbitration in a multi-master system will result in a system interrupt.

Figure 7.3: I²C Master Block Diagram

7.1.5 Protocol

The I²C protocol communication is a two-wire serial transmission. It is a half-duplex protocol where data can be transferred at various baud rates: up to 100Kbps (100Kbps) in standard mode and up to 400Kbps (400Kbps) in fast mode.

Transfer Protocol

Each transfer is made of a start bit followed by one or more sequences of eight data bits, acknowledge bit(s) sent by the receiver, and a stop bit. Data is sent with the most significant bit (MSB) first. Every byte delivered to the SDA line must be eight bits long; however, the number of bytes that can be transmitted per transfer is unrestricted.

Bit Transfer

Both SDA and SCL lines are bi-directional lines connected to a positive supply voltage via a current source or a pullup resistor. When the bus is free, the lines are in high state. The data on the SDA line must be stable when the SCL line is high.

Communication starts when the SDA line switches from high to low state and the SCL line is high. Communication stops when the SDA line switches from low to high state and the SCL line is high. Only the master generates the start and stop conditions. After the start condition and during communication, the bus is considered busy.

Start and Stop Conditions

A high to low transition on the SDA line while SCL is high defines a start condition; a low to high transition on the SDA line while SCL is high defines a stop condition.

Acknowledge (ACK) and Not Acknowledge (NACK)

The acknowledge takes place after every byte after which the receiver signals the transmitter that the byte was successfully received and another byte may be sent. The acknowledge signal operates as follows: the transmitter releases the SDA line during the acknowledge clock pulse so the receiver can pull the SDA line to the low state (it remains stable in the low state during the high period of this clock pulse on the SCL line). Setup and hold times may affect this behavior.

A NACK signal will occur when the SDA remains high during this ninth clock pulse. The I²C master can then generate either a stop condition to abort the transfer or a repeated start condition to start a new transfer. There are five conditions that lead to NACK signal generation:

1. No receiver is present on the bus with the transmitted address so there is no device to respond with an acknowledge signal.
2. The receiver is unable to receive or transmit because it is performing some real-time function and is not ready to start communication with the master.
3. During the transfer, the receiver gets data or commands that it does not understand.
4. During the transfer, the receiver cannot receive any more data bytes.
5. A master-receiver must signal the end of the transfer to the slave transmitter.

Addressing

This block can run in master or slave mode, and in fast or standard mode. The first byte sent defines the type of addressing.

Definition of the First Byte

7-1 bits	R/W bit	Definition
0000 000	0	General Call Address
0000 000	1	START byte
0000 001	X	Reserved (CBUS address)
0000 010	X	Reserved
0000 011	X	Reserved
0000 1XX	X	Reserved
1111 1XX	X	Reserved

General Call Address

The general call address is for addressing every device connected to the I²C bus at the same time. The General Call Address is 0x00 for the first byte transmitted and addresses all devices on the bus. A device that does require data from the general call address acknowledges the address and behaves as a slave-receiver; a device that does not require any of the data supplied within the general call address does not have to acknowledge the command. A General Call Address is followed by a second byte, which initiates the devices.

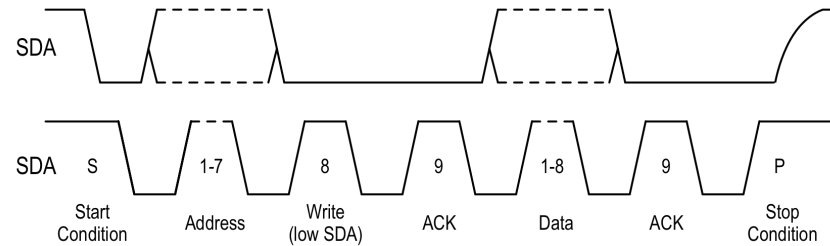


Figure 7.4: I2C Transfer

Read/Write Bit

When the R/W bit is 0, the second byte has the following definition:

- 0x06: Reset and write the programmable part of the slave address by hardware.
- 0x04: Write the programmable part of the slave address by hardware.
- 0x00: This code is not allowed to be used as the second byte.

When the R/W bit is set to 1, the hardware address of the master is written in the second byte.

The General Call Address has no effect on this interface.

START Byte

The START byte is used to initiate communication with slow devices (i.e., one reliant on software polling). The START byte has no effect on this interface.

If two or more slaves are addressed at the same time, only the slave that has the smallest address (described in decimal figures), can exchange data with the master. Communication with the other slaves is cut off.

Clock Synchronization

The I²C protocol accepts multiple masters on the bus. This is the reason why synchronization between the masters' clocks is compulsory. Clock synchronization is performed using the wired-AND connection of SCL.

Bus Arbitration

When the I²C bus is free, the two masters can initiate communication; only one master can make a valid transmission and the other must switch to slave mode. Each master compares the data sent to the SDA line. If the data is different, the master with SDA high state output will switch off its SDA data output. Arbitration occurs until only one master remains.

Master Interrupt

The **MAX32600** I²C Controller contains multiple interrupt sources that are combined into one interrupt request signal to the processor. The source of the interrupt is determined by which bits are set in the [I2CMn_INTEN](#) register.

It is recommended to acknowledge an interrupt before enabling it (unmask operation) to avoid a previous event triggering.

7.1.6 Peripheral Clock Selection and Clock Gating

To initialize the I²C master ports, the system clock source and divider must first be configured. Divider selection is dependent on many variables, including: the targeted bus-speed, the system clock frequency, and the dividers of both the system clock frequency and the I²C peripheral clock (PCLK). The register field [CLKM-AN_CLK_CTRL_6_I2CM.i2cm_clk_scale](#) enables the system clock to the I²C master ports and sets the system clock frequency divider. This is a 4-bit field.

Peripheral Clock Selection

i2cm_clk_scale	Description
0000b	CLK is Disabled
0001b	(System Clock Source / 1)
0010b	(System Clock Source / 2)
0011b	(System Clock Source / 4)
0100b	(System Clock Source / 8)
0101b	(System Clock Source / 16)
0110b	(System Clock Source / 32)
0111b	(System Clock Source / 64)
1000b	(System Clock Source / 128)

i2cm_clk_scale	Description
1001b	(System Clock Source / 256)
other	(System Clock Source / 1)

7.1.6.1 Peripheral Clock Frequency Selection

To set up the SCL frequency, it is necessary to set up four configuration registers to achieve proper operation: the I²C Peripheral Clock, Filter Clock Divisor, SCL low time, and SCL high time. The amount of time required for the SCL low time versus SCL high time (Duty Cycle) is dependent on the specific slave device(s) being communicated with. [SCL Clock Configuration Common Calculations](#) shows typical target SCL low versus SCL high times for standard slave I²C devices.

Note Pullup delays, input filtering delays, and multi-master clock synchronization affect the observed SCL frequency on the I²C bus.

Full-Speed Target SCL Calculation

$$\text{Target SCL Frequency} = \frac{\text{Peripheral Clock}}{fs_filter_clk_div}$$

Where `fs_filter_clk_div` is an 8-bit number to divide down the [Peripheral Clock](#).

SCL Clock Configuration Common Calculations

PCLK	F _{I2C}	I ² C _{DUTY}	F _{HOLD}	T _{RC}	Filt CLK Divisor	SCL Hi	SCL Lo
24	100	0.67	1	1000	12	38	144
24	400	0.67	0.25	300	3	5	36
12	100	0.67	1	1000	6	17	72
12	400	0.67	0.25	300	2	1	18
6	100	0.67	1	1000	3	7	36
3	100	0.67	1	1000	2	1	18

Note Explanation of [SCL Clock Configuration Common Calculations](#) values:

- PCLK = Peripheral Clock (MHz)
- F_{I²C} = I²C Frequency (KHz)
- I²C_{DUTY} = I²C Duty Cycle (H/L)
- F_{HOLD} = I²C Hold (μs)
- T_{RC} = RC Rise Time (ns)
- Filt CLK Divisor = [fs_filter_clk_div](#)
- SCL Hi = [fs_scl_hi_cnt](#)
- SCL Lo = [fs_scl_lo_cnt](#)

7.1.7 Communication and Data Transfer

7.1.7.1 FIFO-Based I²C Master

The FIFO-based I²C engine implements a packetized interface to slave devices. This interface supports multi-master environments, 10-bit addressing, and System Management Bus (SMBus) protocol.

Sixteen-byte FIFOs are provided on each master peripheral for both Tx and Rx. The data is tagged prior to enqueueing to allow decoupling of firmware execution from I²C operation. A full complement of FIFO status is available for firmware monitoring and interrupt generation.

The user must define the filter clock frequency, which is typically two to four times faster than the target SCL frequency. The target SCL frequency and duty cycle are set by defining the low and high system clock limits of the SCL clock (i.e., SCL low time and SCL high time). If operating in a high-speed environment, the user must define two sets of clock control registers for normal-speed as well as high-speed.

Note Pullup delays, input filtering delays, and multi-master clock synchronization affect the observed SCL frequency on the I²C bus.

I²C and SMBus Compliance

SMBus and I²C protocols are essentially the same: an SMBus master is able to control I²C devices and vice versa at the protocol level. The SMBus clock is defined from 10kHz to 100kHz whereas I²C can range from 0Hz to 100kHz or 0Hz to 400kHz depending on the mode. An I²C bus running at less than 10kHz is not SMBus compliant since the SMBus device(s) may time out (see [Timeout Feature](#) below).

Peripheral Management Unit

The I²C supports direct memory access peripheral communication via the [Peripheral Management Unit \(PMU\)](#). This allows the PMU to read and/or write the FIFOs of the I²C device.

To enable the PMU transfers, reference [Peripheral Management Unit \(PMU\)](#). There is no additional configuration for the I²C device: the PMU should be seen as a core (executing software) replacement. The I²C interrupts are not rerouted to the PMU and are still routed to the core (it is up to the software to configure the interrupts according to the PMU usage).

Rx FIFO

Rx FIFO transfers are 8-byte depth. The Rx FIFO is written by hardware when a data has been received from the device. If the Rx FIFO is full, all data received from the slave are lost; a new read operation cannot be started with the Rx FIFO full. Reading the data register is necessary to read the Rx FIFO.

Tx FIFO

Tx FIFO transfers are 8-byte depth. The Tx FIFO is written by software using the data register (a write in the data register writes into the Tx FIFO). The Tx FIFO is read by hardware only when the acknowledge bit has been received (meaning just before the stop/restart bit in case of NACK or just before the first bit of the next data in case of ACK).

Timeout Feature

The SMBus protocol has a timeout feature which resets devices if communication takes too long. The minimum clock frequency of 10kHz for SMBus peripherals prevents locking up the bus erroneously. The I²C can be a DC bus, meaning that a slave device stretches the master clock when performing some routine while the master is accessing it. This notifies the master that the slave is busy but does not want to lose communication. The slave device will allow continuation of the communication after its task is complete. There is no limit to the delay in the I²C bus protocol; for a SMBus system, the delay is limited to 35ms. SMBus protocol assumes if a communication takes too long, there must be a problem on the bus and all devices must reset in order to clear this mode. This is the timeout feature of the SMBus. Slave devices are not then allowed to hold the clock in the low state too long.

7.1.8 Registers (I2CM)

7.1.8.1 Module I2CM Registers

Address	Register	32b Word Len	Description
0x40040000	I2CM0_FS_CLK_DIV	1	I2C Master 0 Full Speed SCL Clock Settings

Address	Register	32b Word Len	Description
0x4004000C	I2CM0_TIMEOUT	1	I2C Master 0 Timeout and Auto-Stop Settings
0x40040010	I2CM0_CTRL	1	I2C Master 0 Control Register
0x40040014	I2CM0_TRANS	1	I2C Master 0 Transaction Start and Status Flags
0x40040018	I2CM0_INTFL	1	I2C Master 0 Interrupt Flags
0x4004001C	I2CM0_INTEN	1	I2C Master 0 Interrupt Enable/Disable Controls
0x40040028	I2CM0_BB	1	I2C Master 0 Bit-Bang Control Register
0x40042000	I2CM1_FS_CLK_DIV	1	I2C Master 1 Full Speed SCL Clock Settings
0x4004200C	I2CM1_TIMEOUT	1	I2C Master 1 Timeout and Auto-Stop Settings
0x40042010	I2CM1_CTRL	1	I2C Master 1 Control Register
0x40042014	I2CM1_TRANS	1	I2C Master 1 Transaction Start and Status Flags
0x40042018	I2CM1_INTFL	1	I2C Master 1 Interrupt Flags
0x4004201C	I2CM1_INTEN	1	I2C Master 1 Interrupt Enable/Disable Controls
0x40042028	I2CM1_BB	1	I2C Master 1 Bit-Bang Control Register
0x40103000	I2CM0_FIFO_TRANS	512	I2C Master 0 Transaction FIFO
0x40103800	I2CM0_FIFO_RSLTS	512	I2C Master 0 Results FIFO
0x4010D000	I2CM1_FIFO_TRANS	512	I2C Master 1 Transaction FIFO
0x4010D800	I2CM1_FIFO_RSLTS	512	I2C Master 1 Results FIFO

7.1.8.1.1 I2CMn_FS_CLK_DIV

I2CMn_FS_CLK_DIV.fs_filter_clk_div

Field	Bits	Default	Access	Description
fs_filter_clk_div	7:0	00000001b	R/W	Full Speed Filter Clock Divisor

Filter frequency = (I2C Master Module Clock)/fs_filter_clk_div. Setting this value to 1 will disable the I2C Master.

I2CMn_FS_CLK_DIV.fs_scl_lo_cnt

Field	Bits	Default	Access	Description
fs_scl_lo_cnt	19:8	12'b0	R/W	Full Speed SCL Low Count

Number of clocks to hold SCL low for clock output

I2CMn_FS_CLK_DIV.fs_scl_hi_cnt

Field	Bits	Default	Access	Description
fs_scl_hi_cnt	31:20	12'b0	R/W	Full Speed SCL High Count

Number of clocks to hold SCL high for clock output

7.1.8.1.2 I2CMn_TIMEOUT

I2CMn_TIMEOUT.tx_timeout

Field	Bits	Default	Access	Description
tx_timeout	23:16	8'b0	R/W	Transaction Timeout Limit

Timeout limit (in mS) for a given transaction; when this timeout expires, master releases the bus and triggers an interrupt.

I2CMn_TIMEOUT.auto_stop_en

Field	Bits	Default	Access	Description
auto_stop_en	24	0	R/W	Auto-Stop Enable

If 1, master automatically issues a Stop when a timeout or unexpected Nack occurs.

7.1.8.1.3 I2CMn_CTRL

I2CMn_CTRL.tx_fifo_en

Field	Bits	Default	Access	Description
tx_fifo_en	2	0	R/W	Master Transaction FIFO Enable

0:Disabled; 1:Enabled

I2CMn_CTRL.rx_fifo_en

Field	Bits	Default	Access	Description
rx_fifo_en	3	0	R/W	Master Results FIFO Enable

0:Disabled; 1:Enabled

I2CMn_CTRL.mstr_reset_en

Field	Bits	Default	Access	Description
mstr_reset_en	7	0	R/W	Master Reset

1:Held in reset

7.1.8.1.4 I2CMn_TRANS

I2CMn_TRANS.tx_start

Field	Bits	Default	Access	Description
tx_start	0	0	R/W	Start Transaction

Write to 1 to begin a master transaction.

I2CMn_TRANS.tx_in_progress

Field	Bits	Default	Access	Description
tx_in_progress	1	0	R/O	Transaction In Progress

Set to 1 by hardware when a new transaction is started; cleared when transaction ends.

I2CMn_TRANS.tx_done

Field	Bits	Default	Access	Description
tx_done	2	0	R/O	Transaction Done

Set to 1 by hardware when a transaction completes; cleared to 0 on transaction Start.

I2CMn_TRANS.tx_nacked

Field	Bits	Default	Access	Description
tx_nacked	3	0	R/O	Transaction Nacked

Set to 1 when a transaction completes due to an unexpected NACK; cleared to 0 on Start.

I2CMn_TRANS.tx_lost_arbitr

Field	Bits	Default	Access	Description
tx_lost_arbitr	4	0	R/O	Transaction Lost Arbitration

Set to 1 when a transaction halts due to an arbitration failure; cleared to 0 on Start.

I2CMn_TRANS.tx_timeout

Field	Bits	Default	Access	Description
tx_timeout	5	0	R/O	Transaction Timed Out

Set to 1 when a transaction halts due to a timeout; cleared to 0 on Start.

7.1.8.1.5 I2CMn_INTFL

I2CMn_INTFL.tx_done

Field	Bits	Default	Access	Description
tx_done	0	0	W1C	Transaction Done Int Status

Write 1 to clear.

Set to 1 by hardware when a transaction completes.

I2CMn_INTFL.tx_nacked

Field	Bits	Default	Access	Description
tx_nacked	1	0	W1C	Transaction NACKed Int Status

Write 1 to clear.

Set to 1 by hardware when a transaction is interrupted due to an unexpected NACK response.

I2CMn_INTFL.tx_lost_arbitr

Field	Bits	Default	Access	Description
tx_lost_arbitr	2	0	W1C	Transaction Lost Arbitration Int Status

Write 1 to clear.

Set to 1 by hardware when a transaction is interrupted due to a lost arbitration failure.

I2CMn_INTFL.tx_timeout

Field	Bits	Default	Access	Description
tx_timeout	3	0	W1C	Transaction Timed Out Int Status

Write 1 to clear.

Set to 1 by hardware when a transaction times out.

I2CMn_INTFL.tx_fifo_empty

Field	Bits	Default	Access	Description
tx_fifo_empty	4	0	W1C	Transaction FIFO Empty Int Status

Write 1 to clear.

Set to 1 by hardware when the transaction FIFO is empty.

I2CMn_INTFL.tx_fifo_3q_empty

Field	Bits	Default	Access	Description
tx_fifo_3q_empty	5	0	W1C	Transaction FIFO 3Q Empty Int Status

Write 1 to clear.

Set to 1 by hardware when the transaction FIFO is three-quarters empty.

I2CMn_INTFL.rx_fifo_not_empty

Field	Bits	Default	Access	Description
rx_fifo_not_empty	6	0	W1C	Results FIFO Not Empty Int Status

Write 1 to clear.

Set to 1 by hardware when the results FIFO is not empty.

I2CMn_INTFL.rx_fifo_2q_full

Field	Bits	Default	Access	Description
rx_fifo_2q_full	7	0	W1C	Results FIFO 2Q Full Int Status

Write 1 to clear.

Set to 1 by hardware when the results FIFO is half full (two quarters).

I2CMn_INTFL.rx_fifo_3q_full

Field	Bits	Default	Access	Description
rx_fifo_3q_full	8	0	W1C	Results FIFO 3Q Full Int Status

Write 1 to clear.

Set to 1 by hardware when the results FIFO is three-quarters full.

I2CMn_INTFL.rx_fifo_full

Field	Bits	Default	Access	Description
rx_fifo_full	9	0	W1C	Results FIFO Full Int Status

Write 1 to clear.

Set to 1 by hardware when the results FIFO is completely full.

7.1.8.1.6 I2CMn_INTEN**I2CMn_INTEN.tx_done**

Field	Bits	Default	Access	Description
tx_done	0	0	R/W	Transaction Done Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.tx_nacked

Field	Bits	Default	Access	Description
tx_nacked	1	0	R/W	Transaction NACKed Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.tx_lost_arbitr

Field	Bits	Default	Access	Description
tx_lost_arbitr	2	0	R/W	Transaction Lost Arbitration IntEnable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.tx_timeout

Field	Bits	Default	Access	Description
tx_timeout	3	0	R/W	Transaction Timed Out Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.tx_fifo_empty

Field	Bits	Default	Access	Description
tx_fifo_empty	4	0	R/W	Transaction FIFO Empty Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.tx_fifo_3q_empty

Field	Bits	Default	Access	Description
tx_fifo_3q_empty	5	0	R/W	Transaction FIFO 3Q Empty Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.rx_fifo_not_empty

Field	Bits	Default	Access	Description
rx_fifo_not_empty	6	0	R/W	Results FIFO Not Empty Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.rx_fifo_2q_full

Field	Bits	Default	Access	Description
rx_fifo_2q_full	7	0	R/W	Results FIFO 2Q Full Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.rx_fifo_3q_full

Field	Bits	Default	Access	Description
rx_fifo_3q_full	8	0	R/W	Results FIFO 3Q Full Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.rx_fifo_full

Field	Bits	Default	Access	Description
rx_fifo_full	9	0	R/W	Results FIFO Full Int Enable

0:Associated int disabled; 1:Interrupt enabled

7.1.8.1.7 I2CMn_BB**I2CMn_BB.bb_scl_out**

Field	Bits	Default	Access	Description
bb_scl_out	0	1	R/W	Bit Bang SCL Output

I2CMn_BB.bb_sda_out

Field	Bits	Default	Access	Description
bb_sda_out	1	1	R/W	Bit Bang SDA Output

I2CMn_BB.bb_scl_in_val

Field	Bits	Default	Access	Description
bb_scl_in_val	2	s	R/O	Bit Bang SCL Input Value

I2CMn_BB.bb_sda_in_val

Field	Bits	Default	Access	Description
bb_sda_in_val	3	s	R/O	Bit Bang SCL Input Value

I2CMn_BB.rx_fifo_cnt

Field	Bits	Default	Access	Description
rx_fifo_cnt	20:16	s	R/O	Results FIFO Data Received Count

7.1.8.1.8 I2CMn_FIFO_TRANS

Default	Access	Description
n/a	R/W	I2C Master 0 Transaction FIFO

Writes to this space result in pushes to the I2C Master Transaction FIFO.

Reads from this space return the FIFO full flag in bit 0, and all other bits are 0.

7.1.8.1.9 I2CMn_FIFO_RSLTS

Default	Access	Description
n/a	R/W	I2C Master 0 Results FIFO

Reads from this space return the next value which is pulled from the Results FIFO.

Writes to this space are ignored.

7.1.9 Registers (I2CS)**7.1.9.1 Module I2CS Registers**

Address	Register	32b Word Len	Description
0x40041000	I2CS0_FS_CLK_DIV	1	Full Speed SCL Clock Settings
0x40041008	I2CS0_DEV_IDS	1	Slave Device ID Settings
0x4004100C	I2CS0_TIMEOUT	1	Timeout and Auto-Stop Settings
0x40041010	I2CS0_CTRL	1	I2C Slave Interface Control Register
0x40041020	I2CS0_INTFL	1	Interrupt Flags
0x40041024	I2CS0_INTEN	1	Interrupt Enable/Disable Controls
0x40041028	I2CS0_BB	1	Bit-Bang Control Register
0x4004102C	I2CS0_SRX_PEEK	1	RX FIFO Peek (Read from end without pull operation)
0x40104000	I2CS0_FIFO_RX_DATA	512	I2C Slave Receive Data FIFO
0x40104800	I2CS0_FIFO_TX_DATA	512	I2C Slave Transmit Data FIFO

7.1.9.1.1 I2CS0_FS_CLK_DIV

I2CS0_FS_CLK_DIV.fs_filter_clk_div

Field	Bits	Default	Access	Description
fs_filter_clk_div	7:0	00000001b	R/W	Full Speed Filter Clock Divisor

Filter frequency = I2C module clock/bits[7:0]; 1=Off

7.1.9.1.2 I2CS0_DEV_IDS

I2CS0_DEV_IDS.dev_id0

Field	Bits	Default	Access	Description
dev_id0	9:0	10'b0	R/W	Slave Device ID 0

The first slave ID which the I2C Slave will respond to.

I2CS0_DEV_IDS.dev_id1

Field	Bits	Default	Access	Description
dev_id1	21:12	10'b0	R/W	Slave Device ID 1

The second slave ID which the I2C Slave will respond to.

I2CS0_DEV_IDS.dev_10b

Field	Bits	Default	Access	Description
dev_10b	24	0	R/W	10-bit Device ID Mode

Controls whether 7-bit or 10-bit addressing is used.

7.1.9.1.3 I2CS0_TIMEOUT**I2CS0_TIMEOUT.clk_stretch_to_limit**

Field	Bits	Default	Access	Description
clk_stretch_to_limit	7:0	8'b0	R/W	Clock-Stretch Timeout Limit

Maximum timeout limit for a slave-controlled clock stretch before the slave will timeout.

I2CS0_TIMEOUT.max_rx_block_size

Field	Bits	Default	Access	Description
max_rx_block_size	15:8	8'b0	R/W	Max RX Block Size

The maximum receive block size which will be accepted by the slave.

I2CS0_TIMEOUT.max_rx_block_size_en

Field	Bits	Default	Access	Description
max_rx_block_size_en	25	0	R/W	Enforce Max RX Block Size

Controls whether the maximum receive block size setting is actually used.

I2CS0_TIMEOUT.tx_arbitr_en

Field	Bits	Default	Access	Description
tx_arbitr_en	26	0	R/W	Enable Slave TX Arbitration

Enables clock arbitration mode to be used by the I2C slave.

I2CS0_TIMEOUT.rx_stop_record_en

Field	Bits	Default	Access	Description
rx_stop_record_en	27	0	R/W	Push RX Stop Record

Enable RX Stop record push to I2C Results FIFO. Otherwise, mark first RX data after Start/Restart.

7.1.9.1.4 I2CS0_CTRL**I2CS0_CTRL.rx_fifo_en**

Field	Bits	Default	Access	Description
rx_fifo_en	0	0	R/W	Slave RX FIFO Enable

Disable to use 1-byte FIFO mode.

I2CS0_CTRL.tx_fifo_en

Field	Bits	Default	Access	Description
tx_fifo_en	1	0	R/W	Slave TX FIFO Enable

I2CS0_CTRL.rx_en

Field	Bits	Default	Access	Description
rx_en	4	0	R/W	Slave RX Enable

Enables I2C slave receive mode.

I2CS0_CTRL.tx_en

Field	Bits	Default	Access	Description
tx_en	5	0	R/W	Slave TX Enable

Enables I2C slave transmit mode.

I2CS0_CTRL.slave_reset_en

Field	Bits	Default	Access	Description
slave_reset_en	6	0	R/W	Slave I2C Reset

If set to 1, holds I2C slave instance in reset.

7.1.9.1.5 I2CS0_INTFL**I2CS0_INTFL.clk_stretch_to**

Field	Bits	Default	Access	Description
clk_stretch_to	0	0	W1C	Clock Stretch Timeout Interrupt Status

Write 1 to clear.

Set to 1 by hardware when a clock stretch timeout event occurs.

I2CS0_INTFL.tx_fifo_empty

Field	Bits	Default	Access	Description
tx_fifo_empty	1	0	W1C	Tx FIFO Empty Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the Tx FIFO is empty.

I2CS0_INTFL.tx_fifo_3q_empty

Field	Bits	Default	Access	Description
tx_fifo_3q_empty	2	0	W1C	Tx FIFO 3Q Empty Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the Tx FIFO is three-quarters empty.

I2CS0_INTFL.rx_fifo_empty

Field	Bits	Default	Access	Description
rx_fifo_empty	3	0	W1C	Rx FIFO Empty Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the Rx FIFO is empty.

I2CS0_INTFL.rx_fifo_2q_full

Field	Bits	Default	Access	Description
rx_fifo_2q_full	4	0	W1C	Rx FIFO 2Q Full Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the Rx FIFO is half full (two quarters).

I2CS0_INTFL.rx_fifo_3q_full

Field	Bits	Default	Access	Description
rx_fifo_3q_full	5	0	W1C	Rx FIFO 3Q Full Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the Rx FIFO is three-quarters full.

I2CS0_INTFL.rx_fifo_full

Field	Bits	Default	Access	Description
rx_fifo_full	6	0	W1C	Rx FIFO Full Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the Rx FIFO is full.

I2CS0_INTFL.rx_clk_stretch

Field	Bits	Default	Access	Description
rx_clk_stretch	7	0	W1C	Rx Clock Stretch Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the I2C slave begins a clock stretch during a receive cycle.

I2CS0_INTFL.tx_clk_stretch_id0

Field	Bits	Default	Access	Description
tx_clk_stretch_id0	8	0	W1C	Tx Clock Stretch (ID0) Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the I2C slave begins a clock stretch during a transmit cycle (ID0).

I2CS0_INTFL.tx_clk_stretch_id1

Field	Bits	Default	Access	Description
tx_clk_stretch_id1	9	0	W1C	Tx Clock Stretch (ID1) Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the I2C slave begins a clock stretch during a transmit cycle (ID1).

I2CS0_INTFL.restart_id0

Field	Bits	Default	Access	Description
restart_id0	10	0	W1C	Restart Detected (ID0) Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the I2C slave detects a restart event (ID0).

I2CS0_INTFL.restart_id1

Field	Bits	Default	Access	Description
restart_id1	11	0	W1C	Restart Detected (ID1) Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the I2C slave detects a restart event (ID1).

I2CS0_INTFL.stop_id0

Field	Bits	Default	Access	Description
stop_id0	12	0	W1C	Stop Detected (ID0) Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the I2C slave detects a Stop event (ID0).

I2CS0_INTFL.stop_id1

Field	Bits	Default	Access	Description
stop_id1	13	0	W1C	Stop Detected (ID1) Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the I2C slave detects a Stop event (ID1).

I2CS0_INTFL.lost_arbitr_id0

Field	Bits	Default	Access	Description
lost_arbitr_id0	14	0	W1C	Lost Arbitration (ID0) Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the I2C slave detects an arbitration loss event (ID0).

I2CS0_INTFL.lost_arbitr_id1

Field	Bits	Default	Access	Description
lost_arbitr_id1	15	0	W1C	Lost Arbitration (ID1) Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the I2C slave detects an arbitration loss event (ID1).

7.1.9.1.6 I2CS0_INTEN

I2CS0_INTEN.clk_stretch_to

Field	Bits	Default	Access	Description
clk_stretch_to	0	0	R/W	Clock Stretch Timeout Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.tx_fifo_empty

Field	Bits	Default	Access	Description
tx_fifo_empty	1	0	R/W	Tx FIFO Empty Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.tx_fifo_3q_empty

Field	Bits	Default	Access	Description
tx_fifo_3q_empty	2	0	R/W	Tx FIFO 3Q Empty Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.rx_fifo_empty

Field	Bits	Default	Access	Description
rx_fifo_empty	3	0	R/W	Rx FIFO Empty Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.rx_fifo_2q_full

Field	Bits	Default	Access	Description
rx_fifo_2q_full	4	0	R/W	Rx FIFO 2Q Full Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.rx_fifo_3q_full

Field	Bits	Default	Access	Description
rx_fifo_3q_full	5	0	R/W	Rx FIFO 3Q Full Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.rx_fifo_full

Field	Bits	Default	Access	Description
rx_fifo_full	6	0	R/W	Rx FIFO Full Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.rx_clk_stretch

Field	Bits	Default	Access	Description
rx_clk_stretch	7	0	R/W	Rx Clock Stretch Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.tx_clk_stretch_id0

Field	Bits	Default	Access	Description
tx_clk_stretch_id0	8	0	R/W	Tx Clock Stretch (ID0) Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.tx_clk_stretch_id1

Field	Bits	Default	Access	Description
tx_clk_stretch_id1	9	0	R/W	Tx Clock Stretch (ID1) Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.restart_id0

Field	Bits	Default	Access	Description
restart_id0	10	0	R/W	Restart Detected (ID0) Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.restart_id1

Field	Bits	Default	Access	Description
restart_id1	11	0	R/W	Restart Detected (ID1) Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.stop_id0

Field	Bits	Default	Access	Description
stop_id0	12	0	R/W	Stop Detected (ID0) Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.stop_id1

Field	Bits	Default	Access	Description
stop_id1	13	0	R/W	Stop Detected (ID1) Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.lost_arbitr_id0

Field	Bits	Default	Access	Description
lost_arbitr_id0	14	0	R/W	Lost Arbitration (ID0) Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

I2CS0_INTEN.lost_arbitr_id1

Field	Bits	Default	Access	Description
lost_arbitr_id1	15	0	R/W	Lost Arbitration (ID1) Interrupt Enable

0:Interrupt is disabled; 1:Interrupt is enabled.

7.1.9.1.7 I2CS0_BB**I2CS0_BB.scl_out**

Field	Bits	Default	Access	Description
scl_out	0	1	R/W	Bit Bang SCL Output

I2CS0_BB.sda_out

Field	Bits	Default	Access	Description
sda_out	1	1	R/W	Bit Bang SDA Output

I2CS0_BB.scl_in_val

Field	Bits	Default	Access	Description
scl_in_val	2	s	R/O	Bit Bang SCL Input Value

I2CS0_BB.sda_in_val

Field	Bits	Default	Access	Description
sda_in_val	3	s	R/O	Bit Bang SDA Input Value

I2CS0_BB.rx_fifo_word_count

Field	Bits	Default	Access	Description
rx_fifo_word_count	21:16	ssssss	R/O	RX FIFO Word Count

7.1.9.1.8 I2CS0_SRX_PEEK**I2CS0_SRX_PEEK.rx_data**

Field	Bits	Default	Access	Description
rx_data	10:0	s	R/O	Slave RX Data Nondestructive

Returns the next value from the RX FIFO without removing the data from the FIFO.

I2CS0_SRX_PEEK.rx_fifo_read_only

Field	Bits	Default	Access	Description
rx_fifo_read_only	11	1	R/O	Slave RX FIFO Empty

Returns 1 if the slave RX FIFO is currently empty.

7.1.9.1.9 I2CS0_FIFO_RX_DATA

Default	Access	Description
n/a	R/W	I2C Slave Receive Data FIFO

Writes to this space are ignored.

Reads from this space return the next value which is pulled from the RX FIFO.

7.1.9.1.10 I2CS0_FIFO_TX_DATA

Default	Access	Description
n/a	R/W	I2C Slave Transmit Data FIFO

Writes to this space result in pushes to the I2C Slave TX FIFO.

Reads from this space return the FIFO full flag in bit 0, and all other bits are 0.

7.2 SPI

7.2.1 Overview

The serial peripheral interface (SPI) module of the **MAX32600** microcontroller provides a highly configurable, flexible, and efficient interface to communicate with a wide variety of SPI slave devices. Three SPI Master ports (SPI0, SPI1, and SPI2) are available on the **MAX32600** and each supports the following features:

- Support of all four [SPI modes \(0, 1, 2, and 3\)](#) for single-bit communication
- 3- or 4-wire mode for single-bit slave device communication
- Full-duplex operation in single-bit mode
- Dual and Quad I/O supported
- Up to five slave select (SS) lines per port with programmable polarity
- Up to two slave ready (SR) lines with programmable polarity
- Programmable interface timing
- Programmable SCK frequency and duty cycle
- Programmable SCK alternate timing
- SS assertion and deassertion timing with respect to leading/trailing SCK edge

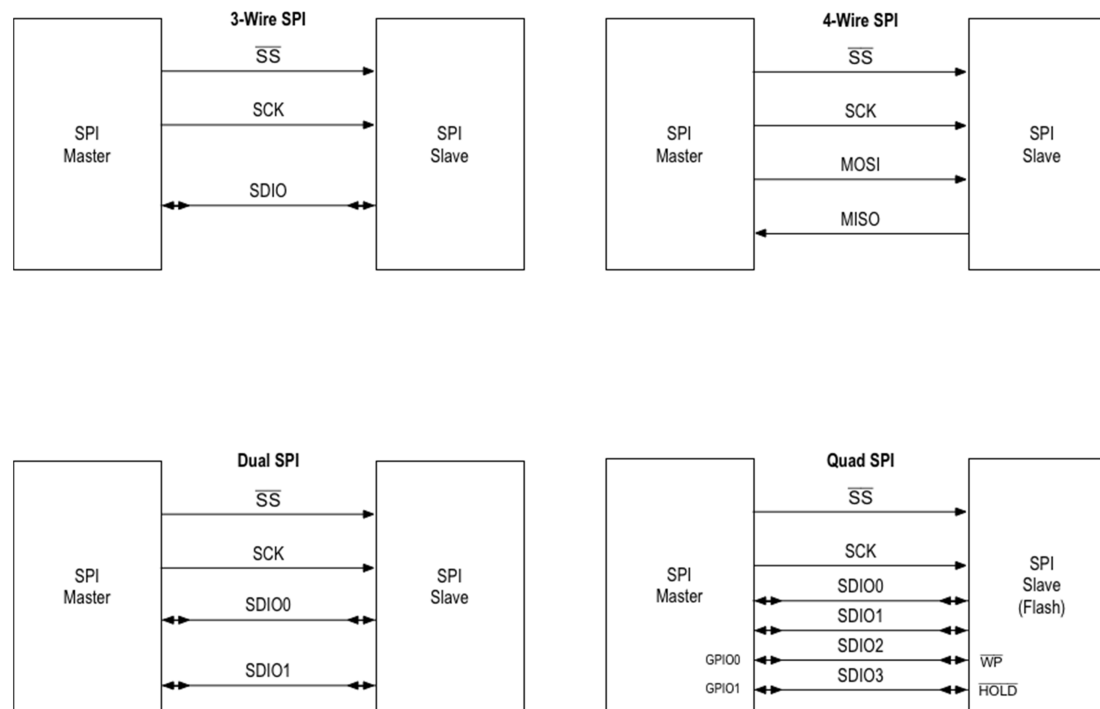


Figure 7.5: Multi I/O SPI Support

7.2.2 SPI Port and Pin Configurations

SPI Wiring Configurations

- **3-Wire SPI:** SS, SCK, SDIO
- **4-Wire SPI:** SS, SCK, MOSI, MISO
- **Dual SPI:** SS, SCK, SDIO0, SDIO1
- **Quad SPI:** SS, SCK, SDIO0, SDIO1, SDIO2, SDIO3

Slave Ready (SR) lines are optional; configuration details can be found in the [Static Configuration](#) section.

Note See [Pin Layout](#) for a detailed mapping of **MAX32600** multiplexed function locations. Functional priority distinction is included in the mapping.

7.2.2.1 Compact Layout (7mm x 7mm) Configuration

The tables below contain the available pin configurations for each of the SPI Master ports (SPI0, SPI1, and SPI2).

SPI0

Logic Signal	Port and Pin
SS	A) P0.3(0), P0.4(1), P0.5(2), P0.6(3), P0.7(4) B) P1.3(0), P1.4(1), P1.5(2), P1.6(3), P1.7(4)
SCK	A) P0.0 B) P1.0
SDIO	A) P0.6(2), P0.7(3) B) P1.6(2), P1.7(3)
SDIO (MOSI)	A) P0.1(0) B) P1.1(0)
SDIO (MISO)	A) P0.2(1) B) P1.2(1)

Optional: Slave Ready

Logic Signal	Port and Pin
SR	A) P0.4(0), P0.5(1) B) P1.4(0), P1.5(1)

SPI1

Logic Signal	Port and Pin
SS	A) P0.7(0), P0.0(1), P0.1(2), P0.2(3), P0.3(4) B) P1.7(0), P1.0(1), P1.1(2), P1.2(3), P1.3(4)
SCK	A) P0.4 B) P1.4
SDIO	A) P0.2(2), P0.3(3) B) P1.2(2), P1.3(3)
SDIO (MOSI)	A) P0.5(0) B) P1.5(0)
SDIO (MISO)	A) P0.6(1) B) P1.6(1)

Optional: Slave Ready

Logic Signal	Port and Pin
SR	A) P0.0(0), P0.1(1) B) P1.0(0), P1.1(1)

SPI2

Logic Signal	Port and Pin
SS	A/B) P2.3(0), P2.4(1), P2.5(2), P2.6(3), P2.7(4)
SCK	A/B) P2.0
SDIO	A/B) P2.6(2), P2.7(3)
SDIO (MOSI)	A/B) P2.1(0)
SDIO (MISO)	A/B) P2.2(1)

Optional: Slave Ready

Logic Signal	Port and Pin
SR	A) P1.7(0) B) P2.4(0) A/B) 2.5(1)

7.2.2.2 Standard Layout (12mm x 12mm) Configuration

The tables below contain the available pin configurations for each of the SPI Master ports (SPI0, SPI1, and SPI2):

SPI0

Logic Signal	Port and Pin
SS	A) P0.3(0), P0.4(1), P0.5(2), P0.6(3), P0.7(4) B) P1.3(0), P2.4(1), P2.5(2), P2.6(3), P2.7(4) C) P6.3(0), P6.4(1), P6.5(2), P6.6(3), P6.7(4)
SCK	A) P0.0 B) P1.0 C) P6.0
SDIO	A) P0.6(2), P0.7(3) B) P2.6(2), P2.7(3) C) P6.6(2), P6.7(3)
SDIO (MOSI)	A) P0.1(0) B) P1.1(0) C) P6.1(0)
SDIO (MISO)	A) P0.2(1) B) P1.2(1) C) P6.2(1)

Optional: Slave Ready

Logic Signal	Port and Pin
SR	A) P0.4(0), P0.5(1) B) P2.4(0), P2.5(1) C) P6.4(0), P6.5(1)

SPI1

Logic Signal	Port and Pin
SS	A) P0.7(0), P0.0(1), P0.1(2), P0.2(3), P0.3(4) B) P2.7(0), P1.0(1), P1.1(2), P1.2(3), P1.3(4) C) P6.7(0), P6.0(1), P6.1(2), P6.2(3), P6.3(4)
SCK	A) P0.4 B) P2.4 C) P6.4
SDIO	A) P0.2(2), P0.3(3) B) P1.2(2), P1.3(3) C) P6.2(2), P6.3(3)
SDIO (MOSI)	A) P0.5(0) B) P2.5(0) C) P6.5(0)
SDIO (MISO)	A) P0.6(1) B) P2.6(1) C) P6.6(1)

Optional: Slave Ready

Logic Signal	Port and Pin
SR	A) P0.0(0), P0.1(1) B) P1.0(0), P1.1(1) C) P6.0(0), P6.1(1)

SPI2

Logic Signal	Port and Pin
SS	A/B) P2.3(0), P1.4(1), P1.5(2), P1.6(3) P1.7(4)
SCK	A/B) P2.0
SDIO	A/B) P1.6(2), P1.7(3)
SDIO (MOSI)	A/B) P2.1(0)
SDIO (MISO)	A/B) P2.2(1)

Optional: Slave Ready

Logic Signal	Port and Pin
SR	A) P2.7(0) B) P1.4(0) A/B) P1.5(1)

7.2.3 Clock Selection and Configuration

The **MAX32600** supports programmable SPI clock rates, which are a divisor of the system clock. Each of the three SPI ports is able to set its clock rate independently. To set the base clock rate, write to the appropriate Clock Control register with the value desired to achieve the ideal clock rate for the slave devices.

SPI Clock Control Registers

Port	SPI Clock Control Register
SPI0	CLKMAN_CLK_CTRL_3_SPI0
SPI1	CLKMAN_CLK_CTRL_4_SPI1
SPI2	CLKMAN_CLK_CTRL_5_SPI2

Setting the SPI Clock Rate - `CLKMAN_CLK_CTRL_[N]_SPI[x]`

[3:0]	SPI Clock Rate
0000b	Disabled
0001b	(System Clock Source / 1)
0010b	(System Clock Source / 2)
0011b	(System Clock Source / 4)

[3:0]	SPI Clock Rate
0100b	(System Clock Source / 8)
0101b	(System Clock Source / 16)
0110b	(System Clock Source / 32)
0111b	(System Clock Source / 64)
1000b	(System Clock Source / 128)
1001b	(System Clock Source / 256)
other	(System Clock Source / 1)

7.2.3.1 Clock Gating

Clock gating for the SPI ports is controlled by the register fields [CLKMAN_CLK_GATE_CTRL1.spi0_clk_gater](#), [CLKMAN_CLK_GATE_CTRL1.spi1_clk_gater](#), and [CLKMAN_CLK_GATE_CTRL1.spi2_clk_gater](#). The table below shows the supported settings for the `spi[x]_clk_gater` register fields and their meanings.

Clock Control - `CLKMAN_CLK_GATE_CTRL1.spi[x]_clk_gater`

SPI[x] Clock Control Value (2b)	Setting
00b	Clock off - SPI[x] disabled
01b	Dynamic Clock Gating Enabled - SPI[x] clock active only when used
10b or 11b	Clock on - SPI[x] enabled at all times

7.2.4 Configuration Modes Overview

Once the main SPI clock is set up for the port, the remainder of the configuration and operation for SPI is mapped into three categories:

- *Static Configuration:* Performed during SPI initial setup and/or when SPI is not active.
 - [SPIn_SS_SR_POLARITY](#)
 - [SPIn_GEN_CTRL](#)
- *Dynamic Configuration:* Configuration required to communicate with a specific slave device, which may take place while the SPI port is active.
 - [SPIn_MSTR_CFG](#)

- *Interrupt Servicing*: Status and Control used by an application either directly or via the Peripheral Management Unit's DMA to efficiently service SPI data transfer.
 - `SPIn_FIFO_CTRL`
 - `SPIn_INTFL`
 - `SPIn_INTEN`

7.2.4.1 Static Configuration

Static configuration should be performed while the SPI port is disabled. Static configuration includes:

- Slave Select signal polarity
- Slave Ready (Flow Control) signal polarity

Slave select polarity is independently configurable for each slave select line for a given SPI port. To set the Slave Select for the SPI port to an Active High State, set the `ss_polarity` field to 0000001b in the register `SPIn_SS_SR_POLARITY`. To set additional slave selects for the same port, set the appropriate bit(s) in the `ss_polarity` field to match the slave select line. By default, the slave selects are set to Active Low.

For the slave ready polarity, each slave ready input signal for a given SPI port is configured in the same manner as the Slave Select. Set the `fc_polarity` bit in the `SPIn_SS_SR_POLARITY` register to either a 1 or a 0.

7.2.4.2 Dynamic Configuration

To begin communicating with a given slave device it is necessary to set up several parameters specific to that slave. All of these settings are controlled by the `SPIn_MSTR_CFG` register.

The `SPIn_MSTR_CFG.slave_sel` field determines which slave select pin is active during the transaction. A total of five possibilities exist for each SPI Port as determined by the GPIO mapping and the user configuration and design.

If the slave device only supports 3-Wire mode, setting `SPIn_MSTR_CFG.three_wire_mode` to 1 puts the SPI Port into 3-Wire mode. For this mode, the corresponding output pin `SDIO[0]` is used for both MOSI and MISO.

Clock Polarity and Phase (Mode Selection)

To select one of the [four supported SPI Modes of operation](#), the `SPIn_MSTR_CFG.spi_mode` field is used. By default, SPI Mode 0 is selected, `spi_mode` = 00b. The upper bit of `spi_mode` is used to control the clock polarity. The figure below shows the two different clock polarity settings and how the settings affect the SPI clock output.

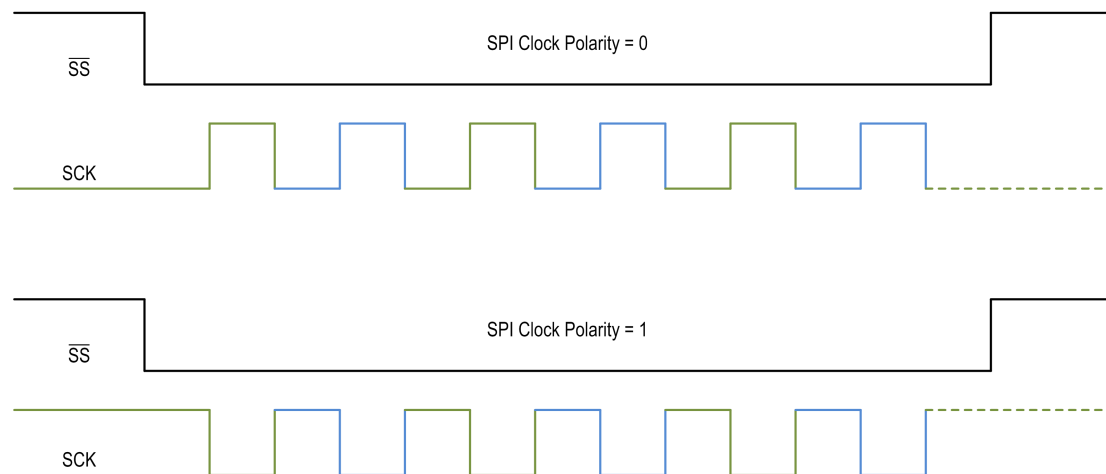


Figure 7.6: SPI Clock Polarity

The low bit of the `spi_mode` field controls the clock phase. The SPI clock phase is used to determine when data is sampled and valid on the MISO/MOSI lines. The default setting is rising edge as shown in the figure below labeled CLK Phase = 0. To set the clock phase to be active on the falling edge of the clock, set the `spi_mode` lower bit to a 1.

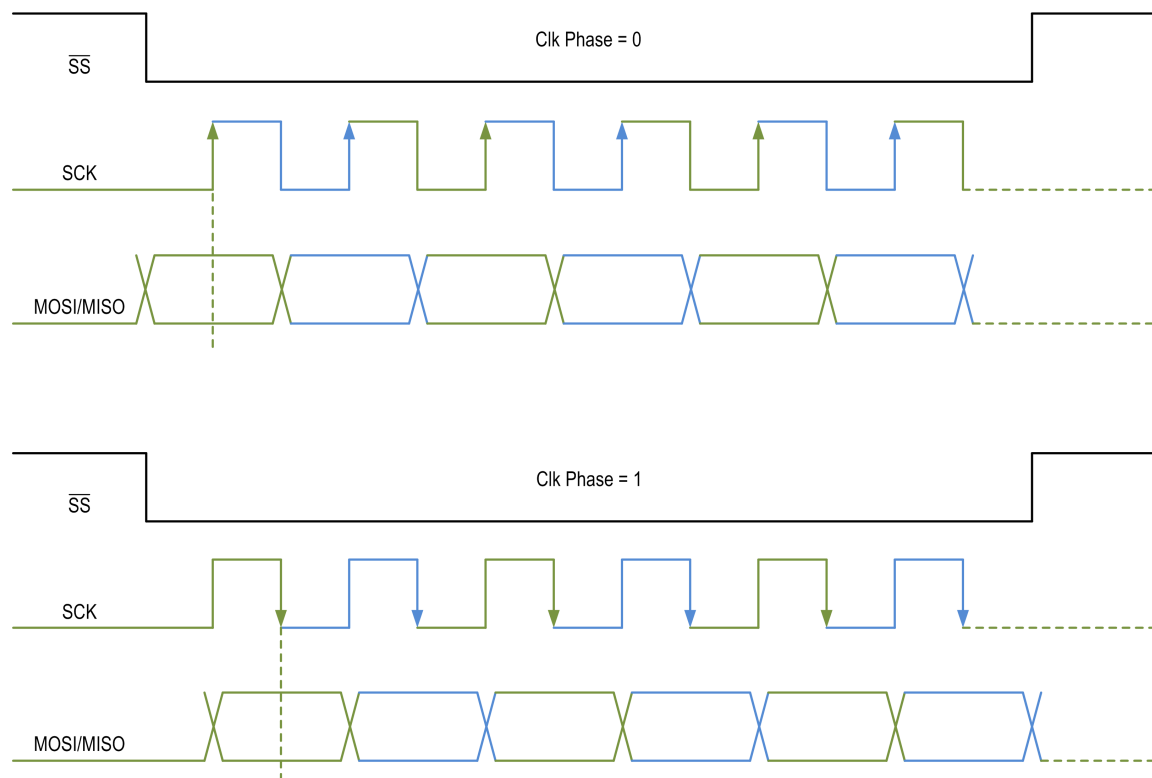


Figure 7.7: SPI Clock Phase

SPI Modes

SPI Mode	spi_mode xxb	SPI Clock State	SPI Sample Clock Edge
0	00	Idle High	Falling Edge
1	01	Idle High	Rising Edge
2	10	Idle Low	Falling edge
3	11	Idle Low	Rising Edge

7.2.5 SPI Fast Mode

Setting up SPI “Fast Mode” is done by setting the [SPIn_MSTR_CFG.sck_hi_clk](#) and [SPIn_MSTR_CFG.sck_lo_clk](#) fields to 0000b. This sets the SPI SCK clock to a gated version of the system clock, based on the SPI Clock Configuration.

Additional configuration options are supported and are detailed in the [SPIn_MSTR_CFG](#) register.

7.2.6 Communication and Data Transfer

Once the SPI has been configured to communicate with a specific slave, SPI transactions are initiated by writing to the SPI Transaction FIFO mapped into the AHB system address map. See [SPI: FIFOs](#) for transmit FIFO details for each SPI port. Prior to initiating a transaction, both the TX and RX SPI FIFOs need to be enabled. To enable the Transmit FIFO, set register field [SPIn_GEN_CTRL.tx_fifo_en](#) for the specific SPI peripheral being used. The Receive FIFO is enabled by setting [SPIn_GEN_CTRL.rx_fifo_en](#) to 1.

The FIFO is 16-bits wide and expects a 16-bit header followed by an optional payload padded out to a 16-bit boundary.

If the transaction generates results data, this data is pushed onto the SPI Results FIFO mapped into the AHB system address map. See [\(SPI: FIFOs\)](#) for receive FIFO details for each SPI port. This FIFO is 8-bits wide and will zero-pad to a byte boundary at the completion of a SPI transaction.

The format of the header is as follows:

SPI Transaction Header

Bit	Mnemonic	Description
[1:0]	Direction	Direction of information transfer with respect to the Master. <ul style="list-style-type: none"> • 0 = None • 1 = TX • 2 = RX • 3 = Both For headers which do not define a transmission (i.e., Direction = None or Rx), no payload is required. Conversely, headers which do not define a reception (i.e., Direction = None or Tx), result in no data being pushed onto the results FIFO.

Bit	Mnemonic	Description
[3:2]	Size Units	Units used to interpret the size field. <ul style="list-style-type: none"> • 0 = Bits • 1 = Bytes • 2 = Pages (See SPIn_MSTR_CFG.page_size for page size definition)
[8:4]	Size	Size of transaction in terms of units. 1 = 1, 2 = 2, ... <i>Note 0 = 32 units</i>
[10:9]	Width	Number of SDIO I/O to use for the transaction. This has no effect on the size of the transaction, just the time required to complete it. <ul style="list-style-type: none"> • <i>Note:</i> • 0 = 1-bit wide • 1 = 2-bits wide • 2 = 4-bits wide
[11]	Alt Timing	RESERVED
[12]	Flow Control	When set to 1, use selected slave flow control input to moderate traffic movement.
[13]	Deassert SS	When set to 1, deassert selected slave select at the completion of this transaction.

7.2.7 Interrupts

Interrupt logic is provided to allow efficient servicing of the SPI Master function by firmware or the [PMU engine](#). Interrupts may be grouped into two categories:

- Keeping the transaction FIFO full.
- Keeping the results FIFO empty. Programmable levels in the FIFO allow interrupt events to be issued if the transaction FIFO falls below a certain level, or if the results FIFO fills above a certain level. See [SPIn_FIFO_CTRL](#) for details.

7.2.8 SPI: FIFOs

SPI Master FIFO AHB Address Ranges

FIFO Write Points for Transaction Setup

SPI[x] FIFO	Start Address	End Address
SPI0_FIFO_TRANS	0x4010_0000	0x4010_07FE
SPI1_FIFO_TRANS	0x4010_1000	0x4010_17FE
SPI2_FIFO_TRANS	0x4010_2000	0x4010_27FE

Writes to this space result in pushes to the SPI Master Transaction FIFO. This space supports single accesses as well as burst accesses. Access widths of 8-bit, 16-bit, and 32-bit are supported. Reads from this space always return zeros. The SPI Master Transaction FIFO is 2-bytes wide. 16-bit writes result in a single push to the FIFO. 32-bit writes result in two FIFO pushes, the LSW is pushed first, followed by the MSW. 8-bit writes must occur in even/odd byte address pairs. The odd byte address data is held by the slave until the even byte address data is written, at which point 2-bytes are pushed to the FIFO.

FIFO Read Points for Results Data

SPI[x] FIFO	Start Address	End Address
SPI0_FIFO_RSLTS	0x4010_0800	0x4010_0FFF
SPI1_FIFO_RSLTS	0x4010_1800	0x4010_1FFF
SPI2_FIFO_RSLTS	0x4010_2800	0x4010_2FFF

Reads from this space pull data from the SPI Master Results FIFO. This space supports single accesses as well as burst accesses. Access widths of 8-bit, 16-bit, and 32-bit are supported. Writes to this space are ignored. The SPI Master Results FIFO is 1-bit wide. 8-bit reads result in a single pull from the FIFO. 16-bit reads result in two FIFO pulls, the LSB is pulled first, followed by the MSB. 32-bit reads result in four FIFO pulls, the LSB is pulled first, continuing until the MSB is pulled last.

7.2.9 Registers (SPI)

7.2.9.1 Module SPI Registers

Address	Register	32b Word Len	Description
0x40030000	SPI0_MSTR_CFG	1	SPI Master 0 Configuration Register
0x40030004	SPI0_SS_SR_POLARITY	1	SPI Master 0 Polarity Control for SS and SR Signals
0x40030008	SPI0_GEN_CTRL	1	SPI Master 0 General Control Register
0x4003000C	SPI0_FIFO_CTRL	1	SPI Master 0 FIFO Control Register
0x40030010	SPI0_SPCL_CTRL	1	SPI Master 0 Special Mode Controls
0x40030014	SPI0_INTFL	1	SPI Master 0 Interrupt Flags

Address	Register	32b Word Len	Description
0x40030018	SPI0_INTEN	1	SPI Master 0 Interrupt Enable/Disable Settings
0x40031000	SPI1_MSTR_CFG	1	SPI Master 1 Configuration Register
0x40031004	SPI1_SS_SR_POLARITY	1	SPI Master 1 Polarity Control for SS and SR Signals
0x40031008	SPI1_GEN_CTRL	1	SPI Master 1 General Control Register
0x4003100C	SPI1_FIFO_CTRL	1	SPI Master 1 FIFO Control Register
0x40031010	SPI1_SPCL_CTRL	1	SPI Master 1 Special Mode Controls
0x40031014	SPI1_INTFL	1	SPI Master 1 Interrupt Flags
0x40031018	SPI1_INTEN	1	SPI Master 1 Interrupt Enable/Disable Settings
0x40032000	SPI2_MSTR_CFG	1	SPI Master 2 Configuration Register
0x40032004	SPI2_SS_SR_POLARITY	1	SPI Master 2 Polarity Control for SS and SR Signals
0x40032008	SPI2_GEN_CTRL	1	SPI Master 2 General Control Register
0x4003200C	SPI2_FIFO_CTRL	1	SPI Master 2 FIFO Control Register
0x40032010	SPI2_SPCL_CTRL	1	SPI Master 2 Special Mode Controls
0x40032014	SPI2_INTFL	1	SPI Master 2 Interrupt Flags
0x40032018	SPI2_INTEN	1	SPI Master 2 Interrupt Enable/Disable Settings
0x40100000	SPI0_FIFO_TRANS	512	SPI Master 0 FIFO Write Space for Transaction Setup
0x40100800	SPI0_FIFO_RSLTS	512	SPI Master 0 FIFO Read Space for Results Data
0x40101000	SPI1_FIFO_TRANS	512	SPI Master 1 FIFO Write Space for Transaction Setup
0x40101800	SPI1_FIFO_RSLTS	512	SPI Master 1 FIFO Read Space for Results Data
0x40102000	SPI2_FIFO_TRANS	512	SPI Master 2 FIFO Write Space for Transaction Setup
0x40102800	SPI2_FIFO_RSLTS	512	SPI Master 2 FIFO Read Space for Results Data

7.2.9.1.1 SPI_n_MSTR_CFG

SPI_n_MSTR_CFG.slave_sel

Field	Bits	Default	Access	Description
slave_sel	2:0	000b	R/W	SPI Slave Select

Selects which SS slave select (out of those which are supported) will be asserted during a SPI transaction.

SPI_n_MSTR_CFG.three_wire_mode

Field	Bits	Default	Access	Description
three_wire_mode	3	0	R/W	3-Wire Mode

- 0: Disabled
- 1: Enabled - SDIO[0] will serve as both MOSI and MISO (half duplex mode)

SPI_n_MSTR_CFG.spi_mode

Field	Bits	Default	Access	Description
spi_mode	5:4	00b	R/W	SPI Mode

Defines Clock Polarity (bit 5) and Clock Phase (bit 4), collectively referred to as SPI Mode.

SPI_n_MSTR_CFG.page_size

Field	Bits	Default	Access	Description
page_size	7:6	00b	R/W	Page Size

Defines number of bytes per page, for transactions that define their length in number of pages.

- 00b: 4 bytes per page
- 01b: 8 bytes per page
- 10b: 16 bytes per page
- 11b: 4 bytes per page

SPI_n_MSTR_CFG.sck_hi_clk

Field	Bits	Default	Access	Description
sck_hi_clk	11:8	0000b	R/W	SCK High Clocks

Number of system clocks SCK will be in the active state (determined by clock polarity setting). If this is set to 0, "Fast Mode" is enabled, in which SCK is a gated version of the system clock.

SPI_n_MSTR_CFG.sck_lo_clk

Field	Bits	Default	Access	Description
sck_lo_clk	15:12	0000b	R/W	SCK Low Clocks

Number of system clocks SCK will be in the INactive state (determined by clock polarity setting). If this is set to 0, "Fast Mode" is enabled, in which SCK is a gated version of the system clock.

SPI_n_MSTR_CFG.act_delay

Field	Bits	Default	Access	Description
act_delay	17:16	00b	R/W	SS Active Timing

Controls the delay from automatic assertion of slave select (SS) to the beginning of the SCK clock pulses as well as the delay from the end of the clock pulses until the automatic deassertion of SS.

- 00b: 0 system clocks
- 01b: 2 system clocks
- 10b: 4 system clocks
- 11b: 8 system clocks

SPIn_MSTR_CFG.inact_delay

Field	Bits	Default	Access	Description
inact_delay	19:18	00b	R/W	SS Inactive Timing

Controls the delay between deassertion of SS at the end of a transaction and reassertion of SS to begin the next transaction, for back-to-back SPI transactions.

- 00b: 0 system clocks
- 01b: 2 system clocks
- 10b: 4 system clocks
- 11b: 8 system clocks

SPIn_MSTR_CFG.alt_sck_hi_clk

Field	Bits	Default	Access	Description
alt_sck_hi_clk	23:20	0000b	R/W	Alt SCK High Clocks

Number of system clocks SCK will be in the active state (determined by clock polarity setting), when alternate timing generation is enabled. If this is set to 0, "Fast Mode" is enabled, in which SCK is a gated version of the system clock.

SPIn_MSTR_CFG.alt_sck_lo_clk

Field	Bits	Default	Access	Description
alt_sck_lo_clk	27:24	0000b	R/W	Alt SCK Low Clocks

Number of system clocks SCK will be in the inactive state (determined by clock polarity setting), when alternate timing generation is enabled. If this is set to 0, "Fast Mode" is enabled, in which SCK is a gated version of the system clock.

7.2.9.1.2 SPI_n_SS_SR_POLARITY**SPI_n_SS_SR_POLARITY.ss_polarity**

Field	Bits	Default	Access	Description
ss_polarity	7:0	00h	R/W	SS Signal Polarity

Defines polarity of each implemented SS slave select signal, where 0=active low, 1=active high.

SPI_n_SS_SR_POLARITY.fc_polarity

Field	Bits	Default	Access	Description
fc_polarity	15:8	00h	R/W	SR Signal Polarity [FC Polarity]

Defines polarity of each implemented SR flow control signal, where 0=active low, 1=active high.

7.2.9.1.3 SPI_n_GEN_CTRL**SPI_n_GEN_CTRL.spi_mstr_en**

Field	Bits	Default	Access	Description
spi_mstr_en	0	0	R/W	Enable/Disable SPI Master

- 0: SPI is disabled and forced to reset state
- 1: SPI is enabled.

SPI_n_GEN_CTRL.tx_fifo_en

Field	Bits	Default	Access	Description
tx_fifo_en	1	0	R/W	Transaction FIFO Enable

- 0: Transaction FIFO is disabled/forced to reset
- 1: Transaction FIFO is enabled

SPI_{IN}_GEN_CTRL.rx_fifo_en

Field	Bits	Default	Access	Description
rx_fifo_en	2	0	R/W	Results FIFO Enable

- 0: Results FIFO is disabled/forced to reset
- 1: Results FIFO is enabled

SPI_{IN}_GEN_CTRL.bit_bang_mode

Field	Bits	Default	Access	Description
bit_bang_mode	3	0	R/W	Bit Bang Mode Enable

- 0: Bit Bang Mode disabled
- 1: Bit Bang Mode enabled

SPI_{IN}_GEN_CTRL.bb_ss_in_out

Field	Bits	Default	Access	Description
bb_ss_in_out	4	0	R/W	Bit Bang SS Input/Output

When written, defines output state of currently selected slave select (Bit Bang Mode only)

When read, returns the current state of the slave select (0=deasserted, 1=asserted)

SPI_{IN}_GEN_CTRL.bb_sr_in

Field	Bits	Default	Access	Description
bb_sr_in	5	0	R/O	Bit Bang SR Input

Writes have no effect.

When read, returns the current state of the flow control (0=deasserted, 1=asserted)

SPI_{IN}_GEN_CTRL.bb_sck_in_out

Field	Bits	Default	Access	Description
bb_sck_in_out	6	0	R/W	Bit Bang SCK Input/Output

When written, defines output state of SPI clock signal SCK (Bit Bang Mode only)

When read, returns the current state of the SCK clock (0=inactive, 1=active)

SPI_{IN}_GEN_CTRL.bb_sdio_in

Field	Bits	Default	Access	Description
bb_sdio_in	11:8	0000b	R/O	Bit Bang SDIO Input

Writes have no effect.

When read, returns the current input state of the SDIO pins

SPI_{IN}_GEN_CTRL.bb_sdio_out

Field	Bits	Default	Access	Description
bb_sdio_out	15:12	0000b	R/W	Bit Bang SDIO Output

Defines output state of SDIO pins (Bit Bang only)

SPI_{IN}_GEN_CTRL.bb_sdio_dr_en

Field	Bits	Default	Access	Description
bb_sdio_dr_en	19:16	0000b	R/W	Bit Bang SDIO Drive Enable

Enables output drive of SDIO pins (Bit Bang only)

7.2.9.1.4 SPI_{IN}_FIFO_CTRL

SPI_{IN}_FIFO_CTRL.tx_fifo_ae_lvl

Field	Bits	Default	Access	Description
tx_fifo_ae_lvl	3:0	15	R/W	Transaction FIFO AE Level

Defines number of unused FIFO entries (words) required to assert Almost Empty flag. FIFO depth is 16 entries.

SPI_{IN}_FIFO_CTRL.tx_fifo_used

Field	Bits	Default	Access	Description
tx_fifo_used	12:8	n/a	R/O	Transaction FIFO Used

Returns number of currently used entries in the Transaction FIFO

SPI_{IN}_FIFO_CTRL.rx_fifo_af_lvl

Field	Bits	Default	Access	Description
rx_fifo_af_lvl	20:16	31	R/W	Results FIFO AF Level

Defines number of used FIFO entries (bytes) required to assert Almost Full flag. FIFO depth is 32 bytes.

SPI_n_FIFO_CTRL.rx_fifo_used

Field	Bits	Default	Access	Description
rx_fifo_used	29:24	n/a	R/O	Results FIFO Used

Returns number of currently used byte entries in the Results FIFO

7.2.9.1.5 SPI_n_SPCL_CTRL

SPI_n_SPCL_CTRL.ss_sample_mode

Field	Bits	Default	Access	Description
ss_sample_mode	0	0	R/W	SS Sample Mode

Enables (when set to 1) the ability to drive SDIO outputs prior to the assertion of Slave Select. This bit should be set when the SPI bus is idle and the transaction FIFO is empty; it will auto-clear when the next slave select assertion occurs.

SPI_n_SPCL_CTRL.miso_fc_en

Field	Bits	Default	Access	Description
miso_fc_en	1	0	R/W	SDIO(1) to SR(0) Mode

When set to 1, routes SDIO(1) input to SR0 for use with devices that use MISO for flow control during writes. Must be cleared to perform reads.

SPI_n_SPCL_CTRL.ss_sa_sdio_out

Field	Bits	Default	Access	Description
ss_sa_sdio_out	7:4	0000b	R/W	SDIO Active Output Value

Defines output mode of SDIO when SS Sample Mode is active.

SPI_n_SPCL_CTRL.ss_sa_sdio_dr_en

Field	Bits	Default	Access	Description
ss_sa_sdio_dr_en	11:8	0000b	R/W	SDIO Active Drive Mode

Defines output drive mode of SDIO when SS Sample Mode is active.

7.2.9.1.6 SPI_n_INTFL

SPI_n_INTFL.tx_stalled

Field	Bits	Default	Access	Description
tx_stalled	0	0	W1C	Transaction Stalled Int Status

Write 1 to clear.

0: Int not active, 1: Interrupt has been triggered Set when transaction FIFO is empty and selected Slave Select is asserted.

SPI_n_INTFL.rx_stalled

Field	Bits	Default	Access	Description
rx_stalled	1	0	W1C	Results Stalled Int Status

Write 1 to clear.

0: Int not active, 1: Interrupt has been triggered Set when results FIFO is full and selected Slave Select is asserted.

SPI_n_INTFL.tx_ready

Field	Bits	Default	Access	Description
tx_ready	2	0	W1C	Transaction Ready Int Status

Write 1 to clear.

0: Int not active, 1: Interrupt has been triggered Set when transaction FIFO is empty and selected Slave Select is deasserted.

SPIn_INTFL.rx_done

Field	Bits	Default	Access	Description
rx_done	3	0	W1C	Results Done Int Status

Write 1 to clear.

0: Int not active, 1: Interrupt has been triggered Set when results FIFO is not empty and selected Slave Select is deasserted.

SPIn_INTFL.tx_fifo_ae

Field	Bits	Default	Access	Description
tx_fifo_ae	4	0	W1C	TXFIFO Almost Empty Int Status

Write 1 to clear.

0: Int not active, 1: Interrupt has been triggered Set when transaction FIFO is in the 'almost empty' state.

SPIn_INTFL.rx_fifo_af

Field	Bits	Default	Access	Description
rx_fifo_af	5	0	W1C	RXFIFO Almost Full Int Status

Write 1 to clear.

0: Int not active, 1: Interrupt has been triggered Set when the results FIFO is in the 'almost full' state.

7.2.9.1.7 SPI_n_INTEN

SPI_n_INTEN.tx_stalled

Field	Bits	Default	Access	Description
tx_stalled	0	0	R/W	Transaction Stalled Int Enable

0: Interrupt source disabled; 1: Interrupt enabled.

SPI_n_INTEN.rx_stalled

Field	Bits	Default	Access	Description
rx_stalled	1	0	R/W	Results Stalled Int Enable

0: Interrupt source disabled; 1: Interrupt enabled.

SPI_n_INTEN.tx_ready

Field	Bits	Default	Access	Description
tx_ready	2	0	R/W	Transaction Ready Int Enable

0: Interrupt source disabled; 1: Interrupt enabled.

SPI_n_INTEN.rx_done

Field	Bits	Default	Access	Description
rx_done	3	0	R/W	Results Done Int Enable

0: Interrupt source disabled; 1:Interrupt enabled.

SPI_n_INTEN.tx_fifo_ae

Field	Bits	Default	Access	Description
tx_fifo_ae	4	0	R/W	TXFIFO Almost Empty Int Enable

0: Interrupt source disabled; 1:Interrupt enabled.

SPI_n_INTEN.rx_fifo_af

Field	Bits	Default	Access	Description
rx_fifo_af	5	0	R/W	RXFIFO Almost Full Int Enable

0: Interrupt source disabled; 1:Interrupt enabled.

7.2.9.1.8 SPI_n_FIFO_TRANS

Default	Access	Description
n/a	R/W	SPI Master 0 FIFO Write Space for Transaction Setup

Writes to this space result in pushes to the SPI Master Transaction FIFO. This write space supports single accesses as well as burst accesses; access widths of 8-bit, 16-bit and 32-bit are supported. Reads from this space always return zeroes.

The SPI Master Transaction FIFO is 16 bits wide and 16 levels deep. Performing a 16-bit write to this space results in a single 16-bit push to the write end of the FIFO. A 32-bit write results in two 16-bit pushes; the least significant word is pushed first, followed by the most significant word. When two pushes occur from a single write, a busy status will be returned by the FIFO to the AHB bus master until both pushes have completed.

If 8-bit writes are used to load the FIFO, these writes must occur in even/odd address pairs. An even byte address write will be held until the corresponding odd byte address write is received, at which point both bytes will be pushed to the FIFO in a single 16-bit push operation (odd byte: MSB, even byte: LSB)

7.2.9.1.9 SPI_n_FIFO_RSLTS

Default	Access	Description
n/a	R/W	SPI Master 0 FIFO Read Space for Results Data

Reads from this space pull data from the SPI Master Results FIFO. This read space supports single accesses as well as burst accesses; access widths of 8-bit, 16-bit and 32-bit are supported. Writes to this space are ignored.

The SPI Master Results FIFO is 8 bits wide and 32 levels deep. Reading an 8-bit value from this space results in a single pull from the read end of the FIFO. Reading a 16-bit value results in two byte pulls (b0 and b1) from the read end of the FIFO; the resulting 16-bit return value is b1:b0 where b1 is the MSB and b0 is the LSB. Reading a 32-bit value results in four byte pulls (b0, b1, b2, b3) from the read end of the FIFO; the resulting 32-bit return value is b3:b2:b1:b0 where b3 is the MSB and b0 is the LSB.

7.3 UART

7.3.1 Overview

The **MAX32600** provides two UART ports which can be used to communicate with external devices requiring an asynchronous serial protocol. Features of the **MAX32600** UARTs:

- Flexible baud rate generation based on the module clock frequency (equal to the system clock source or a subdivide of the system clock source)
- Programmable word size (5- to 8-bits), stop bits, and parity settings
- Automatic parity and framing error detection
- Automatic flow control can be enabled for RTS and CTS lines
- 8-byte FIFO in both directions (separate read/RX and write/TX FIFOs)
- Interrupts available for frame error, parity error, CTS, RX FIFO overrun, and FIFO full/partially full conditions

7.3.2 UART Port and Pin Configurations

UART Wiring Configuration

- **RX, TX, CTS, RTS**

Note See [Pin Layout](#) for a detailed mapping of **MAX32600** multiplexed function locations. Functional priority distinction is included in the mapping.

7.3.2.1 Compact Layout Configuration

The tables below contains the available pin configurations for each of the UART ports (UART0 and UART1):

UART0

Logic Signal	Port and Pin
RX	A) P1.0 B) P2.0 D) P0.0
TX	A) P1.1 B) P2.1 D) P0.1
CTS	A) P1.2 B) P2.4 D) P0.2
RTS	A) P1.3 B) P2.5 D) P0.3

UART1

Logic Signal	Port and Pin
RX	A) P1.2 B) P2.4 D) P1.6
TX	A) P1.3 B) P2.5 D) P1.7
CTS	A) P1.6 B) P2.6
RTS	A) P1.7 B) P2.7

7.3.2.2 Standard Layout Configuration

The tables below contains the available pin configurations for each of the UART ports (UART0 and UART1):

UART0

Logic Signal	Port and Pin
RX	A) P1.0 B) P2.0 C) P7.0 D) P0.0
TX	A) P1.1 B) P2.1 C) P7.1 D) P0.1
CTS	A) P1.2 B) P1.4 C) P7.2 D) P0.2

Logic Signal	Port and Pin
RTS	A) P1.3 B) P1.5 C) P7.3 D) P0.3

UART1

Logic Signal	Port and Pin
RX	A) P1.2 B) P1.4 C) P7.2 D) P2.6
TX	A) P1.3 B) P1.5 C) P7.3 D) P2.7
CTS	A) P2.6 B) P1.6 C) P7.6
RTS	A) P2.7 B) P1.7 C) P7.7

7.3.3 Port Register Blocks

Each UART instance is controlled by a block of registers assigned to that port; all blocks contain identical control, interrupt, status, and FIFO read/write registers. The addresses for each register block are shown in the table below.

