



THE DATASHEET OF MSP430FR6989IPNR

MSP430FR58xx, MSP430FR59xx, and MSP430FR6xx Family

User's Guide



Literature Number: SLAU367P
October 2012–Revised April 2020

Preface	45
1 System Resets, Interrupts, and Operating Modes, System Control Module (SYS)	47
1.1 System Control Module (SYS) Introduction	48
1.2 System Reset and Initialization	48
1.2.1 Device Initial Conditions After System Reset	50
1.3 Interrupts	50
1.3.1 (Non)Maskable Interrupts (NMIs)	51
1.3.2 SNMI Timing	51
1.3.3 Maskable Interrupts	51
1.3.4 Interrupt Processing	52
1.3.5 Interrupt Nesting	53
1.3.6 Interrupt Vectors	53
1.3.7 SYS Interrupt Vector Generators	54
1.4 Operating Modes	56
1.4.1 Low-Power Modes and Clock Requests	58
1.4.2 Entering and Exiting Low-Power Modes LPM0 Through LPM4	59
1.4.3 Low-Power Modes LPM3.5 and LPM4.5 (LPMx.5)	59
1.5 Principles for Low-Power Applications	61
1.6 Connection of Unused Pins	62
1.7 Reset Pin (RST /NMI) Configuration	62
1.8 Configuring JTAG Pins	62
1.9 Vacant Memory Space	63
1.10 Boot Code	63
1.11 Bootloader (BSL)	63
1.12 JTAG Mailbox (JMB) System	63
1.12.1 JMB Configuration	64
1.12.2 JMBOUT0 and JMBOUT1 Outgoing Mailbox	64
1.12.3 JMBIN0 and JMBIN1 Incoming Mailbox	64
1.12.4 JMB NMI Usage	64
1.13 JTAG and SBW Lock Mechanism Using the Electronic Fuse	64
1.13.1 JTAG and SBW Lock Without Password	65
1.13.2 JTAG and SBW Lock With Password	65
1.14 Device Descriptor Table	65
1.14.1 Identifying Device Type	66
1.14.2 TLV Descriptors	67
1.14.3 Calibration Values	68
1.15 SFR Registers	72
1.15.1 SFRIE1 Register	73
1.15.2 SFRIFG1 Register	74
1.15.3 SFRRPCR Register	75
1.16 SYS Registers	76
1.16.1 SYSCTL Register	77
1.16.2 SYSJMBC Register	78
1.16.3 SYSJMBI0 Register	79
1.16.4 SYSJMBI1 Register	79

1.16.5	SYSJMBO0 Register	80
1.16.6	SYSJMBO1 Register	80
1.16.7	SYSUNIV Register	81
1.16.8	SYSSNIV Register	81
1.16.9	SYSRSTIV Register	82
2	Power Management Module (PMM) and Supply Voltage Supervisor (SVS).....	83
2.1	Power Management Module (PMM) Introduction	84
2.2	PMM Operation	85
2.2.1	V _{CORE} and the Regulator	85
2.2.2	Supply Voltage Supervisor	85
2.2.3	Supply Voltage Supervisor - Power-Up	86
2.2.4	LPM3.5 and LPM4.5	86
2.2.5	Brownout Reset (BOR)	86
2.2.6	RST/NMI	87
2.2.7	PMM Interrupts	87
2.2.8	Port I/O Control	87
2.3	PMM Registers	88
2.3.1	PMMCTL0 Register (offset = 00h) [reset = 9640h]	89
2.3.2	PMMCTL1 Register (offset = 02h) [reset = 9600h]	90
2.3.3	PMMIFG Register (offset = 0Ah) [reset = 0000h]	91
2.3.4	PM5CTL0 Register (offset = 10h) [reset = 0001h]	92
3	Clock System (CS) Module	93
3.1	Clock System Introduction	94
3.2	Clock System Operation.....	96
3.2.1	CS Module Features for Low-Power Applications	96
3.2.2	LFXT Oscillator	96
3.2.3	HFXT Oscillator.....	97
3.2.4	Internal Very-Low-Power Low-Frequency Oscillator (VLO).....	98
3.2.5	Module Oscillator (MODOSC)	98
3.2.6	Digitally Controlled Oscillator (DCO).....	98
3.2.7	Operation From Low-Power Modes, Requested by Peripheral Modules	99
3.2.8	CS Module Fail-Safe Operation.....	100
3.2.9	Synchronization of Clock Signals.....	102
3.3	MemoryMap Registers	103
3.3.1	CTL0 Register (Offset = 0h) [reset = 9600h]	104
3.3.2	CTL1 Register (Offset = 2h) [reset = Ch]	105
3.3.3	CTL2 Register (Offset = 4h) [reset = 33h]	106
3.3.4	CTL3 Register (Offset = 6h) [reset = 33h]	107
3.3.5	CTL4 Register (Offset = 8h) [reset = CDC9h]	108
3.3.6	CTL5 Register (Offset = Ah) [reset = 00C5h]	110
3.3.7	CTL6 Register (Offset = Ch) [reset = 7h]	111
4	CPUX	112
4.1	MSP430X CPU (CPUX) Introduction.....	113
4.2	Interrupts	115
4.3	CPU Registers	116
4.3.1	Program Counter (PC)	116
4.3.2	Stack Pointer (SP)	116
4.3.3	Status Register (SR)	117
4.3.4	Constant Generator Registers (CG1 and CG2)	119
4.3.5	General-Purpose Registers (R4 to R15).....	120
4.4	Addressing Modes	122
4.4.1	Register Mode.....	123
4.4.2	Indexed Mode	124

4.4.3	Symbolic Mode	130
4.4.4	Absolute Mode	135
4.4.5	Indirect Register Mode	137
4.4.6	Indirect Autoincrement Mode	138
4.4.7	Immediate Mode	139
4.5	MSP430 and MSP430X Instructions	141
4.5.1	MSP430 Instructions	141
4.5.2	MSP430X Extended Instructions	146
4.6	Instruction Set Description	157
4.6.1	Extended Instruction Binary Descriptions	158
4.6.2	MSP430 Instructions	160
4.6.3	Extended Instructions	212
4.6.4	Address Instructions	255
5	32-Bit Hardware Multiplier (MPY32)	270
5.1	32-Bit Hardware Multiplier (MPY32) Introduction	271
5.2	MPY32 Operation	273
5.2.1	Operand Registers	274
5.2.2	Result Registers	275
5.2.3	Software Examples	276
5.2.4	Fractional Numbers	277
5.2.5	Putting It All Together	280
5.2.6	Indirect Addressing of Result Registers	283
5.2.7	Using Interrupts	283
5.2.8	Using DMA	284
5.3	MPY32 Registers	285
5.3.1	MPY32CTL0 Register	287
6	FRAM Controller Overview	288
6.1	FRAM Controller Overview	288
7	FRAM Controller (FRCTL)	289
7.1	FRAM Introduction	290
7.2	FRAM Organization	290
7.3	FRCTL Module Operation	290
7.4	Programming FRAM Devices	291
7.4.1	Programming FRAM With JTAG or Spy-Bi-Wire	291
7.4.2	Programming FRAM With the Bootloader (BSL)	291
7.4.3	Programming FRAM With a Custom Solution	291
7.5	Wait State Control	291
7.5.1	Wait State and Cache Hit	292
7.6	FRAM ECC	292
7.7	FRAM Write Back	292
7.8	FRAM Power Control	292
7.9	FRAM Cache	293
7.10	FRCTL Registers	294
7.10.1	FRCTL0 Register	295
7.10.2	GCCTL0 Register	296
7.10.3	GCCTL1 Register	297
8	FRAM Controller A (FRCTL_A)	298
8.1	FRAM Controller A (FRCTL_A) Introduction	299
8.2	FRAM Controller A (FRCTL_A) Operation	299
8.2.1	FRCTL_A Error Detection	299
8.2.2	Programming FRAM Memory Devices	300
8.2.3	Access Control	300