Registers for UART0

Address	Register	Details
0x40038000	UART0_CTRL	UART Control Register
0x40038004	UART0_STATUS	UART Status Register
0x40038008	UART0_INTEN	Interrupt Enable/Disable Controls
0x4003800C	UART0_INTFL	Interrupt Flags
0x40038010	UART0_BAUD_INT	Baud Rate Setting (Integer Portion)
0x40038014	UART0_BAUD_DIV_128	Baud Rate Setting (Div 128 Decimal Portion)
0x40038018	UART0_TX_FIFO_OUT	TX FIFO Output End (read-only)
0x4003801C	UART0_HW_FLOW_CTRL	Hardware Flow Control Register
0x40038020	UART0_TX_RX_FIFO	Write to load TX FIFO, Read to unload RX FIFO

Registers for UART1

Address	Register	Details
0x40039000	UART1_CTRL	UART Control Register
0x40039004	UART1_STATUS	UART Status Register
0x40039008	UART1_INTEN	Interrupt Enable/Disable Controls
0x4003900C	UART1_INTFL	Interrupt Flags
0x40039010	UART1_BAUD_INT	Baud Rate Setting (Integer Portion)
0x40039014	UART1_BAUD_DIV_128	Baud Rate Setting (Div 128 Decimal Portion)
0x40039018	UART1_TX_FIFO_OUT	TX FIFO Output End (read-only)
0x4003901C	UART1_HW_FLOW_CTRL	Hardware Flow Control Register
0x40039020	UART1_TX_RX_FIFO	Write to load TX FIFO, Read to unload RX FIFO

7.3.4 UART Clock Selection and Clock Gating

The UARTs in the **MAX32600** use the System Core Clock. To enable a specific UART, it is necessary enable clocks for the port. In addition, the baud clock enable bit must be set in [UARTn_CTRL.baud_clk_en](#) to allow baud clock generation (required for transmit only). There are two options that will ensure the UART clocks are enabled: Dynamic Clock Gating and turning the clock on without clock gating. To enable the UART clocks for a given port, the fields [CLKMAN_CLK_GATE_CTRL0.uart0_clk_gater](#) or [CLKMAN_CLK_GATE_CTRL0.uart1_clk_gater](#) should be set as shown in the table below.

UART Clock Control (2b)	Setting
00b	Clock off - UART disabled
01b	Dynamic Clock Gating Enabled - UART clock active only when used
10b or 11b	Clock on - UART enabled at all times

7.3.5 Format and Baud Rate Selection

The overall baud rate for the UART is based on the nominal peripheral clock frequency, which is derived from the main system clock as controlled by the Clock Manager. To set the baud rate value, write the appropriate divider values to the registers [UARTn_BAUD_INT](#) and [UARTn_BAUD_DIV_128](#) for the UART instance to be configured.

The formulas for calculating the `UARTn_BAUD_INT` and `UARTn_BAUD_DIV_128` are:

$$div = \frac{\text{System Core Clock}}{\left(\frac{\text{DESIRED BAUD RATE}}{100} * 128\right)}$$

- `UARTn_BAUD_INT` is equal to the integer portion of `div`.

$$UARTn_BAUD_INT = \frac{div}{100}$$

- `UARTn_BAUD_DIV_128` is the decimal portion of `div` converted to an integer.

$$UARTn_BAUD_DIV_128 = \frac{((div - UARTn_BAUD_INT) * 100) * 128}{100}$$

Note The PLL output must be selected as the system clock source for proper UART operation. Using the crypto ring or the nano ring as the clock source will not provide sufficient clock accuracy to generate a stable baud rate.

The number of stop bits used and parity calculation mode are all set by writing to the appropriate bits and bit fields in the `UARTn_CTRL` register.

- `UARTn_CTRL.parity_enable` is used to enable parity.
- `UARTn_CTRL.parity_mode` is used to set even or odd parity.
- `UARTn_CTRL.parity_bias` is used to determine if 0s or 1s are used for parity determination.
- `UARTn_CTRL.stop_bit_mode` is used to set the number of stop bits.

To set the transfer size (character length), the `UARTn_CTRL.char_length` field must be set.

UART Character Size (2b)	Setting
00b	5-bit character
01b	6-bit character
01b	7-bit character
11b	8-bit character

7.3.6 Transferring and Receiving Data

Data to be transferred must be written to the TX FIFO by writing to the [UARTn_TX_RX_FIFO](#) register (either directly or using a PMU channel). This data will be transferred out by the hardware automatically a character at a time in the order that it was written to the FIFO. The status flags and associated UART interrupts can be used to monitor the FIFO status and determine when the transfer cycle or cycles have completed.

As data is received, it is loaded into the RX FIFO, and it can then be unloaded by reading from the [UARTn_TX_RX_FIFO](#) register. If eight characters are received and are not unloaded by the CPU or PMU controller, and another character arrives on RX, an overrun error will occur and the new character will be lost.

Additionally, the TX and RX FIFOs can be flushed manually. To flush the TX FIFO, write a 1 to the [UARTn_CTRL.tx_fifo_flush](#). To flush the RX FIFO, write a 1 to the [UARTn_CTRL.rx_fifo_flush](#).

7.3.7 UART: Interrupts

Interrupts can be generated by each UART instance for one or more of the following conditions:

- A framing error occurs on a received character (RX transitions out of sync to the baud clock which is synchronized to the initial RX falling edge).
- The received character has an invalid parity bit according to the chosen parity settings.
- The level on the CTS line changes (when hardware flow control is disabled).
- The number of bytes in the FIFO RX buffer reaches a configurable threshold level.
- An overrun error occurs on the FIFO RX buffer.
- The FIFO TX buffer reaches a half-empty level.
- The FIFO TX buffer has only one byte remaining.

7.3.8 Hardware Flow Control

When hardware flow control is enabled, the CTS and RTS lines are controlled and sampled directly by hardware. With hardware flow control disabled, the bits in the UART Pin Control register can be used to control CTS and RTS manually.

7.3.8.1 Clear to Send

The CTS (Clear to Send) line is an input to the UART that is driven by an external device. It is used by the external device to notify the UART whether or not the external device is ready to receive additional data. With hardware flow control enabled, the CTS input will automatically pause transmission from the TX FIFO when CTS is driven high (active low input). The UART samples CTS just before a new character is transmitted from the TX FIFO. If CTS is low, the character transmission continues. If CTS is high, the UART pauses and waits for CTS to return to a low level before continuing the transmission.

7.3.8.2 Ready to Send

The RTS (Ready to Send) line is an output from the UART that notifies an external device whether or not the UART is ready to receive more data. When hardware flow control is enabled, the RTS output is automatically driven low (active state) whenever the RX FIFO on the UART is not full. When the RX FIFO becomes full, the RTS output will be driven high to indicate to the external device that it should pause its data transmission until the UART has more buffer space available to receive data.

7.3.9 Registers (UART)

7.3.9.1 Module UART Registers

Address	Register	32b Word Len	Description
0x40038000	UART0_CTRL	1	UART 0 Control Register
0x40038004	UART0_STATUS	1	UART 0 Status Register
0x40038008	UART0_INTEN	1	UART 0 Interrupt Enable/Disable Controls
0x4003800C	UART0_INTFL	1	UART 0 Interrupt Flags
0x40038010	UART0_BAUD_INT	1	UART 0 Baud Rate Setting (Integer Portion)
0x40038014	UART0_BAUD_DIV_128	1	UART 0 Baud Rate Setting (Div 128 Decimal Portion)
0x40038018	UART0_TX_FIFO_OUT	1	UART 0 TX FIFO Output End (read-only)
0x4003801C	UART0_HW_FLOW_CTRL	1	UART 0 Hardware Flow Control Register
0x40038020	UART0_TX_RX_FIFO	1	UART 0 Write to load TX FIFO, Read to unload RX FIFO
0x40039000	UART1_CTRL	1	UART 1 Control Register
0x40039004	UART1_STATUS	1	UART 1 Status Register
0x40039008	UART1_INTEN	1	UART 1 Interrupt Enable/Disable Controls
0x4003900C	UART1_INTFL	1	UART 1 Interrupt Flags
0x40039010	UART1_BAUD_INT	1	UART 1 Baud Rate Setting (Integer Portion)
0x40039014	UART1_BAUD_DIV_128	1	UART 1 Baud Rate Setting (Div 128 Decimal Portion)
0x40039018	UART1_TX_FIFO_OUT	1	UART 1 TX FIFO Output End (read-only)
0x4003901C	UART1_HW_FLOW_CTRL	1	UART 1 Hardware Flow Control Register
0x40039020	UART1_TX_RX_FIFO	1	UART 1 Write to load TX FIFO, Read to unload RX FIFO

7.3.9.1.1 UARTn_CTRL

UARTn_CTRL.rx_threshold

Field	Bits	Default	Access	Description
rx_threshold	2:0	100b	R/W	Receive FIFO Interrupt Threshold

Specifies the depth of receive FIFO that triggers an interrupt. Valid settings are 1-7.

UARTn_CTRL.parity_enable

Field	Bits	Default	Access	Description
parity_enable	4	0	R/W	Parity Enable

- 0:Disable parity
- 1:Enable parity

UARTn_CTRL.parity_mode

Field	Bits	Default	Access	Description
parity_mode	5	0	R/W	Parity Mode Select

- 0:Even parity mode selected
- 1:Odd parity mode selected

UARTn_CTRL.parity_bias

Field	Bits	Default	Access	Description
parity_bias	6	0	R/W	Parity Basis Select

- 0:Parity is based on number of 1 bits in the frame.
- 1:Parity is based on number of 0 bits in the frame.

UARTn_CTRL.tx_fifo_flush

Field	Bits	Default	Access	Description
tx_fifo_flush	8	0	W/O	Transmit FIFO Flush

Write to 1 to flush the transmit FIFO.

This bit is cleared to zero by hardware after the flush operation has completed.

UARTn_CTRL.rx_fifo_flush

Field	Bits	Default	Access	Description
rx_fifo_flush	9	0	W/O	Receive FIFO Flush

Write to 1 to flush the receive FIFO.

This bit is cleared to zero by hardware after the flush operation has completed.

UARTn_CTRL.char_length

Field	Bits	Default	Access	Description
char_length	11:10	00b	R/W	Character Transfer Bit Length

- 00:5-bit character transfer

- 01:6-bit character transfer
- 10:7-bit character transfer
- 11:8-bit character transfer

UARTn_CTRL.stop_bit_mode

Field	Bits	Default	Access	Description
stop_bit_mode	12	0	R/W	Stop Bit Mode Select

- 0:One stop bit is generated.
- 1:Either 1.5 or 2 stop bits are generated.

UARTn_CTRL.hw_flow_ctrl_en

Field	Bits	Default	Access	Description
hw_flow_ctrl_en	13	0	R/W	Hardware Flow Control Enable

- 0:Hardware flow control (RTS/CTS) is disabled.
- 1:Hardware flow control (RTS/CTS) is enabled.

UARTn_CTRL.baud_clk_en

Field	Bits	Default	Access	Description
baud_clk_en	14	0	R/W	Baud Clock Generation Enable

- 0:Baud clock generation is disabled.

- 1: Baud clock generation enabled.

7.3.9.1.2 UARTn_STATUS

UARTn_STATUS.tx_busy

Field	Bits	Default	Access	Description
tx_busy	0	0	R/O	TX Busy

1: The UART is transmitting data.

UARTn_STATUS.rx_busy

Field	Bits	Default	Access	Description
rx_busy	1	0	R/O	RX Busy

1: The UART is receiving data.

UARTn_STATUS.rx_fifo_empty

Field	Bits	Default	Access	Description
rx_fifo_empty	4	1	R/O	RX FIFO Empty

1: The Receive FIFO is empty.

UARTn_STATUS.rx_fifo_full

Field	Bits	Default	Access	Description
rx_fifo_full	5	0	R/O	RX FIFO Full

1: The Receive FIFO is full.

UARTn_STATUS.tx_fifo_empty

Field	Bits	Default	Access	Description
tx_fifo_empty	6	1	R/O	TX FIFO Empty

1: The Transmit FIFO is empty.

UARTn_STATUS.tx_fifo_full

Field	Bits	Default	Access	Description
tx_fifo_full	7	0	R/O	TX FIFO Full

1: The Transmit FIFO is full.

UARTn_STATUS.rx_fifo_chars

Field	Bits	Default	Access	Description
rx_fifo_chars	11:8	0	R/O	RX FIFO Chars Used

Number of characters currently in the RX FIFO.

UARTn_STATUS.tx_fifo_chars

Field	Bits	Default	Access	Description
tx_fifo_chars	15:12	0	R/O	TX FIFO Chars Used

Number of characters currently in the TX FIFO.

7.3.9.1.3 UARTn_INTEN**UARTn_INTEN.rx_frame_error**

Field	Bits	Default	Access	Description
rx_frame_error	0	0	R/W	RX Frame Error Interrupt Enable

0:Interrupt is disabled, 1:Interrupt is enabled

UARTn_INTEN.rx_parity_error

Field	Bits	Default	Access	Description
rx_parity_error	1	0	R/W	RX Parity Error Interrupt Enable

0:Interrupt is disabled, 1:Interrupt is enabled

UARTn_INTEN.cts_change

Field	Bits	Default	Access	Description
cts_change	2	0	R/W	CTS Value Change Interrupt Enable

0:Interrupt is disabled, 1:Interrupt is enabled

UARTn_INTEN.rx_overrun

Field	Bits	Default	Access	Description
rx_overrun	3	0	R/W	RX Overrun Interrupt Enable

0:Interrupt is disabled, 1:Interrupt is enabled

UARTn_INTEN.rx_over_threshold

Field	Bits	Default	Access	Description
rx_over_threshold	4	0	R/W	RX FIFO Over Threshold Int Enable

0:Interrupt is disabled, 1:Interrupt is enabled

UARTn_INTEN.tx_almost_empty

Field	Bits	Default	Access	Description
tx_almost_empty	5	0	R/W	TX FIFO Almost Empty Int Enable

0:Interrupt is disabled, 1:Interrupt is enabled

UARTn_INTEN.tx_half_empty

Field	Bits	Default	Access	Description
tx_half_empty	6	0	R/W	TX FIFO Half Empty Int Enable

0:Interrupt is disabled, 1:Interrupt is enabled

7.3.9.1.4 UARTn_INTFL**UARTn_INTFL.rx_frame_error**

Field	Bits	Default	Access	Description
rx_frame_error	0	0	W0C	RX Frame Error Interrupt Status

Write 0 to clear.

Set to 1 by hardware when int condition occurs.

UARTn_INTFL.rx_parity_error

Field	Bits	Default	Access	Description
rx_parity_error	1	0	W0C	RX Parity Error Interrupt Status

Write 0 to clear.

Set to 1 by hardware when int condition occurs.

UARTn_INTFL.cts_change

Field	Bits	Default	Access	Description
cts_change	2	0	W0C	CTS Value Change Interrupt Status

Write 0 to clear.

Set to 1 by hardware when int condition occurs.

UARTn_INTFL.rx_overrun

Field	Bits	Default	Access	Description
rx_overrun	3	0	W0C	RX Overrun Interrupt Status

Write 0 to clear.

Set to 1 by hardware when int condition occurs.

UARTn_INTFL.rx_over_threshold

Field	Bits	Default	Access	Description
rx_over_threshold	4	0	W0C	RX FIFO Over Threshold Int Status

Write 0 to clear.

Set to 1 by hardware when int condition occurs.

UARTn_INTFL.tx_almost_empty

Field	Bits	Default	Access	Description
tx_almost_empty	5	0	W0C	TX FIFO Almost Empty Int Status

Write 0 to clear.

Set to 1 by hardware when int condition occurs.

UARTn_INTFL.tx_half_empty

Field	Bits	Default	Access	Description
tx_half_empty	6	0	W0C	TX FIFO Half Empty Int Status

Write 0 to clear.

Set to 1 by hardware when int condition occurs.

7.3.9.1.5 UARTn_BAUD_INT

UARTn_BAUD_INT.fbaud

Field	Bits	Default	Access	Description
fbaud	11:0	0	R/W	Integer portion of baudrate DIV.

Fbaud=(UART clock / (128 * DIV))

7.3.9.1.6 UARTn_BAUD_DIV_128**UARTn_BAUD_DIV_128.div**

Field	Bits	Default	Access	Description
div	6:0	0	R/W	Decimal portion of baudrate DIV.

$DIV = \text{UARTn_BAUD_INT}[11:0] + (\text{UARTn_BAUD_DIV_128}[6:0]/128)$

7.3.9.1.7 UARTn_TX_FIFO_OUT**UARTn_TX_FIFO_OUT.tx_fifo**

Field	Bits	Default	Access	Description
tx_fifo	7:0	n/a	R/O	TX FIFO Output

Writes have no effect.

Reading from this register returns the current value at the output of the TX FIFO, without changing the contents of the TX FIFO.

7.3.9.1.8 UARTn_HW_FLOW_CTRL**UARTn_HW_FLOW_CTRL.cts_input**

Field	Bits	Default	Access	Description
cts_input	0	1	R/O	CTS Input Value

Returns the current value of the CTS I/O signal.

UARTn_HW_FLOW_CTRL.rts_output

Field	Bits	Default	Access	Description
rts_output	1	1	R/W	RTS Output Value

This bit is only used when hardware flow control is disabled.

- 0: RTS I/O is set to low level.
- 1: RTS I/O is set to high level.

7.3.9.1.9 UARTn_TX_RX_FIFO

UARTn_TX_RX_FIFO.fifo_data

Field	Bits	Default	Access	Description
fifo_data	7:0	n/a	R/W	TX/RX FIFO Data

Writing to this register loads a character into the TX FIFO.

Reading from this register unloads the next character from the RX FIFO and returns its value.

UARTn_TX_RX_FIFO.parity_error

Field	Bits	Default	Access	Description
parity_error	8	n/a	R/O	Parity Error Flag

When reading, if a parity error occurred during the reception of the character being returned, this bit is set to 1; otherwise, set to 0.

7.4 USB Device Interface

7.4.1 Overview

The **MAX32600** includes a Universal Serial Bus (USB) peripheral dedicated to device operation. The peripheral is a USB 2.0 compliant, full-speed device and includes a Serial Interface Engine (SIE) that connects to the internal USB transceiver and pin drivers.

7.4.2 Operation

The USB device controller supports a total of eight endpoints with programmable configuration.

7.4.2.1 USB Reset Definitions

The USB device is reset under the following conditions:

- **USB Bus Reset:** Host issues a bus reset.
- **USB Device Reset:** Firmware changes the [USB_CN.usb_en](#) bit from 0, disabled, to 1, enabled.
- **System Reset:** System undergoing a reset.

Note In this document a reset condition without any qualifier refers to a System Reset. The term *USB reset* refers to either a USB Bus Reset or a USB Device Reset.

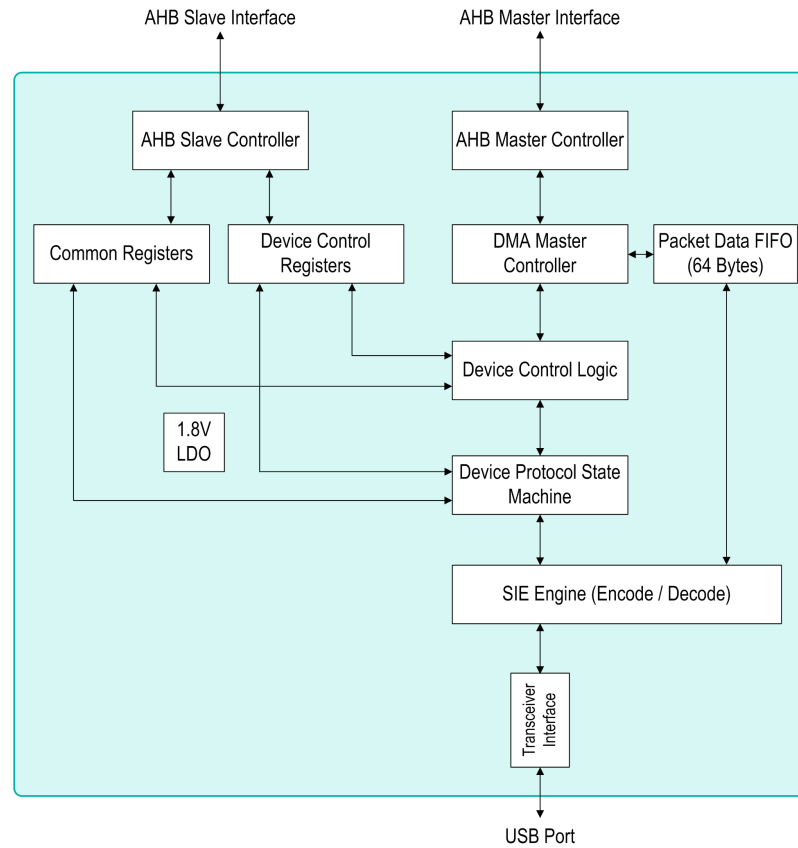


Figure 7.8: USB Device Block Diagram

7.4.3 USB Endpoints

The **MAX32600** supports eight USB endpoints that can be individually configured.

Each endpoint supports:

- Single / Double buffer
- Programmable buffer starting address
- Programmable interrupt generation
- Ability to STALL a packet
- Bulk and Interrupt transfer
- Control transfer (endpoint 0 only)
- Configurable response to Status Stage of Control transfer.

Each endpoint includes:

- Endpoint Control Register
- Endpoint Buffer Descriptor
- Endpoint Buffer Address

7.4.3.1 Endpoint Control Register

The USB endpoint control registers `USB_EP` are used to define the endpoint general characteristics.

- Disable an endpoint `ep_dir = 0`
- Direction of packet transfer `ep_dir`
 - Endpoint 0 is the only CONTROL mode endpoint
 - All other Endpoints can be set to IN or OUT
- Single or double buffering, `ep_buf2`
- Reset data buffer to 0 for double buffered streams `ep_dt`
 - Write to 1 resets the current buffer to 0; writes to 0 are ignored. Cleared to 0 by hardware after the data clear / reset process has been completed.
- Interrupt generation

- Enable for IN or OUT data transfers, [ep_int_en](#)
- Enable for NAK handshakes sent to host [ep_nak_en](#)
- Endpoint STALL status [ep_stall](#)
- Send a STALL to host for status stage [ep_st_stall](#)
- Send an ACK to the host for status stage [ep_st_ack](#)

7.4.3.2 Endpoint Buffer Descriptor

The endpoint buffer descriptor is used as a communication port between the application firmware and the SIE in system memory. The endpoint buffer descriptor starting address is specified by the USB endpoint descriptor base address register [USB_EP_BASE.ep_base](#).

The endpoint data toggle value and the buffer are maintained internally by the SIE. The data toggle value is initialized to DATA0 upon USB reset. The buffer will also be reset to buffer 0 upon USB reset. When an endpoint is double-buffered, the SIE will use the two buffers in ping-pong fashion. Firmware can reset the active data buffer by writing 1 to [ep_dt](#).

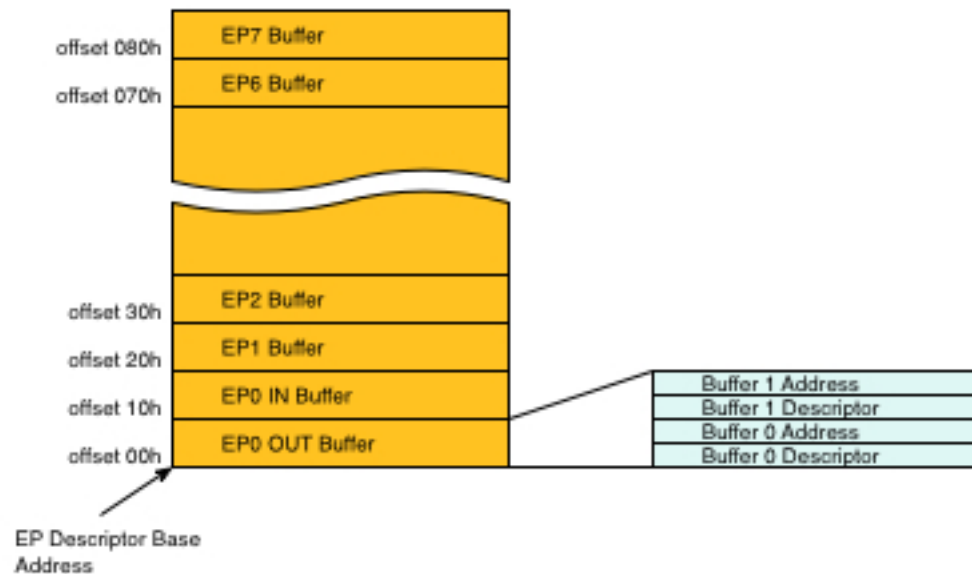


Figure 7.9: USB Endpoint Buffer Descriptor Memory

7.4.4 Registers (USB)

7.4.4.1 Module USB Registers

Address	Register	32b Word Len	Description
0x4010C000	USB_CN	1	USB Control Register
0x4010C200	USB_DEV_ADDR	1	USB Device Address Register
0x4010C204	USB_DEV_CN	1	USB Device Control Register
0x4010C208	USB_DEV_INTFL	1	USB Device Interrupt
0x4010C20C	USB_DEV_INTEN	1	USB Device Interrupt Enable
0x4010C220	USB_EP_BASE	1	USB Endpoint Descriptor Table Base Address
0x4010C224	USB_CUR_BUF	1	USB Current Endpoint Buffer Register
0x4010C228	USB_IN_OWNER	1	USB IN Endpoint Buffer Owner Register
0x4010C22C	USB_OUT_OWNER	1	USB OUT Endpoint Buffer Owner Register
0x4010C230	USB_IN_INT	1	USB IN Endpoint Buffer Available Interrupt
0x4010C234	USB_OUT_INT	1	USB OUT Endpoint Data Available Interrupt
0x4010C238	USB_NAK_INT	1	USB IN Endpoint NAK Interrupt
0x4010C23C	USB_DMA_ERR_INT	1	USB DMA Error Interrupt
0x4010C240	USB_BUF_OVR_INT	1	USB Buffer Overflow Interrupt
0x4010C260	USB_SETUP0	1	USB SETUP Packet Bytes 0 to 3
0x4010C264	USB_SETUP1	1	USB SETUP Packet Bytes 4 to 7
0x4010C280	USB_EP	8	USB Endpoint[n] Control Register

7.4.4.1.1 USB_CN

USB_CN.usb_en

Field	Bits	Default	Access	Description
usb_en	0	0	R/W	USB Device Interface Enable

Enabling the USB Device interface (setting this bit from 0 to 1) causes a reset of the USB Device internal state.

- 0: Disabled
- 1: Enabled

USB_CN.host

Field	Bits	Default	Access	Description
host	1	0	R/W	USB Host Mode Select

Reserved field; only Device mode is implemented.

7.4.4.1.2 USB_DEV_ADDR

USB_DEV_ADDR.dev_addr

Field	Bits	Default	Access	Description
dev_addr	6:0	0000000b	R/O	USB Device Address

Set by the USB SIE hardware during host enumeration as the result of a SetAddress() request.

7.4.4.1.3 USB_DEV_CN

USB_DEV_CN.sigrwu

Field	Bits	Default	Access	Description
sigrwu	2	0	R/W	USB Signal Remote Wakeup

- 0: Do not send remote wakeup signal
- 1: Send remote wakeup signal to USB bus host.

USB_DEV_CN.connect

Field	Bits	Default	Access	Description
connect	3	0	R/W	Connect to USB

- 0: Disconnect internal pullup resistor between DPLUS and VBUS.
- 1: Connect internal pullup resistor between DPLUS and VBUS.

USB_DEV_CN.ulpm

Field	Bits	Default	Access	Description
ulpm	4	0	R/W	USB Low Power Mode

- 0: USB transceiver in normal operation
- 1: USB transceiver will enter a low power mode

USB_DEV_CN.urst

Field	Bits	Default	Access	Description
urst	5	0	R/W	USB Device Controller Reset

- 0: USB device controller is released to run normally.
- 1: USB device controller is held in reset (until this bit is cleared back to 0 by firmware)

USB_DEV_CN.vbgate

Field	Bits	Default	Access	Description
vbgate	6	0	R/W	VBUS Gate

- 0: CONNECT operation is independent of VBUS status
- 1: CONNECT operation is performed conditionally depending on presence of VBUS voltage

USB_DEV_CN.oscen

Field	Bits	Default	Access	Description
oscen	7	0	R/W	OSC En

Reserved for test; do not modify the value of this bit.

USB_DEV_CN.bact_oe

Field	Bits	Default	Access	Description
bact_oe	8	0	R/W	BACT OE

Reserved for test; do not modify the value of this bit.

USB_DEV_CN.fifo_mode

Field	Bits	Default	Access	Description
fifo_mode	9	0	R/W	FIFO Mode

This register bit, when set, configures the device controller to respond to an incoming IN request as soon as the device controller's FIFO become non-empty, instead of waiting for the full packet to be received by the FIFO. Setting this bit to 1 reduces the likelihood of returning NAK. This is the recommended setting.

7.4.4.1.4 USB_DEV_INTFL**USB_DEV_INTFL.dpact**

Field	Bits	Default	Access	Description
dpact	0	0	W1C	DPLUS Activity Interrupt Flag

This interrupt flag indicates that activity has been detected on the DPLUS (D+) USB interface pin. Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.rwu_dn

Field	Bits	Default	Access	Description
rwu_dn	1	0	W1C	Remote Wakeup Done Interrupt Flag

This interrupt flag indicates that the remote wakeup signalling to the USB bus host has been completed. Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.bact

Field	Bits	Default	Access	Description
bact	2	0	W1C	USB Bus Activity Interrupt Flag

This interrupt flag indicates that USB bus activity has been detected (the USB controller has received a SYNC field). Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.brst

Field	Bits	Default	Access	Description
brst	3	0	W1C	USB Bus Reset In Progress Interrupt Flag

This interrupt flag indicates that a USB bus reset condition has been detected, which causes all USB registers to be cleared to their default states. Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.susp

Field	Bits	Default	Access	Description
susp	4	0	W1C	USB Suspend Interrupt Flag

This interrupt flag indicates that a USB bus suspend condition has been detected. Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.no_vbus

Field	Bits	Default	Access	Description
no_vbus	5	0	W1C	No VBUS Interrupt Flag

This interrupt flag indicates that VBUS has powered down (level transition from 1 to 0). Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.vbus

Field	Bits	Default	Access	Description
vbus	6	0	W1C	VBUS Detect Interrupt Flag

This interrupt flag indicates that VBUS has powered up (level transition from 0 to 1).

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.brst_dn

Field	Bits	Default	Access	Description
brst_dn	7	0	W1C	USB Bus Reset Completed Interrupt Flag

This interrupt flag indicates that a USB bus reset condition has completed.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.setup

Field	Bits	Default	Access	Description
setup	8	0	W1C	Setup Packet Interrupt Flag

This interrupt flag indicates that a SETUP packet has been received by Endpoint 0.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.ep_in

Field	Bits	Default	Access	Description
ep_in	9	0	W1C	Endpoint IN Interrupt Flag

This interrupt flag indicates that an Endpoint IN Interrupt is pending at one or more endpoints.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.ep_out

Field	Bits	Default	Access	Description
ep_out	10	0	W1C	Endpoint OUT Interrupt Flag

This interrupt flag indicates that an Endpoint OUT Interrupt is pending at one or more endpoints. Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.ep_nak

Field	Bits	Default	Access	Description
ep_nak	11	0	W1C	Endpoint NAK Interrupt Flag

This interrupt flag indicates that an Endpoint NAK Interrupt is pending at one or more endpoints. Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.dma_err

Field	Bits	Default	Access	Description
dma_err	12	0	W1C	DMA Error Interrupt Flag

This interrupt flag indicates that a DMA error has been detected by one or more endpoints. Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.buf_ovr

Field	Bits	Default	Access	Description
buf_ovr	13	0	W1C	Buffer Overflow Interrupt Flag

This interrupt flag indicates that a buffer overflow error has been detected by one or more endpoints.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.vbus_st

Field	Bits	Default	Access	Description
vbus_st	16	0	R/O	VBUS Status

This status flag indicates the current VBUS level (0-low, 1-high).

7.4.4.1.5 USB_DEV_INTEN

USB_DEV_INTEN.dpact

Field	Bits	Default	Access	Description
dpact	0	0	R/W	DPLUS Activity Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.rwu_dn

Field	Bits	Default	Access	Description
rwu_dn	1	0	R/W	Remote Wakeup Done Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.bact

Field	Bits	Default	Access	Description
bact	2	0	R/W	USB Bus Activity Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.brst

Field	Bits	Default	Access	Description
brst	3	0	R/W	USB Bus Reset In Progress Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.susp

Field	Bits	Default	Access	Description
susp	4	0	R/W	USB Suspend Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.no_vbus

Field	Bits	Default	Access	Description
no_vbus	5	0	R/W	No VBUS Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.vbus

Field	Bits	Default	Access	Description
vbus	6	0	R/W	VBUS Detect Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.brst_dn

Field	Bits	Default	Access	Description
brst_dn	7	0	R/W	USB Bus Reset Completed Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.setup

Field	Bits	Default	Access	Description
setup	8	0	R/W	Setup Packet Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.ep_in

Field	Bits	Default	Access	Description
ep_in	9	0	R/W	Endpoint IN Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.ep_out

Field	Bits	Default	Access	Description
ep_out	10	0	R/W	Endpoint OUT Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.ep_nak

Field	Bits	Default	Access	Description
ep_nak	11	0	R/W	Endpoint NAK Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.dma_err

Field	Bits	Default	Access	Description
dma_err	12	0	R/W	DMA Error Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.buf_ovr

Field	Bits	Default	Access	Description
buf_ovr	13	0	R/W	Buffer Overflow Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

7.4.4.1.6 USB_EP_BASE

USB_EP_BASE.ep_base

Field	Bits	Default	Access	Description
ep_base	31:9	23'b0	R/W	USB Endpoint Descriptor Table Base Address

This field represents the base address of the endpoint descriptor table in RAM, with the physical byte address given as (EP_BASE[31:9] * 512).

7.4.4.1.7 USB_CUR_BUF

USB_CUR_BUF.out_buf

Field	Bits	Default	Access	Description
out_buf	15:0	00000000h	R/O	OUT Transfer Current Buffers

For double-buffered endpoints, the low 8 bits of this field indicate the current buffer that the USB controller will use for an OUT transfer, as follows:

- bit is set to 0: buffer 0 will be used for this endpoint
- bit is set to 1: buffer 1 will be used for this endpoint

For single-buffered endpoints, the corresponding bit in this field will always be zero.

- bit 0 : current buffer for OUT transfers on Endpoint 0
- bit 1 : current buffer for OUT transfers on Endpoint 1
- bit 2 : current buffer for OUT transfers on Endpoint 2
- bit 3 : current buffer for OUT transfers on Endpoint 3
- bit 4 : current buffer for OUT transfers on Endpoint 4
- bit 5 : current buffer for OUT transfers on Endpoint 5
- bit 6 : current buffer for OUT transfers on Endpoint 6
- bit 7 : current buffer for OUT transfers on Endpoint 7

USB_CUR_BUF.in_buf

Field	Bits	Default	Access	Description
in_buf	31:16	00000000h	R/O	IN Transfer Current Buffers

For double-buffered endpoints, the low 8 bits of this field indicate the current buffer that the USB controller will use for an IN transfer, as follows:

- bit is set to 0: buffer 0 will be used for this endpoint
- bit is set to 1: buffer 1 will be used for this endpoint

For single-buffered endpoints, the corresponding bit in this field will always be zero.

- bit 0 : current buffer for IN transfers on Endpoint 0

- bit 1 : current buffer for IN transfers on Endpoint 1
- bit 2 : current buffer for IN transfers on Endpoint 2
- bit 3 : current buffer for IN transfers on Endpoint 3
- bit 4 : current buffer for IN transfers on Endpoint 4
- bit 5 : current buffer for IN transfers on Endpoint 5
- bit 6 : current buffer for IN transfers on Endpoint 6
- bit 7 : current buffer for IN transfers on Endpoint 7

7.4.4.1.8 USB_IN_OWNER

USB_IN_OWNER.buf0_owner

Field	Bits	Default	Access	Description
buf0_owner	15:0	00000000h	R/W	Owner for IN Buffer 0 for Endpoints

Bits 0 to 7 indicate the owner of the corresponding Endpoint IN buffer 0 as follows:

- 0 - Endpoint buffer is owned by application software
- 1 - Endpoint buffer is owned by the USB controller hardware
- bit 0 : owner for Endpoint 0 IN buffer 0
- bit 1 : owner for Endpoint 1 IN buffer 0
- bit 2 : owner for Endpoint 2 IN buffer 0
- bit 3 : owner for Endpoint 3 IN buffer 0
- bit 4 : owner for Endpoint 4 IN buffer 0
- bit 5 : owner for Endpoint 5 IN buffer 0
- bit 6 : owner for Endpoint 6 IN buffer 0
- bit 7 : owner for Endpoint 7 IN buffer 0

USB_IN_OWNER.buf1_owner

Field	Bits	Default	Access	Description
buf1_owner	31:16	00000000h	R/W	Owner for IN Buffer 1 for Endpoints

Bits 0 to 7 indicate the owner of the corresponding Endpoint IN buffer 1 as follows:

- 0 - Endpoint buffer is owned by application software
- 1 - Endpoint buffer is owned by the USB controller hardware
- bit 0 : owner for Endpoint 0 IN buffer 1
- bit 1 : owner for Endpoint 1 IN buffer 1
- bit 2 : owner for Endpoint 2 IN buffer 1
- bit 3 : owner for Endpoint 3 IN buffer 1
- bit 4 : owner for Endpoint 4 IN buffer 1
- bit 5 : owner for Endpoint 5 IN buffer 1
- bit 6 : owner for Endpoint 6 IN buffer 1
- bit 7 : owner for Endpoint 7 IN buffer 1

7.4.4.1.9 USB_OUT_OWNER**USB_OUT_OWNER.buf0_owner**

Field	Bits	Default	Access	Description
buf0_owner	15:0	00000000h	R/W	Owner for OUT Buffer 0 for Endpoints

Bits 0 to 7 indicate the owner of the corresponding Endpoint OUT buffer 0 as follows:

- 0 - Endpoint buffer is owned by application software
- 1 - Endpoint buffer is owned by the USB controller hardware

- bit 0 : owner for Endpoint 0 OUT buffer 0
- bit 1 : owner for Endpoint 1 OUT buffer 0
- bit 2 : owner for Endpoint 2 OUT buffer 0
- bit 3 : owner for Endpoint 3 OUT buffer 0
- bit 4 : owner for Endpoint 4 OUT buffer 0
- bit 5 : owner for Endpoint 5 OUT buffer 0
- bit 6 : owner for Endpoint 6 OUT buffer 0
- bit 7 : owner for Endpoint 7 OUT buffer 0

USB_OUT_OWNER.buf1_owner

Field	Bits	Default	Access	Description
buf1_owner	31:16	00000000h	R/W	Owner for OUT Buffer 1 for Endpoints

Bits 0 to 7 indicate the owner of the corresponding Endpoint OUT buffer 1 as follows:

- 0 - Endpoint buffer is owned by application software
- 1 - Endpoint buffer is owned by the USB controller hardware
- bit 0 : owner for Endpoint 0 OUT buffer 1
- bit 1 : owner for Endpoint 1 OUT buffer 1
- bit 2 : owner for Endpoint 2 OUT buffer 1
- bit 3 : owner for Endpoint 3 OUT buffer 1
- bit 4 : owner for Endpoint 4 OUT buffer 1
- bit 5 : owner for Endpoint 5 OUT buffer 1
- bit 6 : owner for Endpoint 6 OUT buffer 1
- bit 7 : owner for Endpoint 7 OUT buffer 1

7.4.4.1.10 USB_IN_INT**USB_IN_INT.inbav**

Field	Bits	Default	Access	Description
inbav	7:0	00000000b	W1C	Endpoint Buffer Available Interrupt Flags

These interrupt flags are set by the USB controller after it has successfully transferred an IN packet to the USB host and has received an ACK handshake in reply. This indicates that the endpoint buffer is now available to be written by software.

- bit 0: flag for Endpoint 0 buffer
- bit 1: flag for Endpoint 1 buffer
- bit 2: flag for Endpoint 2 buffer
- bit 3: flag for Endpoint 3 buffer
- bit 4: flag for Endpoint 4 buffer
- bit 5: flag for Endpoint 5 buffer
- bit 6: flag for Endpoint 6 buffer
- bit 7: flag for Endpoint 7 buffer

Write 1 to clear.

7.4.4.1.11 USB_OUT_INT**USB_OUT_INT.outdav**

Field	Bits	Default	Access	Description
outdav	7:0	00000000b	W1C	Endpoint Data Available Interrupt Flags

These interrupt flags are set by the USB controller when it has successfully received an OUT packet from the host and has loaded it into the designated endpoint buffer. This indicates that the packet data is available to be read by software.

- bit 0: flag for Endpoint 0
- bit 1: flag for Endpoint 1
- bit 2: flag for Endpoint 2
- bit 3: flag for Endpoint 3
- bit 4: flag for Endpoint 4
- bit 5: flag for Endpoint 5
- bit 6: flag for Endpoint 6
- bit 7: flag for Endpoint 7

Write 1 to clear.

7.4.4.1.12 USB_NAK_INT

USB_NAK_INT.nak

Field	Bits	Default	Access	Description
nak	7:0	00000000b	W1C	Endpoint NAK Interrupt Flags

These int flags are set by the USB controller after it sends a NAK handshake to the host in response to an IN request. This indicates that the endpoint buffer has no data available to be transmitted to the USB host.

- bit 0: flag for Endpoint 0
- bit 1: flag for Endpoint 1
- bit 2: flag for Endpoint 2
- bit 3: flag for Endpoint 3
- bit 4: flag for Endpoint 4
- bit 5: flag for Endpoint 5
- bit 6: flag for Endpoint 6

- bit 7: flag for Endpoint 7

Write 1 to clear.

7.4.4.1.13 USB_DMA_ERR_INT

USB_DMA_ERR_INT.dma_err

Field	Bits	Default	Access	Description
dma_err	7:0	00000000b	W1C	Endpoint DMA Error Interrupt Flags

These int flags are set by the USB controller when a USB DMA error occurs, meaning that the USB controller was unable to transfer data to or from the corresponding endpoint over the AHB bus.

- bit 0: flag for Endpoint 0
- bit 1: flag for Endpoint 1
- bit 2: flag for Endpoint 2
- bit 3: flag for Endpoint 3
- bit 4: flag for Endpoint 4
- bit 5: flag for Endpoint 5
- bit 6: flag for Endpoint 6
- bit 7: flag for Endpoint 7

Write 1 to clear.

7.4.4.1.14 USB_BUF_OVR_INT

USB_BUF_OVR_INT.buf_ovr

Field	Bits	Default	Access	Description
buf_ovr	7:0	00000000b	W1C	Endpoint Buffer Overflow Interrupt Flags

These int flags are set by the USB controller when a data packet to or from the USB host is larger than the size of the corresponding endpoint buffer in memory.

- bit 0: flag for Endpoint 0
- bit 1: flag for Endpoint 1
- bit 2: flag for Endpoint 2
- bit 3: flag for Endpoint 3
- bit 4: flag for Endpoint 4
- bit 5: flag for Endpoint 5
- bit 6: flag for Endpoint 6
- bit 7: flag for Endpoint 7

Write 1 to clear.

7.4.4.1.15 USB_SETUP0

USB_SETUP0.byte0

Field	Bits	Default	Access	Description
byte0	7:0	00h	R/O	SETUP Packet Byte 0

Byte 0 of the last SETUP packet received by the USB controller.

USB_SETUP0.byte1

Field	Bits	Default	Access	Description
byte1	15:8	00h	R/O	SETUP Packet Byte 1

Byte 1 of the last SETUP packet received by the USB controller.

USB_SETUP0.byte2

Field	Bits	Default	Access	Description
byte2	23:16	00h	R/O	SETUP Packet Byte 2

Byte 2 of the last SETUP packet received by the USB controller.

USB_SETUP0.byte3

Field	Bits	Default	Access	Description
byte3	31:24	00h	R/O	SETUP Packet Byte 3

Byte 3 of the last SETUP packet received by the USB controller.

7.4.4.1.16 USB_SETUP1**USB_SETUP1.byte0**

Field	Bits	Default	Access	Description
byte0	7:0	00h	R/O	SETUP Packet Byte 4

Byte 4 of the last SETUP packet received by the USB controller.

USB_SETUP1.byte1

Field	Bits	Default	Access	Description
byte1	15:8	00h	R/O	SETUP Packet Byte 5

Byte 5 of the last SETUP packet received by the USB controller.

USB_SETUP1.byte2

Field	Bits	Default	Access	Description
byte2	23:16	00h	R/O	SETUP Packet Byte 6

Byte 6 of the last SETUP packet received by the USB controller.

USB_SETUP1.byte3

Field	Bits	Default	Access	Description
byte3	31:24	00h	R/O	SETUP Packet Byte 7

Byte 7 of the last SETUP packet received by the USB controller.

7.4.4.1.17 USB_EP**USB_EP.ep_dir**

Field	Bits	Default	Access	Description
ep_dir	1:0	11b	R/W	Endpoint Direction

For Endpoint 0 - Read-only; always set to 11b (CONTROL pipe accepting IN, OUT, and STATUS packets) for Endpoint 0.

For other endpoints (1 through 7) -

- 00b: Endpoint is disabled.
- 01b: Endpoint is configured for OUT transfers.
- 10b: Endpoint is configured for IN transfers.
- 11b: Reserved (only EP0 can be set to CONTROL mode).

USB_EP.ep_buf2

Field	Bits	Default	Access	Description
ep_buf2	3	0	R/W	Endpoint Double Buffered Enable

- 0: Endpoint is single buffered.
- 1: Endpoint is double buffered.

USB_EP.ep_int_en

Field	Bits	Default	Access	Description
ep_int_en	4	0	R/W	Endpoint Transfer Complete Interrupt Enable

1: Enable generation of interrupts for this endpoint upon completion of an IN or OUT data transfer.

USB_EP.ep_nak_en

Field	Bits	Default	Access	Description
ep_nak_en	5	0	R/W	Endpoint NAK Interrupt Enable

1: Enable generation of interrupts for this endpoint when a NAK handshake is returned to the host.

USB_EP.ep_dt

Field	Bits	Default	Access	Description
ep_dt	6	0	R/W	Endpoint Data Toggle Clear

Write to 1 to reset the data toggle field to DATA0 and reset the current buffer to buffer 0. In addition, the IN and OUT Buffer Owner bits for this endpoint will be cleared as well. Writes to 0 are ignored.

Cleared to 0 by hardware after the data clear / reset process has been completed.

USB_EP.ep_stall

Field	Bits	Default	Access	Description
ep_stall	8	0	R/W	Endpoint Stall

- 0: Endpoint is not stalled.
- 1: Endpoint is stalled.

Upon receiving a SETUP packet, this bit is automatically cleared to 0.

USB_EP.ep_st_stall

Field	Bits	Default	Access	Description
ep_st_stall	9	0	R/W	Endpoint Stall Status Stage of Control Transfer

- 0: Do not send STALL.
- 1: Send STALL to host for status stage.

Upon receiving a SETUP packet, this bit is automatically cleared to 0.

USB_EP.ep_st_ack

Field	Bits	Default	Access	Description
ep_st_ack	10	0	R/W	Endpoint Acknowledge Status Stage of Control Transfer

- 0: Do not send ACK.
- 1: Send ACK to host for status stage.

Upon receiving a SETUP packet, this bit is automatically cleared to 0.

8 Analog Front End

Analog Front End

The analog front end on the **MAX32600** includes a wide variety of input and output peripherals and blocks which allow sensing of input analog voltages (e.g., with the ADC or analog-to-digital converter block) and driving of output analog voltages (e.g., with the DAC or digital-to-analog output channels) to connect digital firmware to the outside analog world.

In addition to the main ADC and DAC analog peripherals, the analog front end includes a variety of smaller function blocks that can be used in either dedicated (standalone) modes or in combination with other analog functions. These analog function blocks include:

- Four operational amplifier (op amp) blocks with built-in comparator modes
- Four dedicated low-power comparators which can be used in parallel with the op amp blocks
- Internal temperature sensor which can be scanned by internally connecting it with an ADC channel
- Four single-pole-single-throw (SPST) analog switches which can be toggled either manually by firmware or automatically using a preconfigured digital pulse train output
- Optical drive modes on pairs of GPIO port pin pads to allow high-current LED drive options using different configurations (including H-Bridge options)

A large analog interconnection matrix allows many of the input and output ports of the ADC, DAC and various other analog and digital blocks to be rerouted to each other using internal connection points and analog multiplexer blocks. This allows for the configuration of more complex analog functions (such as voltage followers and other types of internal amplifiers and buffers) which include ADC or DAC voltages as an input or output, without having to add physical connections external to the device.

One key feature of the analog front end functions on the **MAX32600** is that they can be set up to run largely without CPU intervention, allowing sequences of ADC readings and/or sequences of DAC channel output voltages to run while the main Cortex-M3 CPU core is in a [low-power sleep mode](#). The [Peripheral Management Unit \(PMU\)](#) on the **MAX32600** can be used instead of the higher-power CPU to perform tasks that must take place during a long sequence of analog operations, such as unloading samples from the ADC FIFO to the main data RAM or loading new sample sequences into the DAC FIFO for output generation. Certain analog functions (e.g., the ADC) require a low-noise environment with the main CPU in sleep mode in order to attain the highest possible precision during data collection operations.

8.1 AFE Overview

This section covers a number of blocks and smaller analog peripherals that are included in the AFE along with the main ADC and DAC peripherals. Some of these analog features can be used independently of the ADC and DAC, while others can only be used in combination with the ADC/DAC or certain other analog functions.

8.1.1 AFE Analog Function Blocks

The following analog function blocks are considered part of the AFE and are controlled (in most cases) by registers included in the AFE register module.

Current Drive Mode (LED Drive) for GPIO Pins

Certain pairs of GPIO pins can be configured to operate as high current source/drain pairs for LED drive output or similar current drive applications. This function may also involve an op amp and DAC in a current controlled feedback circuit.

Op Amps

The AFE includes four uncommitted op amps which can be used for a variety of purposes either in standalone configuration or as part of a more complex setup involving the ADC and/or DAC. Each op amp can be used in either a general purpose amplifier mode or a dedicated comparator mode.

Low-Power Comparators

The AFE includes four low-power comparators which can be used in parallel with the main op amps. These comparators are specifically optimized for low power consumption, making them ideal to use as wakeup sources to bring the **MAX32600** out of a low power state.

SPST Switches

Four single-pole, single-throw (SPST) analog switches are included in the AFE. These switches can be used in a standalone mode, or as part of a larger analog setup. The state of each switch (open or closed) can be controlled either directly by firmware using register bits, or automatically using certain pulse train outputs. When used with an external capacitor, these switches can be used to build a low cost sample and hold circuit.

ADC and DAC References

Two programmable reference levels (one used by the ADC, one used by the DACs) are included, and each can be individually set to one of four output levels: 1.024V, 1.5V, 2.048V or 2.5V.

An external reference can also be provided at the REFADJ pin; if this feature is used, the external reference voltage will be used in place of the 1.23V bandgap output, and the programmable output levels for the ADC and DAC references will shift accordingly.

Analog Matrix

The analog matrix is a set of interlinked switch arrays in the AFE which allows a large number of possible connections between different input and output ports of various components in the AFE.

8.2 AFE Reconfiguration Matrix

8.2.1 AFE Reconfiguration Matrix Overview

A large analog reconfiguration matrix allows many of the input and output ports of the ADC, DAC, and various other analog and digital blocks to be dynamically connected to each other using internal connection points and analog multiplexer blocks. This allows for the configuration of highly complex analog functions (e.g., voltage followers and other types of internal amplifiers and buffers), which include ADC or DAC voltages as an input or output, without having to add physical connections external to the device.

Many of the analog front end functions on the **MAX32600** can be configured to run primarily without CPU intervention. The [Peripheral Management Unit \(PMU\)](#) on the **MAX32600** can be used instead of the CPU to perform tasks that must take place during long sequences of analog operations. This allows sequences of ADC readings and/or sequences of DAC channel output voltages to run while the Cortex-M3 core is in a low-power sleep mode (reference [Power Ecosystem](#) for more details about **MAX32600** power modes). More specifically, this includes operations such as unloading samples from the ADC FIFO to the main data RAM or loading new sample sequences into the DAC FIFO for output generation. The PMU's functionality is particularly useful for analog functions (e.g., the ADC) that require a low-noise environment to attain the highest possible precision during data collection operations; the PMU provides this environment by operating independent of the Cortex-M3 core. The [Analog Reconfiguration Sub-Matrix Diagrams](#) illustrate the configurability between the:

- Op amps
- Low-power comparators
- DAC outputs
- Op amp output pins
- SPST switch external pins
 - **NOTE:** The number of available SPST switch external pins is package dependent; refer to the **MAX32600** data sheet for the pin out configuration for each of the available package types
- LED drive and observe ports pins

The multiplexers (mux) allow a variety of possible connections to implement features including an op amp follower to buffer a DAC output, a second-order Sallen-key filter, an LED driver control loop, among many others. The flexibility of the analog configuration matrix enables maximum configuration options of the **MAX32600**

analog front end. Prior to configuring the Analog Reconfiguration Matrix for a specific application, the user must thoroughly understand the configuration options to prevent setting up an undesirable configuration.

Note It is possible to configure the matrix in ways that short pads together or create undesired current levels or loads. Careful attention to all the register settings that control the matrix is imperative.

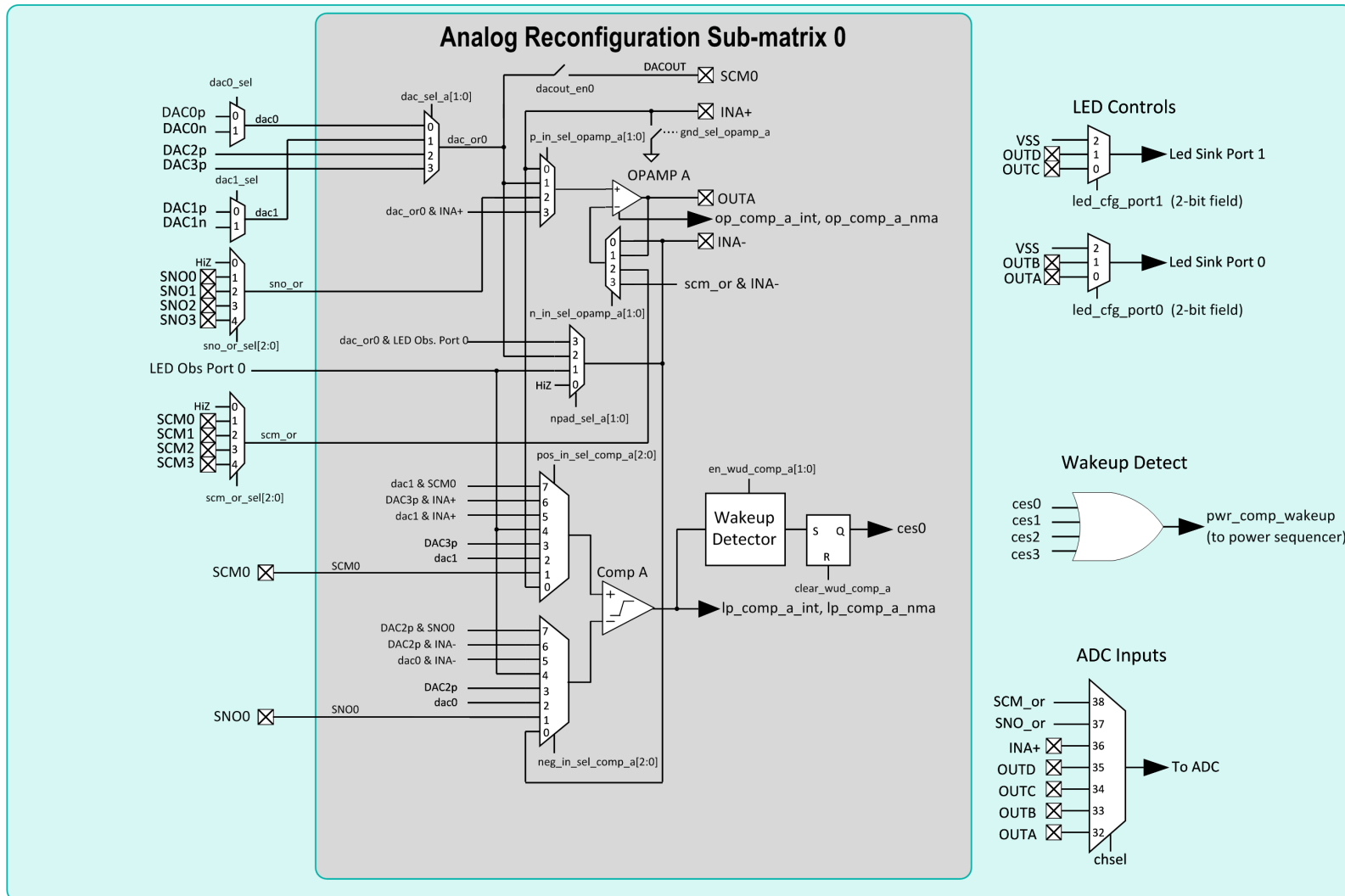


Figure 8.1: Analog Reconfiguration Sub-Matrix 0 Diagram

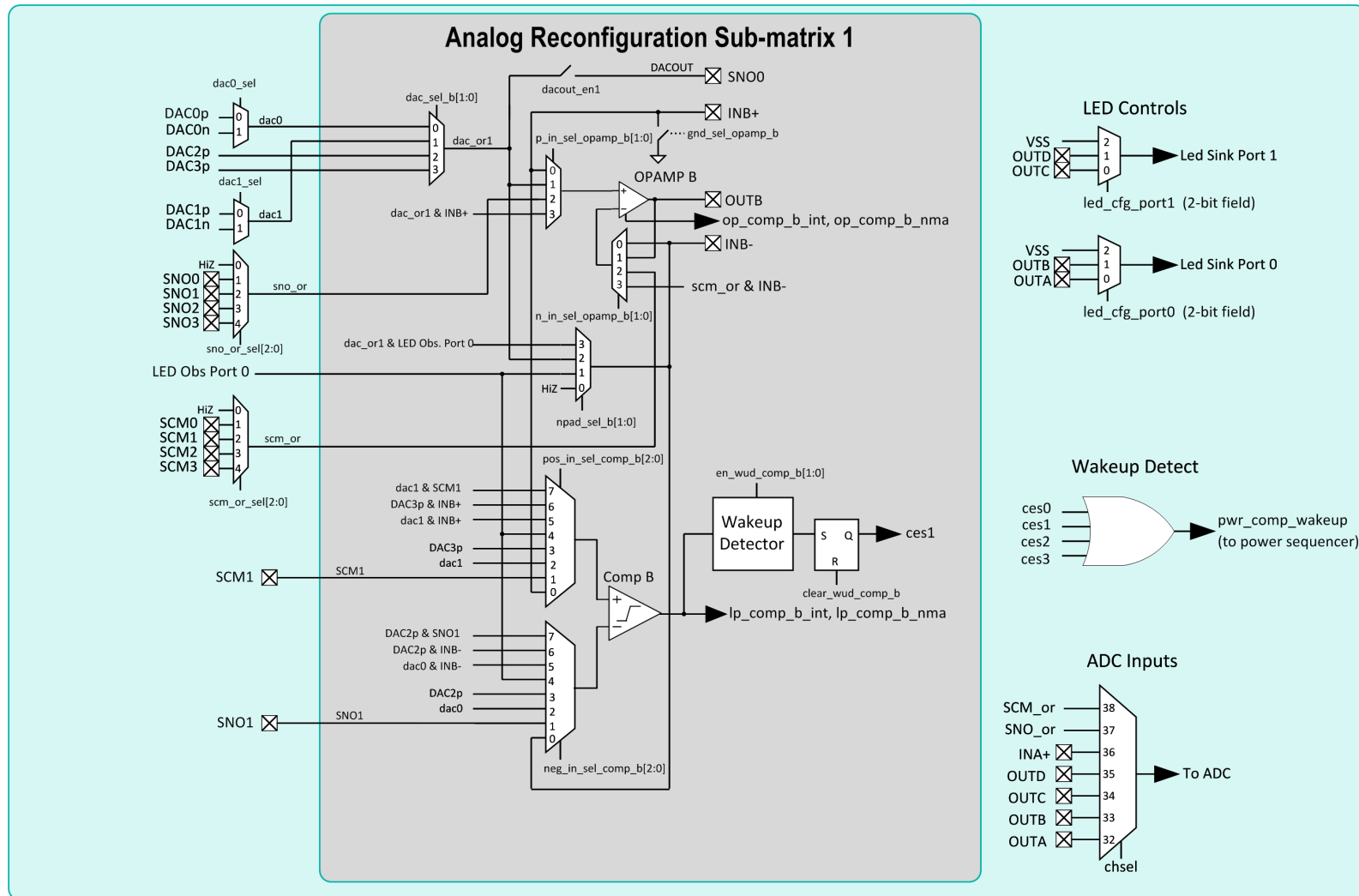


Figure 8.2: Analog Reconfiguration Sub-Matrix 1 Diagram

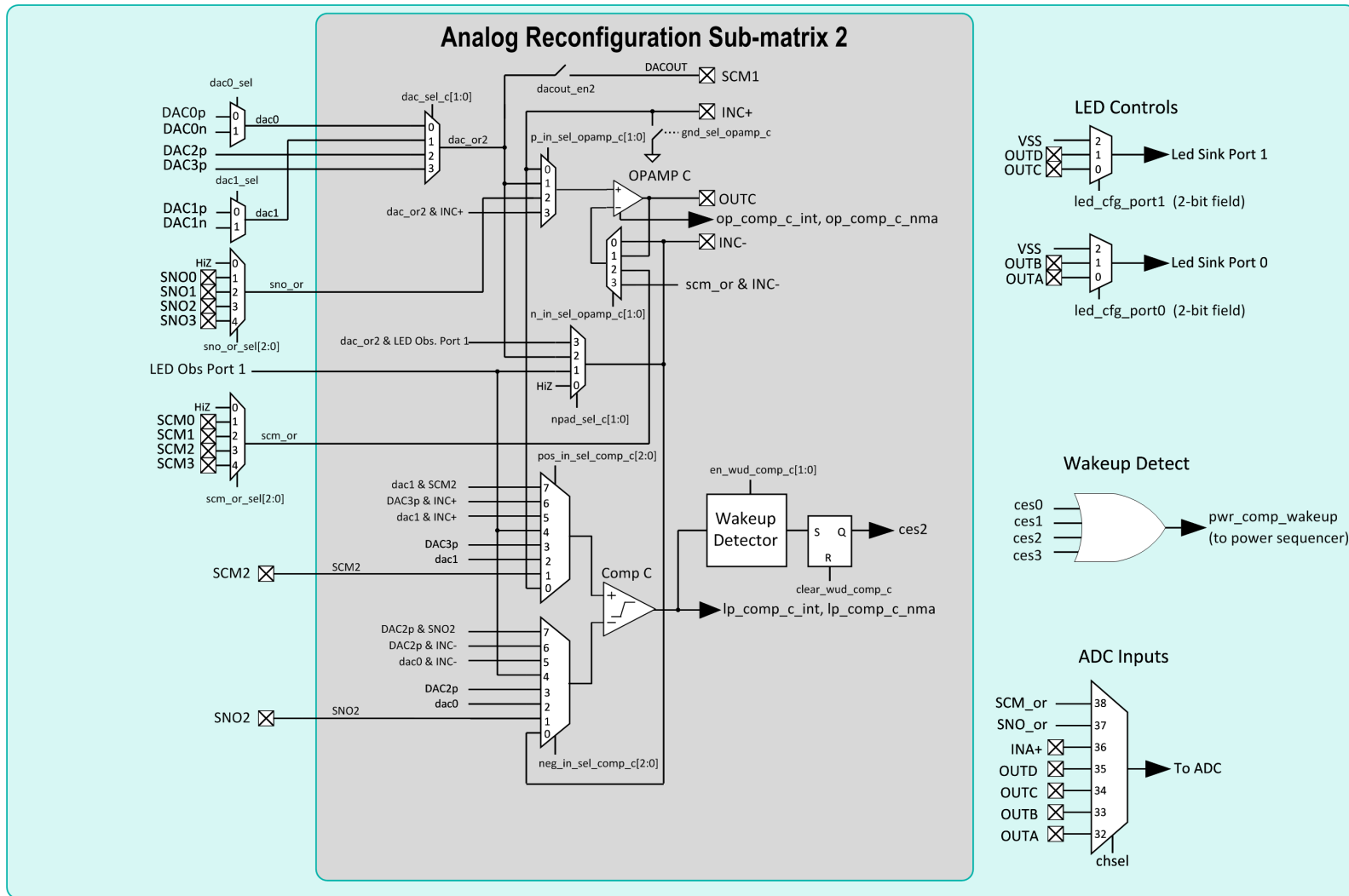


Figure 8.3: Analog Reconfiguration Sub-Matrix 2 Diagram

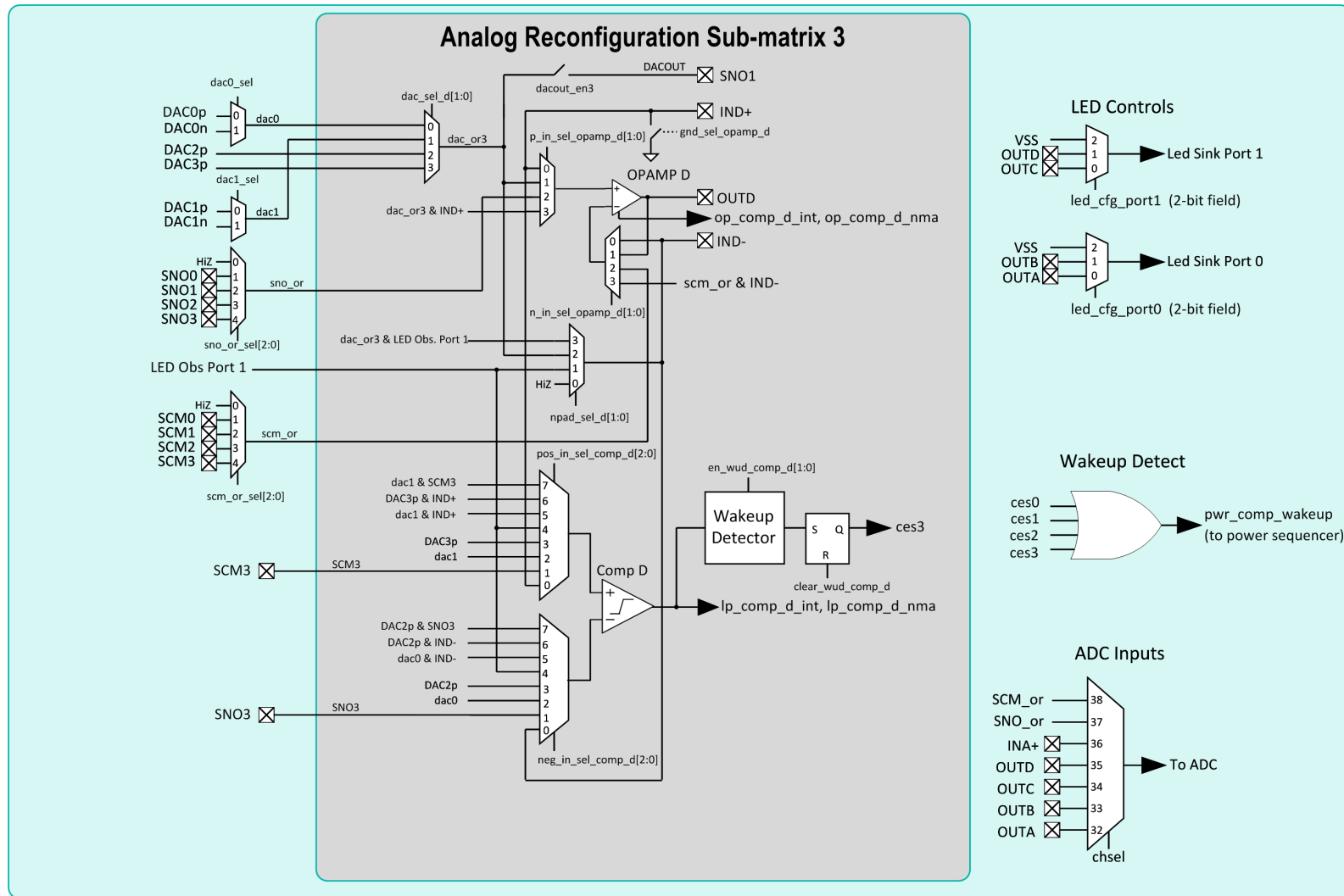


Figure 8.4: Analog Reconfiguration Sub-Matrix 3 Diagram

Reconfiguration Matrix Inputs

The register fields that control each mux for the inputs to the Analog Reconfiguration Sub-Matrix are:

- [dac0_sel](#) - Top-Level Multiplexer
- [dac1_sel](#) - Top-Level Multiplexer
- [sno_or_sel](#) - Single-Pole, Single-Throw (SPST)
- [scm_or_sel](#) - Single-Pole, Single-Throw (SPST)

Each of the above register fields are shown on the control input of a mux to select the inputs to the [Analog Reconfiguration Sub-Matrix](#). Input options to the matrix come from:

- DAC outputs
- SPST Switches (either four or eight pins depending on the package option)
- Op amp input and output pins
- LED observe port pins

Reconfiguration Matrix Outputs

Outputs from the matrix go to four SPST switch pins (common to all package types), op amp IN+ and IN- pins, an analog bus input to the ADC, four comparator outputs to the core, and two LED drive port pins sent to the gate voltage of the LED drivers in the GPIO module. The Analog Reconfiguration Sub-Matrices are grouped together by an op amp and a standard comparator to create sub-matrices 0, 1, 2, and 3. Sub-matrix 0 includes op amp A and comparator A; the others are grouped similarly. All four sub-matrices have identical topologies and similar mux control inputs that differ only in a 0-3 or A-D suffix denoted with an N or an X throughout this document when referring to the sub-matrices in general. The components shown outside the [Analog Reconfiguration Sub-Matrix](#) are common to all of the sub-matrices.

Top-Level Multiplexers

The top-level multiplexers consist of two 2:1 multiplexers that select the two 12-bit DAC positive or negative outputs, DAC0p/n and DAC1p/n, for the two internal signals dac0 and dac1 that drive into each of the four analog reconfiguration matrices. To select DAC0p/n on the first mux, refer to register field [dac0_sel](#); for DAC1p/n on the second mux, refer to register field [dac1_sel](#).

Note These top level multiplexers control which signal is *internally* connected to all four of the analog reconfiguration matrices; this does not limit which signal are connected to the op amps. If different signals are to be used, connect the external pins in the circuit. Here, it is possible to select DAC0p for op amp A and DAC1n for op amp B as the dac0 and dac1 signals are independent of one another. The 8-bit DAC outputs DAC2p and DAC3p drive directly into the four analog reconfiguration matrices. The 8-bit DACs do not have negative outputs.

Single-Pole, Single-Throw (SPST)

Similarly the four SPST pins SNO0/1/2/3 are selected by a 4:1 mux using register field `sno_or_sel` creating the signal `sno_or`. `scm_or_sel` selects between the four SPST pins SCM0/1/2/3 creating the signal `scm_or`. SPST switches 2 and 3 are only available in the 12mm x 12mm package and cannot be selected in the 7mm x 7mm or WLP packages.

The `sno_or` signal is common to all four Analog Reconfiguration Matrices. The SPST switches must be disabled if these pins are to be used as matrix inputs and not as SPST switch pins as signals may be shorted to one another in undesirable ways. Writing 0 to the following registers opens (disables) the corresponding SPST switches: `close_spst0`, `close_spst1`, `close_spst2`, and `close_spst3`. It is possible, however, to use one side of an SPST switch as an op amp input in which case the switch could be closed and still be used in the matrices. The user must understand the nature of each of the signals connected to the matrix to avoid situations where the impedance of the closed switch might create undesired currents or loads.

LED Observe Ports

The LED observe ports 0 and 1 are available as Analog Reconfiguration matrix inputs. Two LED drive control multiplexers allow op amp outputs OUTA or OUTB to drive LED Sink Port 0 and OUTC or OUTD to drive LED Sink Port 1. A third mux state drives V_{SS} onto LED Sink Port 0 or LED Sink Port 1 disabling the LED sink devices. This is the power-up default condition for the part.

Channel Select Register

The `chsel` register can select various signals to drive into the ADC input as noted in the [Analog Reconfiguration Sub-Matrix Diagram](#). Specifically, each of the four op amp outputs OUTA/B/C/D, INA+, and the mux switch pins `scm_or` or `sno_or`. The `chsel` register can also select a number of other signals for the ADC as described in the [ADC Input Multiplexer Configuration Section](#).

External Pins

The above mentioned `dac0`, `dac1`, `sno_or`, `scm_or`, and LED Observe Port 0 and 1 signals along with the two 8-bit DAC outputs DAC2p and DAC3p comprise the eight signals that drive into all four of the analog reconfiguration matrices. The 16 external pins, however, are specific to each of the four analog reconfiguration matrices as follows:

Analog Reconfiguration Matrix External Pin Table

Analog Reconfiguration Matrix	DACOUT	Op Amp Positive Input	Op Amp Negative Input	Op Amp Output	LED Observe Port Input	SCM Pin	SNO Pin
0	SCM0	INA+	INA-	OUTA	0	SCM0	SNO0
1	SNO0	INB+	INB-	OUTB	0	SCM1	SNO1
2	SCM1	INC+	INC-	OUTC	1	SCM2	SNO2
3	SNO1	IND+	IND-	OUTD	1	SCM3	SNO3

Note **DACOUT:** Any DAC output can also go to the IN- pin of an op amp.

Within each analog reconfiguration matrix, there are six multiplexers and one switch that control the op amp and comparator inputs as well as the SPST switch and op amp negative input external pins. Note that external pins can be both inputs and outputs; care must be taken in programming the matrix to avoid undesired loops or short circuits.

8.2.1.1 DAC

The four DAC inputs—dac0, dac1 (previously multiplexed at the matrix top-level) and DAC2p, DAC3p (directly from the 8-bit DACs)—drive a 4:1 mux that outputs the dac_orN signal (where N=0,1,2,3 depending on the particular analog reconfiguration matrix). The dac_orN signal can be brought out to the SPST external pin via the [dacout_en0](#), [dacout_en1](#), [dacout_en2](#), and [dacout_en3](#) register fields and is also used as an input to the op amp positive input mux and op amp negative external pin mux.

8.2.1.2 Op Amps

The op amp positive input is driven by a 4:1 mux selecting either the INX+ external pin or the internally derived signals dac_orN and sno_or. A fourth state selects the dac_orN signal and also sends dac_orN to the INX+ external pin. Note that in this configuration any resistive load on the INX+ external pin will load the DAC output. All the multiplexers are implemented as analog switches and normally have only one switch active at any moment; in this case, two of the analog switches are selected so that a three-way connection is possible. The three-way connections only occur with external pin connections for the purpose of internal observability.

The op amp negative input is driven by one of three sources: the INX- external pin, the op amp output OUTX, or the scm_or input. A fourth state selects both scm_or and the INX- pin.

The op amp negative external pin can be driven by the locally generated dac_orN signal described above, the LED Observe Port input, or both. A fourth state disconnects the mux (this is denoted as a HiZ signal in the [diagram](#)) so that the INX- pin can be driven externally. This is the default condition for the mux. Note that because of the various mux options, the “op amp negative pin” INX- is not always the negative input to the op amp.

Additionally, there is a ground select switch on each of the op amp positive input external pins INX+ that shorts the pin to V_{SS} with an impedance of 30 Ohms. To choose this option, use register fields [gnd_sel_opamp_a](#), [gnd_sel_opamp_b](#), [gnd_sel_opamp_c](#), and [gnd_sel_opamp_d](#). Note that the op amp positive input mux must be set appropriately for this to affect the op amp positive input.