8.3	FRAM ECC	302
8.4	FRAM Power Control.....	302
8.5	FRAM Cache	303
8.6	FRCTL_A Registers	304
8.6.1	FRCTL0 Register (Offset = 0h) [reset = 9600h]	305
8.6.2	GCCTL0 Register (Offset = 4h) [reset = 4h]	307
8.6.3	GCCTL1 Register (Offset = 6h) [reset = 0h]	309
9	Memory Protection Unit (MPU)	311
9.1	Memory Protection Unit (MPU) Introduction	312
9.2	MPU Segments	313
9.2.1	Main Memory Segments	313
9.2.2	IP Encapsulation Segment	314
9.2.3	Segment Border Setting	315
9.2.4	IP Encapsulation Border Settings.....	316
9.2.5	Information Memory	317
9.3	MPU Access Management Settings.....	317
9.4	MPU Violations.....	318
9.4.1	Interrupt Vector Table and Reset Vector	318
9.4.2	Violation Handling	318
9.5	MPU Lock.....	318
9.6	How to Enable MPU and IPE Segments	318
9.6.1	IP Encapsulation (IPE) Instantiation Using IPE Signatures	319
9.6.2	IP Encapsulation Removal	320
9.7	MPU Registers	321
9.7.1	MPUCTL0 Register	322
9.7.2	MPUCTL1 Register	323
9.7.3	MPUSEGB2 Register	324
9.7.4	MPUSEGB1 Register	325
9.7.5	MPUSAM Register.....	326
9.7.6	MPUIPC0 Register	328
9.7.7	MPUIPSEGB2 Register	329
9.7.8	MPUIPSEGB1 Register	330
10	RAM Controller (RAMCTL)	331
10.1	RAM Controller (RAMCTL) Introduction	332
10.2	RAMCTL Operation.....	332
10.2.1	Considerations for Complete Power Down.....	332
10.2.2	DACCESSIE and DACCESSIFG Bits in RCCTL1 Register.....	332
10.3	RAMCTL Registers	334
10.3.1	CTL0 Register (Offset = 0h) [reset = 6900h]	335
10.3.2	CTL1 Register (Offset = 2h) [reset = 0h]	337
11	DMA Controller.....	338
11.1	Direct Memory Access (DMA) Introduction.....	339
11.2	DMA Operation.....	341
11.2.1	DMA Addressing Modes	341
11.2.2	DMA Transfer Modes	342
11.2.3	Initiating DMA Transfers	348
11.2.4	Halting Executing Instructions for DMA Transfers.....	349
11.2.5	Stopping DMA Transfers.....	349
11.2.6	DMA Channel Priorities	349
11.2.7	DMA Transfer Cycle Time	350
11.2.8	Using DMA With System Interrupts	350
11.2.9	DMA Controller Interrupts	350
11.2.10	Using the eUSCI_B I ² C Module With the DMA Controller	352

11.2.11	Using ADC12 With the DMA Controller	352
11.3	DMA Registers	353
11.3.1	DMACTL0 Register	355
11.3.2	DMACTL1 Register	356
11.3.3	DMACTL2 Register	357
11.3.4	DMACTL3 Register	358
11.3.5	DMACTL4 Register	359
11.3.6	DMAxCTL Register	360
11.3.7	DMAxSA Register	362
11.3.8	DMAxDA Register	363
11.3.9	DMAxSZ Register	364
11.3.10	DMAIV Register	365
12	Digital I/O	366
12.1	Digital I/O Introduction	367
12.2	Digital I/O Operation	368
12.2.1	Input Registers (PxIN)	368
12.2.2	Output Registers (PxOUT)	368
12.2.3	Direction Registers (PxDIR)	368
12.2.4	Pullup or Pulldown Resistor Enable Registers (PxREN)	368
12.2.5	Function Select Registers (PxSEL0, PxSEL1)	369
12.2.6	Port Interrupts	369
12.3	I/O Configuration	371
12.3.1	Configuration After Reset	371
12.3.2	Configuration of Unused Port Pins	372
12.3.3	Configuration for LPMx.5 Low-Power Modes	372
12.4	Digital I/O Registers	374
12.4.1	PxIV Register	388
12.4.2	PxIN Register	389
12.4.3	PxOUT Register	389
12.4.4	PxDIR Register	389
12.4.5	PxREN Register	390
12.4.6	PxSEL0 Register	390
12.4.7	PxSEL1 Register	390
12.4.8	PxSELC Register	391
12.4.9	PxIES Register	391
12.4.10	PxIE Register	391
12.4.11	PxIFG Register	392
13	Capacitive Touch I/O	393
13.1	Capacitive Touch I/O Introduction	394
13.2	Capacitive Touch I/O Operation	395
13.3	CapTouch Registers	396
13.3.1	CAPTIOxCTL Register (offset = 0Eh) [reset = 0000h]	397
14	AES256 Accelerator	398
14.1	AES Accelerator Introduction	399
14.2	AES Accelerator Operation	400
14.2.1	Load the Key (128-Bit, 192-Bit, or 256-Bit Key Length)	401
14.2.2	Load the Data (128-Bit State)	401
14.2.3	Read the Data (128-Bit State)	402
14.2.4	Trigger an Encryption or Decryption	402
14.2.5	Encryption	403
14.2.6	Decryption	404
14.2.7	Decryption Key Generation	405