Op Amp Control

Each op amp has four control signals set by the corresponding AFE control register bits. Bit `pu_opamp_[X]` controls the power-up of each op amp where X=A,B,C,D. This bit must be set for the op amp to operate. If 0, the OUTX output pin goes into a high-impedance state. The op amp has two independent input stages, one n-channel and one p-channel, that together allow rail-to-rail operation. In cases where the op amp input bias operates over a restricted range, better THD performance and bias-independent offset can be achieved if only one of the two input pairs is selected. Setting control bit `en_pch_opamp[X]` to 1 enables the p-channel input, and setting control bit `en_nch_opamp[X]` to 1 enables the n-channel input (if neither input is selected the op amp will automatically power off). For an input bias voltage between V_{SS} and $V_{DDA} - 1.05V$, the p-channel input should be selected; for an input bias between 0.95V and V_{DDA} , the n-channel input should be selected. If the op amp is configured for a single-polarity input and used outside the specified operating range, a dramatic decrease in gain will be seen and the op amp will become non-functional. For the general case, both `en_pch_opamp` and `en_nch_opamp` should be set to 1s for rail-to-rail operation.

Input bias voltage range	Input Stage
$V_{SS} < V_{in} < (V_{DDA} - 1.05V)$	p-channel only
$0.95V < V_{in} < V_{DDA}$	n-channel only
$V_{SS} < V_{in} < V_{DDA}$	Both n-channel and p-channel

Op Amp Comparator Configuration

In general, if a comparator function is needed, the low-power comparators should be used for reduced power; however, if more than four comparators are needed, the op amp may be used as a comparator. Setting control bit `op_cmp_X` to 1 will disable the internal compensation capacitor, improving response time. If `op_cmpX` is set to 0 instead, the comparator will be subject to the same slew rate limitation of the op amp. If the op amp is used as a feedback op amp, this bit should not be set or instability will occur. Also, when `op_cmp_X` is set selecting comparator mode, the op amp output `op_cmpX` drives into the core. Thus, the user needs to ensure that when `op_cmp_X` is set that the op amp output will be either V_{SS} or V_{DDA3} or crowbar current (current resulting from a mid-rail analog voltage applied to a logic gate input) may result. Output `op_cmpX` is a buffered version of OUTX that is disabled when `op_cmp_X` is set to 0 to prevent crowbar current in normal op amp mode.

8.2.1.3 Comparators

Each comparator input is driven by an 8:1 mux. The positive comparator input can be driven by the INX+ or SCMN external pins, the dac1 signal from the top-level of the matrix, the DAC3p DAC output, or the LED observe port as noted in the [Analog Reconfiguration Matrix External Pin Table](#). There are three additional states that allow either dac1 or DAC3p to be selected and also observed on otherwise unused external pins: dac1 and INX+, DAC3p and INX+, or dac1 and SCMN. Register fields `pos_in_sel_comp_a`, `pos_in_sel_comp_b`, `pos_in_sel_comp_c`, and `pos_in_sel_comp_d` are used to select which option is passed to the positive comparator input. The user must ensure that these pins are not used for any other purpose.

The negative comparator input can be driven by the INX- or SNON external pins, the dac0 signals from the matrix top-level, the DAC output DAC2p, or the LED observe port as noted in the [Analog Reconfiguration Matrix External Pin Table](#). Three additional states allow the dac0 or DAC2p signals to be selected and also routed to external pins: dac0 and INX-; DAC2p and INX-; or DAC2p and SNON, again to allow external pin observability. Register fields `neg_in_sel_comp_a`,

[neg_in_sel_comp_b](#), [neg_in_sel_comp_c](#), and [neg_in_sel_comp_d](#) are used to select which option is passed to the negative comparator input. As with the positive comparator input, the user must ensure that these pins are not used for any other purpose.

Comparator Hysteresis Comparator hysteresis prevents unwanted transitioning from occurring. Without hysteresis, the comparator output would transition whenever inputs are equal (i.e., $V_P=V_N$), regardless of whether the input is transitioning (from low to high or from high to low).

For example, without hysteresis, if $V_N=0.5V$ and $V_P=0V$, the comparator output would be 0 and would not switch high until V_P is increased to the point where $V_P=V_N$. Further, as V_P is increased so $V_P > V_N$, the comparator output stays high; then, as V_P is decreased back to the $V_P=V_N$ point, the comparator output transitions to a low level and stays at a low level as V_P continues to be lowered. In this situation, if the levels are kept at $V_P=V_N$, the output would oscillate as the high gain of the circuit amplifies noise on the input.

Adding hysteresis to the comparator avoids this behavior: the trip point on a low to high input transition is higher than that of a high to low input transition. This trip point difference is called the hysteresis voltage (V_{hyst}) and can be programmed from 0-30mV in the [hyst_comp_X\[1:0\]](#) register field.

The register field [hy_pol_comp_X](#) sets the hysteresis polarity:

- [hy_pol_comp_X](#) = 0, the comparator output will transition from low to high when $V_P > (V_N + V_{hyst})$ on an input low to high transition, but the output will not go low until V_P is lowered to the point where $V_P < V_N$.
- [hy_pol_comp_X](#) = 1, the comparator output will transition from low to high when $V_P > V_N$ on an input low to high transition, but the output will not go low until V_P is lowered to the point where $V_P < (V_N - V_{hyst})$.

Low-Power Comparator Control

Each low power comparator has seven control bits:

- [poweru_comp_X](#) - Powers on comparator when set to 1
- [hyst_comp_X\[1:0\]](#) – Sets the magnitude of the comparator input hysteresis (V_{hyst}) from 0-30mV (as shown in the register details and section [above](#))
- [hy_pol_comp_a](#) – Sets the hysteresis polarity:
 - 0: Comparator output trips high when $V_P > (V_N + V_{hyst})$ and trips low when $V_P < V_N$
 - 1: Comparator output trips high when $V_P > V_N$ and trips low when $V_P < (V_N - V_{hyst})$
- [bias_mode_comp_X\[1:0\]](#) – Allows user to trade off comparator power for speed with four different bias current settings:
 - 0: $0.52\mu A$, $4.0\mu s$ delay (default)
 - 1: $1.4\mu A$, $1.7\mu s$ delay
 - 2: $2.8\mu A$, $1.1\mu s$ delay
 - 3: $5.1\mu A$, $0.7\mu s$ delay

- `in_mode_comp_X[1:0]` – Allows either or both n-channel and p-channel input stages to be selected for the low-power comparator:
 - 0: Both n-channel and p-channel input stages selected (default)
 - 1: Only n-channel stage input stage selected
 - 2: Only p-channel stage input stage selected
 - 3: Reserved
- `clear_wud_comp_X` – Clears the corresponding wake-up detector for the specified comparator when set to 1
- `en_wud_comp_X[1:0]` – Sets the comparator wakeup detector:
 - 0: Wakeup detector is inactive (idle state)
 - 1: Activates wakeup detector with low input level
 - 2: Activates wakeup detector with high input level
 - 3: Reserved

Each comparator has a direct output `lpcmp[X]` that drives into the **MAX32600** logic core. The primary purpose of the low-power comparators is to provide a low-power means for detecting external events and wake the device from low-power sleep or stop modes. Each comparator has an associated wake-up detector, set by the `en_wud_comp_X[1:0]` bits and cleared with the `clear_wud_comp_X` bit. `in_mode_comp_X[1:0]` has three states that control the mode of the wakeup detector:

Wakeup Detector Set Modes

<code>in_mode_compX[1:0]</code>	Mode	Notes
00	Idle (default)	Useful for rail-to-tail input voltages
01	Only n-channel input stage selected	Useful for minimizing input offset voltage for input voltages restricted to $\sim 0.9\text{ V} < (V_P/V_N) < V_{DDA}$
10	Only p-channel input stage selected	Useful for minimizing input offset voltage for input voltages restricted to $V_{SS} < (V_P/V_N) < (V_{DDA} - \sim 0.9\text{V})$
11	Reserved	

Setting `clear_wud_comp_X` to 1 forces the wake-up detector comparator event seen (`ces`) output to 0. With `en_wud_comp_X[1:0]` set to 00b, the wakeup detector is inactive and the wakeup detector `ces` output stays at 0. With `en_wud_comp_X[1:0]` set to 01b, `ces` (low level) will go high when `lpcmp` is low, and similarly with `en_wud_comp_X[1:0]` set to 10b, `ces` (high level) will go high when `lpcmp` is high. Note that the wakeup detector is level sensitive, not edge sensitive. Typically, the wakeup detector is programmed before the wakeup event has happened. Here, the 01 state will detect when the comparator switches low, but if the comparator output is already low when `en_wud_comp_X[1:0]` changes to the 01 state, `ces` will immediately go high.

At the top-level of the matrix hierarchy, the four `ces[D:A]` signals from each of the analog reconfiguration matrices are OR'd together to create the `pwr_comp_wakup` signal that drives into the power sequencer. This signal will go high when any of the four wakeup detectors activate.

8.2.2 Registers (AFE)

8.2.2.1 Module AFE Registers

Address	Register	32b Word Len	Description
0x4005401C	AFE_INTR	1	Analog Front End Interrupt Flags and Enable/Disable
0x40054020	AFE_CTRL0	1	Analog Front End Control 0
0x40054024	AFE_CTRL1	1	Analog Front End Control 1
0x40054028	AFE_CTRL2	1	Analog Front End Control 2
0x4005402C	AFE_CTRL3	1	Analog Front End Control 3
0x40054030	AFE_CTRL4	1	Analog Front End Control 4
0x40054034	AFE_CTRL5	1	Analog Front End Control 5

8.2.2.1.1 AFE_INTR

AFE_INTR.op_comp_a_int

Field	Bits	Default	Access	Description
op_comp_a_int	0	0	W1C	Op Amp A Comparator Event Interrupt Flag

Set by hardware when the associated interrupt condition occurs.

Write 1 to clear.

AFE_INTR.op_comp_b_int

Field	Bits	Default	Access	Description
op_comp_b_int	1	0	W1C	Op Amp B Comparator Event Interrupt Flag

Set by hardware when the associated interrupt condition occurs.

Write 1 to clear.

AFE_INTR.op_comp_c_int

Field	Bits	Default	Access	Description
op_comp_c_int	2	0	W1C	Op Amp C Comparator Event Interrupt Flag

Set by hardware when the associated interrupt condition occurs.

Write 1 to clear.

AFE_INTR.op_comp_d_int

Field	Bits	Default	Access	Description
op_comp_d_int	3	0	W1C	Op Amp D Comparator Event Interrupt Flag

Set by hardware when the associated interrupt condition occurs.

Write 1 to clear.

AFE_INTR.lp_comp_a_int

Field	Bits	Default	Access	Description
lp_comp_a_int	4	0	W1C	Low-Power Comparator A Event Interrupt Flag

Set by hardware when the associated interrupt condition occurs.

Write 1 to clear.

AFE_INTR.lp_comp_b_int

Field	Bits	Default	Access	Description
lp_comp_b_int	5	0	W1C	Low-Power Comparator B Event Interrupt Flag

Set by hardware when the associated interrupt condition occurs.

Write 1 to clear.

AFE_INTR.lp_comp_c_int

Field	Bits	Default	Access	Description
lp_comp_c_int	6	0	W1C	Low-Power Comparator C Event Interrupt Flag

Set by hardware when the associated interrupt condition occurs.

Write 1 to clear.

AFE_INTR.lp_comp_d_int

Field	Bits	Default	Access	Description
lp_comp_d_int	7	0	W1C	Low-Power Comparator D Event Interrupt Flag

Set by hardware when the associated interrupt condition occurs.

Write 1 to clear.

AFE_INTR.op_comp_a_nma

Field	Bits	Default	Access	Description
op_comp_a_nma	8	0	W1C	Op Amp Comparator A Non-Maskable Event Flag

Set by hardware when the associated interrupt condition occurs. This is not a standard interrupt flag; instead it is a non-maskable, asynchronous output that goes directly to the associated interrupt detector in the PMU. This will occur even if the AFE/ADC AMBA bus clock has been turned off for power savings reasons.

Write 1 to clear.

AFE_INTR.op_comp_b_nma

Field	Bits	Default	Access	Description
op_comp_b_nma	9	0	W1C	Op Amp Comparator B Non-Maskable Event Flag

Set by hardware when the associated interrupt condition occurs. This is not a standard interrupt flag; instead it is a non-maskable, asynchronous output that goes directly to the associated interrupt detector in the PMU. This will occur even if the AFE/ADC AMBA bus clock has been turned off for power savings reasons.

Write 1 to clear.

AFE_INTR.op_comp_c_nma

Field	Bits	Default	Access	Description
op_comp_c_nma	10	0	W1C	Op Amp Comparator C Non-Maskable Event Flag

Set by hardware when the associated interrupt condition occurs. This is not a standard interrupt flag; instead it is a non-maskable, asynchronous output that goes directly to the associated interrupt detector in the PMU. This will occur even if the AFE/ADC AMBA bus clock has been turned off for power savings reasons.

Write 1 to clear.

AFE_INTR.op_comp_d_nma

Field	Bits	Default	Access	Description
op_comp_d_nma	11	0	W1C	Op Amp Comparator D Non-Maskable Event Flag

Set by hardware when the associated interrupt condition occurs. This is not a standard interrupt flag; instead it is a non-maskable, asynchronous output that goes directly to the associated interrupt detector in the PMU. This will occur even if the AFE/ADC AMBA bus clock has been turned off for power savings reasons.

Write 1 to clear.

AFE_INTR.lp_comp_a_nma

Field	Bits	Default	Access	Description
lp_comp_a_nma	12	0	W1C	Low-Power Comparator A Non-Maskable Event Flag

Set by hardware when the associated interrupt condition occurs. This is not a standard interrupt flag; instead it is a non-maskable, asynchronous output that goes directly to the associated interrupt detector in the PMU. This will occur even if the AFE/ADC AMBA bus clock has been turned off for power savings reasons.

Write 1 to clear.

AFE_INTR.lp_comp_b_nma

Field	Bits	Default	Access	Description
lp_comp_b_nma	13	0	W1C	Low-Power Comparator B Non-Maskable Event Flag

Set by hardware when the associated interrupt condition occurs. This is not a standard interrupt flag; instead it is a non-maskable, asynchronous output that goes directly to the associated interrupt detector in the PMU. This will occur even if the AFE/ADC AMBA bus clock has been turned off for power savings reasons.

Write 1 to clear.

AFE_INTR.lp_comp_c_nma

Field	Bits	Default	Access	Description
lp_comp_c_nma	14	0	W1C	Low-Power Comparator C Non-Maskable Event Flag

Set by hardware when the associated interrupt condition occurs. This is not a standard interrupt flag; instead it is a non-maskable, asynchronous output that goes directly to the associated interrupt detector in the PMU. This will occur even if the AFE/ADC AMBA bus clock has been turned off for power savings reasons.

Write 1 to clear.

AFE_INTR.lp_comp_d_nma

Field	Bits	Default	Access	Description
lp_comp_d_nma	15	0	W1C	Low-Power Comparator D Non-Maskable Event Flag

Set by hardware when the associated interrupt condition occurs. This is not a standard interrupt flag; instead it is a non-maskable, asynchronous output that goes directly to the associated interrupt detector in the PMU. This will occur even if the AFE/ADC AMBA bus clock has been turned off for power savings reasons.

Write 1 to clear.

AFE_INTR.op_comp_a_pol

Field	Bits	Default	Access	Description
op_comp_a_pol	16	0	R/W	Op Amp Comparator A Polarity Select

- 0: Rising Edge
- 1: Falling Edge

AFE_INTR.op_comp_b_pol

Field	Bits	Default	Access	Description
op_comp_b_pol	17	0	R/W	Op Amp Comparator B Polarity Select

- 0: Rising Edge
- 1: Falling Edge

AFE_INTR.op_comp_c_pol

Field	Bits	Default	Access	Description
op_comp_c_pol	18	0	R/W	Op Amp Comparator C Polarity Select

- 0: Rising Edge
- 1: Falling Edge

AFE_INTR.op_comp_d_pol

Field	Bits	Default	Access	Description
op_comp_d_pol	19	0	R/W	Op Amp Comparator D Polarity Select

- 0: Rising Edge
- 1: Falling Edge

AFE_INTR.lp_comp_a_pol

Field	Bits	Default	Access	Description
lp_comp_a_pol	20	0	R/W	Low-Power Comparator A Polarity Select

- 0: Rising Edge
- 1: Falling Edge

AFE_INTR.lp_comp_b_pol

Field	Bits	Default	Access	Description
lp_comp_b_pol	21	0	R/W	Low-Power Comparator B Polarity Select

- 0: Rising Edge
- 1: Falling Edge

AFE_INTR.lp_comp_c_pol

Field	Bits	Default	Access	Description
lp_comp_c_pol	22	0	R/W	Low-Power Comparator C Polarity Select

- 0: Rising Edge
- 1: Falling Edge

AFE_INTR.lp_comp_d_pol

Field	Bits	Default	Access	Description
lp_comp_d_pol	23	0	R/W	Low-Power Comparator D Polarity Select

- 0: Rising Edge
- 1: Falling Edge

AFE_INTR.op_comp_a_en

Field	Bits	Default	Access	Description
op_comp_a_en	24	0	R/W	Op Amp Comparator A Interrupt Enable/–Disable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

AFE_INTR.op_comp_b_en

Field	Bits	Default	Access	Description
op_comp_b_en	25	0	R/W	Op Amp Comparator B Interrupt Enable/–Disable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

AFE_INTR.op_comp_c_en

Field	Bits	Default	Access	Description
op_comp_c_en	26	0	R/W	Op Amp Comparator C Interrupt Enable/–Disable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

AFE_INTR.op_comp_d_en

Field	Bits	Default	Access	Description
op_comp_d_en	27	0	R/W	Op Amp Comparator D Interrupt Enable/–Disable

- 0: Interrupt is disabled

- 1: Interrupt is enabled

AFE_INTR.lp_comp_a_en

Field	Bits	Default	Access	Description
lp_comp_a_en	28	0	R/W	Low-Power Comparator A Interrupt Enable/–Disable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

AFE_INTR.lp_comp_b_en

Field	Bits	Default	Access	Description
lp_comp_b_en	29	0	R/W	Low-Power Comparator B Interrupt Enable/–Disable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

AFE_INTR.lp_comp_c_en

Field	Bits	Default	Access	Description
lp_comp_c_en	30	0	R/W	Low-Power Comparator C Interrupt Enable/–Disable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

AFE_INTR.lp_comp_d_en

Field	Bits	Default	Access	Description
lp_comp_d_en	31	0	R/W	Low-Power Comparator D Interrupt Enable/Disable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

8.2.2.1.2 AFE_CTRL0**AFE_CTRL0.led_cfg_port0**

Field	Bits	Default	Access	Description
led_cfg_port0	1:0	10b	R/W	LED Port 0 Drive Configuration

- 00b: drive LED Sink Port 0 from OUTA
- 01b: drive LED Sink Port 0 from OUTB
- 1xb: disable LED Sink Port 0

AFE_CTRL0.led_cfg_port1

Field	Bits	Default	Access	Description
led_cfg_port1	3:2	10b	R/W	LED Port 1 Drive Configuration

- 00b: drive LED SinkPort 1 from OUTC
- 01b: drive LED Sink Port 1 from OUTD
- 1xb: disable LED Sink Port 1

AFE_CTRL0.clear_wud_comp_a

Field	Bits	Default	Access	Description
clear_wud_comp_a	4	0	R/W	Clear Wakeup Detect for Low-Power Comparator A

- 0: No operation
- 1: Clear wakeup detection mode for comparator

AFE_CTRL0.clear_wud_comp_b

Field	Bits	Default	Access	Description
clear_wud_comp_b	5	0	R/W	Clear Wakeup Detect for Low-Power Comparator B

- 0: No operation
- 1: Clear wakeup detection mode for comparator

AFE_CTRL0.clear_wud_comp_c

Field	Bits	Default	Access	Description
clear_wud_comp_c	6	0	R/W	Clear Wakeup Detect for Low-Power Comparator C

- 0: No operation
- 1: Clear wakeup detection mode for comparator

AFE_CTRL0.clear_wud_comp_d

Field	Bits	Default	Access	Description
clear_wud_comp_d	7	0	R/W	Clear Wakeup Detect for Low-Power Comparator D

- 0: No operation
- 1: Clear wakeup detection mode for comparator

AFE_CTRL0.en_wud_comp_a

Field	Bits	Default	Access	Description
en_wud_comp_a	9:8	00b	R/W	Set Wakeup Detect for Low-Power Comparator A

- 0xb: No operation
- 10b: Set falling edge wakeup detect mode for comparator
- 11b: Set rising edge wakeup detect mode for comparator

AFE_CTRL0.en_wud_comp_b

Field	Bits	Default	Access	Description
en_wud_comp_b	11:10	00b	R/W	Set Wakeup Detect for Low-Power Comparator B

- 0xb: No operation
- 10b: Set falling edge wakeup detect mode for comparator
- 11b: Set rising edge wakeup detect mode for comparator

AFE_CTRL0.en_wud_comp_c

Field	Bits	Default	Access	Description
en_wud_comp_c	13:12	00b	R/W	Set Wakeup Detect for Low-Power Comparator C

- 0xb: No operation
- 10b: Set falling edge wakeup detect mode for comparator
- 11b: Set rising edge wakeup detect mode for comparator

AFE_CTRL0.en_wud_comp_d

Field	Bits	Default	Access	Description
en_wud_comp_d	15:14	00b	R/W	Set Wakeup Detect for Low-Power Comparator D

- 0xb: No operation
- 10b: Set falling edge wakeup detect mode for comparator
- 11b: Set rising edge wakeup detect mode for comparator

AFE_CTRL0.in_mode_comp_a

Field	Bits	Default	Access	Description
in_mode_comp_a	17:16	00b	R/W	Low-Power Comparator A Input Mode

- 00b: Enable both N-channel and P-channel
- 01b: Enable N-channel only

- 10b: Enable P-channel only
- 11b: Reserved

AFE_CTRL0.in_mode_comp_b

Field	Bits	Default	Access	Description
in_mode_comp_b	19:18	00b	R/W	Low-Power Comparator A Input Mode

- 00b: Enable both N-channel and P-channel
- 01b: Enable N-channel only
- 10b: Enable P-channel only
- 11b: Reserved

AFE_CTRL0.in_mode_comp_c

Field	Bits	Default	Access	Description
in_mode_comp_c	21:20	00b	R/W	Low-Power Comparator C Input Mode

- 00b: Enable both N-channel and P-channel
- 01b: Enable N-channel only
- 10b: Enable P-channel only
- 11b: Reserved

AFE_CTRL0.in_mode_comp_d

Field	Bits	Default	Access	Description
in_mode_comp_d	23:22	00b	R/W	Low-Power Comparator D Input Mode

- 00b: Enable both N-channel and P-channel
- 01b: Enable N-channel only
- 10b: Enable P-channel only
- 11b: Reserved

AFE_CTRL0.bias_mode_comp_a

Field	Bits	Default	Access	Description
bias_mode_comp_a	25:24	00b	R/W	Low-Power Comparator A Bias Mode

- 00b: 0.52uA, delay 4.0us
- 01b: 1.4uA, delay 1.7us
- 10b: 2.8uA, delay 1.1us
- 11b: 5.1uA, delay 0.7us

AFE_CTRL0.bias_mode_comp_b

Field	Bits	Default	Access	Description
bias_mode_comp_b	27:26	00b	R/W	Low-Power Comparator B Bias Mode

- 00b: 0.52uA, delay 4.0us
- 01b: 1.4uA, delay 1.7us
- 10b: 2.8uA, delay 1.1us

- 11b: 5.1uA, delay 0.7us

AFE_CTRL0.bias_mode_comp_c

Field	Bits	Default	Access	Description
bias_mode_comp_c	29:28	00b	R/W	Low-Power Comparator C Bias Mode

- 00b: 0.52uA, delay 4.0us
- 01b: 1.4uA, delay 1.7us
- 10b: 2.8uA, delay 1.1us
- 11b: 5.1uA, delay 0.7us

AFE_CTRL0.bias_mode_comp_d

Field	Bits	Default	Access	Description
bias_mode_comp_d	31:30	00b	R/W	Low-Power Comparator D Bias Mode

- 00b: 0.52uA, delay 4.0us
- 01b: 1.4uA, delay 1.7us
- 10b: 2.8uA, delay 1.1us
- 11b: 5.1uA, delay 0.7us

8.2.2.1.3 AFE_CTRL1

AFE_CTRL1.tmon_current_en

Field	Bits	Default	Access	Description
tmon_current_en	0	0	R/W	Temperature Sense Current Source Enable

- 0: Disabled
- 1: Enable the current source for the temperature sensor.

AFE_CTRL1.tmon_current_val

Field	Bits	Default	Access	Description
tmon_current_val	2:1	00b	R/W	Temperature Sense Current Source Value

- 00b: 4uA
- 01b: 60uA
- 10b: 64uA
- 11b: 120uA

AFE_CTRL1.refadc_fast_pwrtn_en

Field	Bits	Default	Access	Description
refadc_fast_pwrtn_en	3	0	R/W	ADC Reference Fast Powerdown Enable

- 0: Disabled
- 1: Output powers down in milliseconds instead of seconds. Must be reset to 0 before the ref is re-powered on using refadc_outen. Can also be used to improve the REFADC slew rate (when lowering the adcrefsel setting) by setting to 1 for 10ms.

AFE_CTRL1.refdac_fast_pwrtn_en

Field	Bits	Default	Access	Description
refdac_fast_pwrtn_en	4	0	R/W	DAC Reference Fast Powerdown Enable

- 0: Disabled
- 1: Output powers down in milliseconds instead of seconds. Must be reset to 0 before the ref is re-powered on using refdac_outen. Can also be used to improve the REFDAC slew rate (when lowering the dacrefsel setting) by setting to 1 for 10ms.

AFE_CTRL1.bgextsel

Field	Bits	Default	Access	Description
bgextsel	5	0	R/W	ADC/DAC Reference Select

Selects reference basis used to generating REFADC and REFDAC:

- 0: Internal bandgap used
- 1: External REFADJ input used

AFE_CTRL1.adcrefsel

Field	Bits	Default	Access	Description
adcrefsel	7:6	00b	R/W	ADC Reference Voltage Select

When using the internal bandgap voltage as the reference basis:

- 00b: VREFADC = 1.024V
- 01b: VREFADC = 1.5V
- 10b: VREFADC = 2.048V

- 11b: VREFADC = 2.5V

When using the external VREFADJ voltage as the reference basis:

- 00b: VREFADC = 1.0000 * V1024
- 01b: VREFADC = 1.4648 * V1024
- 10b: VREFADC = 2.0000 * V1024
- 11b: VREFADC = 2.4414 * V1024

where V1024 is determined by the v1extadj field.

AFE_CTRL1.dacrefsel

Field	Bits	Default	Access	Description
dacrefsel	9:8	00b	R/W	DAC Reference Voltage Select

When using the internal bandgap voltage as the reference basis:

- 00b: VREFDAC = 1.024V
- 01b: VREFDAC = 1.5V
- 10b: VREFDAC = 2.048V
- 11b: VREFDAC = 2.5V

When using the external VREFADJ voltage as the reference basis:

- 00b: VREFDAC = 1.0000 * V1024
- 01b: VREFDAC = 1.4648 * V1024
- 10b: VREFDAC = 2.0000 * V1024
- 11b: VREFDAC = 2.4414 * V1024

where V1024 is determined by the v1extadj field.

AFE_CTRL1.dacsel

Field	Bits	Default	Access	Description
dacsel	10	1	R/W	DAC Reference Select

Selects reference for all DAC output levels:

- 0: REFADC
- 1: REFDAC

AFE_CTRL1.refadc_outen

Field	Bits	Default	Access	Description
refadc_outen	11	1	R/W	Internal ADC Reference Output Enable

- 0: Internal ADC reference powered down; REFADC may be driven externally
- 1: Internal ADC reference powered up; REFADC is driven by internal ADC reference output

AFE_CTRL1.refdac_outen

Field	Bits	Default	Access	Description
refdac_outen	12	1	R/W	DAC Reference Powerup

- 0: Internal DAC reference powered down; REFDAC may be driven externally
- 1: Internal DAC reference powered up; REFDAC is driven by internal DAC reference output

AFE_CTRL1.ref_pu

Field	Bits	Default	Access	Description
ref_pu	13	1	R/W	Reference Block Powerup

- 0: Reference block powered down
- 1: Reference block powered up

AFE_CTRL1.refadc_cp

Field	Bits	Default	Access	Description
refadc_cp	14	0	R/W	ADC Reference Compensation Enable

- 0: No additional pole
- 1: Additional pole enabled to compensate zero from external resistance

AFE_CTRL1.refdac_cp

Field	Bits	Default	Access	Description
refdac_cp	15	0	R/W	DAC Reference Compensation Enable

- 0: No additional pole
- 1: Additional pole enabled to compensate zero from external resistance

AFE_CTRL1.refadc_gain

Field	Bits	Default	Access	Description
refadc_gain	17:16	00b	R/W	Reserved Field, Do Not Modify

This field should not be modified by the user. Leave at default setting (zero) for proper operation.

AFE_CTRL1.refdac_gain

Field	Bits	Default	Access	Description
refdac_gain	19:18	00b	R/W	Reserved Field, Do Not Modify

This field should not be modified by the user. Leave at default setting (zero) for proper operation.

AFE_CTRL1.abus_page_2_0

Field	Bits	Default	Access	Description
abus_page_2_0	22:20	000b	R/W	Reserved Field, Do Not Modify

This field should not be modified by the user. Leave at default setting (zero) for proper operation.

AFE_CTRL1.tm3_tst_ib

Field	Bits	Default	Access	Description
tm3_tst_ib	23	0	R/W	Reserved Field, Do Not Modify

This field should not be modified by the user. Leave at default setting (zero) for proper operation.

AFE_CTRL1.v1extadj

Field	Bits	Default	Access	Description
v1extadj	29:25	00000b	R/W	Adjustable Trim for External Voltage Reference

When the external REFADJ input is used to generate the ADC/DAC voltage references, this value is used to determine the internal VREFADC and VREFDAC voltages that are available for selection.

$V1_{\text{extadj}} = 2$'s complement of this field.

$V1024 = VREFADJ * (87 + V1_{\text{extadj}}) / (104 + V1_{\text{extadj}})$

AFE_CTRL1.tmon_ext_sel

Field	Bits	Default	Access	Description
tmon_ext_sel	30	0	R/W	Temperature Sense Current Source Routing Select

- 0: Output from temperature sense current source is routed to the internal temperature sensor.
- 1: Output from temperature sense current source is routed to the AIN1+ pin for use by an external temperature sensor.

AFE_CTRL1.noise_filter_disable

Field	Bits	Default	Access	Description
noise_filter_disable	31	0	R/W	DAC0 and DAC1 (12bit) Noise Filter Disable

- 0: Enables noise limiting filter in DAC0/DAC1 (default)
- 1: Disables noise limiting filter

8.2.2.1.4 AFE_CTRL2

AFE_CTRL2.hyst_comp_a

Field	Bits	Default	Access	Description
hyst_comp_a	1:0	00b	R/W	Low-Power Comparator A Hysteresis Magnitude Select

Selects hysteresis magnitude setting for comparator.

- 00b: 0mV
- 01b: 7.5mV
- 10b: 15mV
- 11b: 30mV

AFE_CTRL2.hyst_comp_b

Field	Bits	Default	Access	Description
hyst_comp_b	3:2	00b	R/W	Low-Power Comparator B Hysteresis Magnitude Select

Selects hysteresis magnitude setting for comparator.

- 00b: 0mV
- 01b: 7.5mV
- 10b: 15mV
- 11b: 30mV

AFE_CTRL2.hyst_comp_c

Field	Bits	Default	Access	Description
hyst_comp_c	5:4	00b	R/W	Low-Power Comparator C Hysteresis Magnitude Select

Selects hysteresis magnitude setting for comparator.

- 00b: 0mV

- 01b: 7.5mV
- 10b: 15mV
- 11b: 30mV

AFE_CTRL2.hyst_comp_d

Field	Bits	Default	Access	Description
hyst_comp_d	7:6	00b	R/W	Low-Power Comparator D Hysteresis Magnitude Select

Selects hysteresis magnitude setting for comparator.

- 00b: 0mV
- 01b: 7.5mV
- 10b: 15mV
- 11b: 30mV

AFE_CTRL2.hy_pol_comp_a

Field	Bits	Default	Access	Description
hy_pol_comp_a	8	0	R/W	Low-Power Comparator A Hysteresis Polarity Select

Selects hysteresis polarity setting for comparator.

- 0: $V_p > V_n + V_{hys}$
- 1: $V_p < V_n - V_{hys}$

AFE_CTRL2.hy_pol_comp_b

Field	Bits	Default	Access	Description
hy_pol_comp_b	9	0	R/W	Low-Power Comparator B Hysteresis Polarity Select

Selects hysteresis polarity setting for comparator.

- 0: $V_p > V_n + V_{hys}$
- 1: $V_p < V_n - V_{hys}$

AFE_CTRL2.hy_pol_comp_c

Field	Bits	Default	Access	Description
hy_pol_comp_c	10	0	R/W	Low-Power Comparator C Hysteresis Polarity Select

Selects hysteresis polarity setting for comparator.

- 0: $V_p > V_n + V_{hys}$
- 1: $V_p < V_n - V_{hys}$

AFE_CTRL2.hy_pol_comp_d

Field	Bits	Default	Access	Description
hy_pol_comp_d	11	0	R/W	Low-Power Comparator D Hysteresis Polarity Select

Selects hysteresis polarity setting for comparator.

- 0: $V_p > V_n + V_{hys}$
- 1: $V_p < V_n - V_{hys}$

AFE_CTRL2.poweru_comp_a

Field	Bits	Default	Access	Description
poweru_comp_a	12	0	R/W	Low-Power Comparator A Powerup/Enable

Powerup/enable selection for comparator.

- 0: Disabled (powered down)
- 1: Enabled (powered up)

AFE_CTRL2.poweru_comp_b

Field	Bits	Default	Access	Description
poweru_comp_b	13	0	R/W	Low-Power Comparator B Powerup/Enable

Powerup/enable selection for comparator.

- 0: Disabled (powered down)
- 1: Enabled (powered up)

AFE_CTRL2.poweru_comp_c

Field	Bits	Default	Access	Description
poweru_comp_c	14	0	R/W	Low-Power Comparator C Powerup/Enable

Powerup/enable selection for comparator.

- 0: Disabled (powered down)
- 1: Enabled (powered up)

AFE_CTRL2.poweru_comp_d

Field	Bits	Default	Access	Description
poweru_comp_d	15	0	R/W	Low-Power Comparator D Powerup/Enable

Powerup/enable selection for comparator.

- 0: Disabled (powered down)
- 1: Enabled (powered up)

AFE_CTRL2.dacout_en0

Field	Bits	Default	Access	Description
dacout_en0	16	0	R/W	Connects dac_or0 output (selected by dac_sel_a) to SCM0 pin.

- 0: No connect
- 1: Connect dac_or0 -> SCM0

AFE_CTRL2.dacout_en1

Field	Bits	Default	Access	Description
dacout_en1	17	0	R/W	Connects dac_or1 output (selected by dac_sel_b) to SNO0 pin.

- 0: No connect
- 1: Connect dac_or1 -> SNO0

AFE_CTRL2.dacout_en2

Field	Bits	Default	Access	Description
dacout_en2	18	0	R/W	Connects dac_or2 output (selected by dac_sel_c) to SCM1 pin.

- 0: No connect
- 1: Connect dac_or2 -> SCM1

AFE_CTRL2.dacout_en3

Field	Bits	Default	Access	Description
dacout_en3	19	0	R/W	Connects dac_or3 output (selected by dac_sel_d) to SNO1 pin.

- 0: No connect
- 1: Connect dac_or3 -> SNO1

AFE_CTRL2.scm_or_sel

Field	Bits	Default	Access	Description
scm_or_sel	22:20	000b	R/W	Selects source for SCM_or signal

SCMx pad select into SCM_or signal feeding OPAMP negative input MUX, comparator positive input, and OPAMP negative input pad

- 001b: Connect SCM0 -> SCM_or
- 010b: Connect SCM1 -> SCM_or
- 011b: Connect SCM2 -> SCM_or
- 100b: Connect SCM3 -> SCM_or
- else: High impedance -> SCM_or

AFE_CTRL2.sno_or_sel

Field	Bits	Default	Access	Description
sno_or_sel	25:23	000b	R/W	Selects source for SNO_or signal

SNOx pad select into SNO_or signal

AFE_CTRL2.dac0_sel

Field	Bits	Default	Access	Description
dac0_sel	26	0	R/W	Internal input stage select for dac0

Connects DAC0p or DAC0n to dac0 input used on DAC_or select MUX and low-power comparator positive input MUX

- 0: DAC0p -> dac0
- 1: DAC0n -> dac0

AFE_CTRL2.dac1_sel

Field	Bits	Default	Access	Description
dac1_sel	27	0	R/W	Internal input stage select for dac1

Connects DAC1P or DAC1N to dac1 input used on DAC_or select MUX and low-power comparator positive input MUX

- 0: DAC1p -> dac1
- 1: DAC1n -> dac1

8.2.2.1.5 AFE_CTRL3**AFE_CTRL3.pu_opamp_a**

Field	Bits	Default	Access	Description
pu_opamp_a	12	0	R/W	Op Amp A Power Up

- 0: Op amp is powered down
- 1: Op amp is powered up

AFE_CTRL3.pu_opamp_b

Field	Bits	Default	Access	Description
pu_opamp_b	13	0	R/W	Op Amp B Power Up

- 0: Op amp is powered down
- 1: Op amp is powered up

AFE_CTRL3.pu_opamp_c

Field	Bits	Default	Access	Description
pu_opamp_c	14	0	R/W	Op Amp C Power Up

- 0: Op amp is powered down
- 1: Op amp is powered up

AFE_CTRL3.pu_opamp_d

Field	Bits	Default	Access	Description
pu_opamp_d	15	0	R/W	Op Amp D Power Up

- 0: Op amp is powered down
- 1: Op amp is powered up

AFE_CTRL3.gnd_sel_opamp_a

Field	Bits	Default	Access	Description
gnd_sel_opamp_a	16	0	R/W	Op Amp A Positive Input Ground Select

- 0: Internal ground switch for op amp A disabled
- 1: Ground switch is enabled, connecting INA+ to ground internally.

AFE_CTRL3.gnd_sel_opamp_b

Field	Bits	Default	Access	Description
gnd_sel_opamp_b	17	0	R/W	Op Amp B Positive Input Ground Select

- 0: Internal ground switch for op amp B disabled
- 1: Ground switch is enabled, connecting INB+ to ground internally.

AFE_CTRL3.gnd_sel_opamp_c

Field	Bits	Default	Access	Description
gnd_sel_opamp_c	18	0	R/W	Op Amp C Positive Input Ground Select

- 0: Internal ground switch for op amp C disabled
- 1: Ground switch is enabled, connecting INC+ to ground internally.

AFE_CTRL3.gnd_sel_opamp_d

Field	Bits	Default	Access	Description
gnd_sel_opamp_d	19	0	R/W	Op Amp D Positive Input Ground Select

- 0: Internal ground switch for op amp D disabled
- 1: Ground switch is enabled, connecting IND+ to ground internally.

AFE_CTRL3.close_spst0

Field	Bits	Default	Access	Description
close_spst0	20	0	R/W	Switch Close for SPST0

If ADC_INTR.spst_sw0_ctrl = 0, this bit controls the state of SPST switch 0 (from SNO0 to SCM0) as follows:

- 0: Switch open
- 1: Switch closed

If ADC_INTR.spst_sw0_ctrl = 1, the setting of this bit has no effect, and the state of SPST switch 0 is controlled by the output of pulse train PT8.

AFE_CTRL3.close_spst1

Field	Bits	Default	Access	Description
close_spst1	21	0	R/W	Switch Close for SPST1

If ADC_INTR.spst_sw1_ctrl = 0, this bit controls the state of SPST switch 1 (from SNO1 to SCM1) as follows:

- 0: Switch open
- 1: Switch closed

If ADC_INTR.spst_sw1_ctrl = 1, the setting of this bit has no effect, and the state of SPST switch 1 is controlled by the output of pulse train PT9.

AFE_CTRL3.close_spst2

Field	Bits	Default	Access	Description
close_spst2	22	0	R/W	Switch Close for SPST2

If ADC_INTR.spst_sw2_ctrl = 0, this bit controls the state of SPST switch 2 (from SNO2 to SCM2) as follows:

- 0: Switch open
- 1: Switch closed

If ADC_INTR.spst_sw2_ctrl = 1, the setting of this bit has no effect, and the state of SPST switch 2 is controlled by the output of pulse train PT10.

AFE_CTRL3.close_spst3

Field	Bits	Default	Access	Description
close_spst3	23	0	R/W	Switch Close for SPST3

If ADC_INTR.spst_sw3_ctrl = 0, this bit controls the state of SPST switch 3 (from SNO3 to SCM3) as follows:

- 0: Switch open
- 1: Switch closed

If ADC_INTR.spst_sw3_ctrl = 1, the setting of this bit has no effect, and the state of SPST switch 3 is controlled by the output of pulse train PT11.

AFE_CTRL3.en_pch_opamp_a

Field	Bits	Default	Access	Description
en_pch_opamp_a	24	0	R/W	Op Amp A P-Channel Input Stage Enable

- 0: P-Channel input stage for this op amp is disabled.
- 1: P-Channel input stage for this op amp is enabled.

AFE_CTRL3.en_pch_opamp_b

Field	Bits	Default	Access	Description
en_pch_opamp_b	25	0	R/W	Op Amp B P-Channel Input Stage Enable

- 0: P-Channel input stage for this op amp is disabled.
- 1: P-Channel input stage for this op amp is enabled.

AFE_CTRL3.en_pch_opamp_c

Field	Bits	Default	Access	Description
en_pch_opamp_c	26	0	R/W	Op Amp C P-Channel Input Stage Enable

- 0: P-Channel input stage for this op amp is disabled.
- 1: P-Channel input stage for this op amp is enabled.

AFE_CTRL3.en_pch_opamp_d

Field	Bits	Default	Access	Description
en_pch_opamp_d	27	0	R/W	Op Amp D P-Channel Input Stage Enable

- 0: P-Channel input stage for this op amp is disabled.
- 1: P-Channel input stage for this op amp is enabled.

AFE_CTRL3.en_nch_opamp_a

Field	Bits	Default	Access	Description
en_nch_opamp_a	28	0	R/W	Op Amp A N-Channel Input Stage Enable

- 0: N-Channel input stage for this op amp is disabled.
- 1: N-Channel input stage for this op amp is enabled.

AFE_CTRL3.en_nch_opamp_b

Field	Bits	Default	Access	Description
en_nch_opamp_b	29	0	R/W	Op Amp B N-Channel Input Stage Enable

- 0: N-Channel input stage for this op amp is disabled.
- 1: N-Channel input stage for this op amp is enabled.

AFE_CTRL3.en_nch_opamp_c

Field	Bits	Default	Access	Description
en_nch_opamp_c	30	0	R/W	Op Amp C N-Channel Input Stage Enable

- 0: N-Channel input stage for this op amp is disabled.
- 1: N-Channel input stage for this op amp is enabled.

AFE_CTRL3.en_nch_opamp_d

Field	Bits	Default	Access	Description
en_nch_opamp_d	31	0	R/W	Op Amp D N-Channel Input Stage Enable

- 0: N-Channel input stage for this op amp is disabled.
- 1: N-Channel input stage for this op amp is enabled.

8.2.2.1.6 AFE_CTRL4

AFE_CTRL4.p_in_sel_opamp_a

Field	Bits	Default	Access	Description
p_in_sel_opamp_a	1:0	00b	R/W	Op Amp A Positive Input Select

Connects the positive (noninverting) input of op amp A internally as follows.

- 0: Connected to pin INA+
- 1: Connected to the output from DAC0–3
- 2: Connected to pin SNO[0-3] or to high impedance (as selected by AFE_CTRL2.sno_or_sel)
- 3: Connected to both pin INA+ AND the DAC[0-3] output

Note Setting 3 connects an unbuffered DAC output directly to pin INA+. This setting must not be used when AFE_CTRL3.gnd_sel_opamp_a is set to 1, since this connects INA+ directly to ground and so would effectively short the DAC output to ground.

AFE_CTRL4.p_in_sel_opamp_b

Field	Bits	Default	Access	Description
p_in_sel_opamp_b	3:2	00b	R/W	Op Amp B Positive Input Select

Connects the positive (noninverting) input of op amp B internally as follows.

- 0: Connected to pin INB+
- 1: Connected to the output from DAC0–3
- 2: Connected to pin SNO[0-3] or to high impedance (as selected by AFE_CTRL2.sno_or_sel)
- 3: Connected to both pin INB+ AND the DAC[0-3] output

Note Setting 3 connects an unbuffered DAC output directly to pin INB+. This setting must not be used when AFE_CTRL3.gnd_sel_opamp_b is set to 1, since this connects INB+ directly to ground and so would effectively short the DAC output to ground.

AFE_CTRL4.p_in_sel_opamp_c

Field	Bits	Default	Access	Description
p_in_sel_opamp_c	5:4	00b	R/W	Op Amp C Positive Input Select

Connects the positive (noninverting) input of op amp C internally as follows.

- 0: Connected to pin INC+
- 1: Connected to the output from DAC0–3
- 2: Connected to pin SNO[0-3] or to high impedance (as selected by AFE_CTRL2.sno_or_sel)
- 3: Connected to both pin INC+ AND the DAC[0-3] output

Note Setting 3 connects an unbuffered DAC output directly to pin INC+. This setting must not be used when AFE_CTRL3.gnd_sel_opamp_c is set to 1, since this connects INC+ directly to ground and so would effectively short the DAC output to ground.

AFE_CTRL4.p_in_sel_opamp_d

Field	Bits	Default	Access	Description
p_in_sel_opamp_d	7:6	00b	R/W	Op Amp D Positive Input Select

Connects the positive (noninverting) input of op amp D internally as follows.

- 0: Connected to pin IND+
- 1: Connected to the output from DAC0–3
- 2: Connected to pin SNO[0-3] or to high impedance (as selected by AFE_CTRL2.sno_or_sel)
- 3: Connected to both pin IND+ AND the DAC[0-3] output

Note Setting 3 connects an unbuffered DAC output directly to pin IND+. This setting must not be used when AFE_CTRL3.gnd_sel_opamp_d is set to 1, since this connects IND+ directly to ground and so would effectively short the DAC output to ground.

AFE_CTRL4.n_in_sel_opamp_a

Field	Bits	Default	Access	Description
n_in_sel_opamp_a	9:8	00b	R/W	Op Amp A Negative Input Select

Connects the negative (inverting) input of op amp A internally as follows.

- 0: Connected to pin INA-
- 1: Connected to pin OUTA (voltage follower mode)
- 2: Connected to pin SCM[0-3] or to high impedance (as selected by AFE_CTRL2.scm_or_sel)
- 3: Connected to both pin SCM[0-3]/high impedance AND INA-

AFE_CTRL4.n_in_sel_opamp_b

Field	Bits	Default	Access	Description
n_in_sel_opamp_b	11:10	00b	R/W	Op Amp B Negative Input Select

Connects the negative (inverting) input of op amp B internally as follows.

- 0: Connected to pin INB-
- 1: Connected to pin OUTB (voltage follower mode)
- 2: Connected to pin SCM[0-3] or to high impedance (as selected by AFE_CTRL2.scm_or_sel)
- 3: Connected to both pin SCM[0-3]/high impedance AND INB-

AFE_CTRL4.n_in_sel_opamp_c

Field	Bits	Default	Access	Description
n_in_sel_opamp_c	13:12	00b	R/W	Op Amp C Negative Input Select

Connects the negative (inverting) input of op amp C internally as follows.

- 0: Connected to pin INC-
- 1: Connected to pin OUTC (voltage follower mode)

- 2: Connected to pin SCM[0-3] or to high impedance (as selected by AFE_CTRL2.scm_or_sel)
- 3: Connected to both pin SCM[0-3]/high impedance AND INC-

AFE_CTRL4.n_in_sel_opamp_d

Field	Bits	Default	Access	Description
n_in_sel_opamp_d	15:14	00b	R/W	Op Amp D Negative Input Select

Connects the negative (inverting) input of op amp D internally as follows.

- 0: Connected to pin IND-
- 1: Connected to pin OUTD (voltage follower mode)
- 2: Connected to pin SCM[0-3] or to high impedance (as selected by AFE_CTRL2.scm_or_sel)
- 3: Connected to both pin SCM[0-3]/high impedance AND IND-

AFE_CTRL4.dac_sel_a

Field	Bits	Default	Access	Description
dac_sel_a	17:16	00b	R/W	DAC Output Mux Select to dac_or0 Stage

MUX that selects 1 of 4 DAC outputs for internal signal DAC_or0, which can be further selected for the OPAMPA positive input, OPAMPA negative pad, or SNO0 pad via other MUX selections

- 2'd0: dac0 -> DAC_or0 (dac0 controlled by DAC0_sel)
- 2'd1: dac1 -> DAC_or0 (dac1 controlled by DAC1_sel)
- 2'd2: DAC2p1 -> DAC_or0
- 2'd3: DAC3p1 -> DAC_or0

AFE_CTRL4.dac_sel_b

Field	Bits	Default	Access	Description
dac_sel_b	19:18	00b	R/W	DAC Output Mux Select to dac_or1 Stage

MUX that selects 1 of 4 DAC outputs for internal signal DAC_or1, which can be further selected for the OPAMPB positive input, OPAMPB negative pad, or SNO1 pad via other MUX selections

- 2'd0: dac0 -> DAC_or1 (dac0 controlled by DAC0_sel)
- 2'd1: dac1 -> DAC_or1 (dac1 controlled by DAC1_sel)
- 2'd2: DAC2p1 -> DAC_or1
- 2'd3: DAC3p1 -> DAC_or1

AFE_CTRL4.dac_sel_c

Field	Bits	Default	Access	Description
dac_sel_c	21:20	00b	R/W	DAC Output Mux Select to dac_or2 Stage

MUX that selects 1 of 4 DAC outputs for internal signal DAC_or2, which can be further selected for the OPAMPC positive input, OPAMPC negative pad, or SNO2 pad via other MUX selections

- 2'd0: dac0 -> DAC_or2 (dac0 controlled by DAC0_sel)
- 2'd1: dac1 -> DAC_or2 (dac1 controlled by DAC1_sel)
- 2'd2: DAC2p1 -> DAC_or2
- 2'd3: DAC3p1 -> DAC_or2

AFE_CTRL4.dac_sel_d

Field	Bits	Default	Access	Description
dac_sel_d	23:22	00b	R/W	DAC Output Mux Select to dac_or3 Stage

MUX that selects 1 of 4 DAC outputs for internal signal DAC_or3, which can be further selected for the OPAMPD positive input, OPAMPD negative pad, or SNO3 pad via other MUX selections

- 2'd0: dac0 -> DAC_or3 (dac0 controlled by DAC0_sel)
- 2'd1: dac1 -> DAC_or3 (dac1 controlled by DAC1_sel)
- 2'd2: DAC2p1 -> DAC_or3
- 2'd3: DAC3p1 -> DAC_or3

AFE_CTRL4.npad_sel_a

Field	Bits	Default	Access	Description
npad_sel_a	25:24	00b	R/W	Pad Internal Connect to INA-

INA- pad select (states 1 and 3 connect two pads together leading to potentially large current flow)

- 2'd0: hiZ -> INA-
- 2'd1: LED Observe Port 0 <-> INA-
- 2'd2: DAC_or0 -> INA-
- 2'd3: DAC_or0 -> INA- <-> LED Observe Port 0

AFE_CTRL4.npad_sel_b

Field	Bits	Default	Access	Description
npad_sel_b	27:26	00b	R/W	Pad Internal Connect to INB-

INB- pad select (states 1 and 3 connect two pads together leading to potentially large current flow)

- 2'd0: hiZ -> INB-
- 2'd1: LED Observe Port 0 <-> INB-
- 2'd2: DAC_or1 -> INB-
- 2'd3: DAC_or1 -> INB- <-> LED Observe Port 0

AFE_CTRL4.npad_sel_c

Field	Bits	Default	Access	Description
npad_sel_c	29:28	00b	R/W	Pad Internal Connect to INC-

INC- pad select (states 1 and 3 connect two pads together leading to potentially large current flow)

- 2'd0: hiZ -> INC-
- 2'd1: LED Observe Port 1 <-> INC-
- 2'd2: DAC_or2 -> INC-
- 2'd3: DAC_or2 -> INC- <-> LED Observe Port 1

AFE_CTRL4.npad_sel_d

Field	Bits	Default	Access	Description
npad_sel_d	31:30	00b	R/W	Pad Internal Connect to IND-

IND- pad select (states 1 and 3 connect two pads together leading to potentially large current flow)

- 2'd0: hiZ -> IND-

- 2'd1: LED Observe Port 1 <-> IND-
- 2'd2: DAC_or3 -> IND-
- 2'd3: DAC_or3 -> IND- <-> LED Observe Port 1

8.2.2.1.7 AFE_CTRL5

AFE_CTRL5.pos_in_sel_comp_a

Field	Bits	Default	Access	Description
pos_in_sel_comp_a	2:0	000b	R/W	Low-Power Comparator A Positive Input Select

- 0: INA+
- 1: SCM0-3 pin
- 2: DAC1 positive/negative output
- 3: DAC3 output
- 4: LED Observe Port 0
- 5: DAC1 positive/negative output AND INA+
- 6: DAC3 output AND INA+
- 7: DAC1 positive/negative output AND SCM0

AFE_CTRL5.pos_in_sel_comp_b

Field	Bits	Default	Access	Description
pos_in_sel_comp_b	5:3	000b	R/W	Low-Power Comparator B Positive Input Select

- 0: INB+

- 1: SCM0-3 pin
- 2: DAC1 positive/negative output
- 3: DAC3 output
- 4: LED Observe Port 0
- 5: DAC1 positive/negative output AND INB+
- 6: DAC3 output AND INB+
- 7: DAC1 positive/negative output AND SCM1

AFE_CTRL5.pos_in_sel_comp_c

Field	Bits	Default	Access	Description
pos_in_sel_comp_c	8:6	000b	R/W	Low-Power Comparator C Positive Input Select

- 0: INC+
- 1: SCM0-3 pin
- 2: DAC1 positive/negative output
- 3: DAC3 output
- 4: LED Observe Port 1
- 5: DAC1 positive/negative output AND INC+
- 6: DAC3 output AND INC+
- 7: DAC1 positive/negative output AND SCM2

AFE_CTRL5.pos_in_sel_comp_d

Field	Bits	Default	Access	Description
pos_in_sel_comp_d	11:9	000b	R/W	Low-Power Comparator D Positive Input Select

- 0: IND+
- 1: SCM0-3 pin
- 2: DAC1 positive/negative output
- 3: DAC3 output
- 4: LED Observe Port 1
- 5: DAC1 positive/negative output AND IND+
- 6: DAC3 output AND IND+
- 7: DAC1 positive/negative output AND SCM3

AFE_CTRL5.neg_in_sel_comp_a

Field	Bits	Default	Access	Description
neg_in_sel_comp_a	14:12	000b	R/W	Low-Power Comparator A Negative Input Select

- 0: INA-
- 1: SNO0-3 pin
- 2: DAC0 positive/negative output
- 3: DAC2 output
- 4: LED Observe Port 0
- 5: DAC0 positive/negative output AND INA-
- 6: DAC2 output AND INA-

- 7: DAC2 output AND SNO0

AFE_CTRL5.neg_in_sel_comp_b

Field	Bits	Default	Access	Description
neg_in_sel_comp_b	17:15	000b	R/W	Low-Power Comparator B Negative Input Select

- 0: INB-
- 1: SNO0-3 pin
- 2: DAC0 positive/negative output
- 3: DAC2 output
- 4: LED Observe Port 0
- 5: DAC0 positive/negative output AND INB-
- 6: DAC2 output AND INB-
- 7: DAC2 output AND SNO1

AFE_CTRL5.neg_in_sel_comp_c

Field	Bits	Default	Access	Description
neg_in_sel_comp_c	20:18	000b	R/W	Low-Power Comparator C Negative Input Select

- 0: INC-
- 1: SNO0-3 pin
- 2: DAC0 positive/negative output

- 3: DAC2 output
- 4: LED Observe Port 1
- 5: DAC0 positive/negative output AND INC-
- 6: DAC2 output AND INC-
- 7: DAC2 output AND SNO2

AFE_CTRL5.neg_in_sel_comp_d

Field	Bits	Default	Access	Description
neg_in_sel_comp_d	23:21	000b	R/W	Low-Power Comparator D Negative Input Select

- 0: IND-
- 1: SNO0-3 pin
- 2: DAC0 positive/negative output
- 3: DAC2 output
- 4: LED Observe Port 1
- 5: DAC0 positive/negative output AND IND-
- 6: DAC2 output AND IND-
- 7: DAC2 output AND SNO3

AFE_CTRL5.op_cmp_a

Field	Bits	Default	Access	Description
op_cmp_a	24	0	R/W	Mode Select for Op Amp A

- 0: Op Amp Mode
- 1: Comparator Mode

AFE_CTRL5.op_cmp_b

Field	Bits	Default	Access	Description
op_cmp_b	25	0	R/W	Mode Select for Op Amp B

- 0: Op Amp Mode
- 1: Comparator Mode

AFE_CTRL5.op_cmp_c

Field	Bits	Default	Access	Description
op_cmp_c	26	0	R/W	Mode Select for Op Amp C

- 0: Op Amp Mode
- 1: Comparator Mode

AFE_CTRL5.op_cmp_d

Field	Bits	Default	Access	Description
op_cmp_d	27	0	R/W	Mode Select for Op Amp D

- 0: Op Amp Mode
- 1: Comparator Mode

8.3 ADC

8.3.1 ADC Overview

The **MAX32600** features a 16-bit analog-to-digital converter (ADC) with a 16-channel analog input multiplexer as shown in the [ADC Signal Chain Diagram](#) below. This multiplexer selects input from any one of 16 input lines in single-ended mode or two of eight input pairs in differential mode. The differential mode supports fully differential signal inputs.

The Programmable Gain Amplifier (PGA) allows programmable gain settings of 1X, 2X, 4X, and 8X before the input sample is converted. Optionally, the PGA can be bypassed so that the multiplexer outputs are passed directly to the ADC to save power.

The **MAX32600** 16-bit ADC supports the following features:

- Configurable clock rate
- Sample rate control up to 500ksps
- Single-ended and differential input modes with unipolar and bipolar signal conditioning
 - Bipolar input ranges ($\pm V_{REF}$, $\pm V_{REF}/2$)
 - Unipolar input range (0 to V_{REF})
- Over-sampling and decimation filtering
- Ability to synchronize to DAC waveform generation
- Programmable out of range detection

8.3.2 ADC Architecture

The ADC on the **MAX32600** is a cyclic analog-to-digital converter. This architecture - sometimes referred to as a recycling architecture - does not alter the reference voltage; rather, any error or residue of the amplifier is doubled. There are certain similarities with a Pipeline ADC: each cycle calculates 1-bit. However, unlike a Pipeline ADC, the amplified value of the input is summed to a reference voltage and residue voltage from this process is further amplified during each successive

cycle until the desired resolution is reached. This corresponds with the output value of the ADC.

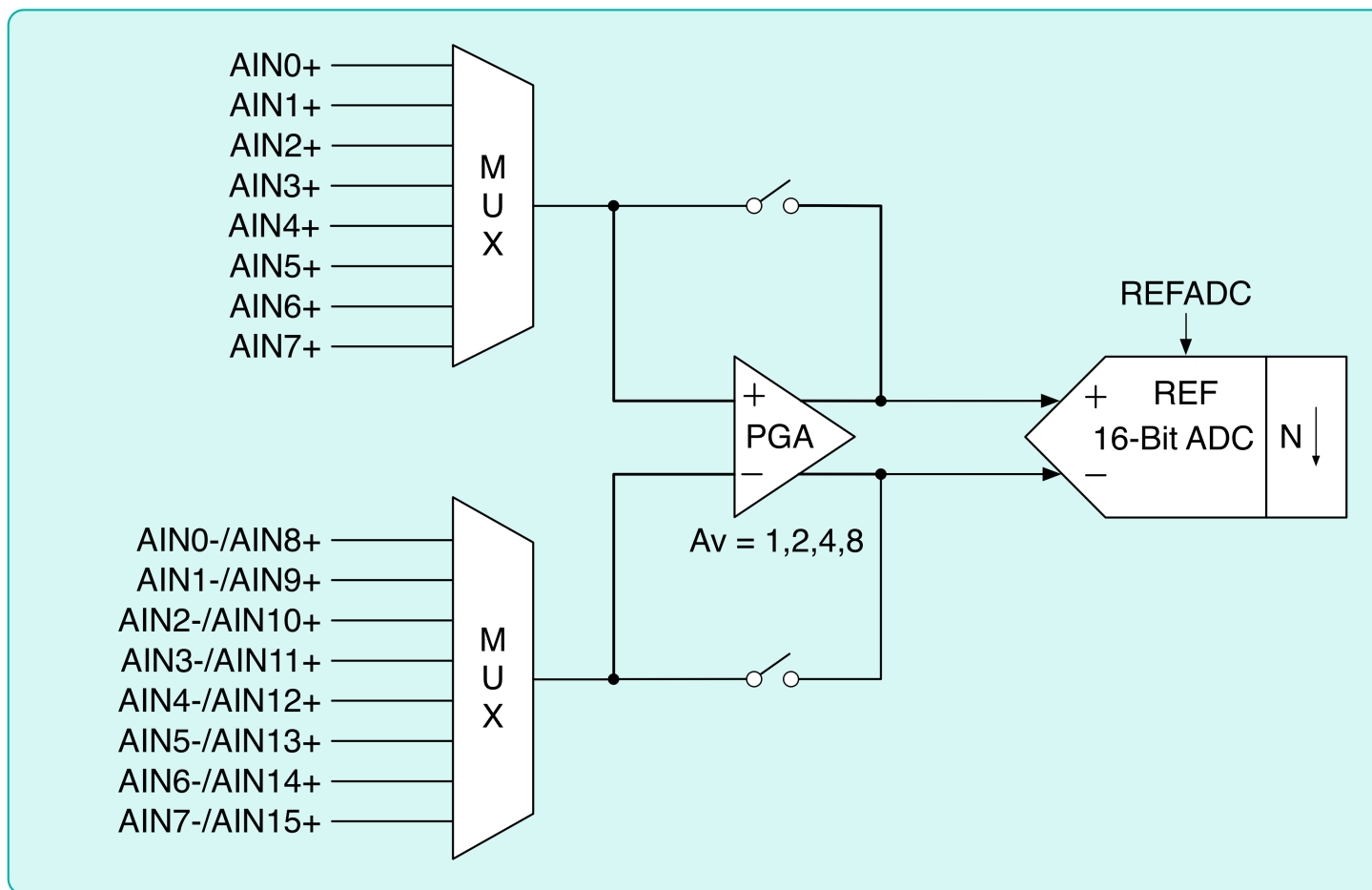


Figure 8.5: ADC Signal Chain Diagram

8.3.3 ADC Operation

The control registers for the ADC can be accessed directly by the CPU to setup, begin, and retrieve the results of ADC conversion operations. For higher performance, it is often required (depending on the application) to put the CPU into the lower power mode [Low Power Mode 2 \(LP2: Peripheral Management Unit\)](#) to reduce overall clock noise to achieve the highest possible accuracy of the measured signal.

When performing ADC operations with the CPU in LP2: PMU, the [Peripheral Management Unit \(PMU\)](#) can be used to monitor the progress of ADC conversion sequences, set up conditions for different portions of an analog conversion operation, transfer resulting sample data from the ADC output FIFO to the main system RAM, and many other functions. Since the PMU is capable of operating independently while the Cortex-M3 CPU is in Sleep mode, it is possible to perform a wide variety of complex analog tasks without needing to wake up the CPU during the process. See [Power Ecosystem and Operating Modes](#) for further information.

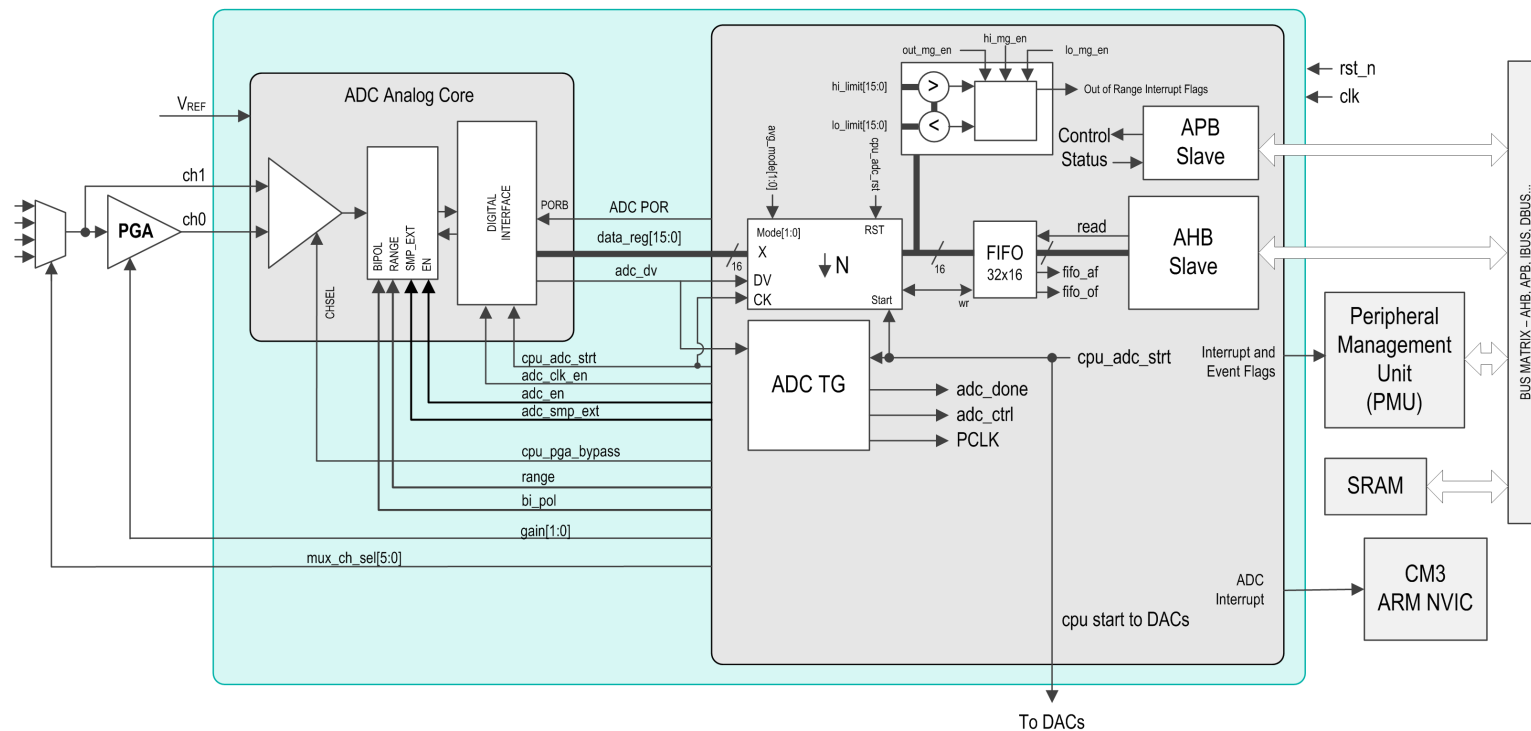


Figure 8.6: ADC Block Diagram

Some of the parameters that can be set for ADC conversion operations in PMU or CPU control modes include:

- Sample rate control
- Oversampling and decimation filter operations
 - The burst sample rate is always the maximum sample rate
 - Programmable burst length for averaging operations
- Channel scan capability, allowing the ADC to switch between multiple channel sources in a conversion sequence
 - Up to eight scan descriptors can be used, allowing eight different channel configurations in a sequence
 - Separate channel select, single-ended or differential mode select, and PGA gain settings for each scan descriptor
- Continuous or Pulsed scanning modes (Pulsed includes a sleep period between conversion bursts)

8.3.4 ADC Configuration

Using the ADC requires setting up the ADC for the specific mode required for the measurement. The following are the steps required to configure and use the ADC. Each of these are described below with details on configuration options.

1. [Set up the ADC Peripheral Clock](#)
2. [Select a reference voltage](#)
3. [Set up input mode \(i.e., single-ended or differential mode\)](#)
4. [Configure the Input Mux for the channels that will be measured](#)
5. [Set up the ADC scan mode \(single, continuous, burst, etc.\)](#)
6. [Configure ADC interrupts](#)
7. [Configure the PGA \(bypass, gain, etc.\)](#)
8. [Set up the ADC sample rate](#)
9. [Start the measurement](#)

8.3.4.1 Peripheral Clock Configuration

The **MAX32600** supports multiple clock source options to operate the ADC.

To use the ADC, it is necessary to select the source for the ADC Peripheral Clock, select the clock divisor, and disable clock gating to generate the ADC Peripheral Clock (PCLK).

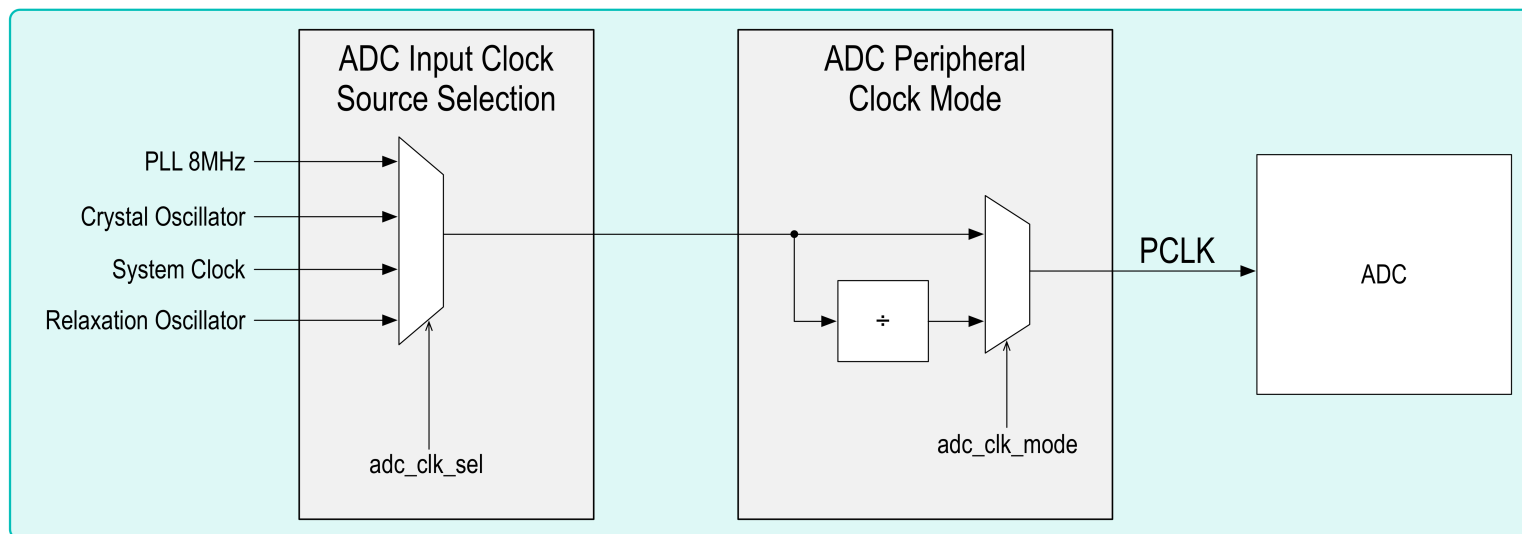


Figure 8.7: ADC Peripheral Clock Configuration

Input Clock Source Selection

The input clock to the ADC can be selected from the system clock, PLL generated clock, external high-frequency clock, external crystal direct input, or internal 24MHz relaxation oscillator. Using the external 8MHz crystal input directly will provide the optimal clock source and highest accuracy with the ADC. If an external 8MHz crystal is not available, it is recommended to use the PLL 8MHz output. Lower sampling rate(s) and input frequency signal(s) are less susceptible to aperture jitter; thus, the relaxation oscillator can be used as the clock source in these applications. To select the ADC input clock source, set the register [CLKMAN_CLK_CTL.adc_source_select](#) as shown in the table below.

adc_source_select	ADC Input Clock Source
00b	System Clock

adc_source_select	ADC Input Clock Source
01b	PLL 8MHz Output
10b	External High Frequency Clock
11b	24MHz Relaxation Oscillator

Peripheral Clock Mode (Clock Divisor)

The ADC peripheral clock, PCLK, is generated based on the ADC input clock source as set in the previous section. The ADC Peripheral Clock directly clocks the ADC internally and is based on a divisor of the ADC Input Clock Source Selection. To set PCLK, register field [ADC_CTRL0.adc_clk_mode](#) is used.

adc_clk_mode	PCLK	Notes
000b	PCLK = Input Clock / 1	PLL generates 8MHz clock
001b	PCLK = Input Clock / 2	
010b	PCLK = Input Clock / 3	Default State
011b	PCLK = Input Clock / 4	
100b	PCLK = Input Clock / 6	
101b	PCLK = Input Clock / 8	
110b	PCLK = Input Clock / 12	
111b	Reserved	

Clock Gating

Prior to using the ADC, clock gating must be turned off by setting [CLKMAN_CLK_CTRL.adc_gate_n](#) to 1. When the ADC is not being used, [CLKMAN_CLK_CTRL.adc_gate_n](#) should be set to 0 to ensure lowest power operation.

Clock Enable

Once all of the clock settings are set up (and prior to taking a measurement), set [ADC_CTRL0.adc_clk_en](#) to 1. This enables the ADC clock tree and PCLK generation.

8.3.4.2 Reference Voltage Configuration

The **MAX32600** supports internal or external ADC reference voltage options. The [AFE_CTRL1.bgextsel](#) register selects the reference source (REFADC). Setting the register field to 0 uses internal bandgap; setting the register field to 1 selects external reference (REFADJ).

It is required to power up the internal reference by setting [AFE_CTRL1.refadc_outen](#) register field to 1. If using an external reference, this field must be set to 0 to allow selection of an external reference voltage.

If [AFE_CTRL1.bgextsel](#) register is set to 1, the adjustable external ADC reference voltage must be defined by selecting the desired [AFE_CTRL1.adcrefsel](#) bit field setting.

adcrefsel	ADC reference voltage output
00b	1.024V
01b	1.5V
10b	2.048V
11b	2.5V

8.3.4.3 Input Modes

The ADC allows for both single-ended and differential input measurement modes. Internally, all ADC conversions are differential at the hardware level; the single-ended/differential mode select determines which of the external analog input signals are used as part of the analog-to-digital conversion process.

When single-ended mode is used, the negative input to the ADC is internally grounded to V_{AGND} . This is to minimize error against the reference ground. The input mux setting for the ADC determines which signal will be connected to the ADC positive input port.

In differential mode, the ADC input mux setting controls two signal connections to the ADC positive and negative inputs. If the negative input signal pin is then grounded, this setup becomes identical to the single-ended mode configuration.

Selection of single-ended or differential input modes is shown in [ADC Input Mux Selection Table](#).

8.3.4.3.1 Signal Conditioning

The **MAX32600** ADC features two signal conditioning modes: bipolar and unipolar mode. Bipolar mode must be used if the negative input could exceed the positive input. Unipolar mode can be used if the positive input will always be greater than the negative input. When using bipolar mode, it is optimal to enable differential input mode; using bipolar mode with single-ended input mode degrades signal resolution. In most cases, unipolar mode will be used in tandem with single-ended input mode.

The register field [ADC_CTRL0.bi_pol](#) is used to select either unipolar or bipolar operation. Setting this field to 0, the default state, indicates unipolar mode; setting it to 1 indicates bipolar operation.

When using bipolar mode, the input range must also be configured using the register field [ADC_CTRL0.range](#). A value of 0, default state, indicates a range of $-V_{REF}/2$

to $+V_{REF}/2$. Setting the `ADC_CTRL0.range` field to 1 indicates a range of $-V_{REF}$ to $+V_{REF}$.

Note There are instances, however, when the signal-to-noise ratio (SNR) can be improved by using differential input mode with unipolar signal conditioning. For example, if $V_{REF} = 2.0V$ and `AIN0+` is used to measure a signal between 2.0 to 2.5V, unipolar mode and single-ended input mode are not valid because the reference is out of range. A valid measurement can be taken if `AIN0-` is DC biased at 2.0V and unipolar and differential modes are enabled. A PGA gain of 4X would give a full-scale result.

8.3.4.4 Input Multiplexer

The Input Mux controls which analog input (or pair of inputs when operating in differential input mode) is routed to the PGA (or directly to the ADC when the PGA is set to bypass mode). There are 16 external analog inputs, which can be used as single-ended inputs or as differential pairs. Any mixture of single-ended and differential is also supported. When used in single-ended mode, the negative input of the PGA and/or ADC is internally connected to V_{AGND} .

In addition to the external analog inputs, there are several different internal analog inputs and input pairs available for selection. These include:

- $\pm V_{DDA3}$: Enables measurement of a voltage that is greater than the reference voltage. The input voltage must still be $\leq V_{DDA3}$.
- Internal temperature sensor, V_{BE} , and resistor taps.
- Op amp outputs: `OUTA`, `OUTB`, `OUTC`, and `OUTD`.
- `INA+`
- `V_{REFDAC}` and `V_{REFADC}` output buffers
- $(V_{DDA3ADC})/4$
- Internal Temperature Sensor (`TMON`)
- `scm_or`
- `sno_or`

Note All internal Input Mux connections (such as `OUTA`, `INA+`) should only be used with signal frequencies less than 10kHz, otherwise THD and SINAD could be adversely affected.

Input Mux Selection

The fields `ADC_PGA_CTRL.mux_ch_sel` and `ADC_PGA_CTRL.mux_diff` select the analog input to be used and, when appropriate, whether single-ended or differential mode will be used. The following table describes all input possibilities; "X" in this table represents a "don't care" value.

ADC Input Mux Selection Table

mux_ch_sel	mux_diff	ADC+	ADC-	Notes
0x00	0	AIN0	V _{SSADC}	Single-ended input
0x01	0	AIN1	V _{AGND}	Single-ended input
0x02	0	AIN2	V _{AGND}	Single-ended input
0x03	0	AIN3	V _{AGND}	Single-ended input
0x04	0	AIN4	V _{AGND}	Single-ended input
0x05	0	AIN5	V _{AGND}	Single-ended input
0x06	0	AIN6	V _{AGND}	Single-ended input
0x07	0	AIN7	V _{AGND}	Single-ended input
0x08	0	AIN8	V _{AGND}	Single-ended input
0x09	0	AIN9	V _{AGND}	Single-ended input
0x0A	0	AIN10	V _{AGND}	Single-ended input
0x0B	0	AIN11	V _{AGND}	Single-ended input
0x0C	0	AIN12	V _{AGND}	Single-ended input
0x0D	0	AIN13	V _{AGND}	Single-ended input
0x0E	0	AIN14	V _{AGND}	Single-ended input
0x0F	0	AIN15	V _{AGND}	Single-ended input
0x11	0	TMON_R	V _{AGND}	Resistor measurement for Temperature Sensor
0x10	X	N/A	N/A	Reserved for future use
0x12	X	(V _{DDA3ADC})/4	V _{AGND}	V _{DDA3ADC} supply, divided by 4
0x13	X	N/A	N/A	Reserved for future use
0x14	X	AIN0/5	V _{AGND}	AIN0, divided by 5
0x15..0x1F	X	N/A	N/A	Reserved for future use
0x20	X	OUTA	V _{AGND}	SINAD through this path cannot be guaranteed.
0x21	X	OUTB	V _{AGND}	SINAD through this path cannot be guaranteed.
0x22	X	OUTC	V _{AGND}	SINAD through this path cannot be guaranteed.
0x23	X	OUTD	V _{AGND}	SINAD through this path cannot be guaranteed.
0x24	X	INA+	V _{AGND}	SINAD through this path cannot be guaranteed.
0x25	X	SNO_or	V _{AGND}	Any SNO input, determined by sno_or_sel in AFE_CTRL2
0x26	X	SCM_or	V _{AGND}	Any SCM input, determined by scm_or_sel in AFE_CTRL2
0x27..0x2F	X	N/A	N/A	Reserved for future use.

mux_ch_sel	mux_diff	ADC+	ADC-	Notes
0x31	X	V _{REFDAC}	V _{AGND}	V _{REFDAC} output buffer
0x32	X	V _{REFADC}	V _{AGND}	V _{REFDAC} output buffer
0x00	1	AIN0	AIN8	Differential Input
0x01	1	AIN1	AIN9	Use differential AIN1/9 for external V _{BE} -Resistor temperature sensor
0x02	1	AIN2	AIN10	Differential Input
0x03	1	AIN3	AIN11	Differential Input
0x04	1	AIN4	AIN12	Differential Input
0x05	1	AIN5	AIN13	Differential Input
0x06	1	AIN6	AIN14	Differential Input
0x07	1	AIN7	AIN15	Differential Input
0x08	1	AIN0	AIN8	Differential Input
0x09	1	AIN1	AIN9	Differential Input
0x0A	1	AIN2	AIN10	Differential Input
0x0B	1	AIN3	AIN11	Differential Input
0x0C	1	AIN4	AIN12	Differential Input
0x0D	1	AIN5	AIN13	Differential Input
0x0E	1	AIN6	AIN14	Differential Input
0x0F	1	AIN7	AIN15	Differential Input
0x11	1	TMON_R	TMON_VBE	V _{BE} measurement for Temperature Sensor

Advanced Details

Input Mux Leakage Current: The input mux is structured to minimize the input leakage, typically as low as a few pA. The input switch chosen by [ADC_PGA_CTRL.mux_ch_sel](#) is only closed for the duration of the ADC/PGA acquisition window. During this acquisition period, a large current will flow into the ADC/PGA capacitors. As the capacitive charge settles, the current will return to the original low-leakage value.

Note The longer PGA acquisition time can be useful for weaker inputs.

8.3.4.5 Scan Modes

Input Mux Channel Scanning

Channels can be scanned at any rate up to the maximum 500kHz sample rate supported by the **MAX32600** ADC. In scan mode, a single sample or a burst of samples is taken from each enabled channel in the scan sequence. The input mux and PGA settings are automatically configured to the parameters needed by the next scan channel in the sequence by the ADC state machine.

Scan mode features include:

- Scan with sample rate up to 500kHz
- Up to eight input channels, in any order
- Mix single-ended and differential inputs
- Independent PGA gain setting for each channel

Scan Mode Operation

In scan mode operation, between one and eight channels can be configured for scanning. Channel configuration and settings are managed using the registers [ADCCFG_SCAN1](#) and [ADCCFG_SCAN2](#). Each of these registers hold descriptors for four channels (ADCCFG_SCAN1: adc_scan0 to adc_scan3; ADCCFG_SCAN2: adc_scan4 to adc_scan7). The number of scan channels is set using [ADCCFG_CTRL1.scan_cnt](#). For each channel being configured, the [adc_scanX](#) field must be filled. The contents of the scan descriptors are shown below.

0	1	2	3	4	5	6	7
mux_ch_sel				pga_gain		mux_diff	RESERVED

- PGA bypass or enabled is the same for all scan channels (globally enabled or bypassed)
- Burst mode operations are completed prior to switching to the next channel
- Four scan modes supported using [adc_mode\[4:0\]](#) for mode selection as shown in the [Scan Mode Selection Table](#)

Note Scan Mode Limitations: Scan mode requires the PGA to be enabled for all channels or disabled (bypassed) for all channels. A single set of PGA/ADC timings, determined by [pga_trk_cnt](#), [pga_acq_cnt](#), and [adc_acq_cnt](#) register fields, must be used for all channels in the sequence. These values must be set to be compatible with the highest PGA gain setting that will be used during the sequence.

Scan Mode Selection Table

adc_mode	Mode	Description
1000b	High Performance Channel Scan	Measure adc_smpl_cnt samples one time per channel.
1001b	Low Power Channel Scan	Measure adc_smpl_cnt samples with adc_sleep_cnt delay between channels. Single pass through scan channels.
1010b	High Performance Continuous Channel Scan	High Performance Channel Scan continuously. When all channels have been measured scanning begins again with the first scan channel.
1011b	Low Power Continuous Channel Scan	Low Power Channel Scan continuously. When all channels have been measured scanning begins again with the first scan channel.

8.3.4.6 Interrupts

ADC interrupts are enabled using the [ADC_INTR](#) register. Refer to [System Configuration and Management](#) for details on setting up and configuring interrupts.

Out of Range Interrupts

The **MAX32600** ADC supports an upper and lower set point to generate out of range interrupts if a measured sample goes above or below a configured threshold. The out of range support simplifies monitoring an input signal for a threshold and allows the CPU to only wake when a set condition occurs.

Interrupt support is provided for a measured value being above the upper limit, below the lower limit, or outside the range of the two limits combined. To set the lower threshold, write the desired value to the register field [ADC_LIMIT.lo_limit](#); [ADC_LIMIT.hi_limit](#) is used to set the upper threshold.

The interrupt flag [ADC_INTR.hi_rng](#) is set by hardware when a measured value exceeds the upper limit threshold. Similarly, the [ADC_INTR.lo_rng](#) is set when a measured value falls below the lower limit threshold. The [ADC_INTR.out_rng](#) flag is set when either of the previous conditions occur.

ADC Output FIFO

The **MAX32600** ADC includes a 32-level-deep output FIFO with each entry 16 bits wide. The FIFO is read through the [ADC_OUT.data_reg](#) register. Multiple reads to the register will return the next 16-bit value. The ADC interface can generate a variety of FIFO interrupts if enabled.

There are several status fields and interrupt flags provided to assist the application in handling the ADC Output FIFO:

- [ADC_CTRL0.adc_fifo_full](#) set by hardware when the FIFO level has exceeded the almost full threshold level.
- [ADC_TG_CTRL1.fifo_af_cnt](#) 4-bit field used to set the threshold for almost full warning.
- [ADC_CTRL0.adc_fifo_empty](#) Reading a value of 1 indicates the ADC FIFO is empty.

Interrupt flags are in the ADC_INTR register and include:

- [ADC_INTR.fifo_af](#) - Set when the ADC FIFO exceeds the almost full threshold.
- [ADC_INTR.fifo_uf](#) - Underflow condition occurred.
- [ADC_INTR.fifo_of](#) - Overflow condition occurred. This indicates the ADC FIFO is full and samples are being overwritten.
- [ADC_INTR.fifo_tf](#) - Interrupt flag set when the FIFO reaches three-quarters full.
- [ADC_INTR.fifo_hf](#) - Interrupt flag set when the FIFO reaches half full.
- [ADC_INTR.fifo_qf](#) - Interrupt flag set when the FIFO reaches one-quarter full.

8.3.4.7 Programmable Gain Amplifier

The **MAX32600** ADC signal chain includes a Programmable Gain Amplifier (PGA) between the Input Mux and the ADC. The PGA takes the outputs from the Input Mux and amplifies it prior to processing by the ADC. This improves the signal-to-noise ratio of low amplitude signals. Bypassing the PGA is supported and results in higher sample rates for the ADC. If enabled, the PGA supports gain settings of 1X, 2X, 4X, and 8X.

Note Optimal SNR and THD are generally achieved by bypassing the PGA and sending the input signal directly to the ADC. However, at full sampling rate, the acquisition time of the PGA is much longer than that of the ADC (i.e., 1.5 μ s vs. 125ns); this can allow for more complete settling of the measured signal if the input source impedance is too high.

PGA Configuration

The PGA can be enabled or bypassed as required by the application. The register field [ADC_PGA_CTRL.cpu_pga_bypass](#) is set to 1 to bypass the PGA. The default is for the PGA to be enabled and in the signal chain.

The Gain setting for the PGA is controlled via the [ADC_PGA_CTRL.gain](#) register field:

- 00b = 1X
- 01b = 2X
- 10b = 4X
- 11b = 8X

The Track portion of the acquisition window is controlled via the `ADC_TG_CTRL0.pga_trk_cnt` register field, and the Hold portion is controlled by the `ADC_TG_CTRL1.pga_acq_cnt` field. Refer to [Sample Rate Calculation](#) for details.

When the ADC is operating in bipolar signal conditioning mode, PGA gain can be used in conjunction with the default ADC input range $V_{REF}/2$ (`ADC_CTRL0.range = 0`). This can provide a total effective gain of 16X. Although both PGA and ADC can provide a 2X gain (in bipolar mode), for optimal SNR, the PGA should be used.

Advanced Details

The PGA is a track and hold input device with switched-capacitance architecture. This architecture enables minimal offset and gain errors, rail-to-rail inputs, and high input impedance. It is important that the input signal source to the PGA be capable of fully settling within the track portion of the measurement. To achieve the settling time requirements as the sample rate increases, the source output impedance needs to be low. The sampling capacitance increases with gain as shown in the table below.

Gain	Input Capacitance (typical)
1X	7pF
2X	13pF
4X	25pF
8X	49pF

The PGA input capacitance should be modeled as a fully discharged capacitor that is periodically connected to the input signal during each track point of the measurement window. The input signal must be capable of charging and settling the PGA input capacitor with an accuracy of $\frac{V_{REFADC}}{2^N}$ (with N being the required ADC measurement resolution) to fully settle. For an RC network, this settling time is:

$$t_{SETTLING} = \ln(RC \times 2^N)$$

The output impedance of the signal source must be carefully considered and balanced with the PGA track window time (or the ADC acquisition time if the PGA is bypassed) to meet the settling time.

Input Voltage to the Receive Path

The choices of ADC, PGA, and reference voltage affect the full scale input voltage to the receive path. The table below illustrates these interactions.

ADC Mode	ADC Range	Input Mode	V_I	PGA Mode (Gain, Bypass)	Min V_I	Max V_I	Notes
Unipolar	Not Applicable	Single-Ended	A_i	1X (Bypass)	0	V_{REF}	

ADC Mode	ADC Range	Input Mode	V_I	PGA Mode (Gain, Bypass)	Min V_I	Max V_I	Notes
Unipolar	Not Applicable	Single-Ended	Ai	2X	0	$V_{REF}/2$	
Unipolar	Not Applicable	Single-Ended	Ai	4X	0	$V_{REF}/4$	
Unipolar	Not Applicable	Single-Ended	Ai	8X	0	$V_{REF}/8$	
Unipolar	Not Applicable	Differential	AiP-AiN	1X (Bypass)	0	V_{REF}	AiP>AiN
Unipolar	Not Applicable	Differential	AiP-AiN	2X	0	$V_{REF}/2$	AiP>AiN
Unipolar	Not Applicable	Differential	AiP-AiN	4X	0	$V_{REF}/4$	AiP>AiN
Unipolar	Not Applicable	Differential	AiP-AiN	8X	0	$V_{REF}/8$	AiP>AiN
Bipolar	V_{REF}	Differential	AiP-AiN	1X (Bypass)	$-V_{REF}$	V_{REF}	
Bipolar	V_{REF}	Differential	AiP-AiN	2X	$-V_{REF}/2$	$V_{REF}/2$	
Bipolar	V_{REF}	Differential	AiP-AiN	4X	$-V_{REF}/4$	$V_{REF}/4$	
Bipolar	V_{REF}	Differential	AiP-AiN	8X	$-V_{REF}/8$	$V_{REF}/8$	
Bipolar	$V_{REF}/2$	Differential	AiP-AiN	1X (Bypass)	$-V_{REF}/2$	$V_{REF}/2$	
Bipolar	$V_{REF}/2$	Differential	AiP-AiN	2X	$-V_{REF}/4$	$V_{REF}/4$	
Bipolar	$V_{REF}/2$	Differential	AiP-AiN	4X	$-V_{REF}/8$	$V_{REF}/8$	
Bipolar	$V_{REF}/2$	Differential	AiP-AiN	8X	$-V_{REF}/16$	$V_{REF}/16$	

Note Explanation of [Input Voltage to the Receive Path Table](#) values:

- V_I : Effective Input Voltage
- Minimum V_I with Digital Output 0x0001
- Maximum V_I with Digital Output 0xFFFF

8.3.5 Sample Rate Calculation

The **MAX32600** ADC supports two primary measurement modes for the ADC: Low Power Mode and High Performance Mode. Low Power Mode uses significantly less power than High Performance Mode by putting the ADC and PGA, if enabled, in a sleep state between samples. The maximum sample rate achievable that

allows the ADC to sleep between samples is 180ksps with PGA enabled, or 333ksps with the PGA in bypass. High Performance Mode leaves the ADC powered during the entire measurement window, enabling sample rates as high as 500ksps.

Both Low Power Mode and High Performance Mode support four sampling configurations. The configurations supported are:

- **Sample Mode:** The ADC measures a single input channel until a specified number of samples have been captured, or continuously until turned off by the CPU.
- **Burst/Decimation Mode:** For a single input channel, a specified number of samples are captured back to back and optionally averaged with a configurable delay between samples.
- **Scan Mode:** Samples are taken from each configured input channel in the sequence set up by the application.
- **Scan Mode with Burst/Decimation:** For each channel measured, a specified number of samples are captured back to back and optionally averaged with a configurable delay between samples before moving to the next input channel in the sequence.

For both Low Power Mode and High Performance Mode and each of their sample configurations, there are specific equations used to achieve a target sample rate. The sections below describe which ADC register fields are used to perform these calculations for both measurement modes and their respective sample configurations. In addition, the equations are dependent on the PGA being either enabled or bypassed.

8.3.5.1 Decimation Filter Modes

When the ADC is set to burst mode in either scan or sample modes, the ADC supports decimation filtering. The register [ADC_CTRL0.avg_mode](#) is used to select the decimation filter type. A value of 0 turns off the decimation filtering and the ADC outputs the raw data samples acquired in the ADC output FIFO. A value of 1 outputs the average of the samples taken in burst mode as specified by [ADC_TG_CTRL1.adc_brst_cnt](#). Using decimation mode affects the ADC sample rate calculations. See Sample Rate Calculations below for details.

8.3.5.2 Register Abbreviations and Definitions

8.3.5.2.1 [adc_acq_cnt](#)

Register field [ADC_TG_CTRL1.adc_acq_cnt](#). This field determines the time (in ADC clock periods) that the ADC is in acquisition mode. The optimal setting for this field can be determined based on the PGA mode, as shown in Table 2 below.

8.3.5.2.2 [adc_config_field_adc_brst_cnt](#)

Register field [ADC_TG_CTRL1.adc_brst_cnt](#). This field determines the number of samples which will be taken in a burst ($2^{\text{adc_brst_cnt}}$). It is only used when the ADC is in decimation mode (as determined by [avg_mode](#)).

8.3.5.2.3 `adc_flush`

This parameter is not based on a register field. It is used in the sample rate calculations, however. It should always have a value of 12.

8.3.5.2.4 `adc_sleep_cnt`

Register field `ADC_TG_CTRL1.adc_sleep_cnt`. This field determines the time (in ADC clock periods) between ADC samples or bursts in low power mode.

8.3.5.2.5 `adc_smpl_cnt`

Register field `ADC_TG_CTRL0.adc_smpl_cnt`. This field determines the number of samples that will be taken in Single Acquisition Mode.

8.3.5.2.6 `adc_wake_cnt`

Register field `ADC_CTRL0.adc_wake_cnt`. This field should always be set to the optimized value of 4.

8.3.5.2.7 `avg_mode`

Register field `ADC_CTRL0.avg_mode`. This field determines the decimation mode.

8.3.5.2.8 `mode`

Register field `ADC_CTRL0.mode`. This field determines the acquisition mode (see Table 1 below).

8.3.5.2.9 `pga_acq_cnt`

Register field `ADC_TG_CTRL1.pga_acq_cnt`. This field determines the time (in ADC clock periods) that the PGA is in acquisition mode. The optimal setting for this field can be determined based on the PGA mode, as shown in Table 2 below.

8.3.5.2.10 `pga_trk_cnt`

Register field `ADC_TG_CTRL1.pga_trk_cnt`. When the PGA is enabled, this field determines the time (in ADC clock periods) that the PGA is in tracking mode. The PGA will be in tracking mode while the ADC is converting the previous sample. When the PGA is bypassed, this value can be used to control the time between the end of one ADC acquisition and the beginning of the next ADC acquisition. There is a minimum allowable value for this field, based on the PGA mode (see Table 2 below).

8.3.5.2.11 `pga_wake_cnt`

Register field `ADC_PGA_CTRL.pga_wake_cnt`. This field determines the time allowed for the PGA to power up and track the first sample. The optimal/minimum value for this field is 29.

8.3.5.2.12 scan_cnt

Register field `ADCCFG_CTRL1.scan_cnt`. This field determines the number of channels used in scan mode where the number of channels = ($scan_cnt + 1$).

8.3.5.3 ADC Mode Explanations

Single Acquisition or Continuous Acquisition

Selected by *mode*. The data rate calculations are the same for both.

In **Single Acquisition** mode, the ADC runs until a specified number of samples (*adc_smpl_cnt*) have been collected. If channel scan mode is enabled, then the number of channels given by ($scan_cnt + 1$) will be scanned.

In **Continuous Acquisition** mode, the ADC runs continuously until it is disabled manually. If channel scan mode is enabled, then the number of channels given by ($scan_cnt + 1$) will be scanned.

Low Power or High Performance

Selected by *mode*.

In *Low Power* mode, the ADC goes to sleep for a specified number of ADC clock periods (determined by *adc_sleep_cnt*) between acquisition cycles. *High Performance* mode is when the ADC does not go to sleep between acquisition cycles.

No Channel Scan or Channel Scan

Selected by *mode*.

No Channel Scan mode is when the acquisition is done for one channel. *Channel Scan* mode is when the acquisition is done for the specified channels.

Single Sample or Burst of Samples

Selected by *adc_brst_cnt*.

Single Sample mode (*adc_brst_cnt* set to 1) is where only one sample is taken per channel per acquisition cycle. *Burst of Samples* mode is where the ADC will take some defined number of samples (*adc_brst_cnt*) per channel per acquisition cycle.

ADC_CTRL0.mode	ADC Mode
0000b	Single Acquisition, High Performance, No Channel Scan, ADC_TG_CTRL1.adc_brst_cnt value sets BURST
0001b	Single Acquisition, Low Power, No Channel Scan, ADC_TG_CTRL1.adc_brst_cnt value sets BURST
0010b	Continuous Acquisition, High Performance, No Channel Scan, ADC_TG_CTRL1.adc_brst_cnt value sets BURST

ADC_CTRL0.mode	ADC Mode
0011b	Continuous Acquisition, Low Power, No Channel Scan, ADC_TG_CTRL1.adc_brst_cnt value sets BURST
1000b	Single Acquisition, High Performance, Channel Scan (burst cannot be used in this mode)
1001b	Single Acquisition, Low Power, Channel Scan, ADC_TG_CTRL1.adc_brst_cnt value sets BURST
1010b	Continuous Acquisition, High Performance, Channel Scan (burst cannot be used in this mode)
1011b	Continuous Acquisition, Low Power, Channel Scan, ADC_TG_CTRL1.adc_brst_cnt value sets BURST

Table 1 - Acquisition modes

The Decimation Filter works in all modes if it is enabled (see *avg_mode* details). The ADC FIFO is filled after the decimation filter, so when decimation is on, the ADC FIFO will contain either just the results of the decimation OR the raw data and the results of the decimation depending on the value set with the *avg_mode* bits. Since Burst and Decimation are not defined by the ADC_CTRL0.mode bits, multiple "Modes" specified below may have the same the ADC_CTRL0.mode value.

Register settings and data rate equations for the different ADC Modes

There are minimum values for *pga_trk_cnt* based off the number of cycles required for the ADC conversion. *pga_trk_cnt* can be increased to meet a required sampling rate and/or to increase the PGA's input settling time. There are also specific values required for *pga_acq_cnt*, and *adc_acq_cnt* based off of the PGA settings. In PGA Enabled mode, these values are optimized for the device and should not be changed. In PGA Bypass mode, *adc_acq_cnt* can be increased to allow large settling time into the ADC's input capacitance. These minimum values must be met.

PGA Mode	PGA Gain	<i>pga_trk_cnt</i>	<i>pga_acq_cnt</i>	<i>adc_acq_cnt</i>
Bypass	N.A.	9 min	X	0 min
Enabled	x1 or x2	7 min	1 optimal	1 optimal
Enabled	x4	7 min	1 optimal	2 optimal
Enabled	x8	7 min	2 optimal	2 optimal

Table 2 - Optimal/minimum register values

When bursting and decimation is active, the ADC data rate may not equal the ADC sample rate. The following equations are for the ADC data rate only.

8.3.5.4 Mode: PGA Enabled, High Performance, no Scan, with no Burst

The ADC data rate is set by adjusting `pga_trk_cnt`.

Set `pga_acq_cnt` and `adc_acq_cnt` to their optimized values from Table 2 based off of the PGA gain setting. Calculate `pga_trk_cnt` from the target ADC data rate (F_{TDR}), and the ADC clock rate (F_C).

$$pga_trk_cnt = (F_C / F_{TDR}) - (pga_acq_cnt + 1 + adc_acq_cnt + 1 + adc_wake_cnt + 1) - 1$$

With `adc_wake_cnt` at its optimized value of 4, we simplify to

$$pga_trk_cnt = (F_C / F_{TDR}) - (pga_acq_cnt + adc_acq_cnt + 8)$$

Round `pga_trk_cnt` to the closest integer. In this mode, `pga_trk_cnt` must be ≥ 7 , so if `pga_trk_cnt` is less than 7, set it to 7. Now calculate the actual ADC data rate (F_{DR}) based off of the calculated `pga_trk_cnt`.

$$F_{DR} = F_C / (pga_trk_cnt + 1 + pga_acq_cnt + 1 + adc_acq_cnt + 1 + adc_wake_cnt + 1)$$

With `adc_wake_cnt` at its optimized value of 4, we simplify to

$$F_{DR} = F_C / (pga_trk_cnt + pga_acq_cnt + adc_acq_cnt + 8)$$

Example calculation: For this example, the PGA gain is 4, F_C is 8 MHz, and the target ADC data rate is 11 ksps.

$$pga_trk_cnt = (8 \text{ MHz} / 11 \text{ ksps}) - (1 + 2 + 8) = 716.27$$

Round 716.27 to the nearest integer, 716

$$F_{DR} = 8 \text{ MHz} / (716 + 1 + 2 + 8) = 11.00 \text{ ksps}$$

8.3.5.5 Mode: PGA Enabled, High Performance, No Scan, with Burst and no decimation

This mode is the same as PGA Enabled, High Performance, no Scan, no Burst because without decimation, the ADC data rate is the same as the burst sample frequency. See the equations above.

8.3.5.6 Mode: PGA Enabled, High Performance, No Scan, with Burst and decimation, averaged data output to FIFO

In this mode, the ADC data rate is set by adjusting `pga_trk_cnt` and is a function of the decimation mode and the burst count. The `adc_brst_cnt` bits set the decimation rate.

The calculations below are for `avg_mode = 1`, just the average of the burst data sent to the ADC FIFO.

Set `pga_acq_cnt` and `adc_acq_cnt` to their optimized values from Table 2 based off of the PGA gain setting. Calculate `pga_trk_cnt` from the number of burst samples ($2^{\text{adc_brst_cnt}}$), the target ADC data rate (F_{TDR}), and the ADC clock rate (F_{C}).

$$pga_trk_cnt = \{F_C / [F_{TDR} * (2^{\text{adc_brst_cnt}})]\} - (pga_acq_cnt + 1 + adc_acq_cnt + 1 + adc_wake_cnt + 1) - 1$$

With `adc_wake_cnt` at its optimized value of 4, we simplify to:

$$pga_trk_cnt = \{F_C / [F_{TDR} * (2^{\text{adc_brst_cnt}})]\} - (pga_acq_cnt + adc_acq_cnt + 8)$$

Round `pga_trk_cnt` to the closest integer. In this mode, `pga_trk_cnt` must be ≥ 7 , so if `pga_trk_cnt` is less than 7, set it to 7. Calculate the actual ADC data rate based off of the calculated `pga_trk_cnt`.

$$F_{DR} = F_C / [(2^{\text{adc_brst_cnt}}) * (pga_trk_cnt + 1 + pga_acq_cnt + 1 + adc_acq_cnt + 1 + adc_wake_cnt + 1)]$$

With `adc_wake_cnt` at its optimized value of 4, we simplify to:

$$F_{DR} = F_C / [(2^{\text{adc_brst_cnt}}) * (pga_trk_cnt + pga_acq_cnt + adc_acq_cnt + 8)]$$

Example calculation:

For this example, the PGA gain is 2, F_{C} is 8 MHz, `adc_brst_cnt` is set to 3 (8 samples per burst), and the target "averaged data" data rate is 7 ksps.

$$pga_trk_cnt = \{8 \text{ MHz} / [7 \text{ ksps} * (2^3)]\} - (1 + 1 + 8) = 132.857$$

Round 132.857 to the nearest integer, 133.

$$F_{DR} = 8 \text{ MHz} / ((2^3) * (133 + 1 + 1 + 8)) = 6.99 \text{ ksps}$$

8.3.5.7 Mode: PGA Enabled, High Performance, Scan, with no Burst

In this mode, the ADC data rate is set by adjusting `pga_trk_cnt` and is a function of the number channels in the scan. When channel scanning is enabled, the ADC cycles through `scan_cnt + 1` channels based on the 8 different scan descriptors. Similar to burst mode, the data rate per channel is reduced by the number of channels in the scan. The calculations below are for the per channel data rate.

Set `pga_acq_cnt` and `adc_acq_cnt` to their optimized values from Table 2 based off of the PGA gain setting. Calculate `pga_trk_cnt` from the number of channels in the scan (`scan_cnt + 1`), the target per channel data rate (F_{TDR}), and the ADC clock rate (F_C).

$$pga_trk_cnt = \{F_C / [F_{TDR} * (scan_cnt + 1)]\} - (pga_acq_cnt + 1 + adc_acq_cnt + 1 + adc_wake_cnt + 1) - 1$$

With `adc_wake_cnt` at its optimized value of 4, we simplify to:

$$pga_trk_cnt = \{F_C / [F_{TDR} * (scan_cnt + 1)]\} - (pga_acq_cnt + adc_acq_cnt + 8)$$

Round `pga_trk_cnt` to the closest integer. In this mode, `pga_trk_cnt` must be ≥ 7 , so if `pga_trk_cnt` is less than 7, set it to 7. Calculate the actual ADC data rate based off of the calculated `pga_trk_cnt`.

$$F_{DR} = F_C / [(scan_cnt + 1) * (pga_trk_cnt + 1 + pga_acq_cnt + 1 + adc_acq_cnt + 1 + adc_wake_cnt + 1)]$$

With `adc_wake_cnt` at its optimized value of 4, we simplify to:

$$F_{DR} = F_C / [(scan_cnt + 1) * (pga_trk_cnt + pga_acq_cnt + adc_acq_cnt + 8)]$$

Example calculation:

For this example, there are 5 channels in the scan (`scan_cnt = 4`), the highest PGA gain for all of the channels is 8, F_C is 8 MHz, and the target per channel data rate is 21 ksps.

$$pga_trk_cnt = (8 \text{ MHz} / (21 \text{ ksps} * 5)) - (2 + 2 + 8) = 64.19$$

Round 64.19 to the nearest integer, 64.

$$F_{DR} = 8 \text{ MHz} / (5 * (64 + 2 + 2 + 8)) = 21.05 \text{ ksps}$$

8.3.5.8 Mode: PGA Enabled, Low-Power, no Scan, with no Burst

In Low power mode, after the ADC Acquisition is complete, 17 clocks (*adc_flush*) are required for the final ADC conversion to complete. It will then power down for *adc_slp_cnt* cycles before the PGA is powered up again. For energy optimization, the ADC is powered up *pga_wake_cnt* cycles after the PGA is powered up. *pga_wake_cnt* is used and *pga_trk_cnt* is not used in this mode because only a single sample is taken each time the ADC wakes up from sleep.

Set *pga_acq_cnt* and *adc_acq_cnt* to their optimized values based off of the PGA gain setting. Calculate *adc_slp_cnt* from the target data rate (F_{TDR}), and the ADC clock rate (F_C).

$$adc_slp_cnt = (F_C / F_{TDR}) - (pga_wake_cnt + 1 + adc_wake_cnt + 1 + pga_acq_cnt + 1 + adc_acq_cnt + 1 + adc_flush) - 1$$

With *pga_wake_cnt* at its optimized value of 29, *adc_wake_cnt* at its optimized value of 4, and *adc_flush* at 17, we simplify to:

$$adc_slp_cnt = (F_C / F_{TDR}) - (pga_acq_cnt + adc_acq_cnt + 55)$$

Round *adc_slp_cnt* to the closest integer. Now calculate the actual data rate based off of the calculated *adc_slp_cnt*.

$$F_{DR} = F_C / (pga_wake_cnt + 1 + adc_wake_cnt + 1 + pga_acq_cnt + 1 + adc_acq_cnt + 1 + adc_flush + adc_slp_cnt + 1)$$

With *pga_wake_cnt* at its optimized value of 29, *adc_wake_cnt* at its optimized value of 4, and *adc_flush* at 17, we simplify to:

$$F_{DR} = F_C / (pga_acq_cnt + adc_acq_cnt + adc_slp_cnt + 55)$$

Example calculation:

For this example, the PGA gain is 1, F_C is 8 MHz, and the target data rate is 57 ksps.

$$adc_slp_cnt = (8 \text{ MHz} / 57 \text{ ksps}) - (1 + 1 + 55) = 83.35$$

Round 83.35 to the nearest integer, 83.

$$F_{DR} = 8 \text{ MHz} / (1 + 1 + 83 + 55) = 57.14 \text{ ksps}$$

8.3.5.9 Mode: PGA Enabled, Low-Power, Scan, with no Burst

This mode is similar to "PGA Enabled, Low-Power, no Scan, with no Burst", except $scn_cnt + 1$ samples are collected between each sleep cycle. In this mode, the ADC data rate is set by adjusting adc_slp_cnt and is a function of the scan count.

Set adc_acq_cnt to its optimized value based off of the PGA gain setting. Set pga_trk_cnt to its minimum value of 7. Calculate adc_slp_cnt from the scan count, target data rate (F_{TDR}), and ADC clock rate (F_C).

$$adc_slp_cnt = (F_C / F_{DR}) - ([scn_cnt * (pga_trk_cnt + 1)] + [(scn_cnt + 1) * (adc_wake_cnt + 1 + adc_acq_cnt + 1)] + adc_flush) - 1$$

With pga_trk_cnt at its minimum value of 7, adc_wake_cnt at its optimized value of 4, and $adc_flush = 17$, we simplify to:

$$adc_slp_cnt = (F_C / F_{DR}) - \{(scn_cnt * 8) + [(scn_cnt + 1) * (adc_acq_cnt + 6)] + 18\}$$

Round adc_slp_cnt to the closest integer. Now calculate the actual data rate based off of the calculated adc_slp_cnt .

$$F_{DR} = F_C / ([scn_cnt * (pga_trk_cnt + 1)] + [(scn_cnt + 1) * (adc_wake_cnt + 1 + adc_acq_cnt + 1)] + adc_flush + 1 + adc_slp_cnt + 1)$$

With pga_trk_cnt at its minimum value of 7, adc_wake_cnt at its optimized value of 4, and $adc_flush = 17$, we simplify to:

$$F_{DR} = F_C / [(scn_cnt * 8) + [(scn_cnt + 1) * (adc_acq_cnt + 6)] + adc_slp_cnt + 18]$$

Example calculation:

For this example, scanning 3 channels, the PGA gain is 4, F_C is 8 MHz, and the target data rate is 2 ksps.

$$adc_slp_cnt = (8 \text{ MHz} / 2 \text{ ksps}) - \{(2 * 8) + [(2 + 1) * (2 + 6)] + 18\} = 3942$$

Since 3942 is already an integer and does not need to be rounded:

$$F_{DR} = 8 \text{ MHz} / [(2 * 8) + [(2 + 1) * (2 + 6)] + 3942 + 18] = 2.00 \text{ ksps}$$

8.3.5.10 Mode: PGA Enabled, Low-Power, no Scan, with Burst and decimation, only averaged data output to FIFO

This mode is similar to "PGA Enabled, Low-Power, no Scan, with no Burst", except that $2^{\text{adc_brst_cnt}}$ samples are collected between each sleep cycle. The $2^{\text{adc_brst_cnt}}$ samples will be averaged and the result will be written to the FIFO.

Set pga_acq_cnt and adc_acq_cnt to their optimized values based off of the PGA gain setting. Set pga_wake_cnt to its optimized value of 29. Set pga_trk_cnt to its minimum value of 7. Calculate adc_slp_cnt from the burst count, target data rate (F_{TDR}), and ADC clock rate (F_{C}).

$$\text{adc_slp_cnt} = (F_{\text{C}} / F_{\text{DR}}) - \{ \text{pga_wake_cnt} + 1 + \{ [(2^{\text{adc_brst_cnt}}) - 1] * (\text{pga_trk_cnt} + 1) \} + \{ [(2^{\text{adc_brst_cnt}}) * (\text{adc_wake_cnt} + 1 + \text{pga_acq_cnt} + 1 + \text{adc_acq_cnt} + 1)] + \text{adc_flush} \} - 1$$

With pga_wake_cnt at its optimized value of 29, pga_trk_cnt at its minimum value of 7, adc_wake_cnt at its optimized value of 4, and adc_flush at 17, we simplify to:

$$\text{adc_slp_cnt} = (F_{\text{C}} / F_{\text{DR}}) - \{ \{ [(2^{\text{adc_brst_cnt}}) - 1] * 8 \} + \{ [(2^{\text{adc_brst_cnt}}) * (\text{pga_acq_cnt} + \text{adc_acq_cnt} + 7)] + 48 \}$$

Now calculate the actual data rate based off of the calculated adc_slp_cnt .

$$F_{\text{DR}} = F_{\text{C}} / \{ \text{pga_wake_cnt} + 1 + \{ [(2^{\text{adc_brst_cnt}}) - 1] * (\text{pga_trk_cnt} + 1) \} + \{ [(2^{\text{adc_brst_cnt}}) * (\text{adc_wake_cnt} + 1 + \text{pga_acq_cnt} + 1 + \text{adc_acq_cnt} + 1)] + \text{adc_flush} + \text{adc_slp_cnt} + 1 \}$$

With pga_wake_cnt at its optimized value of 29, pga_trk_cnt at its minimum value of 7, adc_wake_cnt at its optimized value of 4, and adc_flush at 17, we simplify to:

$$F_{\text{DR}} = F_{\text{C}} / \{ \{ [(2^{\text{adc_brst_cnt}}) - 1] * 8 \} + \{ [(2^{\text{adc_brst_cnt}}) * (\text{pga_acq_cnt} + \text{adc_acq_cnt} + 7)] + \text{adc_slp_cnt} + 48 \}$$

Example calculation:

For this example, the PGA gain is 2, F_{C} is 8 MHz, adc_brst_cnt is set to 2 (4 samples per burst), and the target "averaged data" data rate is 13 ksp/s.

$$\text{adc_slp_cnt} = (8 \text{ MHz} / 13 \text{ ksp/s}) - \{ \{ [(2^2) - 1] * 8 \} + \{ [(2^2) * (1 + 1 + 7)] + 48 \} = 507.38$$

Round 507.38 to the nearest integer, 507.

$$F_{\text{DR}} = 8 \text{ MHz} / \{ \{ [(2^2) - 1] * 8 \} + \{ [(2^2) * (1 + 1 + 7)] + 507 + 48 \} = 13.01 \text{ ksp/s}$$

8.3.5.11 Mode: PGA Enabled, Low-Power, Scan, with Burst and decimation, only averaged data output to FIFO

This mode is similar to "PGA Enabled, Low-Power, no Scan, with Burst and decimation, only averaged data output to FIFO", with multiple samples of each scanned channel being collected between each sleep cycle. In this mode, the ADC data rate is set by adjusting `adc_slp_cnt` and is a function of the scan count and burst count.

Set `pga_acq_cnt` and `adc_acq_cnt` to their optimized values based off of the PGA gain setting. Set `pga_trk_cnt` to its minimum value of 7. Calculate `adc_slp_cnt` from the scan count, burst count, target data rate (F_{TDR}), and ADC clock rate (F_C).

$$adc_slp_cnt = (F_C / F_{DR}) - \{pga_wake_cnt + 1 + \{[(scn_cnt + 1) * (2^{adc_burst})] - 1\} * (pga_trk_cnt + 1)\} + \{[(scn_cnt + 1) * (2^{adc_burst})] * (adc_wake_cnt + 1 + pga_acq_cnt + 1 + adc_acq_cnt + 1)\} + adc_flush\} - 1$$

With `pga_wake_cnt` at its optimized value of 29, `pga_trk_cnt` at its minimum value of 7, `adc_wake_cnt` at its optimized value of 4, and `adc_flush` = 17, we simplify to:

$$adc_slp_cnt = (F_C / F_{DR}) - \{[(scn_cnt + 1) * (2^{adc_burst})] - 1\} * 8\} + \{[(scn_cnt + 1) * (2^{adc_burst})] * (pga_acq_cnt + adc_acq_cnt + 6)\} + 48\}$$

Round `adc_slp_cnt` to the closest integer. Now calculate the actual data rate based off of the calculated `adc_slp_cnt`.

$$F_{DR} = F_C / \{pga_wake_cnt + 1 + \{[(scn_cnt + 1) * (2^{adc_burst})] - 1\} * (pga_trk_cnt + 1)\} + \{[(scn_cnt + 1) * (2^{adc_burst})] * (adc_wake_cnt + 1 + pga_acq_cnt + 1 + adc_acq_cnt + 1)\} + adc_flush + adc_slp_cnt + 1\}$$

With `pga_wake_cnt` at its optimized value of 29, `pga_trk_cnt` at its minimum value of 7, `adc_wake_cnt` at its optimized value of 4, and `adc_flush` at 17, we simplify to:

$$F_{DR} = F_C / \{[(scn_cnt + 1) * (2^{adc_burst})] - 1\} * (pga_trk_cnt + 1)\} + \{[(scn_cnt + 1) * (2^{adc_burst})] * (pga_acq_cnt + adc_acq_cnt + 7)\} + adc_slp_cnt + 48\}$$

Example calculation:

For this example, the PGA gain is 8, scanning 4 channels, `adc_brst_cnt` is set to 1 (2 samples per burst), F_C is 8 MHz, and the target "averaged data" data rate is 9 ksps.

$$adc_slp_cnt = (8\text{ MHz} / 9\text{ ksps}) - \{[(3 + 1) * (2^1)] - 1\} * 8\} + \{(3 + 1) * (2^1)\} * (2 + 2 + 6)\} + 48\} = 768.89$$

Round 768.89 to the nearest integer, 769.

$$F_{DR} = 8\text{ MHz} / \{[(3 + 1) * (2^1)] - 1\} * 8\} + \{(3 + 1) * (2^1)\} * (2 + 2 + 6)\} + 769 + 48\} = 9.00\text{ ksps}$$

8.3.5.12 Mode: PGA Bypass, High Performance, no Scan, with no Burst

The ADC data rate is set by adjusting `pga_trk_cnt` and `pga_acq_cnt` is not used in PGA Bypass modes.

Set `adc_acq_cnt` to 1. If you need to run at the fastest rate, it can be set to 0. Calculate `pga_trk_cnt` from the target ADC data rate (F_{TDR}), and the ADC clock rate (F_C).

$$pga_trk_cnt = (F_C / F_{TDR}) - (adc_acq_cnt + 1 + adc_wake_cnt + 1) - 1$$

With `adc_wake_cnt` at its optimized value of 4, we simplify to:

$$pga_trk_cnt = (F_C / F_{TDR}) - (adc_acq_cnt + 7)$$

Round `pga_trk_cnt` to the closest integer. In this mode, `pga_trk_cnt` must be ≥ 9 , so if `pga_trk_cnt` is less than 9, set it to 9. Now calculate the actual ADC data rate (F_{DR}) based off of the calculated `pga_trk_cnt`.

$$F_{DR} = F_C / (pga_trk_cnt + 1 + adc_acq_cnt + 1 + adc_wake_cnt + 1)$$

With `adc_wake_cnt` at its optimized value of 4, we simplify to:

$$F_{DR} = F_C / (pga_trk_cnt + adc_acq_cnt + 7)$$

Example calculation:

For this example, F_C is 8 MHz, and the target ADC data rate is 11 ksps.

$$pga_trk_cnt = (8 \text{ MHz} / 11 \text{ ksps}) - (1 + 7) = 717.27$$

Round 717.27 to the nearest integer, 717.

$$F_{DR} = 8 \text{ MHz} / (717 + 1 + 7) = 11.03 \text{ ksps}$$

8.3.5.13 Mode: PGA Bypass, High Performance, No Scan, with Burst and no decimation

This mode is the same as PGA Bypass, High Performance, no Scan, no Burst because without decimation, the ADC data rate is the same as the burst sample frequency. See the equations above.

8.3.5.14 Mode: PGA Bypass, High Performance, No Scan, with Burst and decimation, only averaged data output to FIFO

In this mode, the ADC data rate is set by adjusting `pga_trk_cnt` and is a function of the decimation mode and the burst count. The `adc_brst_cnt` bits set the decimation rate. The calculations below are for `avg_mode = 1`, just the average of the burst data sent to the ADC FIFO.

Set `adc_acq_cnt` to 1. If you need to run at the fastest rate, it can be set to 0. Calculate `pga_trk_cnt` from the number of burst samples ($2^{\text{adc_brst_cnt}}$), the target ADC data rate (F_{TDR}), and the ADC clock rate (F_C).

$$pga_trk_cnt = \{F_C / [F_{TDR} * (2^{\text{adc_brst_cnt}})]\} - (\text{adc_acq_cnt} + 1 + \text{adc_wake_cnt} + 1) - 1$$

With `adc_wake_cnt` at its optimized value of 4, we simplify to:

$$pga_trk_cnt = \{F_C / [F_{TDR} * (2^{\text{adc_brst_cnt}})]\} - (\text{adc_acq_cnt} + 7)$$

Round `pga_trk_cnt` to the closest integer. In this mode, `pga_trk_cnt` must be ≥ 9 , so if `pga_trk_cnt` is less than 9, set it to 9. Calculate the actual ADC data rate based off of the calculated `pga_trk_cnt`.

$$F_{DR} = F_C / [(2^{\text{adc_brst_cnt}}) * (\text{pga_trk_cnt} + 1 + \text{adc_acq_cnt} + 1 + \text{adc_wake_cnt} + 1)]$$

With `adc_wake_cnt` at its optimized value of 4, we simplify to:

$$F_{DR} = F_C / [(2^{\text{adc_brst_cnt}}) * (\text{pga_trk_cnt} + \text{adc_acq_cnt} + 7)]$$

Example calculation:

For this example, F_C is 8 MHz, `adc_brst_cnt` is set to 3 (8 samples per burst), and the target "averaged data" data rate is 7 ksp/s.

$$pga_trk_cnt = \{8 \text{ MHz} / [7 \text{ ksp/s} * (2^3)]\} - (1 + 7) = 134.857$$

Round 134.857 to the nearest integer, 135.

$$F_{DR} = 8 \text{ MHz} / ((2^3) * (135 + 1 + 7)) = 6.99 \text{ ksp/s}$$

8.3.5.15 Mode: PGA Bypass, High Performance, Scan, with no Burst

In this mode, the ADC data rate is set by adjusting `pga_trk_cnt` and is a function of the number of channels in the scan. When channel scanning is enabled, the ADC cycles through `scan_cnt + 1` channels based on the 8 different scan descriptors. Similar to burst mode, the data rate per channel is reduced by the number of channels in the scan. The calculations below are for the per channel data rate.

Set `adc_acq_cnt` to 1. If you need to run at the fastest rate, it can be set to 0. Calculate `pga_trk_cnt` from the number of channels in the scan (`scan_cnt + 1`), the target per channel data rate (F_{TDR}), and the ADC clock rate (F_C).

$$pga_trk_cnt = \{F_C / [F_{TDR} * (scan_cnt + 1)]\} - (adc_acq_cnt + 1 + adc_wake_cnt + 1) - 1$$

With `adc_wake_cnt` at its optimized value of 4, we simplify to:

$$pga_trk_cnt = \{F_C / [F_{TDR} * (scan_cnt + 1)]\} - (adc_acq_cnt + 7)$$

Round `pga_trk_cnt` to the closest integer. In this mode, `pga_trk_cnt` must be ≥ 9 , so if `pga_trk_cnt` is less than 9, set it to 9. Calculate the actual ADC data rate based off of the calculated `pga_trk_cnt`.

$$F_{DR} = F_C / [(scan_cnt + 1) * (pga_trk_cnt + 1 + adc_acq_cnt + 1 + adc_wake_cnt + 1)]$$

With `adc_wake_cnt` at its optimized value of 4, we simplify to:

$$F_{DR} = F_C / [(scan_cnt + 1) * (pga_trk_cnt + adc_acq_cnt + 7)]$$

Example calculation:

For this example, there are 5 channels in the scan (`scan_cnt = 4`), F_C is 8 MHz, and the target per channel data rate is 21 ksp/s.

$$pga_trk_cnt = \{8 \text{ MHz} / [21 \text{ ksp/s} * (4 + 1)]\} - (1 + 7) = 68.19$$

Round 68.19 to the nearest integer, 68.

$$F_{DR} = 8 \text{ MHz} / [(4 + 1) * (68 + 1 + 7)] = 21.05 \text{ ksp/s}$$

8.3.5.16 Mode: PGA Bypass, Low Power, no Scan, no Burst

In low power mode, after the ADC acquisition is complete, 17 clocks (*adc_flush*) are required for the final ADC conversion to complete. It will then power down for *adc_slp_cnt* cycles before the ADC is powered up again. For energy optimization, the ADC is powered up *pga_wake_cnt* cycles after the PGA is powered up. *pga_wake_cnt* and *pga_trk_cnt* are not used in this mode.

Set *adc_acq_cnt* to 1. If you need to run at the fastest rate, it can be set to 0. Calculate *adc_slp_cnt* from the target data rate (F_{TDR}), and the ADC clock rate (F_C).

$$adc_slp_cnt = (F_C / F_{TDR}) - (adc_wake_cnt + 1 + adc_acq_cnt + 1 + adc_flush) - 1$$

With *adc_wake_cnt* at its optimized value of 4 and *adc_flush* at 17, we simplify to:

$$adc_slp_cnt = (F_C / F_{TDR}) - (adc_acq_cnt + 24)$$

Round *adc_slp_cnt* to the closest integer. Now calculate the actual data rate based off of the calculated *adc_slp_cnt*.

$$F_{DR} = F_C / (adc_wake_cnt + 1 + adc_acq_cnt + 1 + adc_flush + adc_slp_cnt + 1)$$

With *adc_wake_cnt* at its optimized value of 4 and *adc_flush* at 17, we simplify to:

$$F_{DR} = F_C / (adc_acq_cnt + adc_slp_cnt + 24)$$

Example calculation:

For this example, F_C is 8 MHz, and the target data rate is 57 ksps.

$$adc_slp_cnt = (8\text{ MHz} / 57\text{ ksps}) - (1 + 24) = 115.35$$

Round 115.35 to the nearest integer, 115.

$$F_{DR} = 8\text{ MHz} / (1 + 115 + 24) = 57.14\text{ ksps}$$

8.3.5.17 Mode: PGA Bypass, Low Power, Scan, no Burst

This mode is similar to "PGA Bypass, Low-Power, no Scan, with no Burst", but with `scn_cnt + 1` samples collected between each sleep cycle. The ADC data rate is set by adjusting `adc_slp_cnt` and is a function of the scan count.

Set `adc_acq_cnt` to 1. If you need to run at the fastest rate, it can be set to 0. Set `pga_trk_cnt` to its minimum value of 9. Calculate `adc_slp_cnt` from the scan count, target data rate (F_{TDR}), and ADC clock rate (F_C).

$$adc_slp_cnt = (F_C / F_{DR}) - \{[scn_cnt * (pga_trk_cnt + 1)] + [(scn_cnt + 1) * (adc_wake_cnt + 1 + adc_acq_cnt + 1)] + adc_flush\} - 1$$

With `pga_trk_cnt` at its minimum value of 9, `adc_wake_cnt` at its optimized value of 4, and `adc_flush` = 17, we simplify to:

$$adc_slp_cnt = (F_C / F_{DR}) - \{(scn_cnt * 10) + [(scn_cnt + 1) * 7] + 18\}$$

Round `adc_slp_cnt` to the closest integer. Now calculate the actual data rate based off of the calculated `adc_slp_cnt`.

$$F_{DR} = F_C / \{[scn_cnt * (pga_trk_cnt + 1)] + [(scn_cnt + 1) * (adc_wake_cnt + 1 + adc_acq_cnt + 1)] + adc_flush + adc_slp_cnt + 1\}$$

With `pga_trk_cnt` at its minimum value of 9, `adc_wake_cnt` at its optimized value of 4, and `adc_flush` = 17, we simplify to:

$$F_{DR} = F_C / \{(scn_cnt * 10) + [(scn_cnt + 1) * 7] + adc_slp_cnt + 18\}$$

Example calculation:

For this example, scanning 3 channels, F_C is 8 MHz, and the target data rate is 2 ksps.

$$adc_slp_cnt = (8 \text{ MHz} / 2 \text{ ksps}) - \{(2 * 10) + [(2 + 1) * 7] + 18\} = 3941$$

3941 is already an integer, so:

$$F_{DR} = 8 \text{ MHz} / \{(2 * 10) + [(2 + 1) * 7] + 3941 + 18\} = 2.00 \text{ ksps}$$

8.3.5.18 Mode: PGA Bypass, Low-Power, no Scan, with Burst and decimation, only averaged data output to FIFO

This mode is similar to "PGA Bypass, Low-Power, no Scan, with no Burst", but with $2^{\text{adc_brst_cnt}}$ samples collected between each sleep cycle. The $2^{\text{adc_brst_cnt}}$ samples will be averaged and the result will be written to the FIFO. The ADC data rate is set by adjusting adc_slp_cnt and is a function of the burst count.

Set adc_acq_cnt to 1. If you need to run at the fastest rate, it can be set to 0. Set pga_trk_cnt to its minimum value of 9. Calculate adc_slp_cnt from the burst count, target data rate (F_{TDR}), and ADC clock rate (F_C).

$$\text{adc_slp_cnt} = (F_C / F_{\text{DR}}) - \{ \{ (2^{\text{adc_brst_cnt}} - 1) * (\text{pga_trk_cnt} + 1) \} + [(2^{\text{adc_brst_cnt}}) * (\text{adc_wake_cnt} + 1 + \text{adc_acq_cnt} + 1)] + \text{adc_flush} \} - 1$$

With adc_acq_cnt at 1, pga_trk_cnt at its minimum value of 9, adc_wake_cnt at its optimized value of 4, and adc_flush at 17, we simplify to:

$$\text{adc_slp_cnt} = (F_C / F_{\text{DR}}) - \{ \{ (2^{\text{adc_brst_cnt}} - 1) * 10 \} + [(2^{\text{adc_brst_cnt}}) * 7] + 18 \}$$

Now calculate the actual data rate based off of the calculated adc_slp_cnt .

$$F_{\text{DR}} = F_C / \{ \{ (2^{\text{adc_brst_cnt}} - 1) * (\text{pga_trk_cnt} + 1) \} + [(2^{\text{adc_brst_cnt}}) * (\text{adc_wake_cnt} + 1 + \text{adc_acq_cnt} + 1)] + \text{adc_flush} + \text{adc_slp_cnt} + 1 \}$$

With adc_acq_cnt at 1, pga_trk_cnt at its minimum value of 9, adc_wake_cnt at its optimized value of 4, and adc_flush at 17, we simplify to:

$$F_{\text{DR}} = F_C / \{ \{ (2^{\text{adc_brst_cnt}} - 1) * 10 \} + [(2^{\text{adc_brst_cnt}}) * 7] + \text{adc_slp_cnt} + 18 \}$$

Example calculation:

For this example, F_C is 8 MHz, adc_brst_cnt is set to 2 (4 samples per burst), and the target "averaged data" data rate is 13 ksps.

$$\text{adc_slp_cnt} = (8 \text{ MHz} / 13 \text{ ksps}) - \{ \{ (2^2) - 1 \} * 10 \} + [(2^2) * 7] + 18 = 539.38$$

Round 539.38 to the nearest integer, 539.

$$F_{\text{DR}} = 8 \text{ MHz} / \{ \{ (2^2) - 1 \} * 10 \} + [(2^2) * 7] + 539 + 18 = 13.01 \text{ ksps}$$

8.3.5.19 Mode: PGA Bypass, Low-Power, Scan, with Burst and decimation, only averaged data output to FIFO

This mode is similar to "PGA Bypass, Low-Power, no Scan, with Burst and decimation, only averaged data output to FIFO", except multiple samples of each scanned channel are being collected between each sleep cycle. In this mode, the ADC data rate is set by adjusting `adc_slp_cnt` and is a function of the scan count and burst count.

Set `adc_acq_cnt` to 1. If you need to run at the fastest rate, it can be set to 0. Set `pga_trk_cnt` to its minimum value of 9. Calculate `adc_slp_cnt` from the scan count, burst count, target data rate (F_{TDR}), and ADC clock rate (F_C).

$$adc_slp_cnt = (F_C / F_{DR}) - \{ \{ \{ (scn_cnt + 1) * (2^{\wedge} adc_brst_cnt) \} - 1 \} * (pga_trk_cnt + 1) \} + \{ (scn_cnt + 1) * (2^{\wedge} adc_brst_cnt) * (adc_wake_cnt + 1 + adc_acq_cnt + 1) \} + adc_flush \} - 1$$

With `adc_acq_cnt` at 1, `pga_trk_cnt` at its minimum value of 9, `adc_wake_cnt` at its optimized value of 4, and `adc_flush` at 17, we simplify to:

$$adc_slp_cnt = (F_C / F_{DR}) - \{ \{ \{ (scn_cnt + 1) * (2^{\wedge} adc_brst_cnt) \} - 1 \} * 10 \} + \{ (scn_cnt + 1) * (2^{\wedge} adc_brst_cnt) * 7 \} + 18 \}$$

Round `adc_slp_cnt` to the closest integer. Now calculate the actual data rate based off of the calculated `adc_slp_cnt`.

$$F_{DR} = F_C / \{ \{ \{ (scn_cnt + 1) * (2^{\wedge} adc_brst_cnt) \} - 1 \} * (pga_trk_cnt + 1) \} + \{ (scn_cnt + 1) * (2^{\wedge} adc_brst_cnt) * (adc_wake_cnt + 1 + adc_acq_cnt + 1) \} + adc_flush + adc_slp_cnt + 1 \}$$

With `adc_acq_cnt` at 1, `pga_trk_cnt` at its minimum value of 9, `adc_wake_cnt` at its optimized value of 4, and `adc_flush` at 17, we simplify to:

$$F_{DR} = F_C / \{ \{ \{ (scn_cnt + 1) * (2^{\wedge} adc_brst_cnt) \} - 1 \} * 10 \} + \{ (scn_cnt + 1) * (2^{\wedge} adc_brst_cnt) * 7 \} + adc_slp_cnt + 18 \}$$

Example calculation:

For this example, scanning 4 channels, `adc_brst_cnt` is set to 1 (2 samples per burst), F_C is 8 MHz, and the target "averaged data" data rate is 9 ksp/s.

$$adc_slp_cnt = (8 \text{ MHz} / 9 \text{ ksp/s}) - \{ \{ \{ (3 + 1) * (2^{\wedge} 1) \} - 1 \} * 8 \} + \{ (3 + 1) * (2^{\wedge} 1) \} * 7 \} + 18 \} = 758.89$$

Round 758.89 to the nearest integer, 759.

$$F_{DR} = 8 \text{ MHz} / \{ \{ \{ (3 + 1) * (2^{\wedge} 1) \} - 1 \} * 8 \} + \{ (3 + 1) * (2^{\wedge} 1) \} * 7 \} + 759 + 18 \} = 9.00 \text{ ksp/s}$$

8.3.5.20 Start the Measurement

To begin taking measurements, two register fields must be set.

When [ADC_CTRL0.adc_strt_mode](#) is 0, data collection begins when the CPU ADC data collection start register, [ADC_CTRL0.cpu_adc_strt](#), is set to 1. This is an active high start signal that starts the programmed data collection sequence when set to 1. It is self-clearing.

When [ADC_CTRL0.adc_strt_mode](#) is 1, the ADC data collection will start when Pulse Train 15 is high. Care must be taken in the pulse train programming to ensure that Pulse Train 15 is low before the ADC collection is complete or the collection process will restart.

8.3.6 Registers (ADC)

8.3.6.1 Module ADC Registers

Address	Register	32b Word Len	Description
0x40054000	ADC_CTRL0	1	ADC Control Register 0
0x40054004	ADC_PGA_CTRL	1	PGA Control Register
0x40054008	ADC_TG_CTRL0	1	ADC Timing Generator Control 0
0x4005400C	ADC_TG_CTRL1	1	ADC Timing Generator Control 1
0x40054010	ADC_LIMIT	1	ADC Limit Settings
0x40054014	ADC_INTR	1	ADC Interrupt Flags and Enable/Disable Controls
0x40054018	ADC_OUT	1	ADC Output Register
0x40054038	ADCCFG_CTRL1	1	ADC Control Register 1
0x4005403C	ADCCFG_SCAN1	1	ADC Auto-Scan Settings 1
0x40054040	ADCCFG_SCAN2	1	ADC Auto-Scan Settings 2
0x40054044	ADCCFG_RO_CAL0	1	ADC Relaxation Oscillator Calibration 0
0x40054048	ADCCFG_RO_CAL1	1	ADC Relaxation Oscillator Calibration 1
0x40109000	ADC_FIFO_DATA	1	ADC Sample FIFO

8.3.6.1.1 ADC_CTRL0**ADC_CTRL0.adc_wake_cnt**

Field	Bits	Default	Access	Description
adc_wake_cnt	3:0	4	R/W	ADC Wake Count

This field determines the delay (in ADC clock cycles) between powering up the ADC and beginning sample conversion.

This field should always be set to the optimized value of 4 (default value).

ADC_CTRL0.adc_tst_en

Field	Bits	Default	Access	Description
adc_tst_en	4	0	R/W	Reserved Field (Do Not Modify)

This field should not be modified by the user. For proper operation, this field must be left at its default value.

ADC_CTRL0.adc_strt_mode

Field	Bits	Default	Access	Description
adc_strt_mode	5	0	R/W	ADC Start Mode Control

ADC mode control for start of data collection.

- 0: Start when cpu_adc_start is set
- 1: Start controlled by PulseTrain[15] output

ADC_CTRL0.range

Field	Bits	Default	Access	Description
range	6	0	R/W	ADC Range Control (Bipolar Mode Only)

- 0: Bipolar operation from $-V_{ref}/2$ to $+V_{ref}/2$
- 1: Bipolar operation from $-V_{ref}$ to $+V_{ref}$

ADC_CTRL0.bi_pol

Field	Bits	Default	Access	Description
bi_pol	7	0	R/W	ADC Unipolar/Bipolar Mode Select

- 0: Unipolar operation from 0 to V_{ref}
- 1: Bipolar operation as determined by `adc_range`

ADC_CTRL0.adc_lmt_dmode

Field	Bits	Default	Access	Description
adc_lmt_dmode	10	0	R/W	Reserved Field - Do Not Modify

This field should not be modified by the user. For proper operation, this field must be left at its default value.

ADC_CTRL0.adc_smp_ext

Field	Bits	Default	Access	Description
adc_smp_ext	11	0	R/W	Reserved Field - Do Not Modify

This field should not be modified by the user. For proper operation, this field must be left at its default value.

ADC_CTRL0.adc_clk_en

Field	Bits	Default	Access	Description
adc_clk_en	12	0	R/W	ADC Interface Clock Enable

- 0: ADC clock is gated off
- 1: ADC clock is enabled

This bit must be set to 1 at least 4 ADC clock cycles prior to measuring analog data.

ADC_CTRL0.cpu_adc_rst

Field	Bits	Default	Access	Description
cpu_adc_rst	13	0	R/W	CPU ADC Reset

- 0: ADC normal operation
- 1: Resets ADC and clears ADC FIFO. Puts ADC in known state. Hardware resets this bit to 0 after one ADC clock cycle.

It is recommended to use this bit to reset the ADC before the first ADC sample is measured after any system reset.

ADC_CTRL0.cpu_adc_strt

Field	Bits	Default	Access	Description
cpu_adc_strt	14	0	R/W	CPU Start for ADC Data Collection

Active high start signal; used to start the ADC's programmed data collection sequence. Write to 1 to start data collection.

Cleared to 0 by hardware when data collection has completed.

ADC_CTRL0.adc_en

Field	Bits	Default	Access	Description
adc_en	15	0	R/W	CPU ADC Enable

- 0: ADC is powered down
- 1: ADC is powered up

ADC_CTRL0.adc_fifo_full

Field	Bits	Default	Access	Description
adc_fifo_full	18	0	R/O	ADC FIFO Almost Full Warning

- 0: FIFO is not full
- 1: FIFO level is > FIFO Almost Full level as determined by ADC_TG_CTRL1.fifo_af_cnt.

ADC_CTRL0.adc_fifo_empty

Field	Bits	Default	Access	Description
adc_fifo_empty	19	1	R/O	ADC FIFO Empty Flag

- 0: FIFO is not empty
- 1: FIFO is empty

ADC_CTRL0.avg_mode

Field	Bits	Default	Access	Description
avg_mode	21:20	0	R/W	ADC Decimation Filter Mode

- 0: Bypass (decimation filter not used)
- 1: Output average only (only in burst mode)
- 2: Reserved
- 3: Reserved

ADC_CTRL0.cpu_dac_strt

Field	Bits	Default	Access	Description
cpu_dac_strt	22	0	R/W	CPU DAC Sequence Start

Write to 1 to start the programmed DAC pattern generation sequence simultaneously on ADC Start. Phase locked with ADC. Cleared to 0 by hardware when operation has completed.

ADC_CTRL0.adc_clk_mode

Field	Bits	Default	Access	Description
adc_clk_mode	26:24	0	R/W	ADC Clock Mode

- 0: Full rate $\text{AdcClk}=\text{ClkIn}$; PLL generates 8MHz (default)
- 1: Half Rate $\text{AdcClk}=\text{ClkIn} / 2$
- 2: Third Rate $\text{AdcClk}=\text{ClkIn} / 3$

- 3: Quarter Rate $\text{AdcClk} = \text{ClkIn} / 4$
- 4: $\text{AdcClk} = \text{ClkIn} / 6$
- 5: $\text{AdcClk} = \text{ClkIn} / 8$
- 6: $\text{AdcClk} = \text{ClkIn} / 12$
- 7: Reserved

ADC_CTRL0.mode

Field	Bits	Default	Access	Description
mode	31:28	0	R/W	ADC Operating Mode

- 0: Collect ADC_TG_CTRL0.adc_smpl_cnt samples (default)
- 1: Collect adc_smpl_cnt samples with delay between bursts as set by ADC_TG_CTRL1.adc_slp_cnt
- 2: Collect samples continuously until disabled
- 3: Collect samples until disabled, with delay between bursts as set by ADC_TG_CTRL1.adc_slp_cnt
- 4/5/6/7: Manual register control: cpu_adc_start, cpu_adc_en, PGA
- 8: Channel Scan mode, adc_smpl_cnt samples
- 9: Channel Scan mode, adc_smpl_cnt samples with delay between bursts set by ADC_TG_CTRL1.adc_slp_cnt
- 10: Channel Scan mode, collect samples continuously until disabled
- 11: Channel Scan mode, collect samples continuously until disabled with delay between bursts set by ADC_TG_CTRL1.adc_slp_cnt

8.3.6.1.2 ADC_PGA_CTRL

ADC_PGA_CTRL.gain

Field	Bits	Default	Access	Description
gain	1:0	0	R/W	PGA Gain Setting

- 0: Gain x1 (default)
- 1: Gain x2
- 2: Gain x4
- 3: Gain x8

ADC_PGA_CTRL.cpu_pga_rst_clk_en

Field	Bits	Default	Access	Description
cpu_pga_rst_clk_en	2	0	R/W	Reserved Field - Do Not Modify

This field should not be modified by the user. For proper operation, this field must be left at its default value.

ADC_PGA_CTRL.cpu_pga_rst

Field	Bits	Default	Access	Description
cpu_pga_rst	3	0	R/W	Reserved Field - Do Not Modify

This field should not be modified by the user. For proper operation, this field must be left at its default value.

ADC_PGA_CTRL.cpu_pga_trk_dly

Field	Bits	Default	Access	Description
cpu_pga_trk_dly	4	1	R/W	Reserved Field - Do Not Modify

This field should not be modified by the user. For proper operation, this field must be left at its default value.

ADC_PGA_CTRL.cpu_pga_trk

Field	Bits	Default	Access	Description
cpu_pga_trk	5	0	R/W	Reserved Field - Do Not Modify

This field should not be modified by the user. For proper operation, this field must be left at its default value.

ADC_PGA_CTRL.cpu_pga_bypass

Field	Bits	Default	Access	Description
cpu_pga_bypass	6	0	R/W	PGA Bypass Control

- 0: Normal mode, PGA path active
- 1: PGA bypass, input mux drives ADC

ADC_PGA_CTRL.pga_wake_cnt

Field	Bits	Default	Access	Description
pga_wake_cnt	12:8	21	R/W	PGA Wakeup Counter

Length of time (defined in number of ADC clocks) that is allowed for the PGA to turn on. Minimum is 29. Longer values would allow longer PGA acquisition time on first sample or during reduced sample rate modes.

NOTE: The default value for this field should not be used. For optimal performance, firmware MUST set this field to 0x1D (29 decimal).

ADC_PGA_CTRL.mux_sw_ain

Field	Bits	Default	Access	Description
mux_sw_ain	13	0	R/W	Reserved Field - Do Not Modify

This field should not be modified by the user. For proper operation, this field must be left at its default value.

ADC_PGA_CTRL.mux_diff

Field	Bits	Default	Access	Description
mux_diff	14	1	R/W	Differential ADC Mux Select

Combines with mux_ch_sel for single/differential channel selection.

ADC_PGA_CTRL.mux_mode

Field	Bits	Default	Access	Description
mux_mode	15	0	R/W	Reserved Field - Do Not Modify

This field should not be modified by the user. For proper operation, this field must be left at its default value.

ADC_PGA_CTRL.pga_rst_clk_cnt

Field	Bits	Default	Access	Description
pga_rst_clk_cnt	23:20	4'Ah	R/W	PGA Reset Clock Count

Internal parameter used in ADC/PGA timing generation.

NOTE: The default value for this field should not be used. For optimal performance, firmware MUST set this field to 0xE (14 decimal).

ADC_PGA_CTRL.mux_ch_sel

Field	Bits	Default	Access	Description
mux_ch_sel	29:24	0	R/W	ADC Input Mux Select

Positive Input to ADC (decoded in concert with mux_diff)

For mux_ch_sel=0..15, mux_diff=0 (single ended mode):

- 00..15: Vp=AIN[0..15], Vn=VSSADC

For mux_ch_sel=0..15, mux_diff=1 (differential mode):

- 00/08: Vp=AIN0, Vn=AIN8
- 01/09: Vp=AIN1, Vn=AIN9
- 02/10: Vp=AIN2, Vn=AIN10
- 03/11: Vp=AIN3, Vn=AIN11
- 04/12: Vp=AIN4, Vn=AIN12
- 05/13: Vp=AIN5, Vn=AIN13
- 06/14: Vp=AIN6, Vn=AIN14
- 07/15: Vp=AIN7, Vn=AIN15

For other mux_ch_sel selections, mux_diff has no effect:

- 16: Vp=VSSADC, Vn=VSSADC
- 17: Vp=TMON, Vn=VSSADC
- 18: Vp=VDDA/4, Vn=VSSADC
- 19: Vp=PWRMON_TST, Vn=VSSADC
- 20: Vp=Ain0Div, Vn=VSSADC

- 32: Vp=MUXA, Vn=VSSADC
- 33: Vp=OUTA, Vn=VSSADC
- 34: Vp=MUXB, Vn=VSSADC
- 35: Vp=OUTB, Vn=VSSADC
- 36: Vp=MUXC, Vn=VSSADC
- 37: Vp=OUTC, Vn=VSSADC
- 38: Vp=MUXD, Vn=VSSADC
- 39: Vp=OUTD, Vn=VSSADC
- 48: Vp=VREFADC, Vn=VSSREF
- 49: Vp=VREFDAC, Vn=VSSREF
- 50: Vp=VREFADJ, Vn=VSSREF
- 51: Vp=Vdd3xtal Vn=xtaltst

All other values for mux_ch_sel are reserved.

8.3.6.1.3 ADC_TG_CTRL0

ADC_TG_CTRL0.pga_trk_cnt

Field	Bits	Default	Access	Description
pga_trk_cnt	15:0	0007h	R/W	PGA Track Window Count

Used to control the sample rate; track and hold window count for PGA before sample is collected, in units of ADC clock periods.

ADC_TG_CTRL0.adc_smpl_cnt

Field	Bits	Default	Access	Description
adc_smpl_cnt	31:16	0001h	R/W	ADC Sample Count

Field	Bits	Default	Access	Description
-------	------	---------	--------	-------------

Used to control the total number of samples collected when `adc_mode` is 0 or 1. If the decimation filter is enabled, this field still controls the total number of samples (data points) collected, but the sample data output to the ADC FIFO will be a smaller number of samples, depending on the current decimation filter setting.

Minimum valid setting for this field is 0001h. 0000h is not a valid setting for this field.

8.3.6.1.4 ADC_TG_CTRL1

ADC_TG_CTRL1.pga_acq_cnt

Field	Bits	Default	Access	Description
<code>pga_acq_cnt</code>	3:0	1	R/W	PGA Acquisition Window Count

This field sets the number of ADC clocks between the PGA entering its gain phase and the PGA entering its acquisition phase.

ADC_TG_CTRL1.fifo_af_cnt

Field	Bits	Default	Access	Description
<code>fifo_af_cnt</code>	7:4	7	R/W	ADC FIFO Almost Full Threshold Control

Sets the threshold value used to trigger the FIFO Almost Full condition. The actual threshold value used is `fifo_af_cnt+16`.

ADC_TG_CTRL1.adc_brst_cnt

Field	Bits	Default	Access	Description
<code>adc_brst_cnt</code>	10:8	0	R/W	ADC Burst Count Setting

- 0: 1 sample
- 1: 2 samples

- 2: 4 samples
- 3: 8 samples
- 4: 16 samples
- 5: 32 samples
- 6: 64 samples
- 7: 128 samples

ADC_TG_CTRL1.adc_acq_cnt

Field	Bits	Default	Access	Description
adc_acq_cnt	15:12	1	R/W	ADC Acquisition Window Count

Used to control the ADC acquisition window duration, in units of ADC clock periods. The actual window duration is set to `adc_acq_cnt+1`.

ADC_TG_CTRL1.adc_sleep_cnt

Field	Bits	Default	Access	Description
adc_sleep_cnt	31:16	1	R/W	ADC Sleep Count

This field sets the time (in ADC clock periods) between ADC samples or bursts in low power mode.

8.3.6.1.5 ADC_LIMIT**ADC_LIMIT.io_limit**

Field	Bits	Default	Access	Description
io_limit	15:0	8000h	R/W	ADC Sample Lower Limit

Lower limit used by out of range detector. Determines when the out_rng_if and lo_rng_if interrupt flags are set.