14.2.8	AES Key Buffer	406
14.2.9	Using the AES Accelerator With Low-Power Modes.....	406
14.2.10	AES Accelerator Interrupts.....	406
14.2.11	DMA Operation and Implementing Block Cipher Modes.....	406
14.3	AES Accelerator Registers	417
14.3.1	AESACTL0 Register.....	418
14.3.2	AESACTL1 Register.....	420
14.3.3	AESASTAT Register	421
14.3.4	AESAKEY Register.....	422
14.3.5	AESADIN Register	423
14.3.6	AESADOUT Register	424
14.3.7	AESAXDIN Register.....	425
14.3.8	AESAXIN Register.....	426
15	CRC Module	427
15.1	Cyclic Redundancy Check (CRC) Module Introduction.....	428
15.2	CRC Standard and Bit Order.....	428
15.3	CRC Checksum Generation.....	429
15.3.1	CRC Implementation	429
15.3.2	Assembler Examples	430
15.4	CRC Registers	432
15.4.1	CRCDI Register.....	433
15.4.2	CRCDIRB Register	433
15.4.3	CRCINIRES Register.....	434
15.4.4	CRCRESR Register	434
16	CRC32 Module.....	435
16.1	Cyclic Redundancy Check (CRC32) Module Introduction.....	436
16.2	CRC Checksum Generation.....	436
16.2.1	CRC Standard and Bit Order.....	437
16.2.2	CRC Implementation	437
16.2.3	Assembler Examples.....	437
16.3	CRC32 Register Descriptions.....	439
16.3.1	CRC32 Registers	439
17	Low-Energy Accelerator (LEA) for Signal Processing	446
17.1	LEA Introduction	447
17.2	LEA Operation.....	447
17.2.1	Use the LEA in Programs.....	448
17.2.2	Where to Get the DSP Library	449
17.2.3	Where to Start.....	449
17.3	LEA Registers	449
18	Ultrasonic Sensing Solution (USS, USS_A)	450
18.1	Introduction	451
18.2	Operation of the USS Module.....	452
18.2.1	Auto Mode and Register Mode	454
18.2.2	Control Signals.....	455
18.3	Debug Features.....	457
19	Universal USS Power Supply (UUPS)	458
19.1	Introduction	459
19.2	USS Power-up Sequence	460
19.3	USS Power States.....	461
19.4	Interface to the ASQ (Acquisition Sequencer)	463
19.4.1	Start New Measurements.....	463
19.4.2	Stop Measurement Before Completion	464

19.4.3	USS Power State After Completion of Measurements	465
19.4.4	UUPSCTL.USSPWRUP Bit and UUPSCTL.USS_BUSY Bit	465
19.5	Interrupts	466
19.6	Debug Mode	466
19.7	UUPS Registers.....	467
19.7.1	UUPSIIIDX Register (Offset = 0h) [reset = 0h]	468
19.7.2	UUPSMIS Register (Offset = 2h) [reset = 0h].....	469
19.7.3	UUPSRIS Register (Offset = 4h) [reset = 0h].....	470
19.7.4	UUPSIMSC Register (Offset = 6h) [reset = 0h].....	471
19.7.5	UUPSICR Register (Offset = 8h) [reset = 0h].....	472
19.7.6	UUPSISR Register (Offset = Ah) [reset = 0h].....	473
19.7.7	UUPSDESCLO Register (Offset = Ch) [reset = 110h].....	474
19.7.8	UUPSDESCHI Register (Offset = Eh) [reset = BA10h].....	475
19.7.9	UUPSCTL Register (Offset = 10h) [reset = 800h].....	476
20	High-Speed PLL (HSPLL)	478
20.1	Introduction	479
20.2	OSC Control Register (HSPLLUSXTCTL).....	480
20.2.1	OSCEN Bit.....	480
20.2.2	OSCTYPE Bit	480
20.2.3	OSCSTATE Bit	480
20.2.4	XTOUTOFF Bit.....	480
20.3	PLL Control (CTL) Register	481
20.3.1	PLLM[5:0] Bits.....	481
20.3.2	PLLINFREQ Bit	481
20.3.3	PLL_LOCK Bit	481
20.3.4	USSXT Control Register	481
20.4	Start-up Sequence of the USSXT Oscillator	481
20.4.1	USSXT Start-up Behavior	482
20.5	Interrupts	482
20.6	HSPLL Registers.....	483
20.6.1	HSPLLIIDX Register (Offset = 0h) [reset = 0h]	484
20.6.2	HSPLLMIS Register (Offset = 2h) [reset = 0h]	485
20.6.3	HSPLLRIS Register (Offset = 4h) [reset = 0h].....	486
20.6.4	HSPLLIMSC Register (Offset = 6h) [reset = 0h].....	487
20.6.5	HSPLLICR Register (Offset = 8h) [reset = 0h].....	488
20.6.6	HSPLLISR Register (Offset = Ah) [reset = 0h]	489
20.6.7	HSPLLDESCLO Register (Offset = Ch) [reset = 110h].....	490
20.6.8	HSPLLDESCHI Register (Offset = Eh) [reset = BD10h].....	491
20.6.9	HSPLLCTL Register (Offset = 10h) [reset = 4000h]	492
20.6.10	HSPLLUSXTLCTL Register (Offset = 12h) [reset = 100h].....	494
21	Sequencer for Acquisition, Programmable Pulse Generator, and Physical Interface (SAPH, SAPH_A).....	495
21.1	Introduction	496
21.2	Programmable Pulse Generator (PPG or PPG_A) Block	497
21.2.1	Pulse Generation	498
21.2.2	Single Tone Generation.....	498
21.2.3	Dual Tone Generation.....	499
21.2.4	Trill Tone Generation.....	500
21.2.5	Multi Tone Generation	501
21.2.6	Excitation Pulse Frequency on PPG or PPG_A	503
21.2.7	Extra Excitation Pulse Frequency on PPG_A.....	503
21.2.8	Test Tone Generation	503
21.3	Physical Interface (PHY) Block	504