ADC_LIMIT.hi_limit

Field	Bits	Default	Access	Description
hi_limit	31:16	7FFFh	R/W	ADC Sample Upper Limit

Upper limit used by out of range detector. Determines when the out_rng_if and hi_rng_if interrupt flags are set.

8.3.6.1.6 ADC_INTR

ADC_INTR.fifo_af

Field	Bits	Default	Access	Description
fifo_af	6	0	R/O	ADC FIFO Almost Full Interrupt Flag

Set by hardware when the ADC FIFO exceeds the programmable Almost Full threshold level.

Cleared by hardware when the ADC FIFO drops back below the programmable Almost Full threshold level.

ADC_INTR.out_rng

Field	Bits	Default	Access	Description
out_rng	7	0	W1C	ADC Sample Out of Range Interrupt Flag

Set by hardware when a collected ADC sample is out of range on the high side (above the programmable hi_limit threshold) or the low side (below the programmable lo_limit threshold).

Write 1 to clear.

ADC_INTR.hi_rng

Field	Bits	Default	Access	Description
hi_rng	8	0	W1C	ADC Sample Above High Limit Interrupt Flag

Set by hardware when a collected ADC sample is out of range on the high side (above the programmable hi_limit threshold).

Write 1 to clear.

ADC_INTR.lo_rng

Field	Bits	Default	Access	Description
lo_rng	9	0	W1C	ADC Sample Below Low Limit Interrupt Flag

Set by hardware when a collected ADC sample is out of range on the low side (below the programmable lo_limit threshold).

Write 1 to clear.

ADC_INTR.done

Field	Bits	Default	Access	Description
done	10	0	W1C	ADC Data Collection Complete Interrupt Flag

Set by hardware when an ADC data collection cycle has completed.

Write 1 to clear.

ADC_INTR.fifo_uf

Field	Bits	Default	Access	Description
fifo_uf	11	0	W1C	ADC FIFO Underflow Interrupt Flag

Set by hardware when an ADC FIFO underflow condition occurs (empty FIFO is read).

Write 1 to clear.

ADC_INTR.fifo_of

Field	Bits	Default	Access	Description
fifo_of	12	0	W1C	ADC FIFO Overflow Interrupt Flag

Set by hardware when an ADC FIFO overflow condition occurs (full FIFO is written).

Write 1 to clear.

ADC_INTR.fifo_tf

Field	Bits	Default	Access	Description
fifo_tf	13	0	W1C	ADC FIFO Three-Quarters Full Interrupt Flag

Set by hardware when the level of the ADC FIFO reaches the three-quarter full point.

Write 1 to clear.

ADC_INTR.fifo_hf

Field	Bits	Default	Access	Description
fifo_hf	14	0	W1C	ADC FIFO Half Full Interrupt Flag

Set by hardware when the ADC FIFO reaches the half full point.

Write 1 to clear.

ADC_INTR.fifo_qf

Field	Bits	Default	Access	Description
fifo_qf	15	0	W1C	ADC FIFO One-Quarter Full Interrupt Flag

Set by hardware when the ADC FIFO is more than one-quarter full.

Write 1 to clear.

ADC_INTR.spst_sw0_ctrl

Field	Bits	Default	Access	Description
spst_sw0_ctrl	16	0	R/W	SPST0 Switch Control Mode

- 0: Switch is controlled by AFE_CTRL3.close_spst0.
- 1: Switch is controlled by pulse train output PT8.

ADC_INTR.spst_sw1_ctrl

Field	Bits	Default	Access	Description
spst_sw1_ctrl	17	0	R/W	SPST1 Switch Control Mode

- 0: Switch is controlled by AFE_CTRL3.close_spst1.
- 1: Switch is controlled by pulse train output PT9.

ADC_INTR.spst_sw2_ctrl

Field	Bits	Default	Access	Description
spst_sw2_ctrl	18	0	R/W	SPST2 Switch Control Mode

- 0: Switch is controlled by AFE_CTRL3.close_spst2.
- 1: Switch is controlled by pulse train output PT10.

ADC_INTR.spst_sw3_ctrl

Field	Bits	Default	Access	Description
spst_sw3_ctrl	19	0	R/W	SPST3 Switch Control Mode

- 0: Switch is controlled by AFE_CTRL3.close_spst3.
- 1: Switch is controlled by pulse train output PT11.

ADC_INTR.fifo_af_en

Field	Bits	Default	Access	Description
fifo_af_en	22	0	R/W	ADC FIFO Almost Full Interrupt Enable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

ADC_INTR.out_rng_en

Field	Bits	Default	Access	Description
out_rng_en	23	0	R/W	ADC Sample Out of Range Interrupt Enable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

ADC_INTR.hi_rng_en

Field	Bits	Default	Access	Description
hi_rng_en	24	0	R/W	ADC Sample Above High Limit Interrupt Enable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

ADC_INTR.lo_rng_en

Field	Bits	Default	Access	Description
lo_rng_en	25	0	R/W	ADC Sample Below Low Limit Interrupt Enable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

ADC_INTR.done_en

Field	Bits	Default	Access	Description
done_en	26	0	R/W	ADC Data Collection Complete Interrupt Enable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

ADC_INTR.fifo_uf_en

Field	Bits	Default	Access	Description
fifo_uf_en	27	0	R/W	ADC FIFO Underflow Interrupt Enable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

ADC_INTR.fifo_of_en

Field	Bits	Default	Access	Description
fifo_of_en	28	0	R/W	ADC FIFO Overflow Interrupt Enable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

ADC_INTR.fifo_tf_en

Field	Bits	Default	Access	Description
fifo_tf_en	29	0	R/W	ADC FIFO Three-Quarters Full Interrupt Enable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

ADC_INTR.fifo_hf_en

Field	Bits	Default	Access	Description
fifo_hf_en	30	0	R/W	ADC FIFO Half Full Interrupt Enable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

ADC_INTR.fifo_qf_en

Field	Bits	Default	Access	Description
fifo_qf_en	31	0	R/W	ADC FIFO One-Quarter Full Interrupt Enable

- 0: Interrupt is disabled
- 1: Interrupt is enabled

8.3.6.1.7 ADC_OUT

ADC_OUT.data_reg

Field	Bits	Default	Access	Description
data_reg	15:0	0000h	R/O	ADC Current Valid Data Sample

Direct read of the ADC output data; cleared on cpu_adc_strt and registered during adc_dv.

8.3.6.1.8 ADCCFG_CTRL1

ADCCFG_CTRL1.scan_cnt

Field	Bits	Default	Access	Description
scan_cnt	18:16	0	R/W	Channel Scan Count

Number of channels to scan (if an ADC scanning mode is active) is given by the value of this field + 1:

- 0: Scan 1 channel (default)
- 1: Scan 2 channels
- 2: Scan 3 channels
- 3: Scan 4 channels
- 4: Scan 5 channels
- 5: Scan 6 channels
- 6: Scan 7 channels
- 7: Scan 8 channels

8.3.6.1.9 ADCCFG_SCAN1

ADCCFG_SCAN1.adc_scan0

Field	Bits	Default	Access	Description
adc_scan0	7:0	00h	R/W	ADC Scan Configuration - Channel 0

- bits 3:0 - mux_ch_sel setting
- bits 5:4 - pga_gain setting
- bit 6 - mux_diff setting
- bit 7 - Not used, should be set to 0

Note ADC unipolar/bipolar mode and bipolar range must be the same for all channels.

ADCCFG_SCAN1.adc_scan1

Field	Bits	Default	Access	Description
adc_scan1	15:8	00h	R/W	ADC Scan Configuration - Channel 1

- bits 3:0 - mux_ch_sel setting
- bits 5:4 - pga_gain setting
- bit 6 - mux_diff setting
- bit 7 - Not used, should be set to 0

Note ADC unipolar/bipolar mode and bipolar range must be the same for all channels.

ADCCFG_SCAN1.adc_scan2

Field	Bits	Default	Access	Description
adc_scan2	23:16	00h	R/W	ADC Scan Configuration - Channel 2

- bits 3:0 - mux_ch_sel setting
- bits 5:4 - pga_gain setting
- bit 6 - mux_diff setting

- bit 7 - Not used, should be set to 0

Note ADC unipolar/bipolar mode and bipolar range must be the same for all channels.

ADCCFG_SCAN1.adc_scan3

Field	Bits	Default	Access	Description
adc_scan3	31:24	00h	R/W	ADC Scan Configuration - Channel 3

- bits 3:0 - mux_ch_sel setting
- bits 5:4 - pga_gain setting
- bit 6 - mux_diff setting
- bit 7 - Not used, should be set to 0

Note ADC unipolar/bipolar mode and bipolar range must be the same for all channels.

8.3.6.1.10 ADCCFG_SCAN2

ADCCFG_SCAN2.adc_scan4

Field	Bits	Default	Access	Description
adc_scan4	7:0	00h	R/W	ADC Scan Configuration - Channel 4

- bits 3:0 - mux_ch_sel setting
- bits 5:4 - pga_gain setting
- bit 6 - mux_diff setting
- bit 7 - Not used, should be set to 0

Note ADC unipolar/bipolar mode and bipolar range must be the same for all channels.

ADCCFG_SCAN2.adc_scan5

Field	Bits	Default	Access	Description
adc_scan5	15:8	00h	R/W	ADC Scan Configuration - Channel 5

- bits 3:0 - mux_ch_sel setting
- bits 5:4 - pga_gain setting
- bit 6 - mux_diff setting
- bit 7 - Not used, should be set to 0

Note ADC unipolar/bipolar mode and bipolar range must be the same for all channels.

ADCCFG_SCAN2.adc_scan6

Field	Bits	Default	Access	Description
adc_scan6	23:16	00h	R/W	ADC Scan Configuration - Channel 6

- bits 3:0 - mux_ch_sel setting
- bits 5:4 - pga_gain setting
- bit 6 - mux_diff setting
- bit 7 - Not used, should be set to 0

Note ADC unipolar/bipolar mode and bipolar range must be the same for all channels.

ADCCFG_SCAN2.adc_scan7

Field	Bits	Default	Access	Description
adc_scan7	31:24	00h	R/W	ADC Scan Configuration - Channel 7

- bits 3:0 - mux_ch_sel setting
- bits 5:4 - pga_gain setting
- bit 6 - mux_diff setting
- bit 7 - Not used, should be set to 0

Note ADC unipolar/bipolar mode and bipolar range must be the same for all channels.

8.3.6.1.11 ADCCFG_RO_CAL0

ADCCFG_RO_CAL0.ro_cal_en

Field	Bits	Default	Access	Description
ro_cal_en	0	0	R/W	Relaxation Oscillator Auto-Calibration Enable

- 0: Use factory trim setting for high-speed relaxation osc.
- 1: Use auto calibration loop register instead of the factory trim setting.

ADCCFG_RO_CAL0.ro_cal_run

Field	Bits	Default	Access	Description
ro_cal_run	1	0	R/W	Enable Closed Loop Control

- 0: Closed loop control is disabled (default).
- 1: Enable closed loop control (must set ro_cal_en=1 first)

ADCCFG_RO_CAL0.ro_cal_load

Field	Bits	Default	Access	Description
ro_cal_load	2	0	W1S	Load Initial RO Calibration Trim

When a 1 value is written to this bit, the initial relaxation oscillator trim setting will be loaded from the trm_init field. Attempts to write this bit to 0 directly will be ignored.

Once the trim load operation has completed, hardware will automatically clear this bit to 0.

ADCCFG_RO_CAL0.trm_mu

Field	Bits	Default	Access	Description
trm_mu	19:8	000h	R/W	Auto Calibration Loop Gain

ADCCFG_RO_CAL0.ro_trm

Field	Bits	Default	Access	Description
ro_trm	31:23	000000000b	R/O	Auto Calibration Loop Register Readback

8.3.6.1.12 ADCCFG_RO_CAL1**ADCCFG_RO_CAL1.trm_init**

Field	Bits	Default	Access	Description
trm_init	8:0	000000000b	R/W	Initial Setting for RO Trim Calibration

Contains the setting to load to auto calibration register (2's complement format). Used as an initial condition for the closed loop calibration.

ADCCFG_RO_CAL1.trm_min

Field	Bits	Default	Access	Description
trm_min	18:10	100000000b	R/W	Trim Lower Limit

Lower limit on adapted trim value (2's complement format)

ADCCFG_RO_CAL1.trm_max

Field	Bits	Default	Access	Description
trm_max	28:20	011111111b	R/W	Trim Upper Limit

Upper limit on adapted trim value (2's complement format)

8.3.6.1.13 ADC_FIFO_DATA

Default	Access	Description
n/a	R/W	ADC Sample FIFO

Read to pull converted sample data from ADC FIFO.

8.4 DAC

8.4.1 DAC Overview

The **MAX32600** features four voltage output DACs: two with 12-bit resolution and two with 8-bit resolution. Each DAC can be set independently to generate either a static output voltage or to generate a series of preloaded sample outputs at a specified sample rate.

The **MAX32600** DAC modules support the following features:

- Configurable clock rate and output sample rate (per DAC instance).

- Output voltage reference selectable (one setting for all DACs) between ADC and DAC reference levels.
- Each DAC can be set to output a static voltage level, a preset number of samples at a configurable sample rate, or samples continuously at a configurable sample rate.
- Interpolation filter (per DAC instance) allows for linearly interpolated output samples to be generated between each pair of output samples (2 to 1, 4 to 1, or 8 to 1).
 - **NOTE:** This saves power by reducing AHB bus traffic and reduces memory bandwidth by lowering AHB transaction requirements to keep the DAC full. The bus traffic reduction and memory savings are proportional to the interpolation rate.
- DAC output samples are pulled from a FIFO that can be reloaded by either the PMU or CPU while the DAC is running to allow continuous sample output generation with no gaps.
- Ability to start pattern generation synchronized to a common start strobe from the ADC.

8.4.2 DAC Interface

Each DAC instance on the **MAX32600** is configured using a set of control and configuration registers mapped to the APB bus. When generating a series of output voltage samples, the high-speed DAC FIFOs are used to provide the sample values for the DAC; these are loaded over the AHB bus by either the CPU or by the

PMU. Burst transfers can be used to quickly load the DAC FIFOs with sample data stored in RAM or constant flash space.

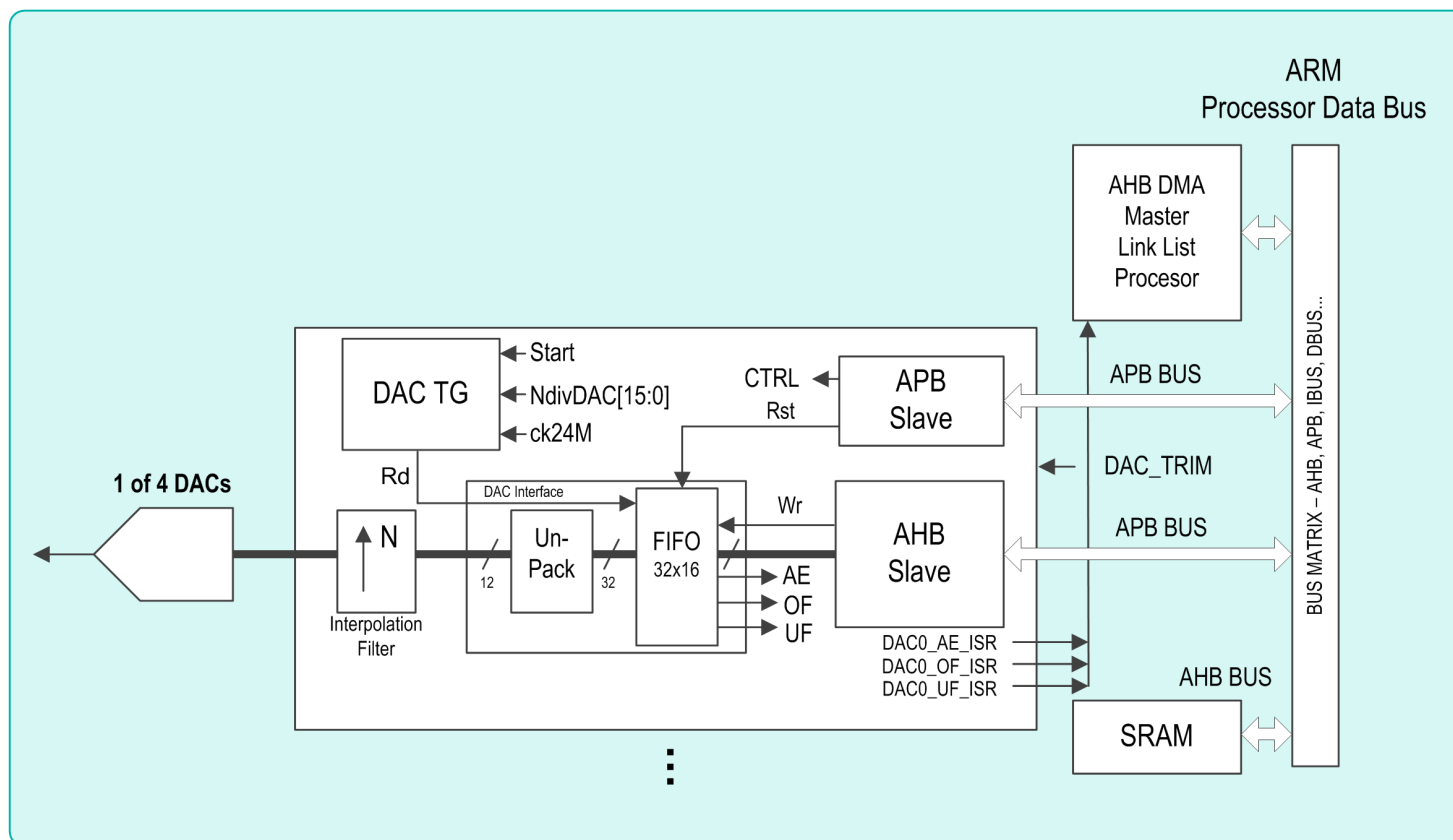


Figure 8.8: DAC Interface Diagram

8.4.3 DAC Operation

Note The following sections will refer to “the DAC”. It is understood that this means any one of the four DACs available on the device.

In order to generate output voltages using the DAC, there are several steps that must be completed. Initially, the DAC must be configured for the desired mode of operation and certain options must be selected. Clocks and power settings/enables that are required for DAC operation must be set as well. Certain settings and options are selected once for all DACs in the system (common DAC configuration settings), while other settings and options are selected independently for each DAC instance (individual DAC configuration settings); these are discussed separately below.

Once all configuration settings and other options have been selected, the DAC can be set to generate the desired output waveform. The procedure for doing so varies depending on the needs of the application; the DAC can be set to generate either a static output voltage (which can be adjusted manually by the CPU or PMU at any time), or the DAC can be set to generate a series of output voltage samples based on values that have been loaded into the DAC FIFO.

8.4.3.1 Common DAC Configuration Settings

Certain pre-operation steps (enabling/powering up modules/functions) and selection of common options must be completed before any of the DACs can be used. Where these settings represent choices between multiple options, these apply to all four of the DAC instances.

Powering Up the Analog Front End

As with other operations/functions involving the analog front end on the **MAX32600**, the AFE must be globally powered up before any of the DACs can be used. This is performed by setting `PWRMAN_PWR_RST_CTRL.afe_powered` to 1.

Configuring the Analog Matrix

For most applications, it will be necessary to configure the analog matrix to route DAC output signals as needed in combination with the on-board op amps, SPST switches, and other analog multiplexing options. This configuration is separate from DAC configuration, but it must be performed as needed before the DAC is used. For more details on this topic, refer to the [AFE Reconfiguration Matrix](#).

DAC Voltage Reference Selection

Output voltages generated by the DACs are based on the currently selected common DAC voltage reference. This reference is selected for all DACs and must be selected before any DAC can be used. The DAC voltage reference is selectable between the two configurable voltage references V_{ADC} and V_{DAC} , which in turn can be based on either an internal bandgap or external reference level. Refer to the [Analog Front End](#) for more information on this topic.

8.4.3.2 Individual DAC Configuration Settings

Once the `shared` settings to all DACs have been configured, there are other individual settings for each DAC that must be set (on a per-instance basis) for each DAC that will be used. These are detailed below.

DAC Clock Enable / Clock Scaling

Each DAC instance runs from an independently scalable clock that is generated from the currently selected system clock source. Because this clock is used by the APB and AHB interface logic as well as the internal DAC state machine logic, the clock must be enabled before attempting to access any of the DAC APB registers or the AHB DAC FIFO.

The clock enable / scaling setting is configured separately for each of the four DAC instances using registers in the Clock Manager module, as shown in the [table](#) below.

DAC Instance	Module Clock Scaling Register
0 (12-bit)	CLKMAN_CLK_CTRL_14_DAC0.dac0_clk_scale
1 (12-bit)	CLKMAN_CLK_CTRL_15_DAC1.dac1_clk_scale
2 (8-bit)	CLKMAN_CLK_CTRL_16_DAC2.dac2_clk_scale
3 (8-bit)	CLKMAN_CLK_CTRL_17_DAC3.dac3_clk_scale

Each of these registers contains a 4-bit `dac[n]_clk_scale` field which is used to enable the module clock for that DAC, and if desired, to scale the module clock frequency by dividing it down from the system clock source, as detailed [below](#).

dac[n]_clk_scale (4b)	DAC Clock Setting
0000b	Disabled
0001b	SysClkSource / 1
0010b	SysClkSource / 2
0011b	SysClkSource / 4
0100b	SysClkSource / 8
0101b	SysClkSource / 16
0110b	SysClkSource / 32
0111b	SysClkSource / 64
1000b	SysClkSource / 128
1001b	SysClkSource / 256
Other settings	Reserved

DAC Power-Up and Enable Sequence

Once the DAC module clock has been enabled, the DAC instance itself may be powered on and enabled for use by the application firmware. In order to do this, the following register settings must be made in the `DACn_CTRL0` register.

Note These writes do not have to be made one at a time; all the fields listed below can be written in a single write to `DACn_CTRL0`.

- Write `DACn_CTRL0.reset` to 1. This will cause the DAC timing generator and other internal logic to be reset (but will not reset the contents of the DAC registers). This bit is self-clearing and will automatically clear back to 0 after the DAC is reset.
- Write `DACn_CTRL0.clock_gate_en` to 1. This will turn on the clock for the DAC logic.
- Write `DACn_CTRL0.power_on` to 1. This powers up the DAC analog circuitry.
- When powering up one of the 12-bit DAC instances (DAC0 or DAC1), the DAC power mode must be selected; this mode is specified by the combination of the `DACn_CTRL0.power_mode_1_0` and `DACn_CTRL0.power_mode_2` fields as shown below. If any power level other than FullPwr is used, then the negative DAC output should be used and other compensations made as described in the [Reduced Power Level Modes for 12-bit DAC Instances](#) section.

power_mode_2 (1b)	power_mode_1_0 (2b)	DAC Power Level (DAC0 and DAC1 only)
0	01b	PwrLvl0 (48 μ A)
0	11b	PwrLvl1 (130 μ A)
1	01b	PwrLvl2 (210 μ A)
1	11b	FullPwr Mode (291 μ A)

Selecting the DAC Sample Rate and Interpolation Mode

Note The settings described below only apply if the DAC will be used to generate a dynamic output voltage pattern by loading a series of sample code values into the DAC FIFO. If the DAC will be used to generate a static output voltage by loading a single value into the DAC FIFO, then these settings do not need to be written.

Along with the DAC module clock, there are two additional parameters which determine the rate at which the voltage value changes at the DAC output.

- The **Rate Count parameter** determines the delay (measured in number of DAC clock cycles) between one output voltage from the DAC and the next.

Note If Interpolation Mode is enabled (as discussed below), this parameter applies to the output rate of values after interpolation. The Rate Count parameter is set by writing to the `DACn_RATE.rate_cnt` field; the actual value of the Rate Count parameter is equal to $(rate_cnt + 2)$. For example, to set the Rate Count to 20 clock cycles, the `rate_cnt` field should be set to 18.

- The optional **Interpolation Mode parameter** can be set to generate linearly interpolated data points between each two sample code values loaded into the FIFO. For example, if the FIFO contains two data points, A1 and A2, and $A2 > A1$, the output sample sequence generated will vary depending on interpolation mode as follows:
 - *No interpolation:*
 - * A1
 - * A2
 - *Interpolation 2-to-1:*
 - * A1
 - * $(A1 + (A2 - A1) \times \frac{1}{2})$
 - * A2
 - *Interpolation 4-to-1:*
 - * A1
 - * $(A1 + (A2 - A1) \times \frac{1}{4})$
 - * $(A1 + (A2 - A1) \times \frac{1}{2})$
 - * $(A1 + (A2 - A1) \times \frac{3}{4})$
 - * A2
 - *Interpolation 8-to-1:*
 - * A1
 - * $(A1 + (A2 - A1) \times \frac{1}{8})$
 - * $(A1 + (A2 - A1) \times \frac{2}{8})$
 - * $(A1 + (A2 - A1) \times \frac{3}{8})$
 - * $(A1 + (A2 - A1) \times \frac{4}{8})$
 - * $(A1 + (A2 - A1) \times \frac{5}{8})$
 - * $(A1 + (A2 - A1) \times \frac{6}{8})$
 - * $(A1 + (A2 - A1) \times \frac{7}{8})$
 - * A2

Increasing interpolation “smooths” the output waveform (as shown below, where the interpolation rate increases from left to right, but the additional data points inserted may not be exactly accurate for the desired waveform. For the highest quality of the generated output waveform, the maximum number of true waveform data points should be loaded via the DAC FIFO. Interpolation allows the bandwidth of data transfer into the DAC FIFO to be reduced (and possibly to reduce the amount of computation that the application has to perform), but comes at a trade-off of a lower-quality wave output in most cases.

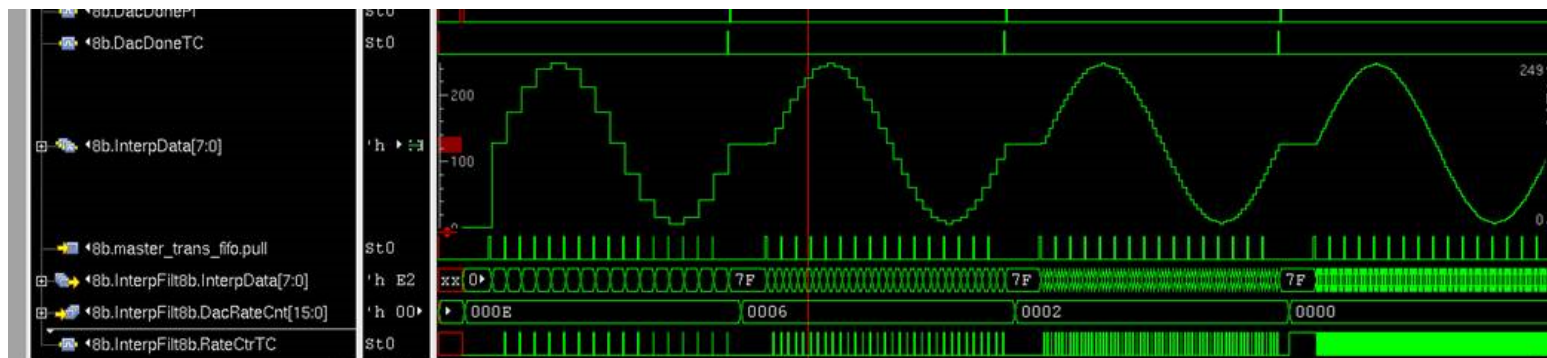


Figure 8.9: DAC Interpolation Waveform

If the output waveform is largely linear slopes (e.g., a triangle or sawtooth wave), then interpolated values may be equivalent to those that the application would have calculated itself. In this case, interpolation can be used with no reduction in waveform quality.

As mentioned above, the Rate Count parameter applies to output sample voltages after interpolation (if used) has taken place. Because increasing interpolation increases the number of output samples, if this is not corrected for, the waveform output duration will increase proportionally. To fix this, the Rate Count must be reduced when increasing interpolation in order to keep the same waveform output frequency, as in the example shown below.

Original Rate Count	Original rate_cnt value	Interpolation Used	New Rate Count Needed	New rate_cnt value
32	30	None	32	30
32	30	2-to-1	16	14
32	30	4-to-1	8	6
32	30	8-to-1	4	2

Setting the DAC Start Mode

The final parameter that must be written to configure the DAC instance is the DAC Start Mode. This setting determines when the DAC will begin generating output voltage data according to the specified settings, based on the voltage samples loaded in the DAC FIFO. To set this parameter, write to the `DACn_CTRL0.start_mode` field; the options are as follows:

- **Start when the FIFO Is Not Empty (00b)** – In this mode, the DAC will begin generating output voltages as soon as there is data available in the FIFO for it to read.
- **Start on DAC Generated Start Strobe (10b)** – In this mode, the DAC output sequence is started by writing the `DACn_CTRL0.cpu_start` bit to 1.
- **Start on ADC Generated Start Strobe (01b)** – In this mode, a common start signal from the ADC (which is issued when the ADC begins to convert samples) is used to trigger the start of the DAC output pattern. This mode should be used when it is necessary to synchronize the start of the DAC output pattern to the start of ADC sample collection, or to synchronize the outputs of two or more DAC instances.

8.4.3.3 DAC Voltage Generation

8.4.3.3.1 Generating DAC Static Voltage Output

To generate a static voltage output with the DAC (which can be changed at any time by loading a new output voltage value), the DAC must be set to the mode “Output New Values from FIFO When Available” by writing 00b to `DACn_CTRL0.op_mode`. With this mode enabled, whenever a sample value is loaded into the DAC FIFO, the DAC will immediately pull the sample value, set the output voltage statically to that value, and then hold that value until a new sample is loaded into the DAC FIFO.

Note When using this mode, the Rate Count and Interpolation Mode parameters do not apply.

After setting the DAC operating mode as described above, the voltage value can be loaded to the FIFO by writing the desired sample output code to the `DACn_FIFO_OUTPUT` address location as follows:

- For an 8-bit DAC instance, the value should be written as a byte (8-bit) write; no padding is needed.
- For a 12-bit DAC instance, the value should be written as a word (16-bit) write. The code value should be MSB aligned (shifted upwards four bits) to occupy the highest 12 bits of the 16-bit word. The lowest four bits of the 16-bit word should be padded to zeroes.

8.4.3.3.2 Generating DAC Dynamic Voltage Output

To generate a dynamic voltage output sequence from the DAC (where voltage output values are generated based on sample code values pulled from the DAC FIFO), the DAC operating mode should be set to one of the following two options by writing to `DACn_CTRL0.op_mode`.

Sample Count Mode

When using this DAC operating mode, the total number of sample code values that the DAC should pull from the FIFO to generate the pattern must be specified. This sample count must be written to `DACn_RATE.sample_cnt`. Once the DAC has pulled this number of samples from the FIFO, the DAC will halt and hold the last output voltage.

Note that the sample count will normally be larger than the number of samples that can fit in the DAC FIFO (since the FIFO is only 16 levels deep for the 8-bit DAC instances DAC2/DAC3, and only 32 levels deep for the 12-bit DAC instances DAC0/DAC1). To prevent gaps in the DAC waveform output, the application must monitor the DAC FIFO status (either directly via CPU interrupt or indirectly using the PMU) and copy additional sample code values into the DAC FIFO when the FIFO nears the empty state. This is done using the DAC FIFO Almost Empty programmable threshold and the DAC FIFO Almost Empty status flag (for use by the PMU) and accompanying interrupt flag and interrupt enable (for use by the CPU).

If Interpolation Mode is being used, it is important to note that the Sample Count parameter only refers to the number of sample code values that the DAC pulls from the FIFO. Using the interpolation function will increase the number of voltage values (each with a delay between them defined by the Rate Count parameter) that are generated at the DAC output, but it does not change the effective Sample Count value.

Continuous Mode

This mode operates in the same manner as Sample Count Mode except that no sample count value is written. Instead, the DAC continues to pull values from the FIFO (at a rate controlled by the Rate Count parameter) until the DAC is disabled, reset, or switched to a different operating mode. This mode is useful when the application needs the DAC to continue to output a repeating voltage pattern indefinitely or until some condition occurs.

As with Sample Count Mode, the interpolation function can be used if needed, and it operates in the same manner.

8.4.4 Additional Topics

Certain DAC operating modes or application / system conditions require special handling or correction / compensation methods to achieve ideal voltage output accuracy.

8.4.4.1 Reduced Power Level Modes for 12-bit DAC Instances

The two 12-bit DAC instances (DAC0 and DAC1) provide a number of lower-power modes that can be used to operate the DAC instance at less than full power. It is important to note that while the current of the DAC can be adjusted using this setting, only the negative DAC output can have its impedance changed. As a result, at any power level other than full power, the negative DAC output should be used instead of the positive one. To correct for this, all voltage output codes used by the DAC should be corrected to:

$$(\text{code used for low power}) = 4095 - (\text{original DAC code used at full power})$$

The DAC outputs are high impedance. Each DAC has a built-in load that is correct on the positive output at full power and is correct on the negative output at all power levels. As power is reduced, though, the linearity and gain accuracy degrade. It is possible to add an external load in some configurations, but this is not advised.

Note Switches are included in the DAC resistor network in order to facilitate proper variable power function.

Power Control Table

Power Mode(pm)	Output to Use	Factor	Output Current μA vs Reference Voltage				Load (Ohms)
			1.024V	1.5V	2.048V	2.5V	
0x01 (PM0)	Negative	1/16X	10.5	15.4	21	25.6	97680
0x03 (PM1)	Negative	5/16X	62.9	92.1	125.8	153.6	16280
0x05 (PM2)	Negative	11/16X	115.3	168.9	230.6	281.5	8880
0x07 (PM3)	Positive	1X	167.7	245.7	335.5	409.5	6105
0x00	Negative	1/95X	6.6				546,510

8.4.4.2 Correcting for Distortion at High Output Frequency

The DAC is vulnerable to second order harmonic behavior while operating at higher frequencies. As operating frequency increases, the time between bounce events decreases until the reference voltage does not have time to recover before the next bounce. When this point is reached, the error current accumulates until the direction of the DAC current changes.

Internal compensation in the DAC block is able to correct for this distortion while the code is rising; however, above 1MHz, the compensation function becomes less and less effective. Another limitation of the internal compensation on the DAC is its inability to go negative; thus, it cannot properly correct for distortion caused by falling edges. The proper procedure to compensate for this is to predistort the input waveform to the DAC as described below.

$$\sin(x) + (k \times \sin(2x + t))$$

Where:

- k is a function of frequency and code, varying from 0 to 0.12
- Suggested setting for k is: $k = (0.005 \times f)$ if code < 2048, otherwise 0 should be used
 - Where f is frequency of DAC in MHz.

8.4.5 Registers (DAC)

8.4.5.1 Module DAC Registers

Address	Register	32b Word Len	Description
0x40050000	DAC0_CTRL0	1	DAC Control Register 0
0x40050004	DAC0_RATE	1	DAC Output Rate/Sample Control
0x40050008	DAC0_CTRL1_INT	1	DAC Control Register 1, Interrupt Flags and Enable/Disable
0x40051000	DAC1_CTRL0	1	DAC Control Register 0
0x40051004	DAC1_RATE	1	DAC Output Rate/Sample Control
0x40051008	DAC1_CTRL1_INT	1	DAC Control Register 1, Interrupt Flags and Enable/Disable
0x40052000	DAC2_CTRL0	1	DAC Control Register 0
0x40052004	DAC2_RATE	1	DAC Output Rate/Sample Control
0x40052008	DAC2_CTRL1_INT	1	DAC Control Register 1, Interrupt Flags and Enable/Disable
0x40053000	DAC3_CTRL0	1	DAC Control Register 0
0x40053004	DAC3_RATE	1	DAC Output Rate/Sample Control
0x40053008	DAC3_CTRL1_INT	1	DAC Control Register 1, Interrupt Flags and Enable/Disable
0x40105000	DAC0_FIFO_OUTPUT	1	DAC Output FIFO
0x40106000	DAC1_FIFO_OUTPUT	1	DAC Output FIFO
0x40107000	DAC2_FIFO_OUTPUT	1	DAC Output FIFO
0x40108000	DAC3_FIFO_OUTPUT	1	DAC Output FIFO

8.4.5.1.1 DACn_CTRL0

DACn_CTRL0.fifo_ae_cnt

Field	Bits	Default	Access	Description
fifo_ae_cnt	3:0	0100b	R/W	DAC FIFO Almost Empty Threshold

Field	Bits	Default	Access	Description
-------	------	---------	--------	-------------

Sets the programmable threshold for detection of the Almost Empty condition for the DAC output FIFO.

The default value for this setting is 4. The allowable range is from 0 to 15, with 8 being a typically used value.

DACn_CTRL0.fifo_almost_full

Field	Bits	Default	Access	Description
fifo_almost_full	5	0	R/O	DAC FIFO Almost Full Flag

This read-only status flag returns 1 when the DAC FIFO is in the Almost Full condition, and returns 0 otherwise.

DACn_CTRL0.fifo_empty

Field	Bits	Default	Access	Description
fifo_empty	6	0	R/O	DAC FIFO Empty Flag

This read-only status flag returns 1 when the DAC FIFO is completely empty, and returns 0 otherwise.

DACn_CTRL0.fifo_almost_empty

Field	Bits	Default	Access	Description
fifo_almost_empty	7	0	R/O	DAC FIFO Almost Empty Flag

This read-only status flag returns 1 when the DAC FIFO is in the Almost Empty condition, and returns 0 otherwise.

DACn_CTRL0.interp_mode

Field	Bits	Default	Access	Description
interp_mode	10:8	000	R/W	DAC Output Interpolation Mode

- 0: Disabled
- 1: 2:1 Interpolation
- 2: 4:1 Interpolation
- 3: 8:1 Interpolation
- Other values are reserved.

DACn_CTRL0.fifo_af_cnt

Field	Bits	Default	Access	Description
fifo_af_cnt	15:12	4'7h	R/W	DAC FIFO Almost Full Threshold

Sets the programmable threshold for detection of the Almost Full condition for the DAC output FIFO.

The actual threshold value is determined by adding 16 to the contents of this field, which gives an allowable range for the threshold value from 16 to 31. The default value for the threshold is 23.

DACn_CTRL0.start_mode

Field	Bits	Default	Access	Description
start_mode	17:16	00b	R/W	DAC Output Start Mode

- 0: Start when FIFO is not empty (default)
- 1: Start on ADC generated Start strobe
- 2: Start when cpu_start is written to 1

- 3: Reserved

DACn_CTRL0.cpu_start

Field	Bits	Default	Access	Description
cpu_start	20	0	R/W	DAC Output CPU Start

Manual control for DAC output sequence start, controlled by CPU (that is, written by application firmware) as opposed to a start condition generated by the ADC or an automatic start triggered by data being loaded into the DAC FIFO. If the start_mode field is set to 10b to enable this mode, then writing cpu_start to 1 will trigger a DAC output sequence.

Automatically cleared to 0 by hardware.

DACn_CTRL0.op_mode

Field	Bits	Default	Access	Description
op_mode	25:24	00b	R/W	DAC Operation Mode

- 0: Whenever there is at least one data point available in the FIFO, output that data point to the DAC immediately. The data point voltage output in this manner will be maintained at the DAC output until another data point is available in the FIFO. This mode is intended for generation of DAC voltage outputs which remain static for relatively long periods of time. Normally, when using this mode, only one data point should be loaded into the FIFO at a time.
- 1: Output DACn_RATE.sample_cnt data points (once) from FIFO at an output rate defined by DACn_RATE.rate_cnt.
- 2: Reserved
- 3: Output DACn_RATE.sample_cnt data points from FIFO at an output rate defined by DACn_RATE.rate_cnt. Repeat this operation (which will eventually require that more data be loaded to the DAC FIFO) until the DAC is disabled or op_mode is changed to a different setting.

DACn_CTRL0.power_mode_1_0

Field	Bits	Default	Access	Description
power_mode_1_0	27:26	00b	R/W	DAC Power Mode (bits 1 and 0)

The entire power_mode field (which consists of bits 30, 27, and 26) sets the power mode for the DAC instance as:

- 001b: PwrLvl0 Mode (48 μ A)
- 011b: PwrLvl1 Mode (130 μ A)
- 101b: PwrLvl2 Mode (210 μ A)
- 111b: PwrLvl3 Mode (291 μ A)

DACn_CTRL0.power_on

Field	Bits	Default	Access	Description
power_on	28	0	R/W	DAC Power On Enable

Write to 1 to power on the DAC circuitry.

DACn_CTRL0.clock_gate_en

Field	Bits	Default	Access	Description
clock_gate_en	29	0	R/W	DAC Interface Clock Gate Enable

Write to 1 to enable clock gating for the DAC interface.

DACn_CTRL0.power_mode_2

Field	Bits	Default	Access	Description
power_mode_2	30	0	R/W	DAC Power Mode (bit 2)

See the description for power_mode_1_0.

DACn_CTRL0.reset

Field	Bits	Default	Access	Description
reset	31	0	R/W	DAC Reset

Write to 1 to reset the DAC.

Automatically cleared to 0 by hardware when operation has completed.

8.4.5.1.2 DACn_RATE

DACn_RATE.rate_cnt

Field	Bits	Default	Access	Description
rate_cnt	15:0	0001h	R/W	DAC Output Rate Control

When op_mode is set to 01b or 11b, this field sets the delay between output samples as

$$T_s = (\text{rate_cnt} + 2) * (1/24\text{MHz})$$

DACn_RATE.sample_cnt

Field	Bits	Default	Access	Description
sample_cnt	31:16	0001h	R/W	DAC Output Sample Count

When `op_mode` is set to 01b and interpolation mode is active, this field sets the total number of data points to output as $(\text{sample_cnt}-1) \times (2^{\text{interp_mode}}) + 1$

When `op_mode` is set to 01b and interpolation is disabled, this field sets the number of data points to output directly.

8.4.5.1.3 DACn_CTRL1_INT

DACn_CTRL1_INT.out_done_if

Field	Bits	Default	Access	Description
out_done_if	0	0	W1C	DAC Output Done Interrupt Flag

Write 1 to clear.

Written to 1 by hardware when DAC output has completed.

DACn_CTRL1_INT.underflow_if

Field	Bits	Default	Access	Description
underflow_if	1	0	W1C	FIFO Underflow Interrupt Flag

Write 1 to clear.

Written to 1 by hardware when FIFO underflow occurs (hardware attempts to pull DAC value when FIFO is empty)

DACn_CTRL1_INT.almost_empty_if

Field	Bits	Default	Access	Description
almost_empty_if	2	0	W1C	FIFO Almost Empty Interrupt Flag

Write 1 to clear.

Set to 1 by hardware when the FIFO reaches the almost empty level.

DACn_CTRL1_INT.underflow

Field	Bits	Default	Access	Description
underflow	3	0	R/O	FIFO Underflow Status

Read-only.

0: FIFO is not in underflow condition 1: FIFO is in underflow condition

DACn_CTRL1_INT.out_done_ie

Field	Bits	Default	Access	Description
out_done_ie	16	0	R/W	DAC Output Done Interrupt Enable

Write 1 to enable interrupt source; 0 to disable.

DACn_CTRL1_INT.underflow_ie

Field	Bits	Default	Access	Description
underflow_ie	17	0	R/W	FIFO Underflow Interrupt Enable

Write 1 to enable interrupt source; 0 to disable.

DACn_CTRL1_INT.almost_empty_ie

Field	Bits	Default	Access	Description
almost_empty_ie	18	0	R/W	FIFO Almost Empty Interrupt Enable

Write 1 to enable interrupt source; 0 to disable.

DACn_CTRL1_INT.ahb_cg_disable

Field	Bits	Default	Access	Description
ahb_cg_disable	28	0	R/W	Local AHB dynamic clock gate disable

Write 1 to disable clock gating; enabled by default (0)

DACn_CTRL1_INT.apb_cg_disable

Field	Bits	Default	Access	Description
apb_cg_disable	29	0	R/W	Local APB dynamic clock gate disable

Write 1 to disable clock gating; enabled by default (0)

8.4.5.1.4 DACn_FIFO_OUTPUT

Default	Access	Description
n/a	R/W	DAC Output FIFO

Write to push values to DAC output FIFO.

For 12-bit DAC instances, the value written to the FIFO should be 16 bits wide, with bits[15:4]=output value, bits[4:0]=0.

For 8-bit DAC instances, the value written to the FIFO should be 8 bits wide.

8.5 LED

8.5.1 LED Overview

The **MAX32600** provides the capability to drive multiple LEDs simultaneously with high output current drive and integrated closed loop current control. Depending on the application, one or more of the various LED configurations will be used. Each particular LED configuration requires specific organization of components both within and external to the chip. The simplest LED configurations require one DAC, one op amp, and one I/O pair on the **MAX32600**. External components—an LED and a sense resistor (Rs)—are also needed, necessitating the use of additional [analog matrix](#) routing resources as well.

Note See [Pin Layout](#) for a detailed mapping of **MAX32600** multiplexed function locations. Functional priority distinction is included in the mapping.

8.5.1.1 LED Driver Details

- Eight switches (CSAx ↔ CSBx)
- Up to two control control loops (one illustrated)
- Register and digital generator control of IO_cfg
- Digital Generators:
 - Pulse Train
 - Square Wave
 - Timers
 - PWM

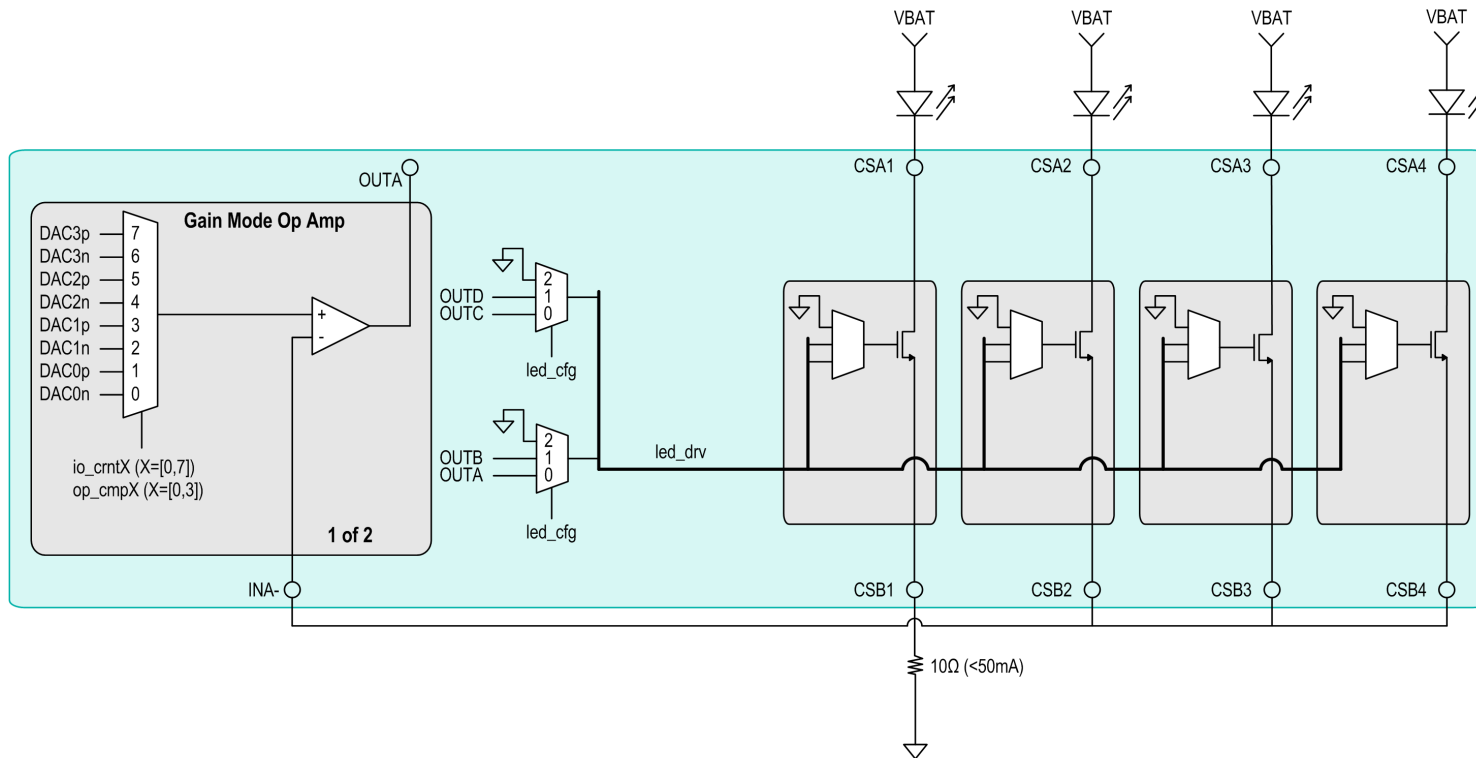


Figure 8.10: LED Block Diagram

8.5.2 LED Configuration

Note In each block diagram below, integrated components are rendered in black and external components are depicted in grey boxes. Also, in configurations with more than one LED, current is not necessarily run simultaneously; internal switches are used to manage current.

8.5.2.1 Basic LED Configuration

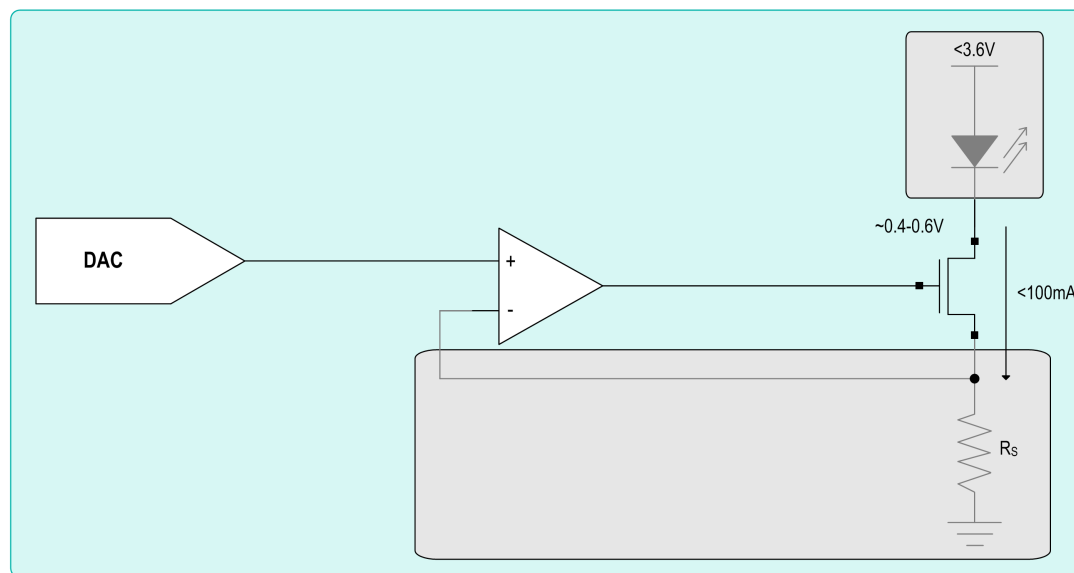


Figure 8.11: Basic LED Configuration

In the most basic LED configuration, an op amp is used to regulate the current (observed as a voltage on an external sense resistor) to match the voltage command from a DAC. The op amp drives the gate of an NMOS switch transistor in the I/O; the I/O regulates digitally whether the switch in the I/O is connected to the op amp or turned off. The **MAX32600** I/O switch can drive up to 100mA. This configuration requires two GPIO pins to implement the switch transistor and activate the LED.

The value of R_s is chosen to trade-off current resolution and available voltage for the LED. As the value of R_s is decreased, the resolution of the current selection is reduced and more voltage is available for the LED; as the value of R_s is increased, the resolution of the current selection is increased and less voltage is available for the LED.

8.5.2.2 Multiple LED Configuration

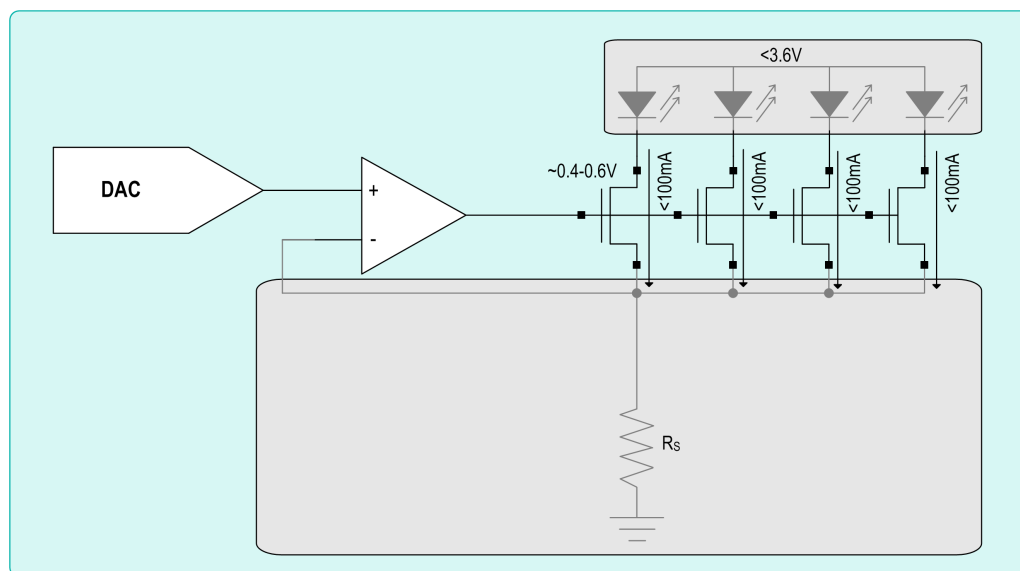


Figure 8.12: Multiple LED Configuration

The multiple LED configuration is an extension of the [Basic LED Configuration](#). The mechanics of the configuration remains the same; the only adaptation is multiple diodes on the control loop. To avoid unequal division of the current, only one diode should be activated at any one time.

Note When switching from all off to one diode on, overshoot on the control voltage must be prevented. The most reliable and effective way of accomplishing this is to switch the negative op amp input from the sense resistor to a voltage higher than the command from the DAC using uncommitted switches.

8.5.2.3 H-bridge LED Configuration

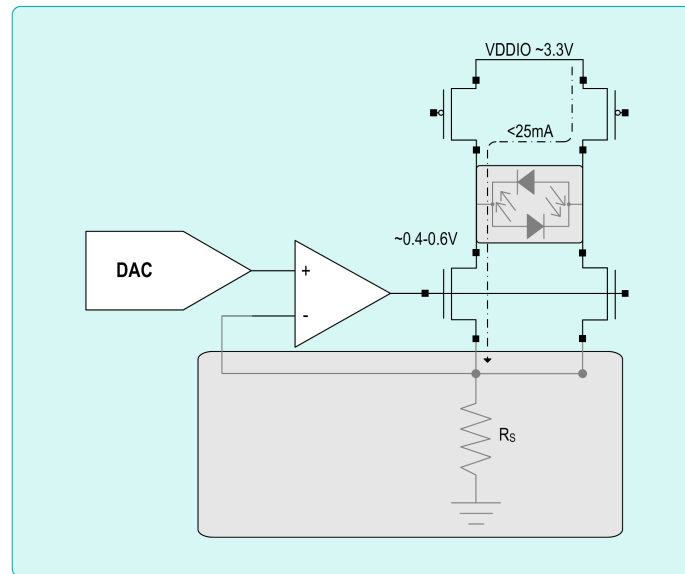


Figure 8.13: H-bridge LED Configuration

The H-bridge LED configuration uses a pair of source/sink LED I/O to drive a pair of diodes connected back-to-back. This design only requires two pins on the LED package and cable; however, it necessitates the use of a pullup device within the **MAX32600**. The onboard pullup devices on the **MAX32600** can usefully source 25mA from diodes with forward voltages as high as 2.5V. Note that capacity is a function of the sense resistor and the minimum available supply voltage. This source capability is not as high as the sink. In cases when more than 25mA is needed, multiple I/O pairs can be hooked up in parallel.

Note Overshoot on the control voltage must be prevented. The most reliable and effective way of accomplishing this is to switch the negative op amp input from the sense resistor to a voltage higher than the command from the DAC using uncommitted switches when enabling the loop.

8.5.2.4 Independent Loop H-bridge LED Configuration

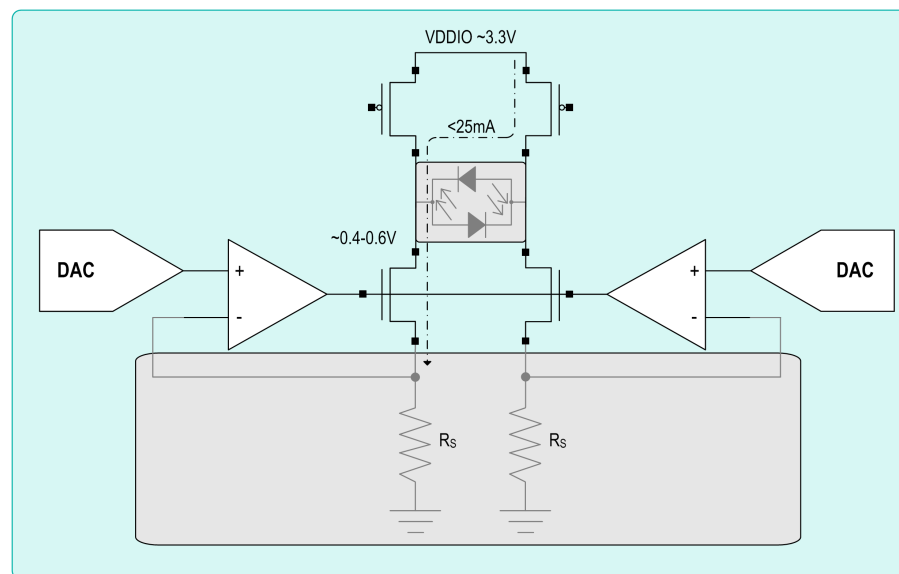


Figure 8.14: Independent Loop H-bridge LED Configuration

When the digital controls switch the loop open then closed again, undesirable loop behavior may occur and must be controlled. Further, only one diode can be activated at any one time. These circumstances may necessitate the use of more than one op amp. Here, the **MAX32600** can provide two independent control loop paths as a possible solution to these problems.

The independent loop H-bridge LED configuration allows two diodes to operate simultaneously on independent control loop paths. This avoids transient interference during operation and enables the loops to be on continuously. Each control loop can be configured with any of the diode drivers. However, only one control loop may be directly managed by pulse trains.

8.5.2.5 Integrated Feedback Loops LED Configuration

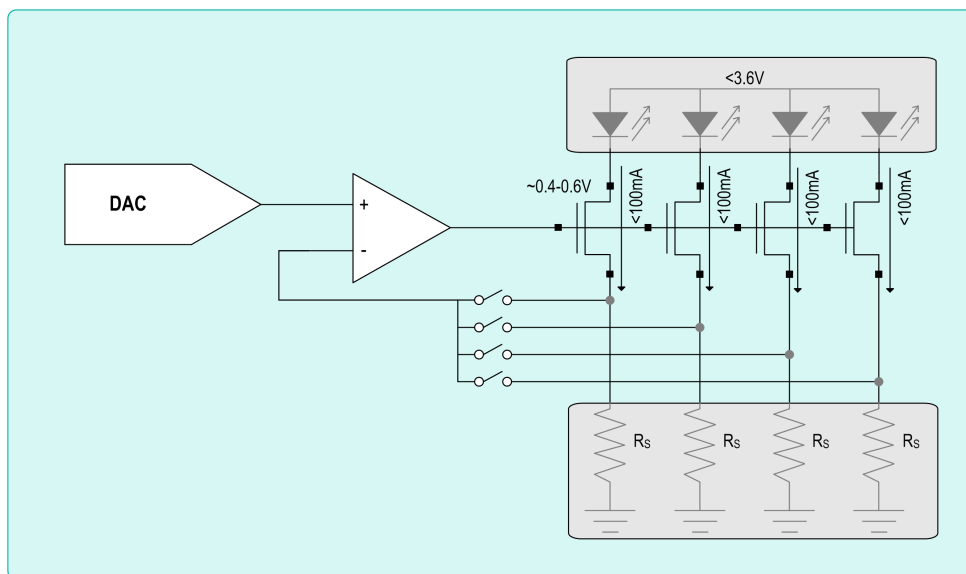


Figure 8.15: Integrated Feedback Loops LED Configuration

The **MAX32600** features the ability to integrate the feedback loop. This configuration enables time-division multiplexing of diodes with separate sense resistors and requires only one uncommitted switch for overshoot control instead of two. These control loops can time multiplex the DAC and op amp resources—creating multiple instances of the above [Basic LED Configuration](#) while using the same amount of on-chip **MAX32600** resources.

Note While any LED driver switch can connect to either of the op amp control lines, each switch can feedback to only one of the loops. Driver/feedback path connectivity is indicated in the [Current Mode Request](#) section below.

8.5.2.6 Internal Feedback with Common Sense Resistor LED Configuration

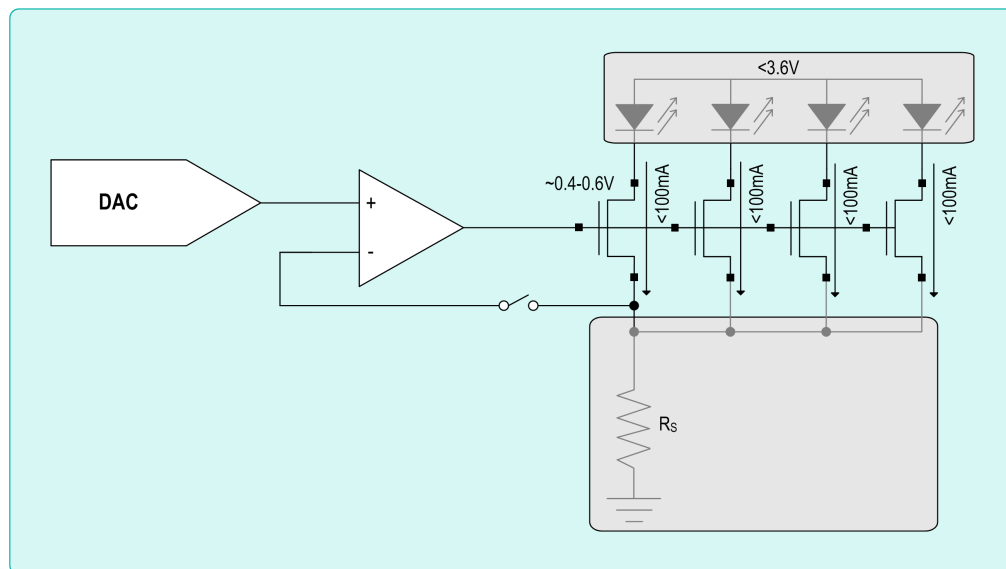


Figure 8.16: Internal Feedback with Common Sense Resistor LED Configuration

An alternate feedback configuration is to feed diodes on a control loop to a common sense resistor. Here, only one source pad needs to be connected back to the op amp—the one pad is common to every diode connected in the control loop. This configuration enables the use of diode drivers that otherwise could not be used with the control loop by themselves; a single sufficient diode driver present on the control loop pad is all that is needed.

8.5.2.7 Double H-bridge LED Configuration

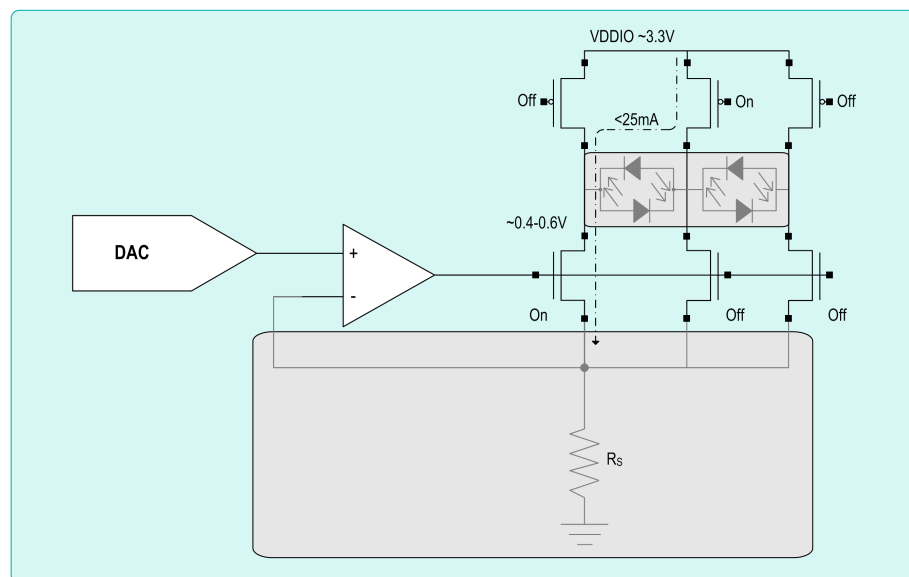


Figure 8.17: Double H-bridge LED Configuration

The H-bridge application can be doubled: one terminal of a back-to-back diode pair shares a terminal of the second diode pair. This extension of the [H-bridge LED Configuration](#) requires three wires and three I/O pairs. This is illustrated in the example [figure](#). Here, the middle PMOS and NMOS are used twice as often as the right and left ones. Alternate applications of this configuration include six diodes with four wires and eight diodes with five wires.

8.5.2.8 Multiple High Voltage LED Control Configuration

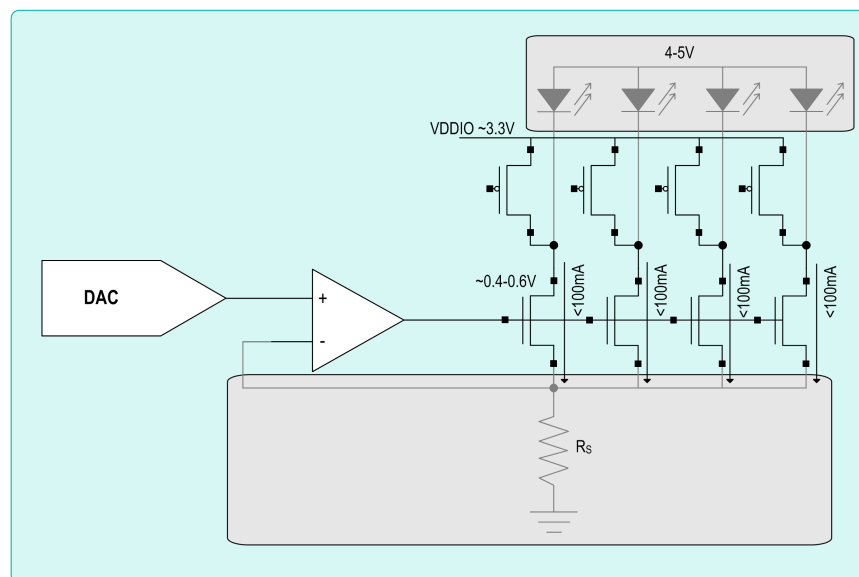


Figure 8.18: Multiple High Voltage LED Control Configuration

This configuration provides for the high drive voltage requirements inherent to some LEDs. LEDs, particularly blue and green, often have high forward voltages; even red LEDs have high drive voltage requirements at high currents due to internal resistances. In these cases, the **MAX32600** 3.6V maximum voltage becomes a limiting factor. However, it is possible to use up to a 5V anode voltage on the LEDs in the multiple LED configuration shown above if the cathode is driven to V_{DDIO} to turn the diode off. When off, the diode still has around 1V on forward bias; however, if that voltage is still on the exponential portion of the diode I-V curve, then the off current is vanishingly small.

Note The anode voltage and the main chip supply should be optimized for the particular diode to maintain desired on voltage and off current.

8.5.2.9 3x3 LED Matrix Configuration

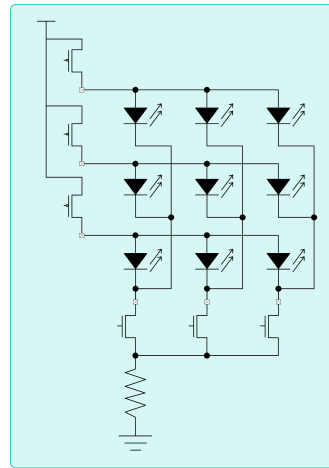


Figure 8.19: 3x3 LED Matrix Configuration

In LED matrix configuration, LED capable I/Os are used at the bottom of the matrix structure, and simple I/Os for the top sides. For proper operation, the top and bottom I/Os must be switched in a time multiplexed manner.

The simplest method of control enables only one LED at a time, but always retains current in the sense resistor. The LEDs are controlled by a combination of the top and bottom drivers and lit in sequence.

The LED capable drivers run at the refresh rate with a 1/3 duty cycle. The top drivers run at three times faster than the refresh frequency with a 1/3 duty cycle. The average brightness is 1/9 the on brightness—as the current is time multiplexed into the nine LEDs. The low average current means that either the peak current needs to be brighter (in this case the bottom drivers run at the refresh rate with a 1/3 duty cycle, and the top drivers must be able to support this) or the low brightness is acceptable.

LED brightness can be addressed in a number of ways. Multiple LEDs could be enabled simultaneously, thus extending the duty cycle. This encounters difficulty, however, because the diodes will tend to split the current unevenly, leading to brightness variations. An alternate way to approach LED brightness is to keep the anode common to all diodes on simultaneously and switching the cathode. This option delivers favorable results because the NMOS switch device operates in saturation and has good output conductance. It must be considered, though, that the pullup device is substantially weaker than the pulldown, so the sum of the currents into the row is limited to $\sim 25\text{mA}$ unless the pullups are operated in parallel. Brightness of individual diodes can be adjusted by fine-tuning the duty cycle.

8.5.2.9.1 LED Matrix Limitations

The matrix system limitations depend in part on the diode; a high peak current rating is required. There are eight available pulldowns and 48 available pullups (if no I/O is reserved for other purposes), yielding a theoretical maximum array size of 384 LEDs. This maximum array, however, would be very dim under any type of control; arrays would have to be significantly smaller to have practical usage.

Typically, small LED module sizes are in multiples of 5x7 pixels and have high peak currents per diode to enable low duty cycle operation. Here, although a 5x7 array only requires five pulldowns and seven pullups for operation, doubling the pullups is recommended for optimal performance. A total of five pulldown pairs and 14 pullups are then needed.

8.5.3 Register Configurations

To configure the **MAX32600** for these [various applications](#), several sets of registers must be written:

- Request current mode usage from I/O mux
- Configure the control loop in the AFE
- If needed, configure any comparators used for fault detection
- If needed, configure TIAs and the ADC for measurement
- Set up the pulse trains or timers

8.5.3.1 Current Mode Request

Note See [Pin Layout](#) for a detailed mapping of **MAX32600** multiplexed function locations. Functional priority distinction is included in the mapping.

Mapping of Logical Current Ports (7mm x 7mm Compact Package)

Sink Port	Switch Drain	Switch Source	Drive	Feedback
CS0	P0.0	P0.1	2X	0
CS1	P0.2	P0.3	2X	1
CS2	P0.4	P0.5	4X	0
CS3	P0.6	P0.7	3X	1

Note The mapping of the WLP Package is equivalent to the 7mm x 7mm Compact package; however, all LED drivers are 2X on the WLP (replace drive figures in [above table](#)).

Mapping of Logical Current Ports (12mm x 12mm Standard Package)

Sink Port	Switch Drain	Switch Source	Drive	Feedback
CS0	P6.0	P6.1	1X	0
CS1	P6.2	P6.3	1X	0
CS2	P6.4	P6.5	1X	1
CS3	P6.6	P6.7	1X	1
CS4	P7.0	P7.1	1X	0
CS5	P7.2	P7.3	1X	0
CS6	P7.4	P7.5	1X	0
CS7	P7.6	P7.7	1X	0

The switch drain and source columns indicate which physical balls are involved in a particular current source function. The feedback column indicates which feedback channel is used to connect the I/O back to the AFE, which subsequently affects which op amps and comparators can be used with that I/O.

To request particular I/O, write to [IOMAN_CRNT_REQ](#) register; each bit corresponds to one sink I/O pair. [IOMAN_CRNT_ACK](#) contains a status flag, and if all requests were granted, should be equal to [IOMAN_CRNT_REQ](#).

Several different sub-modes are available; these modes define the behavior of the 0 and 1 states of the I/O. Each I/O port has a register, [GPIO_OUT_VAL_P\[0:7\]](#) which controls whether the I/O is in the 0 or 1 state. The following table maps the [IOMAN_CRNT_MODE](#) against the two values of the [GPIO_OUT_VAL_Pn](#) register.

LED Reconfiguration Modes

IOMAN_CRNT_MODE	GPIO_OUT_VAL			
	DRAIN		SOURCE	
	LOGIC 1	LOGIC 0	LOGIC 1	LOGIC 0
0x00	N-gate=Op amp A/B	N-gate=GND	N-gate=GND	
0x01	N-gate=Op amp C/D		N-gate=GND	
0x02	N-gate=Op amp A/B		N-gate=GND	

IOMAN_CRNT_MODE	GPIO_OUT_VAL			
	DRAIN		SOURCE	
	LOGIC 1	LOGIC 0	LOGIC 1	LOGIC 0
0x03	N-gate=Op amp C/D		Feedback channel=Source	
0x04	Reserved			
0x05				
0x06				
0x07	N-gate=GND P-gate=On	N-gate=Op amp C/D P-gate=Off	N-gate=GND Feedback channel=Source	
0x08	N-gate=Op amp A/B Feedback channel=drain	N-gate=GND	N-gate=GND	N-gate=GND Feedback channel=Source
0x09	N-gate=Op amp A/B Feedback channel=drain			
0x0A	Reserved			
0x0B				
0x0C				
0x0D				
0x0E				
0x0F				

Note IOMAN_CRNT_MODE 0x7 is not available in the standard (12mm x 12mm) package.

The feedback channel indicates which pad is connected to the I/O feedback channel. Which channel is connected depends on which I/O is used (the details are in [Compact Package](#) and [Standard Package](#) tables). Where not specified, the PMOS source is off and the feedback connection is floating.

To operate the LED I/O in source mode, such as for H-bridge pullups or high voltage applications, the user must switch to digital high, with [GPIO_OUT_MODE_Pn](#) 0x9 (H-bridge) or 0x0 (high voltage). To switch between the modes, write [IOMAN_CRNT_MODE](#) as above for the on state and [GPIO_OUT_MODE_Pn](#) for the off state; use [IOMAN_CRNT_REQ](#) to switch between on and off.

For the compact packages (7mm x 7mm and WLP packages) switching between sink and source is more simple. Set the [IOMAN_CRNT_MODE](#) to 0x7 and use [GPIO_OUT_VAL_Pn](#) to switch between sink and source. Since [GPIO_OUT_VAL_Pn](#) can be controlled by a pulse train, this off loads the PMU and ARM processors and provides deterministic timing. *NOTE:* As is is available only in mode 0x7, op amps B or D must be used. In the WLP package, op amp D is not available externally, so it is the ideal candidate to manage the LED control loops.

8.5.4 Control Loop Setup

An op amp from the AFE must be used to achieve a regulated current in Current Mode. As depicted in the [Basic LED Configuration](#), the source pad is fed back to the op amp's negative input, and the positive input is the command voltage (coming from a DAC in this case). An alternative option is to use the uncommitted switches and ground switches to dynamically switch the command voltage. Additionally, the command voltage can originate from an external source using the positive input pad of the op amp.

First, the connection from the op amp to the appropriate current sink must be established. Modify the [AFE_CTRL0.led_cfg_port0](#) and [AFE_CTRL0.led_cfg_port1](#) fields, as shown below:

AFE_CTRL0.led_cfg	State	Channel 0	Channel 1
1:0	0x00	Op amp A	N/A
	0x01	Op amp B	N/A
	0x02, 0x03	Disabled	N/A
3:2	0x00	N/A	Op amp C
	0x01	N/A	Op amp C
	0x02, 0x03	N/A	Disabled

Note If a control voltage needs to be driven from outside the part, the [AFE_CTRL0.led_cfg_port0](#) and [AFE_CTRL0.led_cfg_port1](#) fields still need to select an op amp, but that op amp must be powered down.