21.3.1	Output Channels (CH0_OUT and CH1_OUT)	504
21.3.2	Trim Registers for the Output Drivers and Termination Resistors	505
21.3.3	Input Channels (CH0_IN and CH1_IN)	506
21.3.4	External Bias (XPB0 and XPB1 on SAPH_A)	510
21.4	Acquisition Sequencer (ASQ)	510
21.4.1	Time Counter	511
21.4.2	Six Time Mark Events	512
21.4.3	Triggering the ASQ	512
21.4.4	Auto Mode and Register Mode	513
21.5	Ultra-Low-Power Bias Mode.....	514
21.6	Interrupts Triggers	515
21.7	DMA Triggers	515
21.8	SAPH and SAPH_A Registers	516
21.8.1	SAPHIIDX/SAPH_AIIDX Register (Offset = 0h) [reset = 0h].....	518
21.8.2	SAPHMIS/SAPH_AMIS Register (Offset = 2h) [reset = 0h].....	519
21.8.3	SAPHRIS/SAPH_ARIS Register (Offset = 4h) [reset = 0h]	520
21.8.4	SAPHIMSC/SAPH_AIMSC Register (Offset = 6h) [reset = 0h].....	521
21.8.5	SAPHICR/SAPH_AICR Register (Offset = 8h) [reset = 0h].....	522
21.8.6	SAPHISR/SAPH_AISR Register (Offset = Ah) [reset = 0h].....	523
21.8.7	SAPHDESCLO/SAPH_ADESCLO Register (Offset = Ch) [reset = 10h]	524
21.8.8	SAPHDESCHI/SAPH_ADESCHI Register (Offset = Eh) [reset = 5553h]	525
21.8.9	SAPHKEY/SAPH_AKEY Register (Offset = 10h) [reset = 0h]	526
21.8.10	SAPHOCTL0/SAPH_AOCTL0 Register (Offset = 12h) [reset = 0h].....	527
21.8.11	SAPHOCTL1/SAPH_AOCTL1 Register (Offset = 14h) [reset = 0h].....	528
21.8.12	SAPHOSEL/SAPH_AOSEL Register (Offset = 16h) [reset = 5h].....	529
21.8.13	SAPHCH0PUT/SAPH_ACH0PUT Register (Offset = 20h) [reset = 0h].....	530
21.8.14	SAPHCH0PDT/SAPH_ACH0PDT Register (Offset = 22h) [reset = 0h].....	531
21.8.15	SAPHCH0TT/SAPH_ACH0TT Register (Offset = 24h) [reset = 0h].....	532
21.8.16	SAPHCH1PUT /SAPH_ACH1PUT Register (Offset = 26h) [reset = 0h].....	533
21.8.17	SAPHCH1PDT/SAPH_ACH1PDT Register (Offset = 28h) [reset = 0h].....	534
21.8.18	SAPHCH1TT/SAPH_ACH1TT Register (Offset = 2Ah) [reset = 0h]	535
21.8.19	SAPHMCNF/SAPH_AMCNF Register (Offset = 2Ch) [reset = 2h]	536
21.8.20	SAPHTACTL/SAPH_ATACTL Register (Offset = 2Eh) [reset = 0h].....	537
21.8.21	SAPHICTL0 /SAPH_AICTL0 Register (Offset = 30h) [reset = 90h]	538
21.8.22	SAPHBCTL/SAPH_ABCTL Register (Offset = 34h) [reset = A1h].....	540
21.8.23	SAPHPGC/SAPH_APGC Register (Offset = 40h) [reset = 0h]	542
21.8.24	SAPHPGLPER/SAPH_APGLPER Register (Offset = 42h) [reset = 0h]	543
21.8.25	SAPHPGHPER/SAPH_APGHPER Register (Offset = 44h) [reset = 0h]	544
21.8.26	SAPHPGCTL/SAPH_APGCTL Register (Offset = 46h) [reset = 11h].....	545
21.8.27	SAPHPGTRIG/SAPH_APPGTRIG Register (Offset = 48h) [reset = 0h].....	547
21.8.28	SAPH_AXPGCTL Register (Offset = 4Ah) [reset = 0h]	548
21.8.29	SAPH_AXPGLPER Register (Offset = 4Ch) [reset = 0h]	549
21.8.30	SAPH_AXPGHPER Register (Offset = 4Eh) [reset = 0h].....	550
21.8.31	SAPHASCTL0/SAPH_AASCTL0 Register (Offset = 60h) [reset = 0h].....	551
21.8.32	SAPHASCTL1/SAPH_AASCTL1 Register (Offset = 62h) [reset = 0h].....	553
21.8.33	SAPHASQTRIG/SAPH_AASQTRIG Register (Offset = 64h) [reset = 0h].....	555
21.8.34	SAPHAPOL/SAPH_AAPOL Register (Offset = 66h) [reset = 0h].....	556
21.8.35	SAPHAPLEV /SAPH_AAPLEV Register (Offset = 68h) [reset = 0h]	557
21.8.36	SAPHAPHIZ /SAPH_AAPHIZ Register (Offset = 6Ah) [reset = 0h]	558
21.8.37	SAPHATM_A/SAPH_AATM_A Register (Offset = 6Eh) [reset = 0h].....	559
21.8.38	SAPHATM_B/SAPH_AATM_B Register (Offset = 70h) [reset = 0h]	560
21.8.39	SAPHATM_C/SAPH_AATM_C Register (Offset = 72h) [reset = 0h].....	561
21.8.40	SAPHATM_D/SAPH_AATM_D Register (Offset = 74h) [reset = 0h].....	562

21.8.41	SAPHATM_E/SAPH_AATM_E Register (Offset = 76h) [reset = 0h]	563
21.8.42	SAPHATM_F/SAPH_AATM_F Register (Offset = 78h) [