In addition to the LED configuration settings, the following components must typically be set up:

- Op amp power mode and source selection
- DAC power and data value selection
- Reference voltage selection

8.5.5 Fault Detection Setup

Fault detection is accomplished by monitoring the source pad of the sink, which is connected to the analog matrix, during operation by a pair of comparators, each with a reference level set by a DAC. The [example below](#) depicts a setup with switch matrix registers configured. In this arrangement, CompA is used as the undershoot or open-circuit detector; it will trip when the feedback node is less than DAC2. CompB is used as the overshoot or short-circuit detector; it will trip when the feedback node is greater than DAC3.

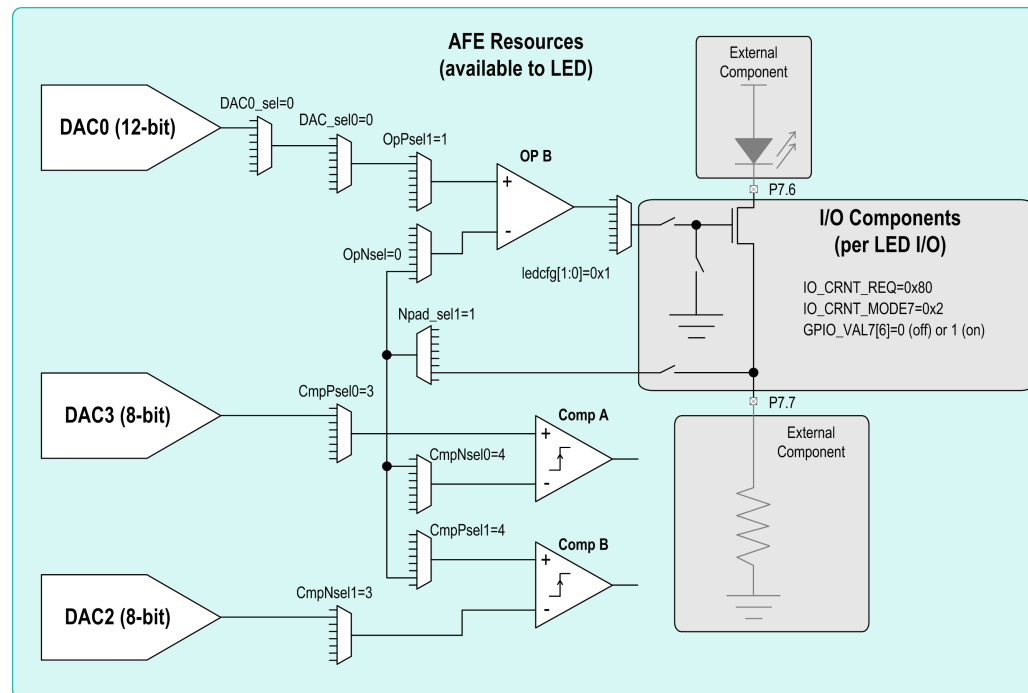


Figure 8.20: Fault Detection Diagram

8.5.6 Timing Configuration

The driver state can be controlled directly by APB writes, pulse trains, or timers. Any port can be controlled by APB writes, any port can access timers 0-3, but the specific pulse trains available for use vary by driver:

Pulse Trains by Current Sink

I/O	PT0	PT1	PT2	PT3	PT4	PT5	PT6	PT7
CS0	X				X			
CS1		X	X			X		
CS2			X		X		X	
CS3				X			X	X
CS4	X				X			
CS5		X	X			X		
CS6			X		X		X	
CS7				X			X	X

The advantages of timer or pulse train control are synchronous control with minimum jitter and simultaneous switch of multiple I/O. Though, pulse trains have limited duty cycle with a maximum length of 32-bits and a minimum of at least 3.125% duty cycle, and timers only operate in a clock-like fashion.

8.5.7 Considerations

8.5.7.1 Design Current

The design process for a precision current LED system involves balancing current resolution, supply voltage, and the operating forward voltage of the LED involved to take advantage of the headroom available.

LED selection is usually determined by the desired wavelength (i.e., color output) and brightness; other components are adapted to fit the necessary requirements.

In many applications, the LED uses the majority of the supply voltage, and the requirement will be to minimize the headroom of the NMOS switch and sense resistor. In the case of an H-bridge configuration, the headroom of the PMOS switch will need to be factored in as well.

Reducing the sense resistor directly results in loss of current resolution and introduces limitations in setting the current. This must be considered as precision is important in certain applications. Also, loss of resolution makes the current loop more susceptible to ground drop. For a 50mA output, 9-bits of resolution yields step sizes of 100 μ A. With the 12-bit DAC as the driver, this means that the sense voltage needs to be at least 12.5% of V_{REFDAC} ; for a 2.5V reference voltage, the sense voltage must be at least 312.5mV.

Ground drop remains a concern and is a key factor of accuracy. To maintain accuracy at the 0.1% level of a precision resistor and 9-bits on the DAC, the ground drop should be less than 0.1% of sense voltage. Also, with currents at 100mA, parasitic resistance can be significant. Typical sense resistors that translate to 0.3-0.6mV

or parasitic resistances on the order of 3-6mOhms keep the effects of parasitic resistance minimized.

8.5.7.2 Average and Peak Current

The absolute maximum steady state current allowed through a single switch is 135mA at 85°C. When multiple switches are arranged in parallel, however, the split of the current between the parallel devices is dependent on threshold and metal resistance differences. Although the division of current between multiple I/Os is constant for a particular part, it is not a controlled or specified parameter.

It is recommended that parts with a high duty cycle and multiple I/O connected in parallel do not exceed an average current of 200mA. Transient currents will be limited by the switch I_{ds} to a point that is safe for the **MAX32600**. Transient currents may damage external components or possibly have safety implications—it is required that transient behavior be controlled within acceptable limits. There are several methods for ensuring that the output current is not out of bounds:

- Use of the comparators for fault detection is recommended; this enables the detection of short-circuit or over-current conditions.
- Powering up or powering down of the loop must be handled correctly.
- When the feedback loop is broken, the negative op amp can be switched above the DAC set-point using one of the uncommitted switches. A small amount of delay is acceptable between breaking the feedback loop and raising the negative voltage is acceptable because the op amp slew rate is limited to several microseconds for rail-to-rail slew.

9 Pulse Train Engine

9.1 Pulse Train Engine (PTE) Overview

The **MAX32600** includes 13 separate pulse train generators. Eight of these (PT0 - PT7) can be used to generate output sequences on [General-Purpose I/O](#) pins. The remaining five are routed to the Analog Front End (AFE). For details on how the pulse trains (PT8 - PT11 and PT15) are used by the AFE, refer to [AFE Overview](#).

Each pulse train generator can be set to output either a square wave or a repeating pattern from 2- to 32-bits in length. The frequency of each enabled pulse train generator is also set separately, based on a divide down (divide by 2, divide by 4, divide by 8, etc.) of the Pulse Train Peripheral Clock.

Any single pulse train generator or any desired group of pulse train generators can be restarted at the beginning of their patterns and synchronized with one another enabling each synchronized pulse train generator to start simultaneously.

9.2 Output Mode Selection

Two modes of operation are supported: [Pulse Train](#) and [Square Wave](#). Both modes use a rate counter that defines the number of system clock cycles that occur before the output state is changed. In both modes, setting the `PTG_CTRL.enable_all` to 0 disables all active pulse trains. See [Enabling and Disabling Pulse Train Outputs](#) for further information.

9.3 Pulse Train Peripheral Clock Rate Configuration

The **MAX32600** supports a programmable Pulse Train Peripheral Clock Rate (PTE_{PCLK}). Each of the 13 pulse trains use the Peripheral Clock as the source to set their individual rate control as described in the [Pulse Train Engine Modes](#) section. To select the peripheral clock rate, write the Pulse Train Clock Control register, `pulse_train_clk_scale`, with the value desired to achieve the ideal clock for your application. The rate is a divisor (4-bits) of the system clock as shown in the [table](#) below.

Pulse Train Peripheral Clock Selection Table

<code>pulse_train_clk_scale</code> (4-bit value)	PT Peripheral Clock Rate (PTE_{PCLK})
0	Disables Pulse Train Peripheral Clock
1	24MHz
2	12MHz
3	6MHz
4	3MHz

pulse_train_clk_scale (4-bit value)	PT Peripheral Clock Rate (PTE _{PCLK})
5	1.5MHz
6	750kHz
7	375kHz
8	187.5kHz
9	93.75kHz
10-15	24MHz

9.4 Enabling and Disabling Pulse Train Outputs

9.4.1 Master Enable/Disable

The Pulse Train Engine supports the ability to start all configured and stop all active pulse trains simultaneously. This enables master control of all pulse train outputs by firmware.

To disable all active pulse trains set the [PTG_CTRL.enable_all](#) to 0. Setting this bit to 1 will start all configured pulse trains simultaneously.

9.4.2 Enabling and Disabling Individual Pulse Train Outputs

For each individual pulse train output, writing a 0 to the [PTn_RATE_LENGTH.mode](#) register disables the specific pulse train output. Details for enabling specific pulse train modes are described [below](#).

9.5 Pulse Train Engine Modes

9.5.1 Pulse Train Mode

In Pulse Train mode, the engine supports lengths between 2- and 32-bits as set in the [PTn_RATE_LENGTH.mode](#) register. After setting the length, the [PTn_TRAIN](#) register defines the sequence of 1s and 0s that will be shifted out (LSB first). To define the rate at which the data will be shifted out, the [PTn_RATE_LENGTH.H.rate_control](#) register is written with a value representing the number of Peripheral Clocks between each bit of data sent. When the pulse sequence defined in the [PTn_TRAIN](#) register has been completely sent out, the process repeats.

Note Setting the [PTn_RATE_LENGTH.mode](#) to 1 results in enabling [Square Wave Mode](#).

Table 9.1: Pulse Train Output Options

	CS0	CS1	CS2	CS3	CS4	CS5	CS6	CS7	SW0	SW1	SW2	SW3	ADC
PT0	X				X								
PT1		X				X							
PT2		X	X			X	X						
PT3				X				X					
PT4	X		X		X		X						
PT5		X				X							
PT6			X	X			X	X					
PT7				X				X					
PT8									X				
PT9										X			
PT10											X		
PT11												X	
PT15													X

9.5.2 Square Wave Mode

In Square Wave mode, the PTE simply toggles the output state when the rate counter, as defined in the `PTn_RATE_LENGTH.rate_control` field, expires. In Square Wave mode, if the rate is set to a value of 1, the Pulse Train Peripheral clock rate is output directly.

Square Wave Mode Rate Control Table

rate_control (27-bit value)	Frequency Out
1	(PTE _{PCLK})
2	(PTE _{PCLK} /2)
3	(PTE _{PCLK} /4)
4	(PTE _{PCLK} /6)
5	(PTE _{PCLK} /8)

In general, for Rates > 1 the output frequency is determined by the equation: $f_{out} = \frac{(PTE_{PCLK})}{(2 \times (rate_control - 1))}$

Where PTE_{PCLK} is the Pulse Train Peripheral Clock and rate_control is a 27-bit value for the divisor set using the `PTn_RATE_LENGTH.rate_control` register field.

9.6 Synchronization

The Pulse Train Engine supports the synchronization of multiple Pulse Train outputs. To synchronize the outputs of one or more Pulse Trains, set the corresponding bit(s) in the `PTG_RESYNC` register. Writing to this register resets the output sequence of all affected pulse trains to the beginning of the pattern defined in the `PTn_TRAIN` register (with the LSB being sent out first), and also resets the output state for any affected pulse trains which are in square wave mode.

Note The Pulse Train Engine configuration registers may be modified at any time by the user. If the PTE is active when these registers are modified, the output state is immediately affected and might result in undesired or unintended effects on the output.

9.7 Registers (PT)

9.7.1 Module PT Registers

Address	Register	32b Word Len	Description
0x40001000	PTG_CTRL	1	Global Pulse Train Enable/Disable
0x40001004	PTG_RESYNC	1	Global Resync (All Pulse Trains) Control
0x40001008	PT0_RATE_LENGTH	1	Pulse Train Configuration
0x4000100C	PT0_TRAIN	1	Pulse Train Output Pattern
0x40001010	PT1_RATE_LENGTH	1	Pulse Train 1 Configuration
0x40001014	PT1_TRAIN	1	Pulse Train 1 Output Pattern
0x40001018	PT2_RATE_LENGTH	1	Pulse Train 2 Configuration
0x4000101C	PT2_TRAIN	1	Pulse Train 2 Output Pattern
0x40001020	PT3_RATE_LENGTH	1	Pulse Train 3 Configuration
0x40001024	PT3_TRAIN	1	Pulse Train 3 Output Pattern
0x40001028	PT4_RATE_LENGTH	1	Pulse Train 4 Configuration
0x4000102C	PT4_TRAIN	1	Pulse Train 4 Output Pattern
0x40001030	PT5_RATE_LENGTH	1	Pulse Train 5 Configuration
0x40001034	PT5_TRAIN	1	Pulse Train 5 Output Pattern
0x40001038	PT6_RATE_LENGTH	1	Pulse Train 6 Configuration
0x4000103C	PT6_TRAIN	1	Pulse Train 6 Output Pattern
0x40001040	PT7_RATE_LENGTH	1	Pulse Train 7 Configuration
0x40001044	PT7_TRAIN	1	Pulse Train 7 Output Pattern
0x40001048	PT8_RATE_LENGTH	1	Pulse Train 8 Configuration
0x4000104C	PT8_TRAIN	1	Pulse Train 8 Output Pattern
0x40001050	PT9_RATE_LENGTH	1	Pulse Train 9 Configuration
0x40001054	PT9_TRAIN	1	Pulse Train 9 Output Pattern
0x40001058	PT10_RATE_LENGTH	1	Pulse Train 10 Configuration
0x4000105C	PT10_TRAIN	1	Pulse Train 10 Output Pattern

Address	Register	32b Word Len	Description
0x40001060	PT11_RATE_LENGTH	1	Pulse Train 11 Configuration
0x40001064	PT11_TRAIN	1	Pulse Train 11 Output Pattern
0x40001068	PT12_RATE_LENGTH	1	Pulse Train 12 Configuration
0x4000106C	PT12_TRAIN	1	Pulse Train 12 Output Pattern
0x40001070	PT13_RATE_LENGTH	1	Pulse Train 13 Configuration
0x40001074	PT13_TRAIN	1	Pulse Train 13 Output Pattern
0x40001078	PT14_RATE_LENGTH	1	Pulse Train 14 Configuration
0x4000107C	PT14_TRAIN	1	Pulse Train 14 Output Pattern
0x40001080	PT15_RATE_LENGTH	1	Pulse Train 15 Configuration
0x40001084	PT15_TRAIN	1	Pulse Train 15 Output Pattern

9.7.1.1 PTG_CTRL

PTG_CTRL.enable_all

Field	Bits	Default	Access	Description
enable_all	1	1	R/W	Enable/disable for all pulse trains

Setting this bit to 0 disables all pulse trains.

9.7.1.2 PTG_RESYNC

PTG_RESYNC.[pt0, pt1, pt2, pt3, pt4, pt5, pt6, pt7]

Field	Bits	Default	Access	Description
pt0	0	0	W/O	Resync control for PT0

Field	Bits	Default	Access	Description
pt1	1	0	W/O	Resync control for PT1
pt2	2	0	W/O	Resync control for PT2
pt3	3	0	W/O	Resync control for PT3
pt4	4	0	W/O	Resync control for PT4
pt5	5	0	W/O	Resync control for PT5
pt6	6	0	W/O	Resync control for PT6
pt7	7	0	W/O	Resync control for PT7

Write 1: Resets the rate counter and pulse train pointer for this pulse train. When in square wave mode, the output mode is also reset to 0.

Self-clearing; always reads 0.

PTG_RESYNC.[pt8, pt9, pt10, pt11, pt12, pt13, pt14, pt15]

Field	Bits	Default	Access	Description
pt8	8	0	W/O	Resync control for PT8
pt9	9	0	W/O	Resync control for PT9
pt10	10	0	W/O	Resync control for PT10
pt11	11	0	W/O	Resync control for PT11
pt12	12	0	W/O	Resync control for PT12
pt13	13	0	W/O	Resync control for PT13
pt14	14	0	W/O	Resync control for PT14
pt15	15	0	W/O	Resync control for PT15

Write 1: Resets the rate counter and pulse train pointer for this pulse train. When in square wave mode, the output mode is also reset to 0.

Self-clearing; always reads 0.

9.7.1.3 PTn_RATE_LENGTH

PTn_RATE_LENGTH.rate_control

Field	Bits	Default	Access	Description
rate_control	26:0	26'b0	R/W	Pulse Train Enable/Rate Control

Defines rate at which the pulse train output changes state. If this field is zero, the pulse train is disabled; otherwise, the output changes to the next state (toggling for square wave mode, or advancing to the next pulse train pattern bit for pulse train output mode) at a rate equal to (Pulse Train Module Clock / Pulse Train Rate).

PTn_RATE_LENGTH.mode

Field	Bits	Default	Access	Description
mode	31:27	00001b	R/W	Pulse Train Output Mode/Train Length

Sets either square wave mode or pulse train mode; for pulse train mode, defines pulse train length.

- 00000b: Pulse train, 32 bits
- 00001b: Square Wave Mode (PTx_TRAIN not used)
- 00010b: Pulse train, 2 bits
- 00011b: Pulse train, 3 bits
- ...
- 11111b: Pulse train, 31 bits

9.7.1.4 PTn_TRAIN

Default	Access	Description
00000000h	R/W	Pulse Train Output Pattern

In square wave mode, this register has no effect. In pulse train mode, this register contains the repeating pattern that will be shifted out as the pulse train output stream (starting with LSB)

10 System Clock, Timers/Counters, Watchdog Timers and Real Time Clock

10.1 System Clock

10.1.1 System Clocks Overview

Note All external clock sources must meet the electrical/timing requirements given in the datasheet.

All functional units in the **MAX32600** are synchronized to the system clock that can be generated from either an external oscillator or internal ring oscillator or with

an external crystal/resonator. A block diagram of the **MAX32600** clock subsystem is provided [below](#).

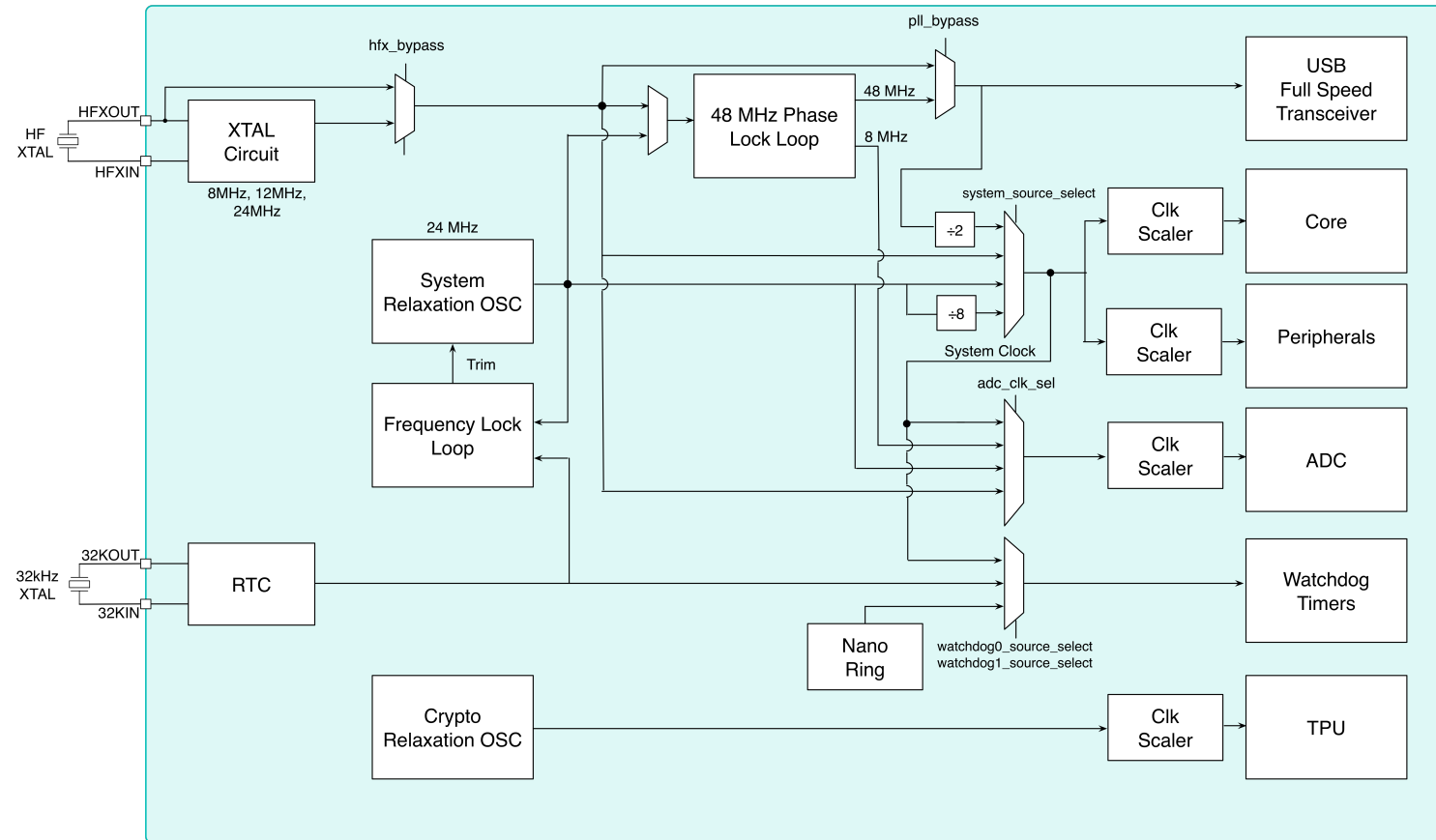


Figure 10.1: Clock System Block Diagram

10.1.1.1 External High-Frequency Crystal Oscillator

The **MAX32600** includes a high-frequency crystal oscillator circuit designed to operate with an external crystal. Fundamental mode crystals can be used with the oscillator circuit up to a frequency of 24MHz. External load capacitors are required depending on the crystal specifications.

An external clock source may also be used by the **MAX32600** in place of a high-frequency crystal. For this configuration, the external clock source is connected to the part on the HFXIN pin, and the HFXOUT pin is left floating.

10.1.1.2 32kHz Crystal Oscillator

A 32kHz crystal oscillator (with the 32kHz crystal connected between the 32KIN and 32KOUT pins) is used to generate the 32kHz clock that is used by the [Real Time Clock \(RTC\)](#) module.

An external clock source may also be used by the **MAX32600** in place of a 32kHz crystal. For this configuration, the external clock source is connected to the part on the 32KIN pin.

10.1.1.3 48MHz USB Clock PLL

The phase locked loop (PLL) clock generation circuit is used to generate a 48MHz clock which is required for proper operation of the [USB Device Interface](#). If the USB interface will not be used, use of the PLL is optional.

The PLL generates a 48MHz clock using a clock multiplier circuit of either 2X, 4X, or 6X. The PLL supports 2X mode for use with a 24MHz external crystal, 4X mode for use with a 12MHz external crystal, and 6X mode for use with an 8MHz external crystal. Additionally, the PLL output can be used as a system clock source (after dividing it by two) with a 24MHz frequency.

10.1.1.4 48MHz Internal 24MHz Trimmed Relaxation Oscillator

An internal trimmed relaxation oscillator generates a 24MHz ($\pm 1\%$) clock that can be used as a system clock source. If calibrated to the 32kHz crystal, this can be used as the input to the PLL to generate the USB clock.

10.1.1.5 Cryptographic Internal Oscillator

The **MAX32600** provides a dedicated on-chip oscillator used to supply a separate isolated clock to reduce opportunities of clock interference attacks. Specifically, timing-based analysis and fault injection attacks on sensitive functions of the device are mitigated with this feature. Vulnerabilities protected by the isolated cryptographic internal oscillator include timing-based and security functions such as the [AES Cryptographic Engine](#) and the [Modular Arithmetic Accelerator \(MAA\)](#) among others.

Effectively decoupling these sensitive functions from the main system clock allows encryption and decryption operations to take a consistent amount of time regardless of the current system clock rate. Also, this provides a more variable (and not externally observable) clock to reduce the opportunity for an attacker to perform power or timing analysis attacks against the cryptographic and security functions.

10.1.2 System Clock Configuration

The **MAX32600** clock subsystem provides a high degree of flexibility to optimize an embedded system implementation. The user has options to trade-off external components depending on the circuitry required by the desired application(s). A block diagram of the **MAX32600** clock subsystem is provided in the [figure](#) above.

10.1.3 System Clock Sources

The internal clock circuitry generates the system clock from one of the three clock sources:

1. External Clock
2. Internal Oscillator with an external crystal or resonator
3. Internal Relaxation Oscillator

The register controls for the clocks are in the Clock Manager ([CLKMAN Registers](#)). There, clock sources can be selected and/or gated off and clock scalars can be selected. In general, most clocks are disabled by default to minimize total system power consumption. Refer to the Clock Manager Register Map for details.

The external clock and crystal are mutually exclusive since they are input via the same clock pin. The clock subsystem is configured through the [CLKMAN_CLK_CTRL](#) register; the [primary clock source selection](#) is made using the [CLKMAN_CLK_CTRL.system_source_select](#) register field.

Field Setting	Primary Clock Source Selection
b00	24Mhz Relaxation Oscillator divided by eight
b01	24Mhz Relaxation Oscillator
b10	High Frequency Crystal Oscillator
b11	48Mhz PLL divided by two

The Relaxation Oscillator accuracy is sufficient to run the core on-chip digital logic and all external peripherals. For [Real Time Clock \(RTC\)](#) functionality, a 32kHz XTAL/Resonator is required. For the USB peripheral, a frequency-accurate clock is required. This functionality can be provided by an external high-frequency crystal oscillator or, if a 32kHz XTAL/Resonator is available, the Relaxation Oscillator can be used after firmware frequency calibration is performed.

An external high-frequency crystal (HFXTAL) may be required to achieve the highest precision ADC measurements. The most accurate ADC performance is obtained when an 8MHz HFXTAL is used as the system clock source. However, in low sample rate ADC applications, the internal Relaxation Oscillator may be adequate.

10.1.3.1 External Clock

The **MAX32600** CPU can obtain the system clock signal directly from an external clock source. In this configuration, the clock generation circuitry is driven directly by an external clock.

To operate the **MAX32600** from an external clock, connect the clock source to the HFXIN pin and enable the HFX Bypass function by setting [CLKMAN_CLK_CONFIG.hfx_bypass](#) bit to 1. Here, the oscillator is disabled and the bypass mode is enabled. The clock source should be driven through a complementary metal-oxide semiconductor (CMOS) driver.

10.1.3.2 External High Frequency Clock (Crystal or Resonator)

An external quartz crystal or resonator can be connected from high-frequency IN to high-frequency OUT (HFXIN to HFXOUT) as the mechanism determining frequency as illustrated in the [figure](#) below. The fundamental mode of the on-chip high-frequency crystal oscillator operates as an inductive reactance in parallel resonance with an external capacitance to the crystal.

Crystal specifications, operating temperature, operating voltage, and parasitic capacitance must be considered when designing or choosing the external oscillator. The **MAX32600** is designed to operate at a maximum frequency of 24MHz; however, the oscillator is not limited to this frequency as the PLL 2X, 4X, and 6X modes are regularly used to achieve higher frequencies. An 8MHz crystal is suggested for lower power and minimal ADC aperture jitter. To further reduce the effects of external noise, place a guard ring around the external oscillator circuitry.

The HFXIN and HFXOUT pins are protected against on-chip electrostatic discharge by clamping devices. These clamping devices are diodes, parasitic to the feedback resistor (Rf) in the inverter circuit of the oscillator. The inverter circuit is presented as a NAND gate which can disable clock generation.

Noise at HFXIN and HFXOUT can adversely affect on-chip clock timing. It is good design practice to place the crystal and capacitors as close to the oscillator circuitry as possible and provide direct traces to the HFXIN, HFXOUT, and ground. The typical values of external capacitors vary with the type of crystal used and should be selected based on the load capacitance as suggested by the crystal manufacturer.

For cost-sensitive applications, a ceramic resonator can be used instead of a crystal. Using the ceramic resonator may require a different circuit configuration and

capacitance value.

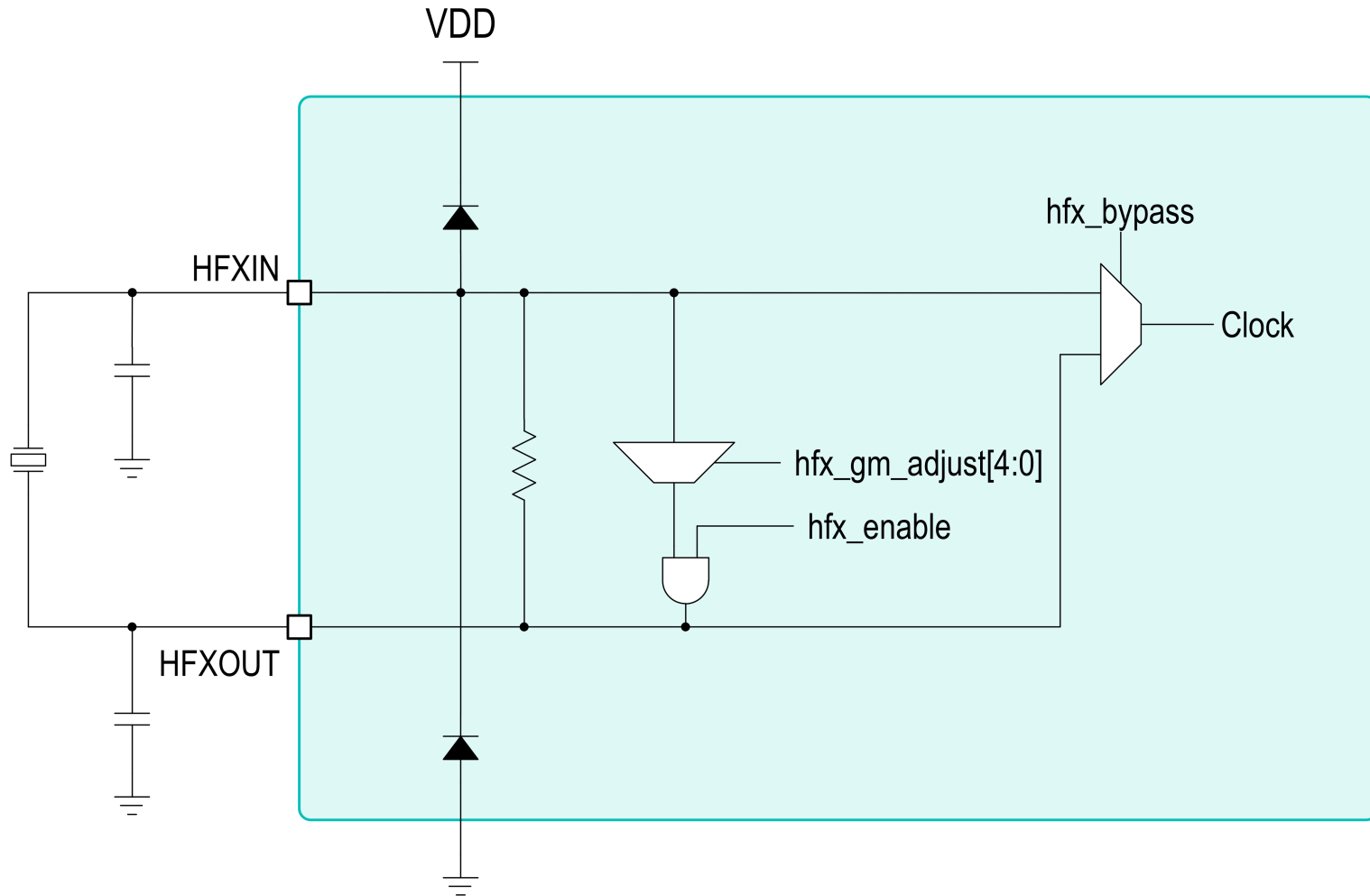


Figure 10.2: On-Chip High-Frequency Crystal Oscillator

Firmware Control of External High Frequency Clock (Crystal or Resonator)

The external clock is controlled by three fields in the [CLKMAN_CLK_CONFIG](#) register. These fields include:

CLKMAN_CLK_CONFIG field	Function
hfx_enable	HFX Enable: enables the GM amplifier for the crystal oscillator
hfx_bypass	HFX Bypass: permits an external clock to be utilized
hfx_gm_adjust	HFX GM Adjust: adjusts the gain current density of the GM amplifier

The following table illustrates the register setting for the most common modes of operation:

Mode	HFX Enable	HFX Bypass	HFX GM Adjust		
			Crystal Frequency		
			8MHz	12MHz	24MHz
XTAL Run Mode	1	0	0x05	0x07	0x0A
Disabled	0	0	0x00		
External Clock Input	0	1	0x00		
Fast XTAL Startup	1	0	0x14		

The Crystal oscillator takes from 0.5ms at 24MHz to 2.5ms at 8MHz to start up. Once running, reducing the GM will reduce the current consumption.

10.1.3.3 External 32kHz Clock (Crystal or Resonator)

The **MAX32600** oscillator drives a standard 32.768kHz watch crystal. These crystals are very high-impedance and do not require a high-current oscillator circuit. The **MAX32600** oscillator is specifically designed to operate with these crystals and is compatible with their high impedance and limited power handling capability. The oscillator power dissipation is very low in order to maximize the lifetime of the RTC battery.

Note The oscillator is designed to be self-biasing. An external resistor should not be connected across the crystal.

The oscillator frequency can be adjusted for 25°C accuracy, as well as for temperature effects. See the [Real Time Clock \(RTC\)](#) section for details.

Firmware Control of the External 32kHz Clock

The 32kHz RTC clock is enabled with the [PWRSEQ_REG0.pwr_rtcent_run](#) bit for LP2: PMU and LP3: RUN and [PWRSEQ_REG0.pwr_rtcent_slp](#) bit for LP0: STOP and LP1: STANDBY.

10.1.3.4 Phase Lock Loop

An on-chip Phase Lock Loop (PLL) is provided to allow for generation of a 48MHz USB clock using lower value external quartz crystals or ceramic resonators. The crystal clock multiplier is controlled via the [CLKMAN_CLK_CONFIG](#) register. Note that proper divisor select is required with the different frequencies of the XTAL reference clocks.

The PLL is selectable between 2X, 4X, and 6X frequency multiplication modes and has a nominal output frequency of 48MHz. Note that the maximum frequency the CPU can be operated at is 24MHz. The PLL's 48MHz output is divided down by two in order to be used as a CPU clock source. To select the multiplier for the PLL, set the [CLKMAN_CLK_CONFIG.pll_divisor_select](#) bit field appropriately.

PLL Frequency Divisor Selection

PLL Divisor Select	PLL Multiplier	Input Clock Freq (HFXIN / Internal Osc)
00b	2X	24MHz
01b	4X	12MHz
1xb	6X	8MHz

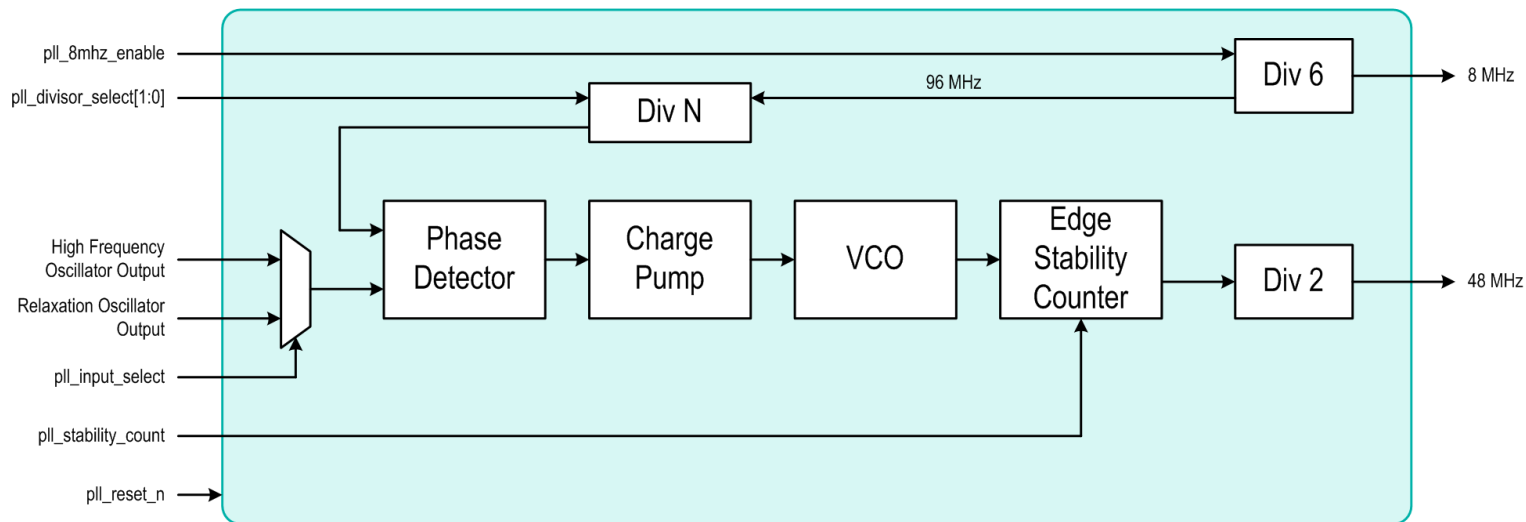


Figure 10.3: Phase Lock Loop Functional Diagram

Firmware Control of the Phase Lock Loop

The PLL is disabled by default. However, the PLL is needed to generate the required 48MHz clock for USB operation. The PLL can multiply an 8MHz, 12MHz, or 24MHz clock up to the required 48MHz. The `CLKMAN_CLK_CONFIG` register settings for the most common applications are detailed in the table below.

PLL Firmware Configuration Table

Mode	PLL Enable	PLL ResetN	PLL Input Select	PLL Divisor Select
Disabled	0	0	0	00b
Relax Oscillator Input	1	1	1	00b
8Mhz Crystal Input	1	1	0	1xb
12Mhz Crystal Input	1	1	0	01b
24Mhz Crystal Input	1	1	0	00b

10.1.3.5 Relaxation Oscillator

The **MAX32600** can source its main system clock directly from an internal 24MHz \pm 1% Relaxation Oscillator. The system Relaxation Oscillator is controlled via [CLKMAN_CLK_CONFIG](#), [PWRSEQ_REG0](#), [ADCCFG_RO_CAL0](#), and [ADCCFG_RO_CAL1](#) registers.

User Trim for \pm 0.25% Accuracy

The Relaxation Oscillator can be used as the USB clock source if its frequency is calibrated to \pm 0.25% using a precision 32kHz Real Time Clock as illustrated [below](#). The frequency calibration must be run before a USB operation to compensate for frequency shifts induced by temperature and supply voltage changes.

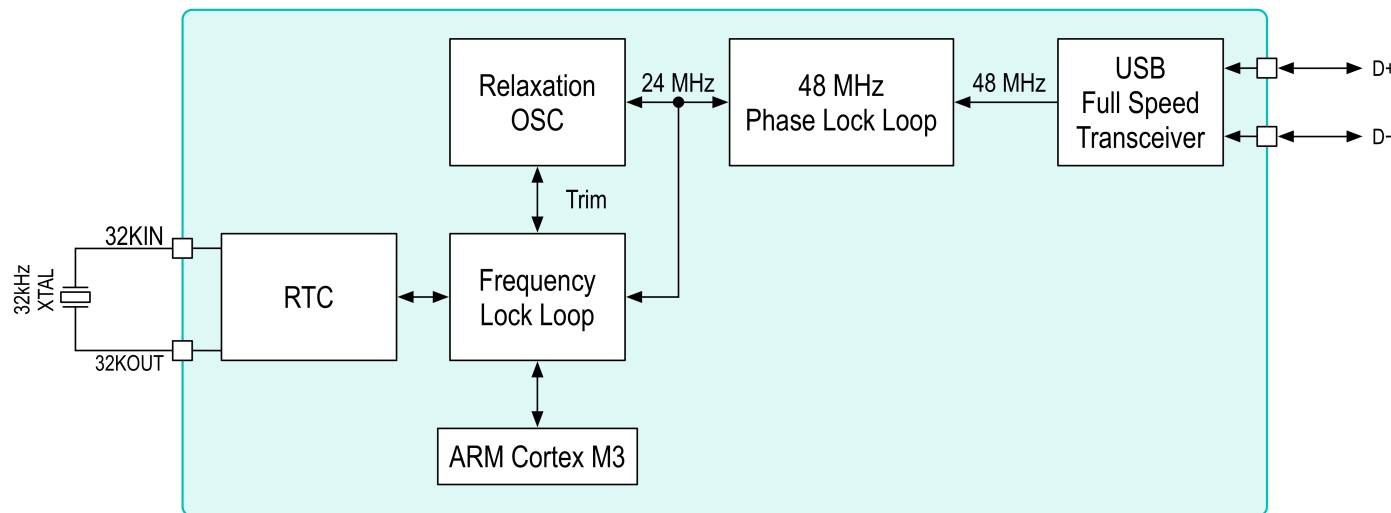


Figure 10.4: Relaxation Oscillator Frequency Calibration

Relaxation Oscillator Operation During a Wake Up Event

The [internal power sequencer](#) enables the Relaxation Oscillator and integrated voltage regulator during transition from a [low power state](#) to [normal run mode](#). For the first 15 μ s after a wake up event, the Relaxation Oscillator is not used by the system. From 15 μ s to 45 μ s, the system is operated using a divided down output of the Relaxation Oscillator (3MHz), and after 45 μ s, the 24MHz output of the Relaxation Oscillator is used as the system clock. Optionally, the relaxation oscillator can

remain powered up in LP1.

Firmware Control of the Relaxation Oscillator

Typically, the Relaxation Oscillator is used for normal system operation due its low power consumption. As described above, this is the default clock after a wake up event. If a different clock source is selected using the field `CLKMAN_CLK_CTRL.system_source_select`, the Relaxation Oscillator can be powered down using `PWRSEQ_REG0.pwr_roen_run`. However, care must be taken to power up the Relaxation Oscillator before it is selected as the system clock or a lock-up condition may occur.

The Relaxation Oscillator can also be used as the input clock to the PLL. This can be used to support USB operation without an external crystal if the relaxation oscillator is frequency calibrated to a quality 32kHz real time clock crystal.

The frequency calibration is performed in the steps outlined in the table below.

Relaxation Oscillator Calibration Procedure

Step	Action	Register I/O
1	Enable 32kHz RTC (if not enabled)	<code>PWRSEQ_REG0.pwr_rtcent_run = 1</code>
2	Read relaxation oscillator flash trim shadow register (RTC Power Domain)	<code>PWRSEQ_REG5.pwr_trim_osc_vref = InitTrim[6:0]</code>
3	Write trim setting to frequency calibration initial condition register (Save it as the 7 MSBs)	<code>ADCCFG_RO_CAL1.trim_init = (InitTrim[6:0] << 2)</code>
4	Load initial trim to active frequency trim register. (write 1 active, self-clearing)	<code>ADCCFG_RO_CAL0.ro_cal_load = 1</code>
5	Enable frequency loop to control relaxation oscillator trim	<code>ADCCFG_RO_CAL0.ro_cal_en = 1</code>
6	Run frequency calibration	<code>ADCCFG_RO_CAL0.ro_cal_run = 1</code>
7	Wait 60ms	N/A
8	Stop frequency calibration	<code>ADCCFG_RO_CAL0.ro_cal_run = 0</code>
9	Read final frequency trim value	<code>ADCCFG_RO_CAL0.ro_trim = FinalTrim[8:0]</code>
10	Write final trim (7 MSBs) to relaxation oscillator flash trim shadow register	<code>PWRSEQ_REG5.pwr_trim_osc_vref = (FinalTrim[8:2] >> 2)</code>
11	Disable RTC (if not needed)	<code>PWRSEQ_REG0.pwr_rtcent_run = 0</code>

10.1.3.6 Crypto Clock Relaxation Oscillator

The **MAX32600** sources the **Trust Protection Unit (TPU)** from an internal 44MHz Relaxation Oscillator (**NOTE:** this is separate from the 24MHz \pm 1% Relaxation Oscillator). A one-time factory trim is utilized to achieve the accuracy over process, temperature, and supply voltage. The Crypto Relaxation Oscillator is controlled

via `CLKMAN_CLK_CONFIG` register.

<code>CLKMAN_CLK_CONFIG</code> field	Function
<code>crypto_enable</code>	Enable the Crypto Relaxation Oscillator
<code>crypto_reset_n</code>	Active low holds the Crypto Relaxation Oscillator in reset state
<code>crypto_stability_count</code>	Crypto Relaxation Oscillator stability select

10.1.4 ADC Clock Source Configuration

The **ADC** requires a clock with a frequency of 8MHz or lower. Additionally, 16-19 clock cycles based on the Programmable Gain Amplifier (PGA) gain are required to perform a conversion. The ADC will operate with a clock frequency less than 8MHz; however, operating the ADC at 8MHz optimizes minimum power consumption (the ADC only consumes power when clocked). The ADC clock selection block is shown [below](#).

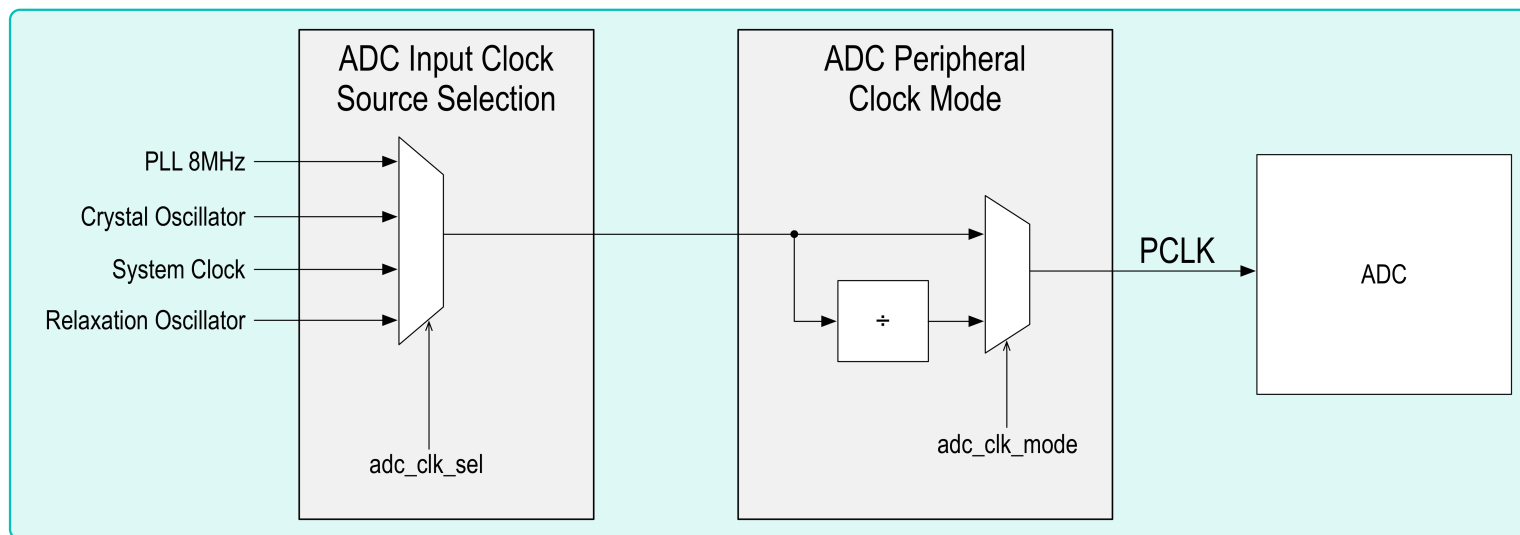


Figure 10.5: ADC Clock Source Block Diagram

Control of the ADC Clock Source Configuration

Clock configuration involves a trade-off with power and performance. The lowest jitter 8MHz clock with the best duty cycle provides the smallest degradation from aperture jitter. For lower ADC sample rates and lower frequency analog inputs this may be less of a consideration. The best performance is achieved with an external 8MHz crystal. The lowest power is achieved using the internal 24MHz Relaxation oscillator.

The following table illustrates the register controls for common ADC clock configurations.

ADC Clock Configuration Table

Mode	CLKMAN_CLK_CTRL.adc_gate_n	CLKMAN_CLK_CTRL.adc_source_select	ADC_CTRL0.adc_clk_mode
Disabled	0	00b	000b
PLL 8MHz Output (PLL configured and enabled)	1	00b	000b
8MHz External Crystal	1	01b	000b
24MHz External Crystal	1	10b	010b
24MHz Relaxation Oscillator	1	11b	010b

10.1.5 Registers (CLKMAN)

10.1.5.1 Module CLKMAN Registers

Address	Register	32b Word Len	Description
0x40090400	CLKMAN_CLK_CONFIG	1	System Clock Configuration
0x40090404	CLKMAN_CLK_CTRL	1	System Clock Controls
0x40090408	CLKMAN_INTFL	1	Interrupt Flags
0x4009040C	CLKMAN_INTEN	1	Interrupt Enable/Disable Controls
0x40090410	CLKMAN_TRIM_CALC	1	Trim Calculation Controls
0x40090424	CLKMAN_I2C_TIMER_CTRL	1	I2C Timer Control
0x40090440	CLKMAN_CLK_CTRL_0_SYSTEM	1	Control Settings for CLK0 - System Clock
0x40090444	CLKMAN_CLK_CTRL_1_GPIO	1	Control Settings for CLK1 - GPIO Module Clock

Address	Register	32b Word Len	Description
0x40090448	CLKMAN_CLK_CTRL_2_PT	1	Control Settings for CLK2 - Pulse Train Module Clock
0x4009044C	CLKMAN_CLK_CTRL_3_SPI0	1	Control Settings for CLK3 - SPI0 Master Clock
0x40090450	CLKMAN_CLK_CTRL_4_SPI1	1	Control Settings for CLK4 - SPI1 Master Clock
0x40090454	CLKMAN_CLK_CTRL_5_SPI2	1	Control Settings for CLK5 - SPI2 Master Clock
0x40090458	CLKMAN_CLK_CTRL_6_I2CM	1	Control Settings for CLK6 - Clock for all I2C Masters
0x4009045C	CLKMAN_CLK_CTRL_7_I2CS	1	Control Settings for CLK7 - I2C Slave Clock
0x40090460	CLKMAN_CLK_CTRL_8_LCD_CHPUMP	1	Control Settings for CLK8 - LCD Charge Pump Clock
0x40090464	CLKMAN_CLK_CTRL_9_PUF	1	Control Settings for CLK9 - PUF Clock
0x40090468	CLKMAN_CLK_CTRL_10_PRNG	1	Control Settings for CLK10 - PRNG Clock
0x4009046C	CLKMAN_CLK_CTRL_11_WDT0	1	Control Settings for CLK11 - Watchdog Timer 0 ScaledSysClk
0x40090470	CLKMAN_CLK_CTRL_12_WDT1	1	Control Settings for CLK12 - Watchdog Timer 1 ScaledSysClk
0x40090474	CLKMAN_CLK_CTRL_13_RTC_INT_SYNC	1	Control Settings for CLK13 - RTC Interrupt Sync Clock
0x40090478	CLKMAN_CLK_CTRL_14_DAC0	1	Control Settings for CLK14 - 12-bit DAC 0 Clock
0x4009047C	CLKMAN_CLK_CTRL_15_DAC1	1	Control Settings for CLK15 - 12-bit DAC 1 Clock
0x40090480	CLKMAN_CLK_CTRL_16_DAC2	1	Control Settings for CLK16 - 8-bit DAC 0 Clock
0x40090484	CLKMAN_CLK_CTRL_17_DAC3	1	Control Settings for CLK17 - 8-bit DAC 1 Clock
0x40090500	CLKMAN_CRYPT_CLK_CTRL_0_AES	1	Control Settings for Crypto Clock 0 - AES
0x40090504	CLKMAN_CRYPT_CLK_CTRL_1_MAA	1	Control Settings for Crypto Clock 1 - MAA
0x40090508	CLKMAN_CRYPT_CLK_CTRL_2_PRNG	1	Control Settings for Crypto Clock 2 - PRNG
0x40090540	CLKMAN_CLK_GATE_CTRL0	1	Dynamic Clock Gating Control Register 0
0x40090544	CLKMAN_CLK_GATE_CTRL1	1	Dynamic Clock Gating Control Register 1

Address	Register	32b Word Len	Description
0x40090548	CLKMAN_CLK_GATE_CTRL2	1	Dynamic Clock Gating Control Register 2

10.1.5.1.1 CLKMAN_CLK_CONFIG

CLKMAN_CLK_CONFIG.hfx_enable

Field	Bits	Default	Access	Description
hfx_enable	0	no effect	R/W	HFX Enable

- 0: Disabled
- 1: High frequency oscillator is enabled

CLKMAN_CLK_CONFIG.hfx_bypass

Field	Bits	Default	Access	Description
hfx_bypass	1	no effect	R/W	HFX Bypass

- 0: Bypass disabled
- 1: Bypass enabled, HFX may be driven by external clock source

CLKMAN_CLK_CONFIG.hfx_test_enable

Field	Bits	Default	Access	Description
hfx_test_enable	2	no effect	R/W	Reserved Field; Do Not Modify

This register field must be left at its default value for proper operation.

CLKMAN_CLK_CONFIG.hfx_gm_adjust

Field	Bits	Default	Access	Description
hfx_gm_adjust	8:4	no effect	R/W	HFX GM Adjust

GM Adjust for Crystal Oscillator Amp

CLKMAN_CLK_CONFIG.hfx_dc_control

Field	Bits	Default	Access	Description
hfx_dc_control	11:9	no effect	R/W	HFX DC Control

Duty Cycle Control for Crystal Oscillator Amp

CLKMAN_CLK_CONFIG.pll_enable

Field	Bits	Default	Access	Description
pll_enable	12	no effect	R/W	PLL Enable

Enable Phase Lock Loop clock source generator

CLKMAN_CLK_CONFIG.pll_reset_n

Field	Bits	Default	Access	Description
pll_reset_n	13	no effect	R/W	PLL ResetN

Active low reset for PLL

CLKMAN_CLK_CONFIG.pll_input_select

Field	Bits	Default	Access	Description
pll_input_select	14	no effect	R/W	PLL Input Select

Selects input clock source for PLL

- 0: Selects HFX output as PLL input clock source
- 1: Selects 24MHz RO output as PLL input clock src

CLKMAN_CLK_CONFIG.pll_divisor_select

Field	Bits	Default	Access	Description
pll_divisor_select	17:16	no effect	R/W	PLL Divisor Select

Must be set to match the PLL input frequency, to allow the intended 48MHz output frequency to be generated.

- 0: PLL input frequency is 24MHz
- 1: PLL input frequency is 12MHz
- 2/3: PLL input frequency is 8MHz

CLKMAN_CLK_CONFIG.pll_8mhz_enable

Field	Bits	Default	Access	Description
pll_8mhz_enable	18	no effect	R/W	PLL 8MHz Enable

- 0: Disabled
- 1: Enable the 8MHz output of the PLL

CLKMAN_CLK_CONFIG.pll_bypass

Field	Bits	Default	Access	Description
pll_bypass	19	no effect	R/W	Reserved Field; Do Not Modify

This register field must be left at its default value for proper operation.

CLKMAN_CLK_CONFIG.pll_stability_count

Field	Bits	Default	Access	Description
pll_stability_count	23:20	no effect	R/W	PLL Stability Count Select

Defines terminal count for PLL stable indicator in terms of PLL clocks as (# of clocks) = $2^{(\text{field value} + 8)}$

- 0: 256 clocks (2^8)
- 1: 512 clocks (2^9)
- 2: 1024 clocks (2^{10})
- ...
- 14: 4194304 clocks (2^{22})
- 15: 8388608 clocks (2^{23})

CLKMAN_CLK_CONFIG.crypto_enable

Field	Bits	Default	Access	Description
crypto_enable	24	0	R/W	Crypto Oscillator Enable

Set to 1 to enable crypto oscillator

CLKMAN_CLK_CONFIG.crypto_reset_n

Field	Bits	Default	Access	Description
crypto_reset_n	25	0	R/W	Crypto Oscillator ResetN

Active low - 0 holds crypto oscillator in reset state

CLKMAN_CLK_CONFIG.crypto_stability_count

Field	Bits	Default	Access	Description
crypto_stability_count	31:28	5	R/W	Crypto Oscillator Stability Select

Defines terminal count for crypto oscillator stable indicator in terms of crypto clocks as (# of clocks) = $2^{(\text{field value} + 8)}$

- 0: 256 clocks (2^8)
- 1: 512 clocks (2^9)
- 2: 1024 clocks (2^{10})
- ...
- 14: 4194304 clocks (2^{22})
- 15: 8388608 clocks (2^{23})

10.1.5.1.2 CLKMAN_CLK_CTRL**CLKMAN_CLK_CTRL.system_source_select**

Field	Bits	Default	Access	Description
system_source_select	2:1	no effect	R/W	System Clock Source Select

- 0: 24MHz Relaxation Osc output (divided by 8)
- 1: 24MHz Relaxation Osc output (undivided)
- 2: HFX Input (intended for test mode only)
- 3: PLL 48MHz output (divided by 2)

Read value is actual mux select, and may lag value written by several clocks due to glitchless clock switching circuit.

CLKMAN_CLK_CTRL.auto_clk_disable

Field	Bits	Default	Access	Description
auto_clk_disable	3	no effect	R/W	Auto Clock Disable

1: Automatically disables HFX and PLL Enables and restores system clock mux to use System RO when going into LP1.

CLKMAN_CLK_CTRL.usb_gate_n

Field	Bits	Default	Access	Description
usb_gate_n	4	0	R/W	USB Clock GateN

Gates off USB clock when asserted to 0 (active low)

CLKMAN_CLK_CTRL.adc_gate_n

Field	Bits	Default	Access	Description
adc_gate_n	8	0	R/W	ADC Clock GateN

Gates off ADC clock when asserted to 0 (active low)

CLKMAN_CLK_CTRL.adc_source_select

Field	Bits	Default	Access	Description
adc_source_select	10:9	00b	R/W	ADC Clock Source Select

Selects the source for the ADC clock:

- 00b: 24MHz system clock
- 01b: PLL 8MHz output
- 10b: HFX clock source
- 11b: 24MHz ring oscillator

CLKMAN_CLK_CTRL.crypto_gate_n

Field	Bits	Default	Access	Description
crypto_gate_n	12	0	R/W	Crypto Clock GateN

Gates off crypto clock when asserted (0;active low)

CLKMAN_CLK_CTRL.watchdog0_gate_n

Field	Bits	Default	Access	Description
watchdog0_gate_n	16	no effect	R/W	Watchdog 0 Clock GateN

Gates off watchdog clock when asserted (active low)

CLKMAN_CLK_CTRL.watchdog0_source_select

Field	Bits	Default	Access	Description
watchdog0_source_select	18:17	no effect	R/W	Watchdog 0 Clock Source Select

Selects the source for the watchdog external clock:

- 00b: Scaled Sys Clock Source (as set by SYS_CLK_CTRL_11)
- 01b: RTC oscillator
- 10b: 24MHz ring oscillator
- 11b: Nano-ring oscillator

Read value is actual mux select, and may lag value written by several clocks due to glitchless clock switching circuit.

CLKMAN_CLK_CTRL.watchdog1_gate_n

Field	Bits	Default	Access	Description
watchdog1_gate_n	20	no effect	R/W	Watchdog 1 Clock GateN

Gates off watchdog clock when asserted (active low)

CLKMAN_CLK_CTRL.watchdog1_source_select

Field	Bits	Default	Access	Description
watchdog1_source_select	22:21	no effect	R/W	Watchdog 1 Clock Source Select

Selects the source for the watchdog external clock:

- 00b: Scaled Sys Clock Source (as set by SYS_CLK_CTRL_12)
- 01b: RTC oscillator
- 10b: 24MHz ring oscillator
- 11b: Nano-ring oscillator

Read value is actual mux select, and may lag value written by several clocks due to glitchless clock switching circuit.

CLKMAN_CLK_CTRL.rtos_mode

Field	Bits	Default	Access	Description
rtos_mode	24	0	R/W	RTOS Mode

- 0: In this mode (default), when the device enters LP2 or LP1 mode, the free-running clock (FCLK) to the ARM core will be shut off. This has two effects. First, the input clock to the SysTick timer will be gated off, which means that SysTick cannot be used to wake up the device. The SysTick timer will not receive clock pulses during LP2 or LP1 even if the 32kHz clock has been selected as the SysTick clock source. Second, the ARM core will not be accessible via the debug port during LP2 or LP1.
- 1: In this mode, the FCLK will continue running to the ARM core even during LP2 or LP1, which results in additional power consumption during these low power modes. With FCLK always free-running, the ARM CPU can still be accessed using the debug port even in LP2 or LP1. If the SysTick timer has been selected to use STCLK (the 32kHz clock) as its input clock source, then in this mode, SysTick will continue running normally in LP2 or LP1. This means that SysTick can be used to wake the device from these low power modes, and also that SysTick can be used to accurately time a period's duration even if the time period includes entry into and/or exit from LP2/LP1.

10.1.5.1.3 CLKMAN_INTFL

CLKMAN_INTFL.ring_stable

Field	Bits	Default	Access	Description
ring_stable	0	0	W1C	24MHz Relaxation Osc Stable Int Flag

Write 1 to clear.

Set to 1 by hardware when the 24MHz relaxation oscillator is considered stable.

CLKMAN_INTFL.pll_stable

Field	Bits	Default	Access	Description
pll_stable	1	0	W1C	PLL Output Stable Int Flag

Write 1 to clear.

Set to 1 by hardware when the PLL output is considered stable.

CLKMAN_INTFL.crypto_stable

Field	Bits	Default	Access	Description
crypto_stable	2	0	W1C	Crypto Oscillator Stable Int Flag

Write 1 to clear.

Set to 1 by hardware when the crypto oscillator is considered stable.

10.1.5.1.4 CLKMAN_INTEN

CLKMAN_INTEN.ring_stable

Field	Bits	Default	Access	Description
ring_stable	0	0	R/W	24MHz Relaxation Oscillator Stable Int Enable

0:Interrupt disabled; 1:Interrupt enabled

CLKMAN_INTEN.pll_stable

Field	Bits	Default	Access	Description
pll_stable	1	0	R/W	PLL Output Stable Int Enable

0:Interrupt disabled; 1:Interrupt enabled

CLKMAN_INTEN.crypto_stable

Field	Bits	Default	Access	Description
crypto_stable	2	0	R/W	Crypto Oscillator Stable Int Enable

0:Interrupt disabled; 1:Interrupt enabled

10.1.5.1.5 CLKMAN_TRIM_CALC

CLKMAN_TRIM_CALC.trim_clk_sel

Field	Bits	Default	Access	Description
trim_clk_sel	0	0	R/W	Trim Clock Select

Selects which trimmable clock to measure for the calculation.

- 0: 24MHz ring oscillator

- 1: Crypto ring oscillator (divided by 2)

CLKMAN_TRIM_CALC.trim_calc_start

Field	Bits	Default	Access	Description
trim_calc_start	1	0	R/W	Start Trim Calculation

Write to 1 to start a new trim calculation; self clearing.

CLKMAN_TRIM_CALC.trim_calc_completed

Field	Bits	Default	Access	Description
trim_calc_completed	2	0	R/O	Trim Calculation Completed

Status bit; this bit is set to 1 by hardware when a trim calculation is completed, and is cleared to 0 by hardware when a new trim calculation is started.

CLKMAN_TRIM_CALC.trim_enable

Field	Bits	Default	Access	Description
trim_enable	3	0	R/W	Trim Logic Enable

- 0: Clear to put trim calculation logic in low-power state.
- 1: Set to this value before using trim calculation.

CLKMAN_TRIM_CALC.trim_calc_results

Field	Bits	Default	Access	Description
trim_calc_results	25:16	n/a	R/O	Trim Calculation Results

Results of trim calculation, valid when the Trim Calculation Completed bit is set to 1. This result consists of the number of trimmable clock (selected by bit 0) edges seen during the time taken for 256 HFX clock periods to elapse.

10.1.5.1.6 CLKMAN_I2C_TIMER_CTRL

CLKMAN_I2C_TIMER_CTRL.i2c_1ms_timer_en

Field	Bits	Default	Access	Description
i2c_1ms_timer_en	0	0	R/W	I2C 1ms Timer Enable

1:Enables the general purpose timer used by the I2C blocks for determining timeout events.

10.1.5.1.7 CLKMAN_CLK_CTRL_0_SYSTEM

CLKMAN_CLK_CTRL_0_SYSTEM.sys_clk_scale

Field	Bits	Default	Access	Description
sys_clk_scale	3:0	0001b	R/W	Control Settings for CLK0 - System Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)

- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is enabled by default following reset.

10.1.5.1.8 CLKMAN_CLK_CTRL_1_GPIO

CLKMAN_CLK_CTRL_1_GPIO.gpio_clk_scale

Field	Bits	Default	Access	Description
gpio_clk_scale	3:0	0000b	R/W	Control Settings for CLK1 - GPIO Module Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.9 CLKMAN_CLK_CTRL_2_PT**CLKMAN_CLK_CTRL_2_PT.pulse_train_clk_scale**

Field	Bits	Default	Access	Description
pulse_train_clk_scale	3:0	0000b	R/W	Control Settings for CLK2 - Pulse Train Module Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.10 CLKMAN_CLK_CTRL_3_SPI0**CLKMAN_CLK_CTRL_3_SPI0.spi0_clk_scale**

Field	Bits	Default	Access	Description
spi0_clk_scale	3:0	0000b	R/W	Control Settings for CLK3 - SPI0 Master Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.11 CLKMAN_CLK_CTRL_4_SPI1

CLKMAN_CLK_CTRL_4_SPI1.spi1_clk_scale

Field	Bits	Default	Access	Description
spi1_clk_scale	3:0	0000b	R/W	Control Settings for CLK4 - SPI1 Master Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)

- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.12 CLKMAN_CLK_CTRL_5_SPI2

CLKMAN_CLK_CTRL_5_SPI2.spi2_clk_scale

Field	Bits	Default	Access	Description
spi2_clk_scale	3:0	0000b	R/W	Control Settings for CLK5 - SPI2 Master Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.13 CLKMAN_CLK_CTRL_6_I2CM**CLKMAN_CLK_CTRL_6_I2CM.i2cm_clk_scale**

Field	Bits	Default	Access	Description
i2cm_clk_scale	3:0	0000b	R/W	Control Settings for CLK6 - Clock for all I2C Masters

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.14 CLKMAN_CLK_CTRL_7_I2CS**CLKMAN_CLK_CTRL_7_I2CS.i2cs_clk_scale**

Field	Bits	Default	Access	Description
i2cs_clk_scale	3:0	0000b	R/W	Control Settings for CLK7 - I2C Slave Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.15 CLKMAN_CLK_CTRL_8_LCD_CHPUMP

CLKMAN_CLK_CTRL_8_LCD_CHPUMP.lcd_clk_scale

Field	Bits	Default	Access	Description
lcd_clk_scale	3:0	0000b	R/W	Control Settings for CLK8 - LCD Charge Pump Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)

- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.16 CLKMAN_CLK_CTRL_9_PUF

CLKMAN_CLK_CTRL_9_PUF.puf_clk_scale

Field	Bits	Default	Access	Description
puf_clk_scale	3:0	0000b	R/W	Control Settings for CLK9 - PUF Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.17 CLKMAN_CLK_CTRL_10_PRNG**CLKMAN_CLK_CTRL_10_PRNG.prng_clk_scale**

Field	Bits	Default	Access	Description
prng_clk_scale	3:0	0000b	R/W	Control Settings for CLK10 - PRNG Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.18 CLKMAN_CLK_CTRL_11_WDT0**CLKMAN_CLK_CTRL_11_WDT0.watchdog0_clk_scale**

Field	Bits	Default	Access	Description
watchdog0_clk_scale	3:0	0000b	R/W	Control Settings for CLK11 - Watchdog Timer 0 ScaledSysClk

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

Note If this clock is used as the WDT0 watchdog timer clock source, a reset will shut down the watchdog timer.

10.1.5.1.19 CLKMAN_CLK_CTRL_12_WDT1

CLKMAN_CLK_CTRL_12_WDT1.watchdog1_clk_scale

Field	Bits	Default	Access	Description
watchdog1_clk_scale	3:0	0000b	R/W	Control Settings for CLK12 - Watchdog Timer 1 ScaledSysClk

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)

- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

Note If this clock is used as the WDT1 watchdog timer clock source, a reset will shut down the watchdog timer.

10.1.5.1.20 CLKMAN_CLK_CTRL_13_RTC_INT_SYNC

CLKMAN_CLK_CTRL_13_RTC_INT_SYNC.rtc_clk_scale

Field	Bits	Default	Access	Description
rtc_clk_scale	3:0	0000b	R/W	Control Settings for CLK13 - RTC Interrupt Sync Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)

- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.21 CLKMAN_CLK_CTRL_14_DAC0

CLKMAN_CLK_CTRL_14_DAC0.dac0_clk_scale

Field	Bits	Default	Access	Description
dac0_clk_scale	3:0	0000b	R/W	Control Settings for CLK14 - 12-bit DAC 0 Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.22 CLKMAN_CLK_CTRL_15_DAC1**CLKMAN_CLK_CTRL_15_DAC1.dac1_clk_scale**

Field	Bits	Default	Access	Description
dac1_clk_scale	3:0	0000b	R/W	Control Settings for CLK15 - 12-bit DAC 1 Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.23 CLKMAN_CLK_CTRL_16_DAC2**CLKMAN_CLK_CTRL_16_DAC2.dac2_clk_scale**

Field	Bits	Default	Access	Description
dac2_clk_scale	3:0	0000b	R/W	Control Settings for CLK16 - 8-bit DAC 0 Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)
- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.24 CLKMAN_CLK_CTRL_17_DAC3

CLKMAN_CLK_CTRL_17_DAC3.dac3_clk_scale

Field	Bits	Default	Access	Description
dac3_clk_scale	3:0	0000b	R/W	Control Settings for CLK17 - 8-bit DAC 1 Clock

- 0000b: CLK is Disabled
- 0001b: CLK = (System Clock Source / 1)
- 0010b: CLK = (System Clock Source / 2)
- 0011b: CLK = (System Clock Source / 4)
- 0100b: CLK = (System Clock Source / 8)
- 0101b: CLK = (System Clock Source / 16)

- 0110b: CLK = (System Clock Source / 32)
- 0111b: CLK = (System Clock Source / 64)
- 1000b: CLK = (System Clock Source / 128)
- 1001b: CLK = (System Clock Source / 256)
- other: CLK = (System Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.25 CLKMAN_CRYPT_CLK_CTRL_0_AES

CLKMAN_CRYPT_CLK_CTRL_0_AES.aes_clk_scale

Field	Bits	Default	Access	Description
aes_clk_scale	3:0	0000b	R/W	Control Settings for Crypto Clock 0 - AES

- 0000b: CLK is Disabled
- 0001b: CLK = (Crypto Clock Source / 1)
- 0010b: CLK = (Crypto Clock Source / 2)
- 0011b: CLK = (Crypto Clock Source / 4)
- 0100b: CLK = (Crypto Clock Source / 8)
- 0101b: CLK = (Crypto Clock Source / 16)
- 0110b: CLK = (Crypto Clock Source / 32)
- 0111b: CLK = (Crypto Clock Source / 64)
- 1000b: CLK = (Crypto Clock Source / 128)
- 1001b: CLK = (Crypto Clock Source / 256)
- other: CLK = (Crypto Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.26 CLKMAN_CRYPT_CLK_CTRL_1_MAA**CLKMAN_CRYPT_CLK_CTRL_1_MAA.uma_clk_scale**

Field	Bits	Default	Access	Description
uma_clk_scale	3:0	0000b	R/W	Control Settings for Crypto Clock 1 - MAA

- 0000b: CLK is Disabled
- 0001b: CLK = (Crypto Clock Source / 1)
- 0010b: CLK = (Crypto Clock Source / 2)
- 0011b: CLK = (Crypto Clock Source / 4)
- 0100b: CLK = (Crypto Clock Source / 8)
- 0101b: CLK = (Crypto Clock Source / 16)
- 0110b: CLK = (Crypto Clock Source / 32)
- 0111b: CLK = (Crypto Clock Source / 64)
- 1000b: CLK = (Crypto Clock Source / 128)
- 1001b: CLK = (Crypto Clock Source / 256)
- other: CLK = (Crypto Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.27 CLKMAN_CRYPT_CLK_CTRL_2_PRNG**CLKMAN_CRYPT_CLK_CTRL_2_PRNG.prng_clk_scale**

Field	Bits	Default	Access	Description
prng_clk_scale	3:0	0000b	R/W	Control Settings for Crypto Clock 2 - PRNG

- 0000b: CLK is Disabled
- 0001b: CLK = (Crypto Clock Source / 1)
- 0010b: CLK = (Crypto Clock Source / 2)
- 0011b: CLK = (Crypto Clock Source / 4)
- 0100b: CLK = (Crypto Clock Source / 8)
- 0101b: CLK = (Crypto Clock Source / 16)
- 0110b: CLK = (Crypto Clock Source / 32)
- 0111b: CLK = (Crypto Clock Source / 64)
- 1000b: CLK = (Crypto Clock Source / 128)
- 1001b: CLK = (Crypto Clock Source / 256)
- other: CLK = (Crypto Clock Source / 1)

This clock is disabled by default following reset.

10.1.5.1.28 CLKMAN_CLK_GATE_CTRL0

CLKMAN_CLK_GATE_CTRL0.cm3_clk_gater

Field	Bits	Default	Access	Description
cm3_clk_gater	1:0	01b	R/W	Clock Gating Control for CM3 CPU

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.sysbus_clk_gater

Field	Bits	Default	Access	Description
sysbus_clk_gater	3:2	01b	R/W	Clock Gating Control for SysBus

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.icache_clk_gater

Field	Bits	Default	Access	Description
icache_clk_gater	5:4	01b	R/W	Clock Gating Control for Instruction Cache

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.flash_clk_gater

Field	Bits	Default	Access	Description
flash_clk_gater	7:6	01b	R/W	Clock Gating Control for Flash Memory

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.sram_clk_gater

Field	Bits	Default	Access	Description
sram_clk_gater	9:8	01b	R/W	Clock Gating Control for SRAM

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.apb_bridge_clk_gater

Field	Bits	Default	Access	Description
apb_bridge_clk_gater	11:10	01b	R/W	Clock Gating Control for AHB-to-APB Bridge

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled

- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.sysman_clk_gater

Field	Bits	Default	Access	Description
sysman_clk_gater	13:12	01b	R/W	Clock Gating Control for Clkman/–Pwrman/IOMan

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.uart0_clk_gater

Field	Bits	Default	Access	Description
uart0_clk_gater	15:14	01b	R/W	Clock Gating Control for UART 0

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.uart1_clk_gater

Field	Bits	Default	Access	Description
uart1_clk_gater	17:16	01b	R/W	Clock Gating Control for UART 1

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.timer0_clk_gater

Field	Bits	Default	Access	Description
timer0_clk_gater	19:18	01b	R/W	Clock Gating Control for Timer Module 0

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.timer1_clk_gater

Field	Bits	Default	Access	Description
timer1_clk_gater	21:20	01b	R/W	Clock Gating Control for Timer Module 1

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.timer2_clk_gater

Field	Bits	Default	Access	Description
timer2_clk_gater	23:22	01b	R/W	Clock Gating Control for Timer Module 2

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.timer3_clk_gater

Field	Bits	Default	Access	Description
timer3_clk_gater	25:24	01b	R/W	Clock Gating Control for Timer Module 3

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.watchdog0_clk_gater

Field	Bits	Default	Access	Description
watchdog0_clk_gater	27:26	01b	R/W	Clock Gating Control for WDT 0

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.watchdog1_clk_gater

Field	Bits	Default	Access	Description
watchdog1_clk_gater	29:28	01b	R/W	Clock Gating Control for WDT 1

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.usb_clk_gater

Field	Bits	Default	Access	Description
usb_clk_gater	31:30	01b	R/W	Clock Gating Control for USB

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

10.1.5.1.29 CLKMAN_CLK_GATE_CTRL1

CLKMAN_CLK_GATE_CTRL1.testacc_clk_gater

Field	Bits	Default	Access	Description
testacc_clk_gater	1:0	01b	R/W	Clock Gating Control for TestAcc

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.adc_clk_gater

Field	Bits	Default	Access	Description
adc_clk_gater	3:2	01b	R/W	Clock Gating Control for ADC

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled

- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.dac12_0_clk_gater

Field	Bits	Default	Access	Description
dac12_0_clk_gater	5:4	01b	R/W	Clock Gating Control for DAC0

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.dac12_1_clk_gater

Field	Bits	Default	Access	Description
dac12_1_clk_gater	7:6	01b	R/W	Clock Gating Control for DAC1

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.dac8_0_clk_gater

Field	Bits	Default	Access	Description
dac8_0_clk_gater	9:8	01b	R/W	Clock Gating Control for DAC2

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.dac8_1_clk_gater

Field	Bits	Default	Access	Description
dac8_1_clk_gater	11:10	01b	R/W	Clock Gating Control for DAC3

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.pmu_clk_gater

Field	Bits	Default	Access	Description
pmu_clk_gater	13:12	01b	R/W	Clock Gating Control for DMA (PMU)

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.lcd_clk_gater

Field	Bits	Default	Access	Description
lcd_clk_gater	15:14	01b	R/W	Clock Gating Control for LCD

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.gpio_clk_gater

Field	Bits	Default	Access	Description
gpio_clk_gater	17:16	01b	R/W	Clock Gating Control for GPIO

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.pulsetrain_clk_gater

Field	Bits	Default	Access	Description
pulsetrain_clk_gater	19:18	01b	R/W	Clock Gating Control for Pulse Train Generators

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.spi0_clk_gater

Field	Bits	Default	Access	Description
spi0_clk_gater	21:20	01b	R/W	Clock Gating Control for SPI 0

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.spi1_clk_gater

Field	Bits	Default	Access	Description
spi1_clk_gater	23:22	01b	R/W	Clock Gating Control for SPI 1

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.spi2_clk_gater

Field	Bits	Default	Access	Description
spi2_clk_gater	25:24	01b	R/W	Clock Gating Control for SPI 2

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.i2cm0_clk_gater

Field	Bits	Default	Access	Description
i2cm0_clk_gater	27:26	01b	R/W	Clock Gating Control for I2C Master 0

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.i2cm1_clk_gater

Field	Bits	Default	Access	Description
i2cm1_clk_gater	29:28	01b	R/W	Clock Gating Control for I2C Master 1

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.i2cs_clk_gater

Field	Bits	Default	Access	Description
i2cs_clk_gater	31:30	01b	R/W	Clock Gating Control for I2C Slave

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

10.1.5.1.30 CLKMAN_CLK_GATE_CTRL2**CLKMAN_CLK_GATE_CTRL2.crc_clk_gater**

Field	Bits	Default	Access	Description
crc_clk_gater	1:0	01b	R/W	Clock Gating Control for CRC16/32 Engine

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL2.tpu_clk_gater

Field	Bits	Default	Access	Description
tpu_clk_gater	3:2	01b	R/W	Clock Gating Control for TPU

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL2.ssbmux_clk_gater

Field	Bits	Default	Access	Description
ssbmux_clk_gater	5:4	01b	R/W	Clock Gating Control for SSB Mux

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL2.pad_clk_gater

Field	Bits	Default	Access	Description
pad_clk_gater	7:6	01b	R/W	Clock Gating Control for Pad Interface

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

10.2 Watchdog Timers

10.2.1 Watchdog Timers Overview

The **MAX32600** features two Windowed Watchdog Timers (WWDT) that protect against corrupt or unreliable software, power faults, and other system-level problems that may place the **MAX32600** into unsuitable operating states. When the application is working correctly, application software will periodically reset the watchdog counter within a specific window of time. If the watchdog timer interrupt is enabled and the software does not reset the counter within the interrupt time period ([int_period](#)), the watchdog timer will generate a watchdog timer interrupt. If the watchdog timer reset is enabled and the software does not reset the counter within the reset time period ([rst_period](#)), the watchdog timer will generate a system reset. In both cases, a watchdog timer interrupt or reset will also occur if the counter is reset during the pre-window.

10.2.2 Clock Source Selection and Gating

The **MAX32600** supports multiple clock source selection options. This enables the use of both WWDTs to ensure that in the event a specific clock source fails, the second WWDT will still catch the failure.

Each Watchdog Timer on the **MAX32600** supports clock source selection from one of three available clocks. To select the clock source, refer to the table below and write to either the [watchdog0_source_select](#) or [watchdog1_source_select](#) field.

Note See [Watchdog Timer value explanation](#) below.

Watchdog Source Selection	Clock Source
00b	System Clock (default)
01b	RTC oscillator
10b	24MHz ring oscillator
11b	Nano-ring oscillator

Once a clock source is set up for the watchdog timer, the clock must be turned on to the Watchdog Timer. By default, the clock is gated off to the block. To turn on the clock and enable the Watchdog Timer Peripheral, write a 1 to the `WDTn_CTRL.en_clock` register.

The default source of the watchdog timer is the system clock and is scalable via the `CLKMAN_CLK_CTRL_11_WDT0.watchdog0_clk_scale` and `CLKMAN_CLK_CTRL_12_WDT1.watchdog1_clk_scale` registers.

Note This does not start the watchdog operation, but enables the selected clock to clock the Watchdog Timer once the Timer is configured and firmware enables the Watchdog. The [configuration procedure](#) is below.

10.2.3 Watchdog Timer Configuration

Each watchdog timer supports independent settings for three independent time delay periods:

1. **Pre-window period:** minimum time interval required between watchdog timer clear events
2. **Interrupt period:** time from watchdog timer clear until an interrupt is triggered by the watchdog timer
3. **Reset period:** time from watchdog timer clear until the system is reset

The Pre-Window period must be less than Interrupt Period which must be less than Reset Period. If the Interrupt period is smaller than the Pre-Window period, an out-of-window interrupt will occur. If the Interrupt period is larger than the Reset period, the interrupt will not happen before the reset system sends a reset signal to the **MAX32600**.

There are 16 choices of time delay periods for the watchdog timer: 2^{16} through 2^{31} clock cycles of the selected clock. The time delay for a specific clock source is calculated as follows:

- **Pre-window period** = Clock Source Period \times `wait_period`
- **Interrupt period** = Clock Source Period \times `int_period`
- **Reset period** = Clock Source Period \times `rst_period`

An illustration of different watchdog timer `int_period` values is shown in the below table.

Clock Source	Interrupt Period	Time Period
24MHz	1111(2 ¹⁶)	2.7ms
24MHz	1011(2 ²⁰)	43.7ms
24MHz	0000(2 ³¹)	89s

10.2.3.1 Locking and Unlocking the Watchdog Timer Configuration

Once the Watchdog Timer is configured, the [Watchdog Timer Control Register](#) can be locked by firmware to further protect against unintended consequences of runaway firmware or system failure. To do this, write the value 0x24 to the [WDTn_LOCK_CTRL.wdlock](#) register field.

If changes to the configuration need to be made by firmware under certain circumstances and the [WDTn_LOCK_CTRL](#) register is already set, it may be unlocked by writing the value 0x42 to the [WDTn_LOCK_CTRL.wdlock](#) register field.

10.2.3.2 Enabling and Disabling the Watchdog Timer Counter

The application software must set the [WDTn_CTRL.en_timer](#) to a 1 to start the Watchdog Timer. The Watchdog Timer is free-running; the following procedure must be followed when enabling to prevent an unintended reset during the enable process.

1. Write 0xA5 to [WDTn_CLEAR](#)
2. Write 0x5A to [WDTn_CLEAR](#)
3. Set [WDTn_CTRL.en_timer](#) bit

The watchdog timer can be disabled in multiple ways:

1. The application software can clear [WDTn_CTRL.en_timer](#) to 0.
2. A POR will clear [WDTn_CTRL.en_timer](#) to 0.
3. The interrupt and reset signals from the watchdog timer can also be enabled or disabled independently through the [WDTn_FLAGS.timeout](#) field.

The [WDTn \[n\]](#) value has the following significance:

- **WDT0** is a "core" watchdog: It can reset the core. (This is not a POR.) Its control register is reset via a system reset.
- **WDT1** is a "system" watchdog: It can reboot the core and results in a POR. Its control register is reset via a POR.
- *Note:* If WDT0 issues a system reset, WDT1 will continue to function because its configuration has not been reset.

10.2.4 Watchdog Timer Operation

Once the Watchdog Timer is enabled and begins counting, firmware is responsible for both periodically "feeding" the watchdog or clearing the watchdog timer to zero by writing the appropriate sequence to the `WDTn_CLEAR` register. Application software needs to ensure this is done within the defined window.

10.2.5 Registers (WDT)

10.2.5.1 Module WDT Registers

Address	Register	32b Word Len	Description
0x40021000	<code>WDT0_CTRL</code>	1	WDT0 - Watchdog Timer Control Register
0x40021004	<code>WDT0_CLEAR</code>	1	WDT0 - Watchdog Clear Register (Feed Dog)
0x40021008	<code>WDT0_FLAGS</code>	1	WDT0 - Watchdog Interrupt and Reset Flags
0x4002100C	<code>WDT0_ENABLE</code>	1	WDT0 - Interrupt/Reset Enable/Disable Controls
0x40021014	<code>WDT0_LOCK_CTRL</code>	1	WDT0 - Register Setting Lock for <code>WDT0_CTRL</code>
0x40022000	<code>WDT1_CTRL</code>	1	WDT1 - Watchdog Timer Control Register
0x40022004	<code>WDT1_CLEAR</code>	1	WDT1 - Watchdog Timer Clear Register (Feed Dog)
0x40022008	<code>WDT1_FLAGS</code>	1	WDT1 - Watchdog Interrupt and Reset Flags
0x4002200C	<code>WDT1_ENABLE</code>	1	WDT1 - Interrupt/Reset Enable/Disable Controls
0x40022014	<code>WDT1_LOCK_CTRL</code>	1	WDT1 - Register Setting Lock for <code>WDT1_CTRL</code>

10.2.5.1.1 WDTn_CTRL

WDTn_CTRL.int_period

Field	Bits	Default	Access	Description
int_period	3:0	Special	R/W	Period from WDT Clear to Interrupt Flag Set

This field sets the duration of the watchdog interrupt period, which is the time period from the beginning of the watchdog timer count (the count resets to zero when the watchdog timer is first enabled, as well as each time the watchdog timer is cleared by writing to [WDTn_CLEAR](#)) until the Watchdog Timeout Interrupt Flag ([wait_period](#) is set).

Defined in terms of a number of watchdog clocks, with the number of clocks given by 2^N , $N=(31 - \text{field value})$, e.g.

- 0h: 2^{31}
- 1h: 2^{30}
- 2h: 2^{29}
-
- Eh: 2^{17}
- Fh: 2^{16}

WDTn_CTRL.rst_period

Field	Bits	Default	Access	Description
rst_period	7:4	.	R/W	Period from WDT Clear to Reset Flag Set

Reset Period - the time period from the beginning of the watchdog timer count (WDT is cleared to zero by being enabled or when the watchdog timer is cleared by writing to the CLEAR register) until the Watchdog Reset Flag is set.

Defined in terms of a number of watchdog clocks, with the number of clocks given by 2^N , $N=(31 - \text{field value})$, e.g.

- 0h: 2^{31}
- 1h: 2^{30}
- 2h: 2^{29}
-
- Eh: 2^{17}
- Fh: 2^{16}

WDTn_CTRL.en_timer

Field	Bits	Default	Access	Description
en_timer	8	.	R/W	Watchdog Timer Enable

- 0: Watchdog timer is disabled and the counter is held at zero.
- 1: Watchdog timer is enabled and counting.

WDTn_CTRL.en_clock

Field	Bits	Default	Access	Description
en_clock	9	.	R/W	Watchdog Clock Gate

Watchdog Clock Gate. This enables the WDC clock which allows flags to be set/cleared (not the same as WDEN which allows the timer to increment).

- 0: Clock to WDT is disabled.
- 1: Clock to WDT is enabled.

WDTn_CTRL.wait_period

Field	Bits	Default	Access	Description
wait_period	15:12	.	R/W	Period from WDT Clear to Clear Window Begin

Pre-Window period. The time period the WDT waits (following WDT enable or reset/feed) before allowing the watchdog to be reset without triggering an out-of-window interrupt.

Defined in terms of a number of watchdog clocks, with the number of clocks given by 2^N , $N=(31 - \text{field value})$, e.g.

- 0h: 2^{31}
- 1h: 2^{30}
- 2h: 2^{29}
-
- Eh: 2^{17}
- Fh: 2^{16}

10.2.5.1.2 WDTn_CLEAR

Default	Access	Description
0000 0000 0000 0000 0000 0000 0000b	R/W	WDT0 - Watchdog Clear Register (Feed Dog)

Write 0xA5,0x5A sequence to clear watchdog timer (feed watchdog)

Reads always return zero.

10.2.5.1.3 WDTn_FLAGS

WDTn_FLAGS.timeout

Field	Bits	Default	Access	Description
timeout	0	0 (POR only)	W1C	Watchdog Timeout Interrupt Flag

Write 1 to clear this flag to 0.

Set to 1 by hardware when the watchdog timer reaches the end of the interrupt period without being cleared.

WDTn_FLAGS.pre_win

Field	Bits	Default	Access	Description
pre_win	1	0 (POR only)	W1C	Watchdog Pre-Window Clear Interrupt Flag

Write 1 to clear this flag to 0.

Set to 1 by hardware when the watchdog timer is cleared before the end of the pre-window period.

WDTn_FLAGS.reset_out

Field	Bits	Default	Access	Description
reset_out	2	0 (POR only)	W1C	Watchdog Reset Flag

Write 1 to clear this flag to 0.

Set to 1 by hardware when the watchdog timer reaches the end of the reset period without being cleared.

10.2.5.1.4 WDTn_ENABLE

WDTn_ENABLE.timeout

Field	Bits	Default	Access	Description
timeout	0	0	R/W	Enable Watchdog Interrupt

- 0: No interrupt will be triggered when the Watchdog Interrupt Flag is set.
- 1: An interrupt will be triggered by the WDT when the Watchdog Interrupt Flag is set to 1.

WDTn_ENABLE.pre_win

Field	Bits	Default	Access	Description
pre_win	1	0	R/W	Enable Watchdog Pre-Window Reset Interrupt

- 0: No pre-window reset interrupt will be triggered.
- 1: An interrupt will be triggered when the Pre-Window Reset Interrupt Flag is set to 1.

WDTn_ENABLE.reset_out

Field	Bits	Default	Access	Description
reset_out	2	0	R/W	Enable Watchdog Reset Output

- 0: No reset will be triggered by this watchdog.
- 1: A system reset (for WDT0) or system reboot (for WDT1) will be triggered when the Watchdog Reset Flag is set to 1.

10.2.5.1.5 WDTn_LOCK_CTRL

WDTn_LOCK_CTRL.wdlock

Field	Bits	Default	Access	Description
wdlock	7:0	0 (POR only)	R/W	Lock for WDT CTRL Register

- If this field reads 0, the CTRL register may be written to.
- If this field reads 1, the CTRL register is read-only.
- Writing 0x24 to this field will set bit 0 to 1 (Lock).
- Writing 0x42 to this field will clear bit 0 to 0 (Unlock).

10.3 Real Time Clock (RTC)

10.3.1 Real Time Clock Overview

The Real Time Clock (RTC) on the **MAX32600** is designed to operate largely independent of other the digital and analog functions on the device. The RTC has its own clock that runs off a 4kHz clock derived from the external crystal 32kHz oscillator output. While one of the main power supplies (V_{DD} or V_{BUS}) is active, the RTC

will run from that supply; however, it can automatically switch to run from the backup supply (V_{RTC}) if the main supply powers down.

The RTC consumes very little power and can operate from its dedicated 4kHz clock base. It is, therefore, possible for the RTC to continue operating while the rest of the **MAX32600** is in the lowest power saving mode, **LP0: STOP**. The RTC can be used to wake the device automatically from **LP0: STOP** or **LP1: STANDBY** after a preprogrammed time interval (using the time of day alarm function). It can also be used to maintain a stable time value that continues to keep counting time properly even when the rest of the system has been powered down or power has been removed entirely. As long as the backup supply V_{RTC} is present, the Real Time Clock can continue operating normally.

Note Reference [Power Ecosystem](#) for further information about **MAX32600** power modes.

10.3.1.1 Real Time Clock Features

The Real Time Clock on the **MAX32600** includes the following features:

- Dedicated low-frequency, low-power 4kHz clock source (derived from 32kHz external crystal oscillator)
- Continued operation when main portion of system is powered off
 - Automatic switchover to operate from either main power supply or V_{RTC} backup power supply
- 32-bit RTC timer
- Integrated prescaler determines interval between RTC timer ticks: from 244 microseconds (fastest RTC tick rate) to eight seconds (slowest RTC tick rate)
- Automatic synchronization of timers, configuration registers, and status/interrupt flags between high-frequency and 4kHz clock domains
- Two separate 32-bit timer compare registers allow a wakeup and/or interrupt event to occur when the RTC timer reaches a predetermined alarm value
- Integrated prescaler compare mask allows a wakeup and/or interrupt event to occur at a regular sub-RTC-tick interval (subsecond/interval alarm)

10.3.2 RTC Resets

The standard System Reset and Power-on Reset (POR) sources do not halt the operation of the RTC nor clear its register contents. Once the RTC has been configured properly and the timer is running, the RTC will continue normal operation during System Reset and/or POR conditions.

Since the RTC has its own backup power supply (V_{RTC}), it will only reset in the event that power fails on both the main supply (both V_{DD} and V_{BUS}) and the V_{RTC} supply. In this case, the RTC will be reset and all associated RTC registers will be cleared to the default state. In the register description sections, this reset condition is listed as "RTC POR".

The RSTN power sequencer reset pin does not affect the operation of the RTC and does not clear any **RTCTMR** or **RTCCFG** register contents.

Simply resetting the RTC oscillator (e.g., setting the **RTCCFG_OSC_CTRL.osc_bypass** to 1) does not reset the RTC as a whole nor clear RTC register contents.

The RTC as a whole will be reset when it exits a shutdown state; the RTC may be shut down during certain power modes as controlled by the power sequencer. See information in [Reset Pins](#) for further details.

10.3.3 RTC Interrupts

The RTC module reports interrupts to the CPU core using the following interrupt vector channels.

Interrupt Number	Vector	Interrupt Source	Description
26	0xa8	RTC Interrupt 0	RTC Time of Day Alarm 0 Interrupt (match between COMP0 and RTC timer)
27	0xac	RTC Interrupt 1	RTC Time of Day Alarm 1 Interrupt (match between COMP1 and RTC timer)
28	0xb0	RTC Interrupt 2	RTC Subsecond/Interval Alarm Interrupt (masked PRESCALE matches zero)
29	0xb4	RTC Interrupt 3	RTC Timer Overflow Interrupt, RTC Trim Adjust Interrupt

Although the 4kHz clock domain interrupt flag can be set while the main system is powered off (e.g., if a time of day alarm is matched), main power must be present in order for the system to wake up and service the interrupt. The CPU cannot operate from the V_{RTC} backup supply without the main 3V supply (V_{DD} and/or V_{BUS}) active.

RTC interrupts to the ARM core are set by the [CLKMAN_CLK_CTRL_13_RTC_INT_SYNC.rtc_clk_scale](#) register. By default, RTC interrupts are disabled. To enable CPU interrupts generated from the RTC block, the register needs to be set to a non-zero number.

rtc_clk_scale write	RTC Interrupt Clock Sync Setting
0000b	CLK is Disabled (Default)
0001b	CLK = (System Clock Source / 1)
0010b	CLK = (System Clock Source / 2)
0011b	CLK = (System Clock Source / 4)
0100b	CLK = (System Clock Source / 8)
0101b	CLK = (System Clock Source / 16)
0110b	CLK = (System Clock Source / 32)
0111b	CLK = (System Clock Source / 64)
1000b	CLK = (System Clock Source / 128)
1001b	CLK = (System Clock Source / 256)

<code>rtc_clk_scale</code> write	RTC Interrupt Clock Sync Setting
other	CLK = (System Clock Source / 1)

Note This clock is disabled by default following reset.

10.3.4 RTC Configuration

In order for firmware to configure the RTC for a desired application and start the operation of the RTC, the following operations must be performed:

1. The 32kHz crystal oscillator (which provides the time base for the RTC module) must be started by powering on the RTC block.
2. The clock for the RTC synchronizers (which translate signals between the 4kHz and high-frequency clock domains) must be started for CPU interrupts to function properly.
3. *OPTIONAL*: If the RTC digital trim function will be used, the settings for the trim must be loaded with the `RTCTMR_TRIM_VALUE` register. The RTC prescaler must be configured to determine the time period represented by the LSB of the RTC timer. The prescaler must be within 0x3 and 0xC.
4. The RTC can now be enabled by setting `RTCTMR_CTRL.enable` bit to 1.

Turning on the RTC Block and 32kHz Crystal Oscillator

By default, the RTC block and the 32kHz crystal oscillator are disabled following an RTC POR event.

To enable RTC operation during `LP2: PMU` and `LP3: RUN`, set the power sequencer control bit `PWRSEQ_REG0.pwr_rtcent_run` to 1. Alternately, write 1 to `PWRS-EQ_REG0.pwr_rtcent_slp` to enable RTC operation during `LP0: STOP` and `LP1: STANDBY`. This will power on the RTC block and, based on the RTC POR default setting of `RTCCFG_OSC_CTRL`, the RTC oscillator will be enabled to run as well.

Set the RTC timer to a zero starting value

- Write `RTCTMR_TIMER` to 00000000h.
- The `RTCTMR_CTRL.pending` status bit will change to 1 to indicate that a clock domain synchronization is pending; the bit will change back to 0 once the synchronization has completed.
 - *Note*: It is not necessary to wait for the pending bit to go low before writing to other RTC registers; this pending bit is most useful to signal when it is permitted to go to LP0 or LP1.

If the RTC Time of Day Alarm will be used, set the appropriate compare register(s)

- Write `RTCTMR_COMP0` or `RTCTMR_COMP1` to the desired compare value.
- Perform a clear operation on one of the flags in `RTCTMR_FLAGS` to ensure the register write just performed is synchronized correctly.
- The `RTCTMR_CTRL.pending` status bit will change to 1 to indicate that clock domain synchronization is pending; the bit will change back to 0 once the synchronization has completed.

Note After writing to any of the following registers: `RTCTMR_COMP0`, `RTCTMR_COMP1`, or `RTCTMR_TRIM_VALUE`, in order to ensure that the synchronization to the 4kHz clock domain completes correctly, it is necessary to follow the register write with a separate write of a '1' value to the `RTCTMR_FLAGS` register. (Any value can be used that results in a clear or attempted clear of any of the flags in bits [4:0] of this register, even if none of the flags are set; the write is merely needed to trigger the synchronization process.)

Set the prescaler to the desired setting

- Write the `RTCTMR_PRESCALE.width_selection` field to the desired clock rate prescale setting.
- The `RTCTMR_CTRL.pending` status bit will change to 1 to indicate that clock domain synchronization is pending; the bit will change back to 0 once the synchronization has completed.

Determining the Proper RTC Timer Prescale Value

The RTC timer register is always a fixed 32-bits in length; the input to the RTC block is always a fixed 4kHz frequency derived from the output of the 32kHz external crystal oscillator. However, by using the prescaler, it is possible to change the rate at which the RTC timer increments, effectively determining the unit of time assigned to the RTC timer LSB.

As shown in the table below, the assigned prescale value determines the number of 4kHz clock ticks that are needed in order to increment the main RTC timer register by one.

PRESCALE	Prescale Reload	4kHz ticks in LSB	Min Timer Value (s)	Max Timer Value (s)	Max Timer Value in Days	Max Timer Value in Years
0h	000h	1	0.00024	1048576	12	0.0
1h	001h	2	0.00049	2097152	24	0.1
2h	003h	4	0.00098	4194304	49	0.1
3h	007h	8	0.00195	8388608	97	0.3
4h	00Fh	16	0.00391	16777216	194	0.5

PRESCALE	Prescale Reload	4kHz ticks in LSB	Min Timer Value (s)	Max Timer Value (s)	Max Timer Value in Days	Max Timer Value in Years
5h	01Fh	32	0.00781	33554432	388	1.1
6h	03Fh	64	0.01563	67108864	777	2.2
7h	07Fh	128	0.03125	134217728	1553	4.4
8h	0FFh	256	0.06250	268435456	3107	8.7
9h	1FFh	512	0.12500	536870912	6214	17.5
Ah	3FFh	1024	0.25000	1073741824	12428	34.9
Bh	7FFh	2048	0.50000	2147483648	24855	69.8
Ch	FFFh	4096	1.00000	4294967296	49710	139.6

For each `RTCTMR_PRESCALE` setting, the Prescale Reload indicates the reload value that will be loaded to the RTC's internal 12-bit prescaler counter when the RTC timer is first started or when the prescaler counter reaches zero. On each 4kHz tick, the prescaler counter will either be reset with the reload value (if the prescaler counter equals zero) or it will be decremented by one (if the prescaler counter is nonzero). Each time the prescaler counter 'rolls under' at zero and is reloaded, the main RTC timer will be incremented by one.

The `PRESCALE==0h` setting represents the minimum prescale reload value, which means that the RTC timer will be incremented each 4kHz clock period. This is also reflected in the value in the '4kHz ticks in LSB' column above. Setting `PRESCALE==1h` will then give the RTC timer a resolution of (2/4kHz) and a tick frequency of 2048Hz, etc.

Setting `PRESCALE=0h` gives the RTC timer its fastest possible frequency (4096Hz), causing it to reach its maximum value of FFFFFFFFh and roll over in 1048576 seconds ($2^{32} / 4096$), or approximately 12 days. Setting the maximum prescale value of `PRESCALE=Ch` results in the slowest possible RTC timer frequency of 1Hz, causing it to reach its maximum value of FFFFFFFFh and roll over in 2^{32} seconds, or about 139.6 years.

The prescaler range is intentionally wide to allow for a variety of potential applications. If the intent of the RTC for a particular application is to measure time across a long device lifetime, then a long period makes the most sense. If the RTC will instead be used to measure events and time intervals of shorter duration, then a shorter period (and a smaller prescaler value) will reduce the maximum time value the RTC can contain but will increase the effective resolution of the RTC timer to allow shorter intervals of time to be measured (or to allow RTC timer-based wakeup operations to occur more precisely).

Starting the RTC Timer

Once the RTC timer has been properly configured, it can be started by setting the `RTCTMR_CTRL.enable` bit to 1. This will cause the RTC timer to begin counting based on the selected prescaler rate.

10.3.5 Registers (RTCTMR)

10.3.5.1 Module RTCTMR Registers

Address	Register	32b Word Len	Description
0x40090A00	RTCTMR_CTRL	1	RTC Timer Control
0x40090A04	RTCTMR_TIMER	1	RTC Timer Count Value
0x40090A08	RTCTMR_COMP0	1	RTC Time of Day Alarm 0 Compare Register
0x40090A0C	RTCTMR_COMP1	1	RTC Time of Day Alarm 1 Compare Register
0x40090A10	RTCTMR_FLAGS	1	CPU Interrupt and RTC Domain Flags
0x40090A18	RTCTMR_INTEN	1	Interrupt Enable Controls
0x40090A1C	RTCTMR_PRESCALE	1	RTC Timer Prescale Setting
0x40090A24	RTCTMR_PRESCALE_MASK	1	RTC Timer Prescale Compare Mask
0x40090A28	RTCTMR_TRIM_CTRL	1	RTC Timer Trim Controls
0x40090A2C	RTCTMR_TRIM_VALUE	1	RTC Timer Trim Adjustment Interval

10.3.5.1.1 RTCTMR_CTRL

RTCTMR_CTRL.enable

Field	Bits	Default	Access	Description
enable	0	0 (RTC POR only)	R/W	RTC Timer Enable

- 0: RTC Timer increment is disabled
- 1: RTC Timer increments normally (running)

RTCTMR_CTRL.clear

Field	Bits	Default	Access	Description
clear	1	s	W/O	RTC Timer Clear Bit

Write to 1 to clear the RTC timer.

Always reads 0.

RTCTMR_CTRL.pending

Field	Bits	Default	Access	Description
pending	2	s	R/O	RTC Transaction Pending

- 0: No transactions are pending
- 1: One or more RTC transactions are pending as indicated by the associated `_active` flags in this register.

RTCTMR_CTRL.use_async_flags

Field	Bits	Default	Access	Description
use_async_flags	3	0 (RTC POR only)	R/W	Use Async RTC Flags

- 0: (default) Use flags synchronized to the core clock. This is redundant because they will get synchronized again by the digital core.
- 1: Connect flags synchronized to the RTC clock directly to the core, since they will be re-synchronized by the core. (This should have been the default setting but currently is not.)

RTCTMR_CTRL.aggressive_rst

Field	Bits	Default	Access	Description
aggressive_rst	4	0 (RTC POR only)	R/W	Use Aggressive Reset Mode

- 0: (default) When resetting all of the RTC flags using the `async_clear_flags` bit, the reset to the flags will be released on the next edge of the 4kHz RTC clock. In other words, the system clock is used to assert the reset to the flags but the release of the reset is synchronized to the 4kHz RTC clock.
- 1: When resetting all of the RTC flags using `async_clear_flags`, the reset to the flags will be released without regards to the 4kHz RTC clock. In other words, the system clock is used to issue a completely unsynchronized reset pulse to the flags.

The purpose of this bit, `aggressive_rst`, is to allow a user to have control of the release of the reset to the flags when using `async_clear_flags`. This is to accommodate two schools of thought, 1) never release reset on a clock edge and 2) don't care. User 1 may feel that the release of the reset at the same time that a clock edge occurs could cause an unstable condition and thus would not want to set this bit. User 2, who does not feel that this situation would ever be an issue and needs the reset to be released quickly so that the flags will be operational for the next 4KHz RTC clock edge, would want to set this bit.

RTCTMR_CTRL.rtc_enable_active

Field	Bits	Default	Access	Description
rtc_enable_active	16	s	R/O	tmr_en_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.osc_goto_low_active

Field	Bits	Default	Access	Description
osc_goto_low_active	17	s	R/O	osc_goto_low_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.osc_frce_sm_en_active

Field	Bits	Default	Access	Description
osc_frce_sm_en_active	18	s	R/O	osc_frce_sm_en_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.osc_frce_st_active

Field	Bits	Default	Access	Description
osc_frce_st_active	19	s	R/O	osc_frce_st_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.rtc_set_active

Field	Bits	Default	Access	Description
rtc_set_active	20	s	R/O	timer_set_active

This sync transaction with the 4kHz clock domain occurs when a new timer value is written to the [RTCTMR_TIMER](#) register.

Reads 1 when the associated transaction is pending.

RTCTMR_CTRL.rtc_clr_active

Field	Bits	Default	Access	Description
rtc_clr_active	21	s	R/O	timer_clr_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.rollover_clr_active

Field	Bits	Default	Access	Description
rollover_clr_active	22	s	R/O	rollover_clr_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.prescale_cmpr0_active

Field	Bits	Default	Access	Description
prescale_cmpr0_active	23	s	R/O	prescale_cmpr0_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.prescale_update_active

Field	Bits	Default	Access	Description
prescale_update_active	24	s	R/O	prescale_update_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.cmpr1_clr_active

Field	Bits	Default	Access	Description
cmpr1_clr_active	25	s	R/O	cmpr1_clr_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.cmpr0_clr_active

Field	Bits	Default	Access	Description
cmpr0_clr_active	26	s	R/O	cmpr0_clr_active

Reads 1 when the associated transaction is pending

10.3.5.1.2 RTCTMR_TIMER

Default	Access	Description
00000000h (RTC POR only)	R/W	RTC Timer Count Value

This timer value is synchronized with TIMER_A, which is the RTC timer counter in the 4kHz clock domain.

10.3.5.1.3 RTCTMR_COMP

Default	Access	Description
FFFFFFFFh (RTC POR only)	R/W	RTC Time of Day Alarm [0..1] Compare Register

The compare value that causes the COMP{0,1}_FLAG to go active high when COMP{0,1}[31:0] matches TIMER_A[31:0].

10.3.5.1.4 RTCTMR_COMP0

Default	Access	Description
FFFFFFFFh (RTC POR only)	R/W	RTC Time of Day Alarm 0 Compare Register

This value is compared against the RTC timer value in the 4kHz clock domain. When a match occurs, the comp0_flag_a is set.

10.3.5.1.5 RTCTMR_COMP1

Default	Access	Description
FFFFFFFFh (RTC POR only)	R/W	RTC Time of Day Alarm 1 Compare Register

This value is compared against the RTC timer value in the 4kHz clock domain. When a match occurs, the comp1_flag_a is set.

10.3.5.1.6 RTCTMR_FLAGS

RTCTMR_FLAGS.comp0

Field	Bits	Default	Access	Description
comp0	0	0 (RTC POR only)	W1C	RTC Compare 0 Interrupt Status

Write 1 to clear.

Set to 1 by hardware when a match occurs between the RTC timer and the Compare Value 0.

RTCTMR_FLAGS.comp1

Field	Bits	Default	Access	Description
comp1	1	0 (RTC POR only)	W1C	RTC Compare 1 Interrupt Status

Write 1 to clear.

Set to 1 by hardware when a match occurs between the RTC timer and the Compare Value 0.

RTCTMR_FLAGS.prescale_comp

Field	Bits	Default	Access	Description
prescale_comp	2	0 (RTC POR only)	W1C	RTC Prescale Compare Int Status

Write 1 to clear.

Set to 1 by hardware when a match occurs between the prescale counter and zero (bits compared determined by the prescale compare mask register).

RTCTMR_FLAGS.overflow

Field	Bits	Default	Access	Description
overflow	3	0 (RTC POR only)	W1C	RTC Overflow Interrupt Status

Write 1 to clear.

Set to 1 by hardware when the RTC Timer overflows.

RTCTMR_FLAGS.trim

Field	Bits	Default	Access	Description
trim	4	0 (RTC POR only)	W1C	RTC Trim Interrupt Status

Write 1 to clear.

Set to 1 by hardware when a trim adjustment event occurs.

RTCTMR_FLAGS.comp0_flag_a

Field	Bits	Default	Access	Description
comp0_flag_a	8	0 (RTC POR only)	R/O	RTC Compare 0 4kHz Flag

Original event detection flag from 4kHz domain.

RTCTMR_FLAGS.comp1_flag_a

Field	Bits	Default	Access	Description
comp1_flag_a	9	0 (RTC POR only)	R/O	RTC Compare 1 4kHz Flag

Original event detection flag from 4kHz domain.

RTCTMR_FLAGS.prescl_flag_a

Field	Bits	Default	Access	Description
prescl_flag_a	10	0 (RTC POR only)	R/O	RTC Prescale Compare 4kHz Flag

Original event detection flag from 4kHz domain.

RTCTMR_FLAGS.overflow_flag_a

Field	Bits	Default	Access	Description
overflow_flag_a	11	0 (RTC POR only)	R/O	RTC Overflow 4kHz Flag

Original event detection flag from 4kHz domain.

RTCTMR_FLAGS.trim_flag_a

Field	Bits	Default	Access	Description
trim_flag_a	12	0 (RTC POR only)	R/O	RTC Trim Event 4kHz Flag

Original event detection flag from 4kHz domain.

RTCTMR_FLAGS.async_clr_flags

Field	Bits	Default	Access	Description
async_clr_flags	31	0	W/O	Asynchronous RTC Flag Clear

Writing a 1 to this bit triggers a one-shot operation which clears the RTC Compare 0/1 flags and the RTC Prescale Compare flag both in this register and in the 4kHz clock domain.

This bit always reads 0.

10.3.5.1.7 RTCTMR_INTEN

RTCTMR_INTEN.comp0

Field	Bits	Default	Access	Description
comp0	0	0	R/W	RTC Time of Day Alarm (Compare 0) Interrupt Enable

- 0: Disabled.
- 1: The RTC Time of Day Alarm 0 Interrupt is enabled.

RTCTMR_INTEN.comp1

Field	Bits	Default	Access	Description
comp1	1	0	R/W	RTC Time of Day Alarm (Compare 1) Interrupt Enable

- 0: Disabled.
- 1: The RTC Time of Day Alarm 1 Interrupt is enabled.

RTCTMR_INTEN.prescale_comp

Field	Bits	Default	Access	Description
prescale_comp	2	0	R/W	RTC Prescale Compare Int Enable

- 0: Disabled.
- 1: The RTC Prescale Compare (Interval) Interrupt is enabled.

RTCTMR_INTEN.overflow

Field	Bits	Default	Access	Description
overflow	3	0	R/W	RTC Overflow Interrupt Enable

- 0: Disabled.
- 1: The RTC Overflow Interrupt is enabled.

RTCTMR_INTEN.trim

Field	Bits	Default	Access	Description
trim	4	0	R/W	RTC Trim Adjust Event Interrupt Enable

- 0: Disabled.
- 1: The RTC Trim Adjust Event Interrupt is enabled.

10.3.5.1.8 RTCTMR_PRESCALE

RTCTMR_PRESCALE.prescale

Field	Bits	Default	Access	Description
prescale	3:0	0000b (RTC POR only)	R/W	RTC Timer Prescale Setting

This field selects the initial value that is reloaded into the RTC timer prescale counter each time the prescale counter reaches 0 (or when the RTC timer is first initialized). The larger the initial prescale value, the slower the overall RTC timer will run.

- 0000b: 000h (RTC timer runs at 4kHz)
- 0001b: 001h (2kHz)
- 0010b: 003h (1kHz)
- 0011b: 007h (512Hz)
- 0100b: 00Fh (256Hz)
- 0101b: 01Fh (128Hz)
- 0110b: 03Fh (64Hz)
- 0111b: 07Fh (32Hz)
- 1000b: 0FFh (16Hz)
- 1001b: 1FFh (8Hz)
- 1010b: 3FFh (4Hz)
- 1011b: 7FFh (2Hz)
- 1100b: FFFh (1Hz)

10.3.5.1.9 RTCTMR_PRESCALE_MASK

RTCTMR_PRESCALE_MASK.prescale_mask

Field	Bits	Default	Access	Description
prescale_mask	3:0	0000b (RTC POR only)	R/W	RTC Timer Prescale Compare Mask

This mask field determines which bits of the prescale counter will be checked against a zero value; 1 bits cause a check to occur.

- 0000b: 0000h - mask setting
- 0001b: 0001h
- 0010b: 0003h
- 0011b: 0007h
- 0100b: 000Fh
- 0101b: 001Fh
- 0110b: 003Fh
- 0111b: 007Fh
- 1000b: 00FFh
- 1001b: 01FFh
- 1010b: 03FFh
- 1011b: 07FFh
- 1100b: 0FFFh

10.3.5.1.10 RTCTMR_TRIM_CTRL

RTCTMR_TRIM_CTRL.trim_enable_r

Field	Bits	Default	Access	Description
trim_enable_r	0	0 (RTC POR only)	R/W	Enable RTL Trim of RTC Timer

- 0: Trim disabled
- 1: Trim enabled

RTCTMR_TRIM_CTRL.trim_faster_ovr_r

Field	Bits	Default	Access	Description
trim_faster_ovr_r	1	0 (RTC POR only)	R/W	Force RTC Trim to Faster

- 0: Disabled; trim direction controlled by high bit of trim value.
- 1: Force trim to always occur in faster direction.

RTCTMR_TRIM_CTRL.trim_slower_r

Field	Bits	Default	Access	Description
trim_slower_r	2	0 (RTC POR only)	R/O	RTC Trim Direction Status

- 0: Trim adjustments will cause RTC to count faster.
- 1: Trim adjustments will cause RTC to count slower.

10.3.5.1.11 RTCTMR_TRIM_VALUE

RTCTMR_TRIM_VALUE.trim_value

Field	Bits	Default	Access	Description
trim_value	17:0	18'b0 (RTC POR only)	R/W	Trim PPM Value

Only bits 17:8 are writeable; bits 7:0 always read 0.

RTC_TRIM_VALUE[17:0] = 1,000,000/PPM (PPM is the target correction). Minimum PPM adjustment is 4; maximum PPM adjustment is 125.

RTCTMR_TRIM_VALUE.trim_slower_control

Field	Bits	Default	Access	Description
trim_slower_control	18	0 (RTC POR only)	R/W	Trim Direction

Field	Bits	Default	Access	Description
-------	------	---------	--------	-------------

- 0: (default) RTCTMR_TRIM_VALUE.trim_value[18] determines faster or slower
- 1: Trim adjustments increment prescaler (slower) regardless of RTCTMR_TRIM_VALUE.trim_value[18]

10.3.6 Registers (RTCCFG)

10.3.6.1 Module RTCCFG Registers

Address	Register	32b Word Len	Description
0x40090A70	RTCCFG_NANO_CNTR	1	Nano Oscillator Counter Read Register
0x40090A74	RTCCFG_CLK_CTRL	1	RTC Clock Control Settings
0x40090A78	RTCCFG_DSEN_CTRL	1	Dynamic Tamper Sensor Control
0x40090A7C	RTCCFG_OSC_CTRL	1	RTC Oscillator Control

10.3.6.1.1 RTCCFG_NANO_CNTR

RTCCFG_NANO_CNTR.nanoring_counter

Field	Bits	Default	Access	Description
nanoring_counter	15:0	s	R/O	Nano Oscillator Counter

Returns a value (asynchronous read) of a counter clocked by the nano oscillator output.

10.3.6.1.2 RTCCFG_CLK_CTRL

RTCCFG_CLK_CTRL.osc1_en

Field	Bits	Default	Access	Description
osc1_en	0		R/W	osc1_en

RTCCFG_CLK_CTRL.osc2_en

Field	Bits	Default	Access	Description
osc2_en	1		R/W	osc2_en

1:Enable clk_osc1_gated that goes to the dsensor. The source of this clock is clk_nano.

RTCCFG_CLK_CTRL.nano_en

Field	Bits	Default	Access	Description
nano_en	2		R/W	nano_en

10.3.6.1.3 RTCCFG_DSEN_CTRL**RTCCFG_DSEN_CTRL.dsen_disable**

Field	Bits	Default	Access	Description
dsen_disable	0	1	R/W	dsen0_dis_o

- 0: Enable
- 1: Disable (default)

10.3.6.1.4 RTCCFG_OSC_CTRL**RTCCFG_OSC_CTRL.osc_bypass**

Field	Bits	Default	Access	Description
osc_bypass	0	0 (RTC POR)	R/W	osc_bypass

- 0: No effect (default)
- 1: The RTC oscillator is held in reset.

RTCCFG_OSC_CTRL.osc_disable_r

Field	Bits	Default	Access	Description
osc_disable_r	1	0 (RTC POR)	R/W	osc_disable_r

- 0: No effect (default)
- 1: If (osc_disable_sel == 1), the RTC oscillator is held in reset.

RTCCFG_OSC_CTRL.osc_disable_sel

Field	Bits	Default	Access	Description
osc_disable_sel	2	0 (RTC POR)	R/W	osc_disable_sel

- 0: The reset state of the RTC oscillator will be controlled by the power sequencer (default).
- 1: The reset state of the RTC oscillator will be controlled by the osc_disable_r bit.

RTCCFG_OSC_CTRL.osc_disable_o

Field	Bits	Default	Access	Description
osc_disable_o	3	s	R/O	osc_disable_o

Read-only indicator:

- 0: RTC oscillator is not held in reset.
- 1: RTC oscillator is held in reset.

10.4 Timers/Counters

[Registers \(TMR\)}](#)

10.4.1 Timers/Counters Overview

The **MAX32600** supports four 32-bit reloadable timers that can be used for timing, event counting, and generation of pulse-width modulated (PWM) signals. The clock timer input (for all modes) is the ARM Cortex-M3 System Clock. Featured is a programmable prescaler with prescale values from 1 to 4096 from the system clock. Additionally, each of the four timers is able to be split into two 16-bit timers for a possible total of eight timers in the system.

In 32-bit mode, the following modes of operation are supported:

- **One-Shot Mode:** Timer counts to terminal count value then halts
- **Continuous Mode:** Timer counts to terminal count value then resets to 0 and repeats
- **Counter Mode:** counts input edges on the timer I/O pin
- **PWM Mode:** Generates PWM output waveform based on programmable frequency and duty cycle
- **Capture Mode:** Captures a snapshot of the current timer count value based on an input edge of the timer I/O pin
- **Compare Mode:** Toggles output on timer I/O pin when the timer count exceeds the programmable terminal count
- **Gated Mode:** Timer counts only when the input on the timer I/O pin is asserted
- **Capture/Compare Mode:** Timer is enabled and begins counting when the timer I/O input is asserted; the timer count value is captured when the timer I/O input is deasserted

Note Input pin signal frequency is limited to a maximum of 1/4 the currently selected System Clock frequency.

In 16-bit mode, the following modes of operation are supported:

- **One-Shot Mode:** Timer counts to terminal count value then halts
- **Continuous Mode:** Timer counts to terminal count value then resets to 0 and repeats

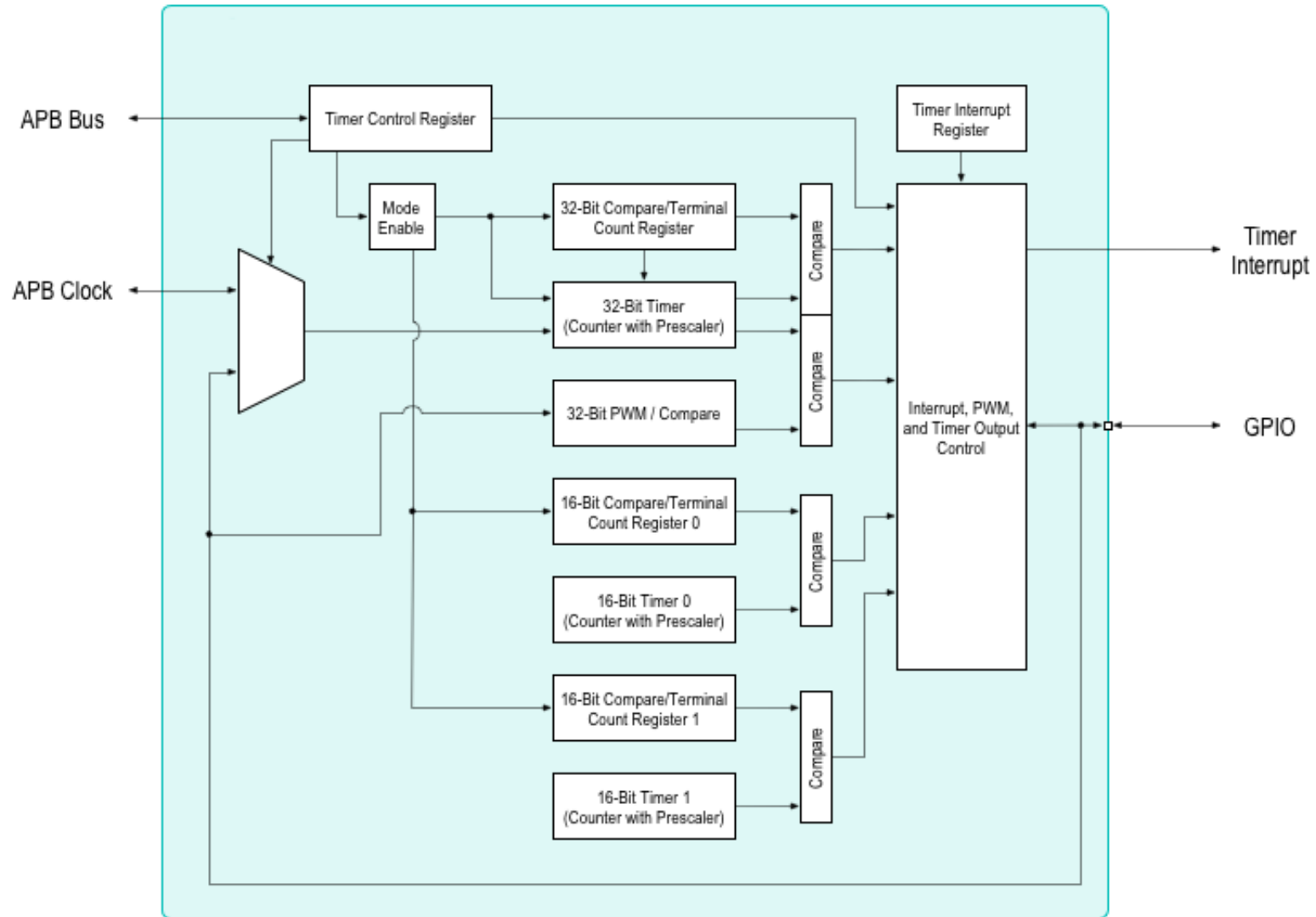


Figure 10.6: Timer Block Diagram

Timer Port and Pin Configurations

The table below contains the available pin configurations for the four timers (TMR0, TMR1, TMR2, and TMR3):

Note See [Pin Configurations, Packages, and Special Function Multiplexing](#) for a detailed mapping of **MAX32600** multiplexed function locations and priority distinction. Timer modules interface with the GPIO pins, which claim the lowest functional priority (see [GPIO Pins and Peripheral Mode Functions](#) for further information).

GPIO Function	Port and Pin
TMR0-3	P0.0-7 P1.0-7 P2.0-7 P6.0-7 P7.0-7

10.4.2 32-bit Mode Timer Operation

The timers are 32-bit up-counter timers. Use [TMRn_CTRL.mode](#) to configure the operating control register for the timer mode. Additional mode information is found below.

Minimum time-out delay is set by:

- Loading the value 0x0000_0001 into the Timer Compare register, [TMRn_TERM_CNT32](#)
- Setting the prescale value to 1 in the [TMRn_CTRL.prescale](#) field

Maximum time-out delay is set by:

- Loading the value 0x0000_0000 into the Timer Compare register, [TMRn_TERM_CNT32](#)
- Setting the prescale value to 4096 in the [TMRn_CTRL.prescale](#) field

If the timer reaches 0xFFFF_FFFF, the timer rolls over to 0x0000_0000 and continues counting.

The current count value, [TMRn_COUNT32](#), in the timers can be read while the timer is counting and enabled. This read action does not affect the timer's operation.

As a rule, the timer output is toggled every time the counter is reloaded.

10.4.2.1 One-Shot Mode

In One-Shot mode, the timer counts from the start count value stored in the [Timer register](#) up to the 32-bit Compare value stored in the [Timer Compare register](#). Upon reaching the Compare value, the timer generates an interrupt and the count value in the Timer register is reset to 0x0000_0001. Then, the timer is automatically

disabled and stops counting. Also, if the Timer Output function is enabled in the GPIO, the Timer output pin will change state for one clock cycle and then return to the polarity value (the `TMRn_CTRL.polarity` bit in the Timer Control register).

The steps for configuring a 32-bit timer for One-Shot mode and initiating the count are as follows:

1. Write the following in the `TMRn_CTRL` register:
 - Disable the timer, `TMRn_CTRL.enable0 = "0"`
 - Select 32-bit timer mode, `TMRn_CTRL.tmr2x16 = "0"`
 - Configure the timer for One-Shot Mode, `TMRn_CTRL.mode = "000"`
 - Set the prescale value, `TMRn_CTRL.prescale`
2. If using the Timer Output function, set the initial output level (High or Low) via `TMRn_CTRL.polarity` field
3. Write the starting count value, `TMRn_COUNT32`
4. Write the Compare/Terminal count value, `TMRn_TERM_CNT32`
5. If desired, enable the timer interrupt, `TMRn_INTEN.timer0`
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function
7. Enable timer and start counting, `TMRn_CTRL.enable0 = "1"`

In One-Shot mode, the Scaled System Clock always provides the timer input. The timer period is given by the following equation:

$$\text{One Shot Timeout Period}(s) = \frac{(\text{Reload} - \text{StartValue})}{\text{SystemClockFrequency}(Hz)} \times \text{Prescale}$$

10.4.2.2 Continuous Mode

In Continuous Mode, the timer counts up to the 32-bit Compare value stored in the Timer Compare register, `TMRn_TERM_CNT32`. The timer input is the Scaled System Clock. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to `0x0000_0001`, and counting resumes. Additionally, if the Timer Output function is enabled in the GPIO, the Timer Output pin changes state from Low to High or from High to Low upon reaching Timer Compare value. The steps for configuring a timer for Continuous Mode and initiating the count are as follows:

1. Write the following in the `TMRn_CTRL` register:
 - Disable the timer, `TMRn_CTRL.enable0 = "0"`
 - Select 32-bit timer mode, `TMRn_CTRL.tmr2x16 = "0"`
 - Configure the timer for Continuous Mode, `TMRn_CTRL.mode = "001"`

- Set the prescale value, `TMRn_CTRL.prescale`
2. If using the Timer Output function, set the initial output level (High or Low) via `TMRn_CTRL.polarity` field
 3. Write to the Timer Count register to set the starting count value `TMRn_COUNT32`
 - This only affects the first pass in Continuous Mode
 - After the first Timer Compare in Continuous Mode, counting always begins at the reset value of 0x0000_0001
 4. Write the Compare Count value, `TMRn_TERM_CNT32`
 5. If desired, enable the timer interrupt, `TMRn_INTEN.timer0`
 6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function

The timer period is given by the following equation:

$$\text{Continuous Timeout Period}(s) = \frac{\text{Reload}}{\text{SystemClockFrequency(Hz)}} \times \text{Prescale}$$

Note If an initial starting value other than 0x0000_0001 is loaded into the Timer register, the **One-Shot Mode** equation must be used to determine the first time-out period.

10.4.2.3 Counter Mode

In Counter Mode, the timer counts input transitions from a GPIO port pin. The timer input is taken from the GPIO port pin Timer Input function. The `TMRn_CTRL.L.polarity` bit in the Timer Control register selects whether the count occurs on the rising edge (polarity = "0") or the falling edge (polarity = "1") of the Timer Input signal.

In Counter Mode, the prescaler is disabled. Any value assigned to `TMRn_CTRL.prescale` in Counter Mode will have no effect. *The input frequency of the Timer Input signal must not exceed one-fourth the currently selected System Clock frequency.* Upon reaching the Compare value stored in the Timer Terminal Count register, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001 and counting resumes. Also, if the Timer Output function is enabled in the GPIO, the Timer Output pin changes state from Low to High or from High to Low at Timer Compare. The steps for configuring a timer for Counter Mode and initiating the count are as follows:

1. Write the following in the `TMRn_CTRL` register:
 - Disable the timer, `TMRn_CTRL.enable0 = "0"`
 - Select 32-bit timer mode, `TMRn_CTRL.tmr2x16 = "0"`
 - Configure the timer for Counter Mode, `TMRn_CTRL.mode = "010"`

- Select either the rising edge or falling edge of the Timer Input signal for the count, [TMRn_CTRL.polarity](#) (**NOTE:** This also sets the initial logic level (High or Low) for the Timer Output function; however, the Timer Output function *does not* have to be enabled via GPIO)
2. Write to the Timer register to set the starting count value, [TMRn_COUNT32](#)
 - This only affects the first pass in Counter Mode
 - After the first Timer Compare in Counter Mode, counting always begins at the reset value of 0x0000_0001
 - In Counter Mode in general, the Timer register should be written with the value 0x0000_0001
 3. Write the Compare Count value, [TMRn_TERM_CNT32](#)
 4. If desired, enable the timer interrupt, [TMRn_INTEN.timer0](#)
 5. Configure the associated GPIO port pin for the Timer Input function
 6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. (See [GPIO Pins and Peripheral Mode Functions](#) for further information.)
 7. Enable timer and start counting, [TMRn_CTRL.enable0](#) = "1"

In Counter Mode, the number of Timer Input transitions since the timer start is given by the following equation:

Counter Mode Timer Input Transition = *CurrentCountValue* – *StartValue*

10.4.2.4 PWM Mode

In PWM Mode, the timer outputs a Pulse-Width Modulated (PWM) output signal through a GPIO port pin. The timer first counts up to the 32-bit PWM compare value stored in the [TMRn_PWM_CAP32](#) register. When the timer count value matches the PWM value, the Timer Output toggles. The timer continues counting until it reaches the Compare value stored in the [TMRn_TERM_CNT32](#). Upon reaching the [TMRn_TERM_CNT32](#) value, the Timer Output signal toggles again, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001, and counting resumes.

When [TMRn_CTRL.polarity](#) = "0": Timer Output signal begins as a Low ("0") and then transitions to a High ("1") when the timer value matches the PWM [TMRn_PWM_CAP32](#) value. The Timer Output signal returns to a Low ("0") after the timer reaches the Compare value, [TMRn_TERM_CNT32](#), which is reset to 0x0000_0001. When [TMRn_CTRL.polarity](#) = "1": Timer Output signal begins as a High ("1") and then transitions to a Low ("0") when the timer value matches the PWM [TMRn_PWM_CAP32](#) Compare/Match value. The Timer Output signal returns to a High ("1") after the timer reaches the Compare value, [TMRn_TERM_CNT32](#), which is reset to 0x0000_0001.

The steps for configuring a timer for PWM mode and initiating the PWM operation are as follows:

1. Write the following in the [TMRn_CTRL](#) register:
 - Disable the timer, [TMRn_CTRL.enable0](#) = "0"

- Select 32-bit timer mode, `TMRn_CTRL.tmr2x16 = "0"`
 - Configure the timer for PWM Mode, `TMRn_CTRL.mode = "011"`
 - Set the prescale value, `TMRn_CTRL.prescale`
 - Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output function, `TMRn_CTRL.polarity`.
2. Write to the Timer register, `TMRn_COUNT32`, to set the starting count value (typically 0x0000_0001)
 - This only affects the first pass in PWM mode
 - This value is only used on the initial PWM signal pass when the PWM starts
 - After the first timer reset in PWM mode, counting always begins at the reset value of 0x0000_0001
 3. Write to the PWM Compare register, `TMRn_PWM_CAP32`, to set the desired match value
 4. Write to the Timer Compare/Match register, `TMRn_TERM_CNT32`, to set the Compare value
 - **Note:** The Compare value must be greater than the PWM value
 5. If desired, enable the timer interrupt `TMRn_INTEN.timer0`
 6. Configure the associated GPIO port pin for the Timer Output function. (See [GPIO Pins and Peripheral Mode Functions](#) for further information.)
 7. Enable timer and start counting, `TMRn_CTRL.enable0 = "1"`

The PWM period is given by the following equation:

$$\text{PWM Period}(s) = \frac{\text{ReloadValue}}{\text{SystemClockFrequency}(Hz)} \times \text{Prescale}$$

Starting count values other than 0x0000_0001

If an initial starting value other than 0x0000_0001 is loaded into the Timer register, the [One-Shot Mode](#) equation must be used to determine the first PWM time-out period.

If `TMRn_CTRL.polarity` is set to "0", the ratio of the PWM output High time to the total period is given by:

$$\text{PWM Output High Time Ratio}(\%) = \frac{(\text{ReloadValue} - \text{PWMValue})}{\text{ReloadValue}} \times 100$$

If `TMRn_CTRL.polarity` is set to "1", the ratio of the PWM output High time to the total period is given by:

$$\text{PWM Output High Time Ratio}(\%) = \frac{\text{PWMValue}}{\text{ReloadValue}} \times 100$$

10.4.2.5 Capture Mode

In Capture Mode, the current timer count value is recorded when the desired external Timer Input transition occurs. The Capture Mode count value is stored in the Timer PWM register. The `TMRn_CTRL.polarity` bit in the Timer Control register determines if the Capture occurs on a rising edge (`TMRn_CTRL.polarity = "0"`) or a falling edge (`TMRn_CTRL.polarity = "1"`) of the Timer Input signal. When the Capture event occurs, an interrupt is generated, and the timer continues counting. The timer continues counting up to the 32-bit compare value stored in the Timer Compare/Match register, `TMRn_TERM_CNT32`. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer count register `TMRn_COUNT32` is reset to `0x0000_0001`, and counting resumes.

The steps for configuring a timer for Capture Mode and initiating the count are as follows:

1. Write the following in the `TMRn_CTRL` register:
 - Disable the timer, `TMRn_CTRL.enable0 = "0"`
 - Select 32-bit timer mode, `TMRn_CTRL.tmr2x16 = "0"`
 - Configure the timer for Capture Mode, `TMRn_CTRL.mode = "100"`
 - Set the prescale value, `TMRn_CTRL.prescale`
 - Set the Capture edge (rising or falling) for the Timer Input, `TMRn_CTRL.polarity`
2. Write to the Timer Count register, `TMRn_COUNT32`, to set the starting count value, typically `0x0000_0001`
3. Write to the Timer Compare register, `TMRn_TERM_CNT32`, to set the Compare value
4. If desired, enable the timer interrupt, `TMRn_INTEN.timer0`
5. Configure the associated GPIO port pin for the Timer Input function
6. Enable timer and start counting, `TMRn_CTRL.enable0 = "1"`

In Capture mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time}(s) = \frac{(\text{CaptureValue} - \text{StartValue})}{\text{SystemClockFrequency}(Hz)} \times \text{Prescale}$$

10.4.2.6 Compare Mode

In Compare Mode, the timer counts up to the compare value stored in the Timer Compare register. Upon reaching the compare value, the timer generates an interrupt and counting continues (the timer value is not reset to `0x0000_0001`). Also, if the Timer Output function is enabled (in the GPIO), the Timer Output pin changes state (from Low to High or from High to Low) upon a Compare Match. When the Timer reaches `0xFFFF_FFFF`, the timer rolls over to `0x0000_0000` and continues counting. The steps for configuring a timer for Compare mode and initiating the count are as follows:

1. Write the following to the `TMRn_CTRL` register:

- Disable the timer, `TMRn_CTRL.enable0 = "0"`
 - Select 32-bit timer mode, `TMRn_CTRL.tmr2x16 = "0"`
 - Configure the timer for Compare Mode, `TMRn_CTRL.mode = "101"`
 - Set the prescale value, `TMRn_CTRL.prescale`
2. If using the Timer Output function, set the initial logic level (High or Low), `TMRn_CTRL.polarity`
 3. Write to the Timer Count register, `TMRn_COUNT32`, to set the starting count value
 4. Write to the Timer Compare register, `TMRn_TERM_CNT32`, to set the Compare value
 5. If desired, enable the timer interrupt, `TMRn_INTEN.timer0`
 6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. (See [GPIO Pins and Peripheral Mode Functions](#) for further information.)
 7. Enable timer and start counting, `TMRn_CTRL.enable0 = "1"`

In Compare mode, the currently selected System Clock always provides the timer input. The Compare time is given by the following equation:

$$\text{Compare Mode Time}(s) = \frac{(\text{CompareValue} - \text{StartValue})}{\text{SystemClockFrequency}(Hz)} \times \text{Prescale}$$

10.4.2.7 Gated Mode

In Gated Mode, the timer counts only when the Timer Input signal is in its active state (asserted), as determined by the `TMRn_CTRL.polarity` bit in the Timer Control register. When the Timer Input signal is asserted, counting begins. A timer interrupt is generated when the Timer Input signal is de-asserted or a Timer Compare occurs. The timer counts up to the 32-bit Compare value stored in the Timer Compare register, `TMRn_TERM_CNT32`. The timer input is the currently selected System Clock. When reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001, and counting resumes (assuming the Timer Input signal is still asserted). Also, if the Timer Output function is enabled (in the GPIO), the Timer Output pin changes state (from Low to High or from High to Low) at Timer Compare. The steps for configuring a timer for Gated Mode and initiating the count are as follows:

1. Write the following to the `TMRn_CTRL` register:
 - Disable the timer, `TMRn_CTRL.enable0 = "0"`
 - Select 32-bit timer mode, `TMRn_CTRL.tmr2x16 = "0"`
 - Configure the timer for Gated Mode, `TMRn_CTRL.mode = "110"`
 - Set the prescale value, `TMRn_CTRL.prescale`
2. Write to the Timer Count register, `TMRn_COUNT32`, to set the starting count value

- This only affects the first pass in Gated Mode
 - Counting always begins at 0x0000_0001 after the first timer reset
3. Write to the Timer Compare register, [TMRn_TERM_CNT32](#), to set the Compare value
 4. If desired, enable the timer interrupt, [TMRn_INTEN.timer0](#)
 5. Configure the associated GPIO port pin for the Timer Input function
 6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. (See [GPIO Pins and Peripheral Mode Functions](#) for further information.)
 7. Enable the timer, [TMRn_CTRL.enable0](#) = "1"
 8. Assert the Timer Input signal to initiate the counting
 - High if [TMRn_CTRL.polarity](#) = "0"
 - Low if [TMRn_CTRL.polarity](#) = "1"

10.4.2.8 Capture/Compare Mode

In Capture/Compare Mode, the timer begins counting after the first desired external Timer Input transition occurs. The selected transition (rising edge or falling edge) is set by the [TMRn_CTRL.polarity](#) bit in the Timer Control Register. The timer input is the currently selected System Clock. Every subsequent transition, after the first transition, captures the current count value. The Captured timer value is written to the Timer PWM register, [TMRn_PWM_CAP32](#). When the Capture event occurs, an interrupt is generated, the count value in the Timer register is reset to 0x0000_0001, and counting resumes. If no Capture event occurs, the timer counts up to the 32-bit Compare value stored in the Timer Compare register, [TMRn_TERM_CNT32](#). Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001, and counting resumes. The steps for configuring a timer for Capture/Compare mode and initiating the count are as follows:

1. Write the following to the [TMRn_CTRL](#) register:
 - Disable the timer, [TMRn_CTRL.enable0](#) = "0"
 - Select 32-bit timer mode, [TMRn_CTRL.tmr2x16](#) = "0"
 - Configure the timer for Capture/Compare mode, [TMRn_CTRL.mode](#) = "111"
 - Set the prescale value, [TMRn_CTRL.prescale](#)
 - Set the Timer Input Capture edge
 - Rising edge: [TMRn_CTRL.polarity](#) = "0"
 - Falling edge: [TMRn_CTRL.polarity](#) = "1"

2. Write to the Timer register to set the starting count value, typically 0x0000_0001
3. Write to the Timer Compare register, `TMRn_COUNT32`, to set the Compare value
4. If desired, enable the timer interrupt, `TMRn_INTEN.timer0`
5. Configure the associated GPIO port pin for the Timer Input function
6. Enable the timer, `TMRn_CTRL.enable0 = "1"`
7. Counting begins after the first desired transition of the Timer Input signal
 - No interrupt is generated by this first edge.

In Capture/Compare mode, the elapsed time from timer start to the Capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time}(s) = \frac{(\text{CaptureValue} - \text{StartValue}) \times \text{Prescale}}{\text{SystemClockFrequency}(Hz)}$$

10.4.3 16 bit Mode Timer Operation

Each of the four 32-bit timers on the **MAX32600** can be split into 2 x 16-bit timers, for a total of up to eight 16-bit timers. Configuration and operation of 16-bit mode is very similar to the 32-bit modes of operation, but when configured as two 16-bit timers, only One-Shot and Continuous Modes of operation are supported.

Minimum time-out delay for each 16-bit timer is set by:

- Loading the value 0x0001 into the Timer Compare register, either the `TMRn_TERM_CNT16_0` or `TMRn_TERM_CNT16_1`
- Setting the prescale value to 1 in the `TMRn_CTRL.prescale` field

Maximum time-out delay is set by:

- Loading the value 0x0000 into the Timer Compare register, `TMRn_TERM_CNT16_0` or `TMRn_TERM_CNT16_1`
- Setting the prescale value to 4096 in the `TMRn_CTRL.prescale`
- If the Timer reaches 0xFFFF, the timer rolls over to 0x0000 and continues counting

The current count value, `TMRn_COUNT16_0.value` or `TMRn_COUNT16_1.value`, in the timers can be read while the timer is counting and enabled. This action does not affect the timer's operation.

As a general rule, the timer output is toggled every time the counter is reloaded.

10.4.3.1 One-Shot Mode

In One-Shot mode, the timer counts from the start count value stored in the Timer register up to the 16-bit Compare value stored in the Timer Compare register. Upon reaching the Compare value, the timer generates an interrupt, and the count value in the Timer register is reset to 0x0001; the timer is then automatically disabled and stops counting.

The steps for configuring a 16-bit timer for One-Shot mode and initiating the count are:

1. Write the following to the `TMRn_CTRL` register:
 - Disable the timer, `TMRn_CTRL.enable0` or `TMRn_CTRL.enable1` = "0"
 - Select the dual 16-bit timer mode, `TMRn_CTRL.tmr2x16` = "1"
 - Configure the timer for One-Shot Mode, `TMRn_CTRL.mode` = "000"
 - Set the prescale value, `TMRn_CTRL.prescale`
2. Write the starting count value, `TMRn_COUNT16_0.value` or `TMRn_COUNT16_1.value`, depending on which one of the 16-bit timers is being used
3. Write the Compare/Terminal count value, `TMRn_TERM_CNT16_0.term_count` or `TMRn_TERM_CNT16_1.term_count`
4. If desired, enable the timer interrupt, `TMRn_INTEN.timer0` or `TMRn_INTEN.timer1`
5. Enable timer and start counting, `TMRn_CTRL.enable0` or `TMRn_CTRL.enable1` = "1"

In One-Shot mode, the System Clock always provides the timer input. The timer period is given by the following equation:

$$\text{One Shot Timeout Period}(s) = \frac{(\text{Reload} - \text{StartValue})}{\text{SystemClockFrequency}(Hz)} \times \text{Prescale}$$

10.4.3.2 Continuous Mode

In Continuous Mode, the timer counts up to the 16-bit Compare value stored in the Timer Compare register, `TMRn_TERM_CNT16_0` or `TMRn_TERM_CNT16_1`. The timer input is the currently selected system clock. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0x0001, and counting resumes. The steps for configuring a timer for Continuous Mode and initiating the count are as follows:

1. Write the following to the `TMRn_CTRL` register:
 - Disable the timer, `TMRn_CTRL.enable0` or `TMRn_CTRL.enable1` = "0"
 - Select 16-bit timer mode, `TMRn_CTRL.tmr2x16` = "1"
 - Configure the timer for Continuous Mode, `TMRn_CTRL.mode` = "001"
 - Set the prescale value, `TMRn_CTRL.prescale`

2. Write to the Timer register to set the starting count value, `TMRn_COUNT16_0.value` or `TMRn_COUNT16_1.value`, depending on which one of the 16-bit timers is being used
 - This only affects the first, starting, operation
 - After the first Timer Compare in Continuous Mode, counting always begins at the reset value of 0x0001
3. Write the Compare Count value, `TMRn_TERM_CNT16_0.term_count` or `TMRn_TERM_CNT16_1.term_count`
4. If desired, enable the timer interrupt, `TMRn_INTEN.timer0` or `TMRn_INTEN.timer1`
5. Enable timer and start counting, `TMRn_CTRL.enable0` or `TMRn_CTRL.enable1 = "1"`

In Continuous Mode, the currently selected System Clock always provides the timer input. The timer period is given by the following equation:

$$\text{Continuous Timeout Period}(s) = \frac{\text{ReloadValue}}{\text{SystemClockFrequency}(Hz)} \times \text{Prescale}$$

If an initial starting value other than 0x0001 is loaded into the Timer register, the [One-Shot Mode](#) equation must be used to determine the first time-out period.

10.4.4 Registers (TMR)

10.4.4.1 Module TMR Registers

Address	Register	32b Word Len	Description
0x40012000	<code>TMR0_CTRL</code>	1	Timer 0 Control Register
0x40012004	<code>TMR0_COUNT32</code>	1	Timer 0 [32 bit] Current Count Value
0x40012008	<code>TMR0_TERM_CNT32</code>	1	Timer 0 [32 bit] Terminal Count Setting
0x4001200C	<code>TMR0_PWM_CAP32</code>	1	Timer 0 [32 bit] PWM Compare Setting or Capture/Measure Value
0x40012010	<code>TMR0_COUNT16_0</code>	1	Timer 0 [16 bit] Current Count Value, 16-bit Timer 0
0x40012014	<code>TMR0_TERM_CNT16_0</code>	1	Timer 0 [16 bit] Terminal Count Setting, 16-bit Timer 0
0x40012018	<code>TMR0_COUNT16_1</code>	1	Timer 0 [16 bit] Current Count Value, 16-bit Timer 1
0x4001201C	<code>TMR0_TERM_CNT16_1</code>	1	Timer 0 [16 bit] Terminal Count Setting, 16-bit Timer 1
0x40012020	<code>TMR0_INTFL</code>	1	Timer 0 Interrupt Flags
0x40012024	<code>TMR0_INTEN</code>	1	Timer 0 Interrupt Enable/Disable Settings

Address	Register	32b Word Len	Description
0x40013000	TMR1_CTRL	1	Timer 1 Control Register
0x40013004	TMR1_COUNT32	1	Timer 1 [32 bit] Current Count Value
0x40013008	TMR1_TERM_CNT32	1	Timer 1 [32 bit] Terminal Count Setting
0x4001300C	TMR1_PWM_CAP32	1	Timer 1 [32 bit] PWM Compare Setting or Capture/Measure Value
0x40013010	TMR1_COUNT16_0	1	Timer 1 [16 bit] Current Count Value, 16-bit Timer 0
0x40013014	TMR1_TERM_CNT16_0	1	Timer 1 [16 bit] Terminal Count Setting, 16-bit Timer 0
0x40013018	TMR1_COUNT16_1	1	Timer 1 [16 bit] Current Count Value, 16-bit Timer 1
0x4001301C	TMR1_TERM_CNT16_1	1	Timer 1 [16 bit] Terminal Count Setting, 16-bit Timer 1
0x40013020	TMR1_INTFL	1	Timer 1 Interrupt Flags
0x40013024	TMR1_INTEN	1	Timer 1 Interrupt Enable/Disable Settings
0x40014000	TMR2_CTRL	1	Timer 2 Control Register
0x40014004	TMR2_COUNT32	1	Timer 2 [32 bit] Current Count Value
0x40014008	TMR2_TERM_CNT32	1	Timer 2 [32 bit] Terminal Count Setting
0x4001400C	TMR2_PWM_CAP32	1	Timer 2 [32 bit] PWM Compare Setting or Capture/Measure Value
0x40014010	TMR2_COUNT16_0	1	Timer 2 [16 bit] Current Count Value, 16-bit Timer 0
0x40014014	TMR2_TERM_CNT16_0	1	Timer 2 [16 bit] Terminal Count Setting, 16-bit Timer 0
0x40014018	TMR2_COUNT16_1	1	Timer 2 [16 bit] Current Count Value, 16-bit Timer 1
0x4001401C	TMR2_TERM_CNT16_1	1	Timer 2 [16 bit] Terminal Count Setting, 16-bit Timer 1
0x40014020	TMR2_INTFL	1	Timer 2 Interrupt Flags
0x40014024	TMR2_INTEN	1	Timer 2 Interrupt Enable/Disable Settings
0x40015000	TMR3_CTRL	1	Timer 3 Control Register
0x40015004	TMR3_COUNT32	1	Timer 3 [32 bit] Current Count Value
0x40015008	TMR3_TERM_CNT32	1	Timer 3 [32 bit] Terminal Count Setting
0x4001500C	TMR3_PWM_CAP32	1	Timer 3 [32 bit] PWM Compare Setting or Capture/Measure Value

Address	Register	32b Word Len	Description
0x40015010	TMR3_COUNT16_0	1	Timer 3 [16 bit] Current Count Value, 16-bit Timer 0
0x40015014	TMR3_TERM_CNT16_0	1	Timer 3 [16 bit] Terminal Count Setting, 16-bit Timer 0
0x40015018	TMR3_COUNT16_1	1	Timer 3 [16 bit] Current Count Value, 16-bit Timer 1
0x4001501C	TMR3_TERM_CNT16_1	1	Timer 3 [16 bit] Terminal Count Setting, 16-bit Timer 1
0x40015020	TMR3_INTFL	1	Timer 3 Interrupt Flags
0x40015024	TMR3_INTEN	1	Timer 3 Interrupt Enable/Disable Settings

10.4.4.1.1 TMRn_CTRL

TMRn_CTRL.mode

Field	Bits	Default	Access	Description
mode	2:0	000b	R/W	Operating Modes for 32-bit/16-bit Timers

For single 32-bit timer mode (tmr2x16=0):

- 000b: One Shot Mode
- 001b: Continuous Mode
- 010b: Counter Mode
- 011b: PWM Mode
- 100b: Capture Mode
- 101b: Compare Mode
- 110b: Gated Mode
- 111b: Measurement Mode

For dual 16-bit timer mode (tmr2x16=1):

Bit 0 controls operating mode for 16-bit timer 0:

- 0: One Shot Mode
- 1: Continuous Mode

Bit 1 controls operating mode for 16-bit timer 1:

- 0: One Shot Mode
- 1: Continuous Mode

Bit 2 is not used in dual 16-bit timer mode.

TMRn_CTRL.tmr2x16

Field	Bits	Default	Access	Description
tmr2x16	3	0	R/W	Dual 16-bit Timer Mode

This bit determines whether this timer instance operates as a single 32-bit timer or dual 16-bit timers.

- 0: Single 32-bit timer
- 1: Dual 16-bit timers

TMRn_CTRL.prescale

Field	Bits	Default	Access	Description
prescale	7:4	0000b	R/W	Timer Clock Prescale Setting

Determines what divide down value (if any) is used when generating the timer clock from the current system clock. This setting is used by both the 32-bit timer and by one or both of the 16-bit timers (when enabled). Disabling the 32-bit timer or both 16-bit timers automatically resets this field to 0 (div by 1).

- 0000b: Divide by 1

- 0001b: Divide by 2
- 0010b: Divide by 4
- 0011b: Divide by 8
- 0100b: Divide by 16
- 0101b: Divide by 32
- 0110b: Divide by 64
- 0111b: Divide by 128
- 1000b: Divide by 256
- 1001b: Divide by 512
- 1010b: Divide by 1024
- 1011b: Divide by 2048
- 1100b: Divide by 4096
- Other: Undefined

TMRn_CTRL.polarity

Field	Bits	Default	Access	Description
polarity	8	0	R/W	Timer I/O Polarity

Polarity control for pad I/O when the 32-bit timer is used in Counter Mode, PWM Mode, Capture Mode, Compare Mode, Gated Mode, or Measurement Mode.,

- 0: Normal polarity
- 1: Inverted polarity

TMRn_CTRL.enable0

Field	Bits	Default	Access	Description
enable0	12	0	R/W	Enable 32-bit timer / 16-bit timer 0

For single 32-bit timer mode (tmr2x16=0):

- 0: 32-bit timer is disabled
- 1: 32-bit timer is enabled

For dual 16-bit timer mode (tmr2x16=1):

- 0: 16-bit timer 0 is disabled
- 1: 16-bit timer 0 is enabled

TMRn_CTRL.enable1

Field	Bits	Default	Access	Description
enable1	13	0	R/W	Enable 16-bit timer 1

(Not used for single 32-bit timer mode)

For dual 16-bit timer mode (tmr2x16=1):

- 0: 16-bit timer 1 is disabled
- 1: 16-bit timer 1 is enabled

10.4.4.1.2 TMRn_COUNT32

Default	Access	Description
0000 0000 0000 0000 0000 0000 0000b	R/W	Timer 0 [32 bit] Current Count Value

In 32-bit timer mode ($\text{tmr2x16}=0$), this register holds the current count value for the 32-bit timer.

In 16-bit timer mode ($\text{tmr2x16}=1$), this register cannot be accessed and all reads from this location will return zero.

10.4.4.1.3 TMRn_TERM_CNT32

Default	Access	Description
0000 0000 0000 0000 0000 0000 0000b	R/W	Timer 0 [32 bit] Terminal Count Setting

In 32-bit timer mode ($\text{tmr2x16}=0$), this register holds the terminal count value for the 32-bit timer.

In 16-bit timer mode ($\text{tmr2x16}=1$), this register cannot be accessed and all reads from this location will return zero.

10.4.4.1.4 TMRn_PWM_CAP32

Default	Access	Description
0000 0000 0000 0000 0000 0000 0000b	R/W	Timer 0 [32 bit] PWM Compare Setting or Capture/Measure Value

In 32-bit timer mode ($\text{tmr2x16}=0$), this register holds the PWM compare setting or the capture/measure value (depending on the operating mode) for the 32-bit timer.

In 16-bit timer mode ($\text{tmr2x16}=1$), this register cannot be accessed and all reads from this location will return zero.

10.4.4.1.5 TMRn_COUNT16_0

TMRn_COUNT16_0.value

Field	Bits	Default	Access	Description
value	15:0	0000h	R/W	Count Value

In 16-bit timer mode ($\text{tmr2x16}=1$), this register holds the current count value for 16-bit timer 0.

In 32-bit timer mode ($\text{tmr2x16}=0$), this register cannot be accessed and all reads from this location will return zero.

10.4.4.1.6 TMRn_TERM_CNT16_0**TMRn_TERM_CNT16_0.term_count**

Field	Bits	Default	Access	Description
term_count	15:0	0000h	R/W	Terminal Count Setting

In 16-bit timer mode (tmr2x16=1), this register holds the terminal count setting for 16-bit timer 0.

In 32-bit timer mode (tmr2x16=0), this register cannot be accessed and all reads from this location will return zero.

10.4.4.1.7 TMRn_COUNT16_1**TMRn_COUNT16_1.value**

Field	Bits	Default	Access	Description
value	15:0	0000h	R/W	Count Value

In 16-bit timer mode (tmr2x16=1), this register holds the current count value for 16-bit timer 1.

In 32-bit timer mode (tmr2x16=0), this register cannot be accessed and all reads from this location will return zero.

10.4.4.1.8 TMRn_TERM_CNT16_1**TMRn_TERM_CNT16_1.term_count**

Field	Bits	Default	Access	Description
term_count	15:0	0000h	R/W	Terminal Count Setting

In 16-bit timer mode (tmr2x16=1), this register holds the terminal count setting for 16-bit timer 1.

In 32-bit timer mode (tmr2x16=0), this register cannot be accessed and all reads from this location will return zero.

10.4.4.1.9 TMRn_INTFL**TMRn_INTFL.timer0**

Field	Bits	Default	Access	Description
timer0	0	0	W1C	Interrupt Flag for 32-bit Timer / 16-bit Timer 0

Hardware sets this flag to 1 when the timer reaches the terminal count or a capture value is obtained or when the input pad is deasserted in Gated mode.

Write 1 to clear this flag to 0.

TMRn_INTFL.timer1

Field	Bits	Default	Access	Description
timer1	1	0	W1C	Interrupt Flag for 16-bit Timer 1

Hardware sets this flag to 1 when the timer reaches the terminal count or a capture value is obtained or when the input pad is deasserted in Gated mode.

Write 1 to clear this flag to 0.

10.4.4.1.10 TMRn_INTEN**TMRn_INTEN.timer0**

Field	Bits	Default	Access	Description
timer0	0	0	R/W	Interrupt Enable for 32-bit Timer / 16-bit Timer 0

Enable/disable setting to allow a timer interrupt to be triggered when the corresponding interrupt flag is set.

0: Interrupt disabled (no interrupt will be triggered) 1: Interrupt enabled (interrupt may trigger normally)

TMRn_INTEN.timer1

Field	Bits	Default	Access	Description
timer1	1	0	R/W	Interrupt Enable for 16-bit Timer 1

Enable/disable setting to allow a timer interrupt to be triggered when the corresponding interrupt flag is set.

0: Interrupt disabled (no interrupt will be triggered) 1: Interrupt enabled (interrupt may trigger normally)

11 Trust Protection Unit (TPU)

Trust Protection Unit (TPU)

The TPU on the **MAX32600** provides support for cryptographic data security and automatic response to external attacks against the device. Although the MAA and the AES engine are covered in separate sections, these are also considered to be part of the TPU as well.

Features included in the TPU:

- Physically Unclonable Feature (PUF), a block which allows a unique random number sequence to be generated, with each physical **MAX32600** part producing a different number sequence. Although the PUF output sequence produced by a particular **MAX32600** will always be repeatable by that physical device, the sequence produced by a particular **MAX32600** cannot be predicted in advance since it depends on manufacturing process variations.
- Pseudo-random number generator, allowing random numbers to be generated by combining a user-selected entropy source with internal clock-based entropy and other sources of hardware random noise.
- A dynamic Tamper Sensor (TSR) generates a randomized signal sequence that can be used to construct a physical enclosure shield that cannot be disrupted by an external attacker. If the signal path in the enclosure shield is broken, the tamper sensor triggers, causing an automatic clear of the "key of keys" registers inside the TPU and also triggering an automatic erasure of all AES memory.

11.1 AES Cryptographic Engine

The **MAX32600** includes an AES cryptographic engine which can perform AES encryption/decryption operations. Supported key lengths are 128-bits, 192-bits, and 256-bits.

11.1.1 Registers (AES)

11.1.1.1 Module AES Registers

Address	Register	32b Word Len	Description
0x40011400	AES_CTRL	1	AES Control and Status
0x40011408	AES_ERASE_ALL	1	Write to Trigger AES Memory Erase
0x4010A000	AES_MEM_INP0	1	AES Input 0 (least significant 32 bits)
0x4010A004	AES_MEM_INP1	1	AES Input 1

Address	Register	32b Word Len	Description
0x4010A008	AES_MEM_INP2	1	AES Input 2
0x4010A00C	AES_MEM_INP3	1	AES Input 3 (most significant 32 bits)
0x4010A010	AES_MEM_KEY0	1	AES Key 0 (least significant 32 bits)
0x4010A014	AES_MEM_KEY1	1	AES Key 1
0x4010A018	AES_MEM_KEY2	1	AES Key 2
0x4010A01C	AES_MEM_KEY3	1	AES Key 3
0x4010A020	AES_MEM_KEY4	1	AES Key 4
0x4010A024	AES_MEM_KEY5	1	AES Key 5
0x4010A028	AES_MEM_KEY6	1	AES Key 6
0x4010A02C	AES_MEM_KEY7	1	AES Key 7 (most significant 32 bits)
0x4010A030	AES_MEM_OUT0	1	AES Output 0 (least significant 32 bits)
0x4010A034	AES_MEM_OUT1	1	AES Output 1
0x4010A038	AES_MEM_OUT2	1	AES Output 2
0x4010A03C	AES_MEM_OUT3	1	AES Output 3 (most significant 32 bits)
0x4010A040	AES_MEM_EXPKEY0	1	AES Expanded Key Data 0
0x4010A044	AES_MEM_EXPKEY1	1	AES Expanded Key Data 1
0x4010A048	AES_MEM_EXPKEY2	1	AES Expanded Key Data 2
0x4010A04C	AES_MEM_EXPKEY3	1	AES Expanded Key Data 3
0x4010A050	AES_MEM_EXPKEY4	1	AES Expanded Key Data 4
0x4010A054	AES_MEM_EXPKEY5	1	AES Expanded Key Data 5
0x4010A058	AES_MEM_EXPKEY6	1	AES Expanded Key Data 6
0x4010A05C	AES_MEM_EXPKEY7	1	AES Expanded Key Data 7

11.1.1.1.1 AES_CTRL**AES_CTRL.start**

Field	Bits	Default	Access	Description
start	0	0	R/W	AES Start/Busy

Writing this bit to 1 initiates an AES operation.

This bit is cleared from 1 to 0 by hardware when an AES operation has completed.

AES_CTRL.crypt_mode

Field	Bits	Default	Access	Description
crypt_mode	1	0	R/W	AES Encrypt/Decrypt Mode

This field can only be written when the AES engine is idle (AES Busy == 0).

- 0: Perform AES encryption operation
- 1: Perform AES decryption operation

AES_CTRL.exp_key_mode

Field	Bits	Default	Access	Description
exp_key_mode	2	0	R/W	AES Expanded Key Mode

This field can only be written when the AES engine is idle (AES Busy == 0).

- 0: Calculate expanded key / round key as needed for this operation.
- 1: Use the expanded key / round key data that was calculated for the previous operation. Only valid until key or enc/dec mode changes.

AES_CTRL.key_size

Field	Bits	Default	Access	Description
key_size	4:3	00b	R/W	AES Key Size Select

This field can only be written when the AES engine is idle (AES Busy == 0). Size of key to use for AES operation.

- 0: 128-bit key size
- 1: 192-bit key size
- 2: 256-bit key size
- 3: Reserved

AES_CTRL.inten

Field	Bits	Default	Access	Description
inten	5	0	R/W	AES Interrupt Enable

- 0: Disable interrupts from the AES
- 1: Allow an interrupt to be triggered by bit 6.

AES_CTRL.intfl

Field	Bits	Default	Access	Description
intfl	6	0	W1C	AES Interrupt Flag

Set by hardware; write 1 to clear.

If AES Interrupt Enable is set, this bit will trigger an AES Interrupt when set to 1.

11.1.1.1.2 AES_ERASE_ALL

Default	Access	Description
0	W/O	Write to Trigger AES Memory Erase

A write to this location triggers an erase of all AES memory locations.

11.1.1.1.3 AES_MEM_INP

Default	Access	Description
0000_0000h	R/W	AES Input (128 bits)

11.1.1.1.4 AES_MEM_INP0

Default	Access	Description
0000_0000h	R/W	AES Input 0 (least significant 32 bits)

11.1.1.1.5 AES_MEM_INP1

Default	Access	Description
0000_0000h	R/W	AES Input 1

11.1.1.1.6 AES_MEM_INP2

Default	Access	Description
0000_0000h	R/W	AES Input 2

11.1.1.1.7 AES_MEM_INP3

Default	Access	Description
0000_0000h	R/W	AES Input 3 (most significant 32 bits)

11.1.1.1.8 AES_MEM_KEY

Default	Access	Description
0000_0000h	R/W	AES Symmetric Key (up to 256 bits)

11.1.1.1.9 AES_MEM_KEY0

Default	Access	Description
0000_0000h	R/W	AES Key 0 (least significant 32 bits)

11.1.1.1.10 AES_MEM_KEY1

Default	Access	Description
0000_0000h	R/W	AES Key 1

11.1.1.1.11 AES_MEM_KEY2

Default	Access	Description
0000_0000h	R/W	AES Key 2

11.1.1.1.12 AES_MEM_KEY3

Default	Access	Description
0000_0000h	R/W	AES Key 3

11.1.1.1.13 AES_MEM_KEY4

Default	Access	Description
0000_0000h	R/W	AES Key 4

11.1.1.1.14 AES_MEM_KEY5

Default	Access	Description
0000_0000h	R/W	AES Key 5

11.1.1.1.15 AES_MEM_KEY6

Default	Access	Description
0000_0000h	R/W	AES Key 6

11.1.1.1.16 AES_MEM_KEY7

Default	Access	Description
0000_0000h	R/W	AES Key 7 (most significant 32 bits)

11.1.1.1.17 AES_MEM_OUT

Default	Access	Description
0000_0000h	R/W	AES Output Data (128 bits)

11.1.1.1.18 AES_MEM_OUT0

Default	Access	Description
0000_0000h	R/W	AES Output 0 (least significant 32 bits)

11.1.1.1.19 AES_MEM_OUT1

Default	Access	Description
0000_0000h	R/W	AES Output 1

11.1.1.1.20 AES_MEM_OUT2

Default	Access	Description
0000_0000h	R/W	AES Output 2

11.1.1.1.21 AES_MEM_OUT3

Default	Access	Description
0000_0000h	R/W	AES Output 3 (most significant 32 bits)

11.1.1.1.22 AES_MEM_EXPKEY

Default	Access	Description
0000_0000h	R/W	AES Expanded Key Data (256 bits)

11.1.1.1.23 AES_MEM_EXPKEY0

Default	Access	Description
0000_0000h	R/W	AES Expanded Key Data 0

11.1.1.1.24 AES_MEM_EXPKEY1

Default	Access	Description
0000_0000h	R/W	AES Expanded Key Data 1

11.1.1.1.25 AES_MEM_EXPKEY2

Default	Access	Description
0000_0000h	R/W	AES Expanded Key Data 2

11.1.1.1.26 AES_MEM_EXPKEY3

Default	Access	Description
0000_0000h	R/W	AES Expanded Key Data 3

11.1.1.1.27 AES_MEM_EXPKEY4

Default	Access	Description
0000_0000h	R/W	AES Expanded Key Data 4

11.1.1.1.28 AES_MEM_EXPKEY5

Default	Access	Description
0000_0000h	R/W	AES Expanded Key Data 5

11.1.1.1.29 AES_MEM_EXPKEY6

Default	Access	Description
0000_0000h	R/W	AES Expanded Key Data 6

11.1.1.1.30 AES_MEM_EXPKEY7

Default	Access	Description
0000_0000h	R/W	AES Expanded Key Data 7

11.2 Modular Arithmetic Accelerator (MAA)

Registers (MAA)}

The Modular Arithmetic Accelerator (MAA) module on the **MAX32600** is used to perform rapid calculation of modular arithmetic operations (with operands up to 512-bits in length). These modular math operations can then be used as the basis for cryptographic operations such as RSA public/private key cryptography.

Operations supported by the MAA (up to 512-bits):

- Modular exponentiation
- Square
- Multiply
- Square plus Multiply
- Add
- Subtract

The MAA also includes support for big-endian or little-endian operand loading/unloading, as well as features to randomize the exact locations of input and output parameters within the MAA memory (shifting parameters within a memory segment) designed to make external analysis or attacks against the content of the MAA memory more difficult.

11.2.1 Registers (MAA)

11.2.1.1 Module MAA Registers

Address	Register	32b Word Len	Description
0x40011800	MAA_CTRL	1	MAA Control, Configuration and Status
0x40011804	MAA_MAWS	1	MAA Word (Operand) Size, Big/Little Endian Mode Select
0x4010A800	MAA_MEM_SEG0	16	[64 bytes] MAA Memory Segment 0
0x4010A840	MAA_MEM_SEG1	16	[64 bytes] MAA Memory Segment 1
0x4010A880	MAA_MEM_SEG2	16	[64 bytes] MAA Memory Segment 2
0x4010A8C0	MAA_MEM_SEG3	16	[64 bytes] MAA Memory Segment 3
0x4010A900	MAA_MEM_SEG4	16	[64 bytes] MAA Memory Segment 4
0x4010A940	MAA_MEM_SEG5	16	[64 bytes] MAA Memory Segment 5

11.2.1.1.1 MAA_CTRL

MAA_CTRL.start

Field	Bits	Default	Access	Description
start	0	0	R/W	Start MAA Calculation

- Write to 1: Start a MAA calculation.
- Write to 0: Reset MAA and stop operation.

Cleared automatically to 0 by hardware when an MAA operation has completed.

MAA_CTRL.opsel

Field	Bits	Default	Access	Description
opsel	3:1	000b	R/W	Select Operation Type

- 0: Exponentiation
- 1: Square operation
- 2: Multiply
- 3: Square followed by multiply
- 4: Addition
- 5: Subtraction

MAA_CTRL.ocalc

Field	Bits	Default	Access	Description
ocalc	4	0	R/W	Optimized Calculation Control

- 0: No optimization is used
- 1: Skip unneeded multiply operations

MAA_CTRL.if_done

Field	Bits	Default	Access	Description
if_done	5	0	R/W	Interrupt Flag - Calculation Done

Write to 0 to clear.

This bit is set to 1 by hardware when an MAA operation has completed successfully.

MAA_CTRL.inten

Field	Bits	Default	Access	Description
inten	6	0	R/W	MAA Interrupt Enable

- 0: MAA Interrupts are disabled
- 1: MAA Interrupt can be triggered on MAA calculation completed or error conditions.

MAA_CTRL.if_error

Field	Bits	Default	Access	Description
if_error	7	0	R/W	Interrupt Flag - Error

Write to 0 to clear.

This bit is set to 1 by hardware when an MAA operation is terminated by an error condition.

MAA_CTRL ofs_a

Field	Bits	Default	Access	Description
ofs_a	9:8	00b	R/W	Operand A Memory Offset Select

These bits select the starting position (offset) of the 'A' parameter within the selected logical segment.

For (MAA Word Size) from 1..256

- x0b: No offset
- x1b: Offset by +16 (increase 0x10) bytes

For (MAA Word Size) from 257..512

- 00b: No offset
- 01b: Offset by +16 (increase 0x10) bytes
- 10b: Offset by +32 (increase 0x20) bytes
- 11b: Offset by +48 (increase 0x30) bytes

MAA_CTRL.ofs_b

Field	Bits	Default	Access	Description
ofs_b	11:10	00b	R/W	Operand B Memory Offset Select

These bits select the starting position of the 'B' parameter within the selected logical segment. (Same as "A" Memory Offset Select)

MAA_CTRL.ofs_exp

Field	Bits	Default	Access	Description
ofs_exp	13:12	00b	R/W	Exponent Memory Offset Select

These bits select the starting position of the 'E' parameter within its logical segment. (Same as "A" Memory Offset Select)

MAA_CTRL.ofs_mod

Field	Bits	Default	Access	Description
ofs_mod	15:14	00b	R/W	Modulus Memory Select

These bits select the starting position of the 'M' parameter within its logical segment. (Same as "A" Memory Offset Select)

MAA_CTRL.seg_a

Field	Bits	Default	Access	Description
seg_a	19:16	0000b	R/W	Operand A Memory Segment Select

These bits select the memory segment for 'A'.

For (MAA Word Size) from 1..256

- 0000b: xA800..xA83F
- 0001b: xA820..xA83F, xA800..xA81F
- 0010b: xA840..xA87F
- 0011b: xA860..xA87F, xA840..xA85F
- 0100b: xA880..xA8BF
- 0101b: xA8A0..xA8BF, xA880..xA89F
- 0110b: xA8C0..xA8FF
- 0111b: xA8E0..xA8FF, xA8C0..xA8DF
- 1000b: xA900..xA93F
- 1001b: xA920..xA93F, xA900..xA91F
- other: reserved

For (MAA Word Size) from 257..512

- 000xb: xA800..xA83F
- 001xb: xA840..xA87F
- 010xb: xA880..xA8BF
- 011xb: xA8C0..xA8FF
- 100xb: xA900..xA93F
- other: reserved

Note that segment 4 values (xA900, xA920) cannot be used for operations when an exponent is needed.

MAA_CTRL.seg_b

Field	Bits	Default	Access	Description
seg_b	23:20	0000b	R/W	Operand B Memory Segment Select

These bits select the memory segment for 'B'. (Same as select settings for Operand A)

MAA_CTRL.seg_res

Field	Bits	Default	Access	Description
seg_res	27:24	0000b	R/W	Result Memory Segment Select

These bits select the memory segment for the results output. (Same as select settings for Operand A)

MAA_CTRL.seg_tmp

Field	Bits	Default	Access	Description
seg_tmp	31:28	0000b	R/W	Temporary Memory Segment Select

These bits select the memory segment which is used as scratch / temporary space by the micro MAA. (Same as select settings for Operand A)

11.2.1.1.2 MAA_MAWS**MAA_MAWS.modlen**

Field	Bits	Default	Access	Description
modlen	9:0	0	R/W	MAA Word Size

Selects bit size for modular operation; valid settings are 1 to 512.

MAA_MAWS.byteswap

Field	Bits	Default	Access	Description
byteswap	15	0	R/W	Big Endian Byte Mode

- 0: Do not swap byte order when transferring to/ from MAA memory space.
- 1: Swap byte order for MAA memory transfers.

11.2.1.1.3 MAA_MEM_SEG0

Default	Access	Description
0000 0000 0000 0000 0000 0000 0000b	R/W	[64 bytes] MAA Memory Segment 0

11.2.1.1.4 MAA_MEM_SEG1

Default	Access	Description
0000 0000 0000 0000 0000 0000 0000b	R/W	[64 bytes] MAA Memory Segment 1

11.2.1.1.5 MAA_MEM_SEG2

Default	Access	Description
0000 0000 0000 0000 0000 0000 0000b	R/W	[64 bytes] MAA Memory Segment 2

11.2.1.1.6 MAA_MEM_SEG3

Default	Access	Description
0000 0000 0000 0000 0000 0000 0000b	R/W	[64 bytes] MAA Memory Segment 3

11.2.1.1.7 MAA_MEM_SEG4

Default	Access	Description
0000 0000 0000 0000 0000 0000 0000b	R/W	[64 bytes] MAA Memory Segment 4

11.2.1.1.8 MAA_MEM_SEG5

Default	Access	Description
0000 0000 0000 0000 0000 0000 0000b	R/W	[64 bytes] MAA Memory Segment 5

11.3 Registers (TPU)

11.3.1 Module TPU Registers

Address	Register	32b Word Len	Description
0x40011200	TPU_PRNG_USER_ENTROPY	1	PRNG User Entropy Value
0x40011204	TPU_PRNG_RND_NUM	1	PRNG Seed Output
0x40011C00	TPU_TSR_STATUS	1	Dynamic Tamper Sensor Status
0x40011C04	TPU_TSR_CTRL0	1	Dynamic Tamper Sensor Control 0
0x40011C08	TPU_TSR_CTRL1	1	Dynamic Tamper Sensor Control 1
0x40011C10	TPU_TSR_SKS0	1	TPU Secure Key Storage Register 0 (Cleared on Tamper Detect)

Address	Register	32b Word Len	Description
0x40011C14	TPU_TSR_SKS1	1	TPU Secure Key Storage Register 1 (Cleared on Tamper Detect)
0x40011C18	TPU_TSR_SKS2	1	TPU Secure Key Storage Register 2 (Cleared on Tamper Detect)
0x40011C1C	TPU_TSR_SKS3	1	TPU Secure Key Storage Register 3 (Cleared on Tamper Detect)

11.3.1.1 TPU_PRNG_USER_ENTROPY

Default	Access	Description
00000000h	R/W	PRNG User Entropy Value

[7:0] Data to combine with clock entropy to generate PRNG.

11.3.1.2 TPU_PRNG_RND_NUM

Default	Access	Description
00000000h	R/W	PRNG Seed Output

[15:0] Seed output value from PRNG.

11.3.1.3 TPU_TSR_STATUS

Default	Access	Description
0000 0000 0000 0000 0000 000s 0000 000sb	R/W	Dynamic Tamper Sensor Status

[bit 0] Set to 1 by hardware when the dynamic tamper sensor is triggered. May be set or cleared by firmware. Cleared to 0 on BOR.

[bit 8] Set to 1 by hardware when a DRS is triggered by the dynamic tamper sensor. Cleared to 0 by BOR; not writeable by firmware.

11.3.1.4 TPU_TSR_CTRL0

TPU_TSR_CTRL0.err_thr

Field	Bits	Default	Access	Description
err_thr	4:0	00000b	R/W	Dynamic Sensor Error Threshold

Sets the error threshold required to trigger a DRS from the tamper sensor. When the error count during an output sample period exceeds this threshold, a tamper event will be triggered. Valid range is 0-31.

Note For output frequency settings of divide by 16, divide by 8, or divide by 4, the error threshold must be less than the divide ratio for proper operation.

TPU_TSR_CTRL0.out_freq

Field	Bits	Default	Access	Description
out_freq	7:5	000b	R/W	Dynamic Sensor Output Frequency

Sets the tamper sensor output frequency based on a divide down of the module clock, as follows:

- 000b: divide by 4 (max error threshold is 3)
- 001b: divide by 8 (max error threshold is 7)
- 010b: divide by 16 (max error threshold is 15)
- 011b: divide by 32
- 100b: divide by 64
- 101b: divide by 128
- 110b: divide by 256
- 111b: reserved

TPU_TSR_CTRL0.clock_div

Field	Bits	Default	Access	Description
clock_div	10:8	000b	R/W	Dynamic Sensor Clock Div

Sets the dynamic sensor module clock based on a divide down of the reference (input) clock source for the dynamic sensor:

- 000b: divide by 1
- 001b: divide by 2
- 010b: divide by 4
- 011b: divide by 8
- 100b: divide by 16
- 101b: divide by 32
- 110b: divide by 64
- 111b: Reserved

TPU_TSR_CTRL0.rtc_tx_busy

Field	Bits	Default	Access	Description
rtc_tx_busy	14	0b	R/O	RTC Domain Tx Busy

- 0: No RTC transfer in progress
- 1: Transfer in progress of this register to the RTC domain

TPU_TSR_CTRL0.lock

Field	Bits	Default	Access	Description
lock	15	0b	R/W	Lock Bit

Once set to 1, can only be cleared by BOR

11.3.1.5 TPU_TSR_CTRL1

Default	Access	Description
0000 0000 0000 0000 0000 0000 0000b	R/W	Dynamic Tamper Sensor Control 1

[bit 0] Dynamic tamper sensor DRS enable; set to 1 to enable a DRS output when the tamper sensor is triggered. Cleared to 0 on BOR only. May be written by firmware only when the lock bit in TPU_TSR_CTRL0 is set to 0.

11.3.1.6 TPU_TSR_SKS0

Default	Access	Description
.	R/W	TPU Secure Key Storage Register 0 (Cleared on Tamper Detect)

11.3.1.7 TPU_TSR_SKS1

Default	Access	Description
.	R/W	TPU Secure Key Storage Register 1 (Cleared on Tamper Detect)

11.3.1.8 TPU_TSR_SKS2

Default	Access	Description
.	R/W	TPU Secure Key Storage Register 2 (Cleared on Tamper Detect)

11.3.1.9 TPU_TSR_SKS3

Default	Access	Description
.	R/W	TPU Secure Key Storage Register 3 (Cleared on Tamper Detect)

12 CRC16 and CRC32 Hardware Accelerator

12.1 Overview

The Cyclic Redundancy Check (CRC) hardware peripheral, incorporated into the **MAX32600**, provides a fast method for calculating 16-bit and 32-bit CRC calculations on 32-bit data values. The CRC peripheral is accessible from the Cortex-M3 processor or via the [Peripheral Management Unit \(PMU\)](#). Both the CRC-16-CCITT and CRC-32 calculations complete in a single clock cycle and use the following polynomials in the calculations.

CRC-16-CCITT:

$$CRC16 = X^{16} + X^{12} + X^5 + 1$$

CRC-32:

$$CRC32 = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$$

12.2 CRC Clock Gating

The CRC hardware peripheral uses the System Core Clock to operate and all CRC calculations complete in a single system clock cycle. The peripheral clock and clock gate are enabled by default for the CRC peripheral. The register field [CLKMAN_CLK_GATE_CTRL2.crc_clk_gater](#) is used to control the peripheral clock and clock gating for the CRC module. The table below shows the modes supported for the CRC peripheral clock.

crc_clk_gater (xxb)	Clock State
00b	Peripheral clock disabled. Lowest power mode when peripheral not in use.
01b	<i>Default:</i> Peripheral clock enabled with Dynamic Clock Gating. Clock is active only when using the peripheral.
1xb	Peripheral clock is enabled and active at all times. Highest performance and highest current option.

12.3 CRC Operation

The **MAX32600** CRC hardware peripheral calculates a 16-bit or 32-bit CRC value based on a 32-bit data value combined with an initial seed value (default is 0xFFFF for 16-bit CRC and 0xFFFFFFFF for 32-bit CRC calculations). For data values that are not 32-bits wide, the upper bytes must be padded with zeroes to the next 32-bit boundary. Data may be written to [CRC_DATA_VALUE16](#) or [CRC_DATA_VALUE32](#) in byte, word, or double word format from either the Cortex-M3 processor or PMU. A user specified initial seed value can be used by writing to the appropriate register ([CRC_SEED16](#) or [CRC_SEED32](#)) in the **MAX32600** hardware peripheral along with setting the appropriate reseed enable bit ([CRC_RESEED.crc16](#) or [CRC_RESEED.crc32](#)).

12.4 CRC-16-CCITT Example Calculation

1. If an initial seed value other than 0xFFFF is desired, write initial seed value into [CRC_SEED16](#).
2. Set [CRC_RESEED.crc16](#) to apply new initial seed value.
3. Write first data value to [CRC_DATA_VALUE16](#). This data value may be 8-bits, 16-bits, or 32-bits wide. Pad upper bits with zeroes if less than 32-bits.
4. Read from [CRC_DATA_VALUE16](#) to obtain the CRC result.
5. *OPTIONAL*: Repeat steps 3 and 4 if additional data needs to be included in the CRC calculation.

Note When computing the CRC over a range of data, step 3 may be repeated until all values have been written prior to proceeding to step 4 and reading the final CRC value from the [CRC_DATA_VALUE32](#) register.

12.5 CRC-32 Example Calculation

1. If an initial seed value other than 0xFFFFFFFF is desired, write initial seed value into [CRC_SEED32](#).
2. Set [CRC_RESEED.crc32](#) to apply new initial seed value.
3. Write first data value to [CRC_DATA_VALUE32](#). This data value may be 8-bits, 16-bit, or 32-bits wide. Pad upper bits with zeroes if less than 32-bits.
4. Read from [CRC_DATA_VALUE32](#) to obtain the CRC result.
5. *OPTIONAL*: Repeat steps 3 and 4 if additional data needs to be included in the CRC calculation.

Note When computing the CRC over a range of data, step 3 may be repeated until all values have been written prior to proceeding to step 4 and reading the final CRC value from the [CRC_DATA_VALUE32](#) register.

12.6 Registers (CRC)

12.6.1 Module CRC Registers

Address	Register	32b Word Len	Description
0x40010000	CRC_RESEED	1	CRC-16/CRC-32 Reseed Controls
0x40010004	CRC_SEED16	1	Reseed Value for CRC-16 Calculations
0x40010008	CRC_SEED32	1	Reseed Value for CRC-32 Calculations
0x4010B000	CRC_DATA_VALUE16	512	Write Next CRC-16 Data Value / Read CRC-16 Result Value
0x4010B800	CRC_DATA_VALUE32	512	Write Next CRC-32 Data Value / Read CRC-32 Result Value

12.6.1.1 CRC_RESEED

CRC_RESEED.crc16

Field	Bits	Default	Access	Description
crc16	0	0	W1C	Reseed CRC16 Generator

Write to 1 to reseed the CRC16 generator.

Cleared to 0 by hardware when reseed operation has completed.

CRC_RESEED.crc32

Field	Bits	Default	Access	Description
crc32	1	0	W1C	Reseed CRC32 Generator

Write to 1 to reseed the CRC32 generator.

Cleared to 0 by hardware when reseed operation has completed.

12.6.1.2 CRC_SEED16

Default	Access	Description
00000000h	R/W	Reseed Value for CRC-16 Calculations

This register contains the value that will be used when a CRC16 reseed operation is triggered.

12.6.1.3 CRC_SEED32

Default	Access	Description
00000000h	R/W	Reseed Value for CRC-32 Calculations

This register contains the value that will be used when a CRC32 reseed operation is triggered.

12.6.1.4 CRC_DATA_VALUE16

Default	Access	Description
00000000h	R/W	Write Next CRC-16 Data Value / Read CRC-16 Result Value

Writing to this register loads the next value into the CRC engine to be processed for the CRC-16 running calculation.

Reading from this register returns the current CRC-16 calculation result.

12.6.1.5 CRC_DATA_VALUE32

Default	Access	Description
00000000h	R/W	Write Next CRC-32 Data Value / Read CRC-32 Result Value

Writing to this register loads the next value into the CRC engine to be processed for the CRC-32 running calculation.

Reading from this register returns the current CRC-32 calculation result.

13 LCD Controller

13.1 LCD Overview

The standard package of the **MAX32600** incorporates an LCD controller with a boost regulator that interfaces with common low-voltage displays ($\pm 3.3V$ or below, AC coupled). An external LCD module is not needed: by incorporating the LCD controller into the microcontroller itself, the design only requires an external LCD glass for full LCD functionality. The contrast adjust feature is used to vary the voltage; the multiplex effectively produces a lower contrast display. Every character in an LCD glass is composed of one or more segments, each of which is activated by internally selecting the appropriate segment and common signal.

The features of the **MAX32600** LCD controller include:

- LCD segment and common-drive signal generation based on multiplex
- Support for four types of display modes
- Integrated boost regulator ensures LCD operation over 2V
- Adjustable frame frequency
- Internal voltage-divider resistors which eliminate requirement for external components
- Internal adjustable resistor allows contrast adjustment without external components
- Autonomous character mapping from LCD local memory to segments

Note LCD functionality is not available on the Compact or WLP pin layouts. See [Pin Configurations, Packages, and Special Function Multiplexing](#) for further information.

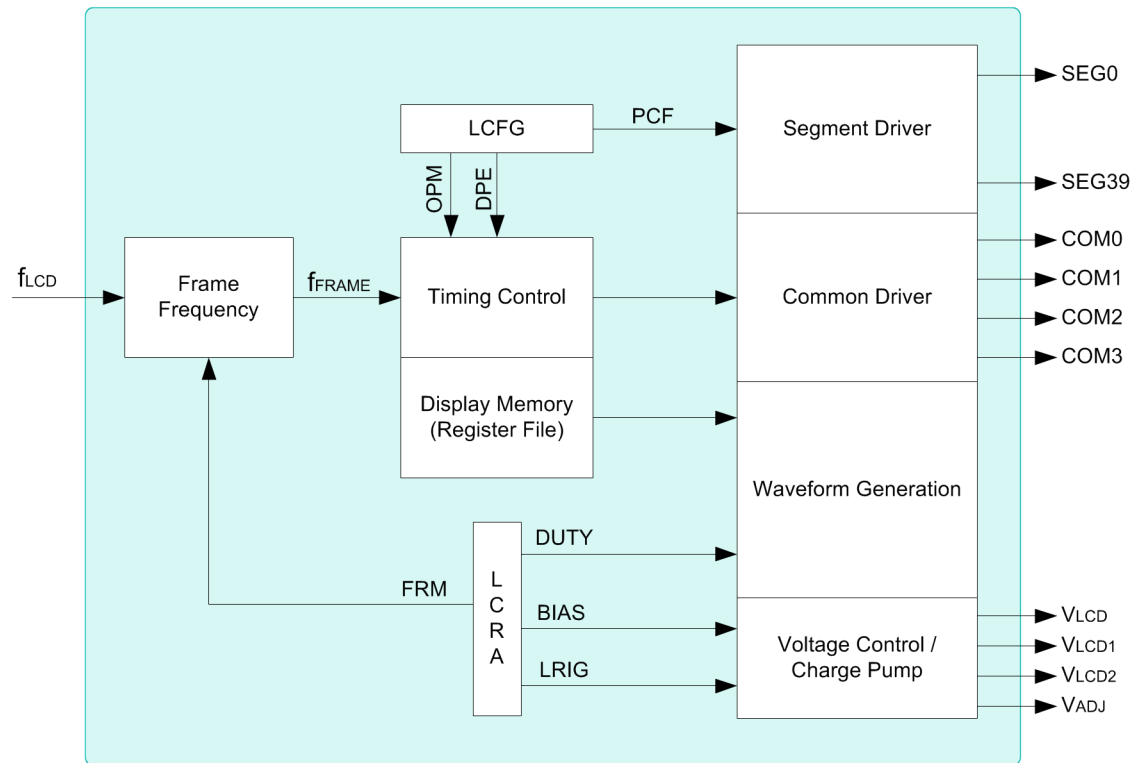


Figure 13.1: LCD Controller Block Diagram

13.2 LCD Operation

Four display modes are supported by the LCD controller:

- Static (COM0)
- 1/2 duty multiplexed with 1/2 bias voltages (COM [0:1])
- 1/3 duty multiplexed with 1/3 bias voltages (COM [0:2])
- 1/4 duty multiplexed with 1/3 bias voltages (COM [0:3])

Note Since the voltages available for LCD drive are V_{LCD} , $V_{LCD} \times \frac{2}{3}$, $V_{LCD} \times \frac{1}{3}$, and V_{ADJ} , the $\frac{1}{2}$ -bias mode (which requires an output level of $V_{LCD} \times \frac{1}{2}$) will require two of the LCD voltage output pins (V_{LCD2} and V_{LCD1}) to be shunted together externally.

Due to the area under the curve, this duty multiplexing produces a lower contrast display in non-static operation. If the hardware has the same amount of time to multiplex two common drivers (COMx) as it does for dedication to one common driver (static operation), the display for the two COMx will be effectively dimmer. Here, the multiplex spends half of its time working on COM0, for instance, and the other half on COM1.

13.3 LCD Configuration

LCD power and clock registers must be configured first to ensure proper device configuration. If a clock is off to the block, writes to the APB registers may not be recognized.

13.3.1 LCD Clock

The **MAX32600** LCD controller uses the device's on-board 32kHz RTC as a source clock. Comprehensive configuration details for the RTC can be found in the [Real Time Clock \(RTC\)](#) section. To enable the RTC for the LCD controller, `PWRSEQ_REG0.pwr_chzyen_run` must be in its default state of 0 and `PWRSEQ_REG0.pwr_rtcen_run` needs to be set to 1 to enable operation in run mode.

Additionally, the clock manager charge pump, `CLKMAN_CLK_CTRL_8_LCD_CHPUMP.lcd_clk_scale`, must be configured. Writing 0001b to this field enables CLK = (System Clock Source / 1). Writing 0000b to this field disables the CLK and is the default following a reset.

13.3.2 LCD Power Control

The power control register of the LCD controller must be enabled in conjunction with the clock charge pump outlined above. Writing 1 to the `LCD_LPWRCTR.L.powerup` field enables power to the controller. Writing 0 to this field powers down the controller.

13.3.3 LCD Configuration Registers

The initial startup of the LCD controller requires two writes to `LCD_LCFG`. First, write 1 to `LCD_LCFG.lcd_en` to enable the LCD display. Second, write 1 to both `LCD_LCFG.operating_mode` and `LCD_LCFG.lcd_en`. The LCD display enable bit must be written twice because the resistor divider can only be configured in

“suspend” mode (i.e., OPM=0) but with the display already enabled (i.e., DPE=1).

Stop mode operation is controlled via the [field_LCD_LCFG_stop_mode_en](#) register "LCD_LCFG.stop_mode_en register". A write to 0 leaves stop mode function unaffected and disabled; a write to 1 enables LCD controller operation in stop mode.

LCD data hold is configured with the [LCD_LCFG.datahold](#) register field. Writing 0 allows display memory changes to propagate to the LCD glass; writing to 1 holds the current data displayed on the LCD COM/SEG regardless of changes to the display memory.

The following table outlines the bit addresses for the LCD configuration registers.

LCD_LCFG Register Details

LCD_LCFG Register	Bit	Operation	Read/Write	Details
lcd_en	0	Display Enable (DPE)	R/W	0 = reset/disable; 1 = enable
operating_mode	1	LCD Operating Mode (OPM)	R/W	0 = reset/suspended; 1 = controller on
stop_mode_en	2	Stop	R/W	0 = reset/stop mode function unaffected; 1 = enable controller operation in stop mode
Reserved	3	Reserved	R/W	Always write 0
Reserved	5:4	Reserved	RO	Always 00
datahold	6	Data Source Select (Hold Data)	R/W	0 = reset/display data from LCD Memory; 1 = continue to display current data
Reserved	15:7	Reserved	RO	Always 0

13.3.4 LCD Internal Register Adjust

The [LCD internal register adjust fields](#) are used to select the ground connect, contrast adjustments, frame rate frequency, and the duty cycle. The internal ground connect is enabled by writing 1 to [LCD_LCRA.gnd_en](#) and disabled by writing 0. This is required to be 1 in anything other than “Static” ([duty_cycle](#) == 0) if no other external components are used as a functional equivalent to the internal ground enable. The LCD register adjust, [LCD_LCRA.reg_adj](#), is used for contrast adjustments when using the internal ground connect by setting the value of the internal LCD ground register from 0-72kOhms. This should only be written when LCD operation is suspended (i.e., OPM=0).

The **MAX32600** LCD output frequency is set with the [LCD_LCRA.frame_rate](#) register. Equations for frame frequencies for the various possible bias modes:

- For 1/3 bias mode: $\frac{2}{3} \times \frac{F_{LCD}}{FRM[4:0]+1}$
- For all other bias modes: $\frac{F_{LCD}}{FRM[4:0]+1}$

Note MAX32600 f_{LCD} is 512Hz.

LCD Frame Frequencies (static, 1/2, 1/4 duty) with $f_{LCD} = 512$ Hz

FRM	f_{FRAME}	FRM	f_{FRAME}	FRM	f_{FRAME}	FRM	f_{FRAME}
0	512	8	57	16	30	24	20
1	256	9	51	17	28	25	20
2	171	10	47	18	27	26	19
3	128	11	43	19	26	27	18
4	102	12	39	20	24	28	18
5	85	13	37	21	23	29	17
6	73	14	34	22	22	30	17
7	64	15	32	23	21	31	16

LCD Frame Frequencies (1/3 duty) with $f_{LCD} = 512$ Hz

FRM	f_{FRAME}	FRM	f_{FRAME}	FRM	f_{FRAME}	FRM	f_{FRAME}
0	341	8	38	16	20	24	14
1	171	9	34	17	19	25	13
2	114	10	31	18	18	26	13
3	85	11	28	19	17	27	12
4	68	12	26	20	16	28	12
5	57	13	24	21	16	29	11
6	49	14	23	22	15	30	11
7	43	15	21	23	14	31	11

Note For most LCD, the best drive performance Frame Frequency, f_{FRAME} , is between 30 Hz and 128 Hz (these instances are in bold in the above tables).

The `LCD_LCRA.duty_cycle` register sets the controller's duty cycle. There are four fixed valid bias (voltage levels) and duty cycle (multiplex) setting choices.

- 00b: Static
- 01b: 1/2 duty (1/2 bias, 2X segment mux)
- 10b: 1/3 duty (1/3 bias, 3X segment mux)
- 11b: 1/4 duty (1/3 bias, 4X segment mux)

The following table outlines the bit addresses for the [LCD Internal Register Adjust registers](#). **Writes to this register can only occur when OPM=0.**

LCD_LCRA Register Details

LCD_LCRA Register	Bit	Operation	Read/Write	Details
reg_adj	3:0	Resistor Adjust	R/W	0 = reset
gnd_en	4	Resistor Internally Grounded	R/W	0 = reset/ disable; 1 = enable internal ground connect
Reserved	5	Reserved	RO	Always 0
Reserved	7:6	Reserved	R/W	Always 00
frame_rate	12:8	Frame Frequency	R/W	0 = reset
Reserved	13	Reserved	RO	Always 0
duty_cycle	15:14	Duty Cycle Select	R/W	0 = reset; 0 = static with 1/2 bias; 1 = 1/2 duty with 1/2 bias; 2 = 1/3 duty with 1/3 bias; 3 = 1/4 duty with 1/3 bias

13.3.5 LCD Port Configuration

To properly configure LCD segment pin operation, LCD and I/O registers must be written via [LCD_LPCF](#) and [IOMAN_LCD](#) registers.

LCD port configuration [LCD_LPCF](#) register bits enable pairs of segment pins when set to 1. The bit/segment pin relationship is as follows:

- Bit 0: Enables SEG0+SEG1
- Bit 1: Enables SEG2+SEG3
- ...

- Bit 19: Enables SEG38+SEG39

Within the LCD I/O management (IOMAN_LCD) registers, multiple fields must be written.

Writing 1 to [IOMAN_LCD_COM_REQ](#) requests LCD COM mode for COM[3:0] (all four simultaneously). [IOMAN_LCD_COM_ACK](#) is read-only acknowledgement from hardware; read of 1 acknowledges LCD COM mode is selected for COM[3:0]. Note that the request may be denied based on port function priority.

I/O management segment requests write to single segment pins and must be a redundancy of the [LCD_LPCF](#) segment request(s). These are accessed with the following registers:

- [IOMAN_LCD_SEG_REQ0.io_req_n](#)
 - Write 1 requests LCD SEG mode for the GPIO selected in the *n* position (applicable to P3.0 – P6.7 [32 segments total])
- [IOMAN_LCD_SEG_REQ1.io_req_n](#)
 - Write 1 requests LCD SEG mode for the GPIO selected in the *n* position (applicable to P7.0 – P7.7 [8 segments total])

The I/O management segment requests also have associated read-only acknowledgements from hardware:

- [IOMAN_LCD_SEG_ACK0.io_ack_n](#)
 - Read of 1 acknowledges SEG mode selected for the GPIO selected in the *n* position (applicable to P3.0 – P6.7 [32 segments total])
- [IOMAN_LCD_SEG_ACK1.io_ack_n](#)
 - Read of 1 acknowledges SEG mode selected for the GPIO selected in the *n* position (applicable to P7.0 – P7.7 [8 segments total])

The following table outlines the bit addresses for the LCD Port Configuration registers. **Writes to this register can only occur when OPM=0.**

LCD_LPCF Register Details

LCD_LPCF Register	Bit	Operation	Read/Write	Details
LCD_LPCF	19:0	Segment Pin Pair Enable	R/W	0 = reset; 1 = enables selected segment pin pairs
LCD_LPCF	31:20	Reserved	RO	always write 0 value

13.3.6 LCD Memory Configuration

LCD glass mapping to `LCD_LCDATA` and `LCD_LCADDR` values are dependent on LCD glass type and its pin interconnection to the **MAX32600** LCD segment pins. Further information can be found in the relevant application note.

LCD_LCDATA / LCD_LCADDR Register Details

Register	Bit	Operation	Read/Write	Details
<code>LCD_LCADDR.addr_sel</code>	4:0	LCD Memory Index	R/W	0 = reset
<code>LCD_LCADDR</code>	15:5	Reserved	RO	Always write 0 value
<code>LCD_LCDATA</code>	7:0	LCD Glass Segment Mapping	R/W	See application note showing LCD_LCDATA bit mapping for all four display mux modes

13.4 Registers (LCD)

13.4.1 Module LCD Registers

Address	Register	32b Word Len	Description
0x40060000	<code>LCD_LCFG</code>	1	LCD Configuration / Control Register
0x40060004	<code>LCD_LCRA</code>	1	LCD Internal Resistor Adjust / Duty Cycle / Bias Mode
0x40060008	<code>LCD_LPCF</code>	1	LCD Port Configuration Register
0x4006000C	<code>LCD_LCADDR</code>	1	LCD Display Memory Address Select Register
0x40060010	<code>LCD_LCDATA</code>	1	LCD Display Memory Data Register
0x40060014	<code>LCD_LPWRCTRL</code>	1	LCD Power Control Register

13.4.1.1 LCD_LCFG

`LCD_LCFG.lcd_en`

Field	Bits	Default	Access	Description
lcd_en	0	0	R/W	LCD Display Enable (DPE)

- 0: LCD display outputs are disabled (default).
- 1: LCD display outputs are enabled.

LCD_LCFG.operating_mode

Field	Bits	Default	Access	Description
operating_mode	1	0	R/W	LCD Operating Mode (OPM)

- 0: LCD controller is disabled (default).
- 1: LCD controller is enabled.

LCD_LCFG.stop_mode_en

Field	Bits	Default	Access	Description
stop_mode_en	2	0	R/W	Stop Mode Operation Enable

- 0: LCD SEG/COM output states will remain static during Stop mode (default).
- 1: LCD pattern generation will continue while Stop mode is active.

LCD_LCFG.autopage_en

Field	Bits	Default	Access	Description
autopage_en	3	0	R/W	Autopage Mode Enable (Not used, do not modify)

This field must remain at its default setting for proper operation of the LCD controller.

LCD_LCFG.page_sel

Field	Bits	Default	Access	Description
page_sel	5:4	0	R/O	Page Select (Not used, do not modify)

This field must remain at its default setting for proper operation of the LCD controller.

LCD_LCFG.datahold

Field	Bits	Default	Access	Description
datahold	6	0	R/W	Display Memory Data Hold

- 0: Changes made to LCD display memory propagate immediately to the LCD SEG/COM output patterns.
- 1: Changes made to LCD display memory are held off until this field is set back to 0. While this field is set to 1, the previous contents of the display memory will be used for LCD SEG/COM output pattern generation.

13.4.1.2 LCD_LCRA

LCD_LCRA.reg_adj

Field	Bits	Default	Access	Description
reg_adj	3:0	0	R/W	LCD Internal Resistor Adjust Value

Sets the value of the internal LCD ground resistor (from 0 to 72k-ohms).

LCD_LCRA.gnd_en

Field	Bits	Default	Access	Description
gnd_en	4	0	R/W	LCD Internal Resistor Ground Connect

- 0: Disabled.
- 1: VADJ is connected to ground internally through the internal LCD ground resistor.

LCD_LCRA.frame_rate

Field	Bits	Default	Access	Description
frame_rate	12:8	0	R/W	Pattern Generation Frame Rate

Selects LCD output frame frequency in Hz. For 1/3-bias mode, the frequency is defined as $((2/3) * F_{lcd}) / (\text{field value} + 1)$. For all other bias modes, the frequency is defined as $F_{lcd} / (\text{field value} + 1)$.

LCD_LCRA.duty_cycle

Field	Bits	Default	Access	Description
duty_cycle	15:14	0	R/W	LCD Output Duty Cycle and Bias Selection

Selects output bias and duty cycle (muxing) modes as follows:

- 0: Static
- 1: Half duty (1/2 bias, x2 segment mux)
- 2: 1/3 duty (1/3 bias, x3 segment mux)

- 3: 1/4 duty (1/3 bias, x4 segment mux)

13.4.1.3 LCD_LPCF

Default	Access	Description
00000000h	R/W	LCD Port Configuration Register

Each bit (when set to 1) enables two corresponding SEG output signals as follows:

- bit 0: set to 1 to enable SEG0 and SEG1
- bit 1: set to 1 to enable SEG2 and SEG3
- bit 2: set to 1 to enable SEG4 and SEG5
- ...
- bit 19: set to 1 to enable SEG38 and SEG39
- bits 20-31: reserved.

13.4.1.4 LCD_LCADDR

LCD_LCADDR.addr_sel

Field	Bits	Default	Access	Description
addr_sel	4:0	0	R/W	Display Memory Address Select

This field selects the byte of LCD display memory that can be read from or written to by reading/writing the LCDATA register.

The valid address range for this field is 0 through 20, since there are 21 bytes of LCD display memory.

LCD_LCADDR.page_sel

Field	Bits	Default	Access	Description
page_sel	7:6	0	R/W	Display Memory Page Select

Reserved. This field must remain set to 0 for proper LCD controller operation.

13.4.1.5 LCD_LCDATA

LCD_LCDATA.data

Field	Bits	Default	Access	Description
data	7:0	0	R/W	LCD Display Memory Data

Reading from this field returns the byte of LCD display memory currently selected by LCADDR.addr_sel.

When a value is written to this field, that value will be loaded into the byte of LCD display memory currently selected by LCADDR.addr_sel.

13.4.1.6 LCD_LPWRCTRL

LCD_LPWRCTRL.powerup

Field	Bits	Default	Access	Description
powerup	0	0	R/W	LCD Powerup

LCD_LPWRCTRL.freerun

Field	Bits	Default	Access	Description
freerun	1	0	R/W	Free Run Mode

LCD_LPWRCTRL.highbat_startup

Field	Bits	Default	Access	Description
highbat_startup	2	0	R/W	Highbat Startup Mode

LCD_LPWRCTRL.bias_powerup

Field	Bits	Default	Access	Description
bias_powerup	3	0	R/W	Bias Powerup

LCD_LPWRCTRL.enbg_4_lcd

Field	Bits	Default	Access	Description
enbg_4_lcd	4	0	R/W	Enable bg_4_lcd

LCD_LPWRCTRL.lcd_vsel

Field	Bits	Default	Access	Description
lcd_vsel	6:5	0	R/W	LCD VSel

LCD_LPWRCTRL.pumpup

Field	Bits	Default	Access	Description
pumpup	24	0	R/O	LCD Voltage Pump Up

LCD_LPWRCTRL.pump_vready

Field	Bits	Default	Access	Description
pump_vready	30	0	R/O	LCD Voltage Pump Ready

Field	Bits	Default	Access	Description
-------	------	---------	--------	-------------

14 Flash Controller and Instruction Cache

The flash memory controller on the **MAX32600** handles control and timing signals for programming and erase operations on both the main program flash memory array and the flash information block. The flash information block is normally written during production test only, and is not generally intended to be modified by the user application. Two exceptions to this are the Destructive Security Bypass access key value and the Auto-Lock option value, which may be set by the user by means of a special procedure even after access to the rest of the information block has been locked out.

Functions provided by the flash controller for use in normal operation include:

- Mass erase of main program flash array
- Page erase of one page in the main program flash array
- Write to one location (programmed 32-bits at a time) in the main program flash array
- Special one-time writes by the user to the DSB Access Key and/or the Auto-Lock option values in the information block

14.1 Registers (FLC)

14.1.1 Module FLC Registers

Address	Register	32b Word Len	Description
0x400F0000	FLC_FADDR	1	Flash Operation Address
0x400F0004	FLC_FCKDIV	1	Flash Clock Pulse Divisor
0x400F0008	FLC_CTRL	1	Flash Control Register
0x400F0024	FLC_INTR	1	Flash Controller Interrupt Flags and Enable/Disable 0
0x400F0030	FLC_FDATA	1	Flash Operation Data Register
0x400F0050	FLC_PERFORM	1	Flash Performance Settings
0x400F0080	FLC_STATUS	1	Security Status Flags
0x400F0088	FLC_SECURITY	1	Flash Controller Security Settings
0x400F009C	FLC_BYPASS	1	Status Flags for DSB Operations
0x400F0100	FLC_USER_OPTION	1	Used to set DSB Access code and Auto-Lock in info block
0x400F0140	FLC_CTRL2	1	Flash Control Register 2

Address	Register	32b Word Len	Description
0x400F0144	FLC_INTFL1	1	Interrupt Flags Register 1
0x400F0148	FLC_INTEN1	1	Interrupt Enable/Disable Register 1

14.1.1.1 FLC_FADDR

FLC_FADDR.faddr

Field	Bits	Default	Access	Description
faddr	17:0	0_0000h	R/W	Flash Operation Address

Byte address for flash write and erase operations.

14.1.1.2 FLC_FCKDIV

FLC_FCKDIV.fckdiv

Field	Bits	Default	Access	Description
fckdiv	4:0	24	R/W	Flash Clock Pulse Divisor

The value of this field is used to generate a 1 microsecond width pulse from the system clock source. This pulse is used when performing write or erase operations on the flash memory. This field should be set to the system clock source frequency in MHz (or as close as possible); for example, if the system clock is running at 24MHz, this field should be set to 24.

14.1.1.3 FLC_CTRL

FLC_CTRL.write

Field	Bits	Default	Access	Description
write	0	0	R/W	Start Flash Write Operation

Writing this bit to 1 attempts to start a flash write operation using the contents of the two registers (which must be written before this bit is set):

- FLC_FADDR = Byte address of location to write.
- FLC_FDATA = Value to write to flash location.

FLC_CTRL.mass_erase

Field	Bits	Default	Access	Description
mass_erase	1	0	R/W	Start Flash Mass Erase Operation

Writing this bit to 1 attempts to start a flash mass erase operation (of the main program flash). The Flash Erase Code must first be written to AAh for this operation to be accepted by the FLC.

FLC_CTRL.page_erase

Field	Bits	Default	Access	Description
page_erase	2	0	R/W	Start Flash Page Erase Operation

Writing this bit to 1 attempts to start a flash page erase operation (in the main program flash for addresses from 00_0000h-1F_FFFFh). The Flash Erase Code must first be written to 55h for this operation to be accepted by the FLC.

FLC_CTRL.erase_code

Field	Bits	Default	Access	Description
erase_code	15:8	00h	R/W	Flash Erase Code

This is a key field that must be written in order to trigger a flash mass erase or page erase operation; the intention is to make it more difficult to trigger these operations by mistake.

- AAh: Enable a mass erase operation
- 55h: Enable a page erase operation

This field is auto-cleared by the hardware to zero following any mass erase or page erase operation.

FLC_CTRL.info_block_unlock

Field	Bits	Default	Access	Description
info_block_unlock	16	0	R/O	Flash Info Block Locked

- 1: Flash info block is accessible (not locked)
- 0: Flash info block is locked - it may not be targeted by write operations, and it is not mapped into AHB memory space.

Note Even if the info block is unlocked, this field will only read 1 if the flash_unlock field has also been set to 0010b.

FLC_CTRL.write_enable

Field	Bits	Default	Access	Description
write_enable	17	0	R/O	Flash Writes Enabled

- 0: Flash writes are not currently enabled.
- 1: Flash writes are currently enabled.

FLC_CTRL.pending

Field	Bits	Default	Access	Description
pending	24	0	R/O	Flash Controller Status

This bit goes to an active high state (1) whenever the flash controller is performing a read, write, or erase operation. Once the operation has completed, the bit will return to 0.

FLC_CTRL.info_block_valid

Field	Bits	Default	Access	Description
info_block_valid	25	0	R/O	Info Block Valid Status

- 0: Flash info block does not contain valid contents. Security, test and trim settings have been set to default values by the FLC.
- 1: Flash info block is valid, and its contents have been loaded into the trim shadow registers.

FLC_CTRL.auto_incre_mode

Field	Bits	Default	Access	Description
auto_incre_mode	27	0	R/W	Address Auto-Increment Mode

- 0: Auto-increment mode disabled.
- 1: Auto-increment mode enabled; flash address will be automatically incremented following a write operation.

FLC_CTRL.flsh_unlock

Field	Bits	Default	Access	Description
flsh_unlock	31:28	0000b	R/W	Flash Write/Erase Enable

0010b/2d: Flash write and erase operations are enabled.

14.1.1.4 FLC_INTR

FLC_INTR.started_if

Field	Bits	Default	Access	Description
started_if	0	0	R/W	Flash Operation Started

Write to zero to clear bit to 0.

Set to 1 by hardware when a flash operation (write or erase) starts.

FLC_INTR.failed_if

Field	Bits	Default	Access	Description
failed_if	1	0	R/W	Flash Operation Failed

Write to zero to clear bit to 0.

Set to 1 by hardware when an error occurs during a flash operation

FLC_INTR.started_ie

Field	Bits	Default	Access	Description
started_ie	8	0	R/W	Flash Operation Started Interrupt Enable

- 0: Interrupt disabled.
- 1: Setting the Flash Operation Started bit to 1 will cause the FLC to generate an interrupt.

FLC_INTR.failed_ie

Field	Bits	Default	Access	Description
failed_ie	9	0	R/W	Flash Operation Failed Interrupt Enable

- 0: Interrupt disabled.
- 1: Setting the Flash Operation Failed bit to 1 will cause the FLC to generate an interrupt.

14.1.1.5 FLC_FDATA

Default	Access	Description
00000000h	R/W	Flash Operation Data Register

This register is used as an input parameter for various flash controller operations.

14.1.1.6 FLC_PERFORM**FLC_PERFORM.delay_se_en**

Field	Bits	Default	Access	Description
delay_se_en	0	1	R/W	Delay SE Enable

- 0: Original flash access design; xaddr and yaddr are activated the same clock edge as SE
- 1: Inserts an additional clock cycle before SE is driven active (safety margin, slows down read accesses) (default)

Note This bit is intended for internal testing purposes only. Application firmware should not modify the default value of this bit.

FLC_PERFORM.fast_read_mode_en

Field	Bits	Default	Access	Description
fast_read_mode_en	8	0	R/W	Fast Read Mode Enable

- 0: Original flash access mode, 8-9 cycles for read access (default)
- 1: Fast read mode - the flash controller will perform back-to-back flash reads for an access time of 2-3 cycles per read cycle (after the 2nd)

Note This bit is intended for internal testing purposes only. Application firmware should not modify the default value of this bit.

14.1.1.7 FLC_STATUS

FLC_STATUS.jtag_lock_window

Field	Bits	Default	Access	Description
jtag_lock_window	0	s	R/O	Debug Locked - Hardware Window

- 0: No lockout from this source
- 1: The debug lockout is being asserted because the debug hardware window lockout feature is active, and the window of 1024 cycles following reset has not yet elapsed.

FLC_STATUS.jtag_lock_static

Field	Bits	Default	Access	Description
jtag_lock_static	1	s	R/O	Debug Locked - Firmware Lockout

- 0: No lockout from this source
- 1: The debug lockout is being asserted because the Disable Debug (bit 0) in the FLC_SECURITY register has been set to 1. This is typically done by firmware as part of a user-defined security scheme.

FLC_STATUS.auto_lock

Field	Bits	Default	Access	Description
auto_lock	3	s	R/O	Debug Locked - Auto Lock

- 0: No lockout from this source
- 1: The debug lockout is being asserted unconditionally because the Auto Lock option has been enabled in the flash info block. The only way to remove this setting is to erase the info block (if the info block has not been locked) or to use the FLC_BYPASS register to perform a Factory Global Erase operation (Super-Wipe).

14.1.1.8 FLC_SECURITY**FLC_SECURITY.debug_disable**

Field	Bits	Default	Access	Description
debug_disable	7:0	SysRst:XXh, POR:00h	R/W	Debug Lockout

This field can be set to two values as follows:

- 00h: Debug access is unlocked; the ARM debugger can be used normally.
- 01h: Debug access is locked out. All access to the ARM debug engine is blocked.

To set this field to 01h (to set debug lockout), write the value A5h to this field. This field cannot be altered if the security_lock field has been set to 1000b.

FLC_SECURITY.mass_erase_lock

Field	Bits	Default	Access	Description
mass_erase_lock	11:8	SysRst:XXXXb, POR:0000b	R/W	Mass Erase Lockout

This field can be set to two values as follows:

- 0000b: Mass erase of the main internal flash memory can be performed in the usual manner.
- 0001b: The mass erase operation is locked out.

To set this field to 0001b (lock out mass erase), write the value Ch to this field. This field cannot be altered if the security_lock field has been set to 1000b.

FLC_SECURITY.security_lock

Field	Bits	Default	Access	Description
security_lock	31:28	SysRst:XXXXb, POR:0000b	R/W	Security Lock

Once this field is set to 1000b, the SECURITY register and the flash page protect registers may no longer be modified.

To set this field to 1000b, write the value 5 to the field. This field can only be set by firmware; it cannot be cleared.

14.1.1.9 FLC_BYPASS

FLC_BYPASS.destruct_bypass_erase

Field	Bits	Default	Access	Description
destruct_bypass_erase	0	0	R/O	Destructive Security Bypass In Progress

1:Operation is in progress.

FLC_BYPASS.superwipe_erase

Field	Bits	Default	Access	Description
superwipe_erase	1	0	R/O	Superwipe Erase In Progress

1:Operation is in progress.

FLC_BYPASS.destruct_bypass_complete

Field	Bits	Default	Access	Description
destruct_bypass_complete	2	0	R/O	Destructive Security Bypass Erase Complete

1:Operation has completed.

FLC_BYPASS.superwipe_complete

Field	Bits	Default	Access	Description
superwipe_complete	3	0	R/O	Superwipe Erase Complete

1:Operation has completed.

14.1.1.10 FLC_USER_OPTION

Default	Access	Description
00000000h	W/O	Used to set DSB Access code and Auto-Lock in info block

To set DESTRUCTIVE_BYPASS_CODE:

- Write your 32 bit DESTRUCTIVE_BYPASS_CODE to FLC_FDATA0
- Write 16'h7502 to this FLC_USER_OPTION[15:0] register.
- Wait for write to flash to complete.
- The code will be saved into the INFO block array of the flash.

- Do this prior to setting AUTO_LOCK.
- This does not set AUTO_LOCK or any other security settings.

To set AUTO_LOCK:

- Write 32'hAE0110EA to FLC_FDATA0
- Write 16'h7502 to this FLC_USER_OPTION[15:0] register.
- Wait for write to flash to complete.
- The AUTO_LOCK bit will be set in the INFO block array.
- Upon reset, the device will automatically lock (no debug).

14.1.1.11 FLC_CTRL2

FLC_CTRL2.flash_lve

Field	Bits	Default	Access	Description
flash_lve	7:0	00h	R/W	Flash LVE Enable

This drives the flash input LVE. Used when device VDD is 1.2V and below. Write 0xC2 to set. Write anything else to clear.

FLC_CTRL2.bypass_ahb_fail

Field	Bits	Default	Access	Description
bypass_ahb_fail	15:8	01h	R/W	AHB Fail Bypass

When using the AHB to write to flash, the fail flag will get set if a read access to the flash occurs before the write to the flash completes. Both AHB writes and AHB reads are arbitrated by the flash controller and the fail flag does not indicate a failed write or read. Thus, this fail flag can be ignored on subsequent writes or reads. If this field is 0 and the fail flag gets set, subsequent flash accesses will be ignored until the fail flag is cleared. Setting this field to 1 removes the fail flag from being evaluated and all subsequent flash accesses will be processed as normal.

Write 0x45 to this field to set the field to 1. Write 0x54 to this field to clear the field to 0.

14.1.1.12 FLC_INTFL1

FLC_INTFL1.sram_addr_wrapped

Field	Bits	Default	Access	Description
sram_addr_wrapped	0	0	W1C	SRAM Address Wrapped Interrupt Flag

FLC_INTFL1.invalid_flash_addr

Field	Bits	Default	Access	Description
invalid_flash_addr	1	0	W1C	Invalid Flash Address Interrupt Flag

FLC_INTFL1.flash_read_locked

Field	Bits	Default	Access	Description
flash_read_locked	2	0	W1C	Flash Read from Locked Area Interrupt Flag

FLC_INTFL1.trim_update_done

Field	Bits	Default	Access	Description
trim_update_done	3	0	W1C	Trim Update Complete Interrupt Flag

14.1.1.13 FLC_INTEN1

FLC_INTEN1.sram_addr_wrapped

Field	Bits	Default	Access	Description
sram_addr_wrapped	0	0	R/W	SRAM Address Wrapped Interrupt Enable/Disable

FLC_INTEN1.invalid_flash_addr

Field	Bits	Default	Access	Description
invalid_flash_addr	1	0	R/W	Invalid Flash Address Interrupt Enable/–Disable

FLC_INTEN1.flash_read_locked

Field	Bits	Default	Access	Description
flash_read_locked	2	0	R/W	Flash Read from Locked Area Interrupt Enable/Disable

FLC_INTEN1.trim_update_done

Field	Bits	Default	Access	Description
trim_update_done	3	0	R/W	Trim Update Complete Interrupt Enable/–Disable

14.1.1.14 FLC_DISABLE_XR0

Default	Access	Description
00000000h	R/W	Disable Flash Page Exec/Read Register 0

Each bit in this register controls the execution/read permission for one page of the main flash memory, as follows:

- 0 - Flash page operates normally.
- 1 - Flash page is removed from memory space; contents of this flash page will not be provided by the flash controller for read access or execution (code access)
- Bit 0 - Page 0 is forced to 0 (read only) since this page is required to be readable.
- ...
- Bit 31 - Page 31

14.1.1.15 FLC_DISABLE_XR1

Default	Access	Description
00000000h	R/W	Disable Flash Page Exec/Read Register 1

Each bit in this register controls the execution/read permission for one page of the main flash memory, as follows:

- 0 - Flash page operates normally.
- 1 - Flash page is removed from memory space; contents of this flash page will not be provided by the flash controller for read access or execution (code access)
- Bit 0 - Page 32
- ...
- Bit 31 - Page 63

14.1.1.16 FLC_DISABLE_XR2

Default	Access	Description
00000000h	R/W	Disable Flash Page Exec/Read Register 2

Each bit in this register controls the execution/read permission for one page of the main flash memory, as follows:

- 0 - Flash page operates normally.
- 1 - Flash page is removed from memory space; contents of this flash page will not be provided by the flash controller for read access or execution (code access)

14.1.1.17 FLC_DISABLE_XR3

Default	Access	Description
00000000h	R/W	Disable Flash Page Exec/Read Register 3

Each bit in this register controls the execution/read permission for one page of the main flash memory, as follows:

- 0 - Flash page operates normally.
- 1 - Flash page is removed from memory space; contents of this flash page will not be provided by the flash controller for read access or execution (code access)

14.1.1.18 FLC_DISABLE_WE0

Default	Access	Description
00000000h	R/W	Disable Flash Page Write/Erase Register 0

Each bit in this register controls the write/erase permission for one page of the main flash memory:

- 0 - Flash page operates normally.
- 1 - Flash page contents may not be modified using write operations or page erase operations.

Pages 0 through 31.

14.1.1.19 FLC_DISABLE_WE1

Default	Access	Description
00000000h	R/W	Disable Flash Page Write/Erase Register 1

Each bit in this register controls the write/erase permission for one page of the main flash memory:

- 0 - Flash page operates normally.
- 1 - Flash page contents may not be modified using write operations or page erase operations.

14.1.1.20 FLC_DISABLE_WE2

Default	Access	Description
00000000h	R/W	Disable Flash Page Write/Erase Register 2

Each bit in this register controls the write/erase permission for one page of the main flash memory:

- 0 - Flash page operates normally.
- 1 - Flash page contents may not be modified using write operations or page erase operations.

14.1.1.21 FLC_DISABLE_WE3

Default	Access	Description
00000000h	R/W	Disable Flash Page Write/Erase Register 3

Each bit in this register controls the write/erase permission for one page of the main flash memory:

- 0 - Flash page operates normally.
- 1 - Flash page contents may not be modified using write operations or page erase operations.

14.2 Registers (ICC)

14.2.1 Module ICC Registers

Address	Register	32b Word Len	Description
0x40080000	ICC_ID	1	Cache ID Register (INTERNAL USE ONLY)
0x40080004	ICC_MEM_CFG	1	Memory Configuration Register
0x40080100	ICC_CTRL_STAT	1	Control and Status
0x40080700	ICC_INVDT_ALL	1	Invalidate (Clear) Cache Control

14.2.1.1 ICC_ID

ICC_ID.rtl_version

Field	Bits	Default	Access	Description
rtl_version	5:0	xxxxxx	R/O	Version

Internal/test use only; not for use by application software.

ICC_ID.part_num

Field	Bits	Default	Access	Description
part_num	9:6	xxxx	R/O	Number ID

Internal/test use only; not for use by application software.

ICC_ID.cache_id

Field	Bits	Default	Access	Description
cache_id	15:10	xxxxxx	R/O	Cache ID

Internal/test use only; not for use by application software.

14.2.1.2 ICC_MEM_CFG

ICC_MEM_CFG.cache_size

Field	Bits	Default	Access	Description
cache_size	15:0	0002h	R/O	Instruction Cache Size

Size of the instruction cache memory (in KB).

For the **MAX32600**, this value is 2KB.

ICC_MEM_CFG.main_memory_size

Field	Bits	Default	Access	Description
main_memory_size	31:16	0008h	R/O	Internal Flash Memory Size

Size of the main internal flash memory (in KB, divided by 32).

For the **MAX32600**, this value is 256KB.

14.2.1.3 ICC_CTRL_STAT

ICC_CTRL_STAT.enable

Field	Bits	Default	Access	Description
enable	0	0	R/W	Cache Enable

- 0: Instruction cache is disabled (bypass mode)
- 1: Instruction cache is enabled

ICC_CTRL_STAT.ready

Field	Bits	Default	Access	Description
ready	16	0	R/O	Cache Ready Status

- 0: Cache is invalidated/in a reset state
- 1: Cache is active and ready for use

14.2.1.4 ICC_INVDT_ALL

Default	Access	Description
00000000h	W/O	Invalidate (Clear) Cache Control

Writing any value to this register triggers a cache invalidate operation. The Cache Ready Status bit will indicate when this has completed.

All reads return 0.

15 Trademarks and Service Marks

The following registered trademarks and registered service marks are referenced in the text of this document:

- ARM is a registered trademark and registered service mark and Cortex is a registered trademark of ARM Limited.
- The Bluetooth word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by Maxim is under license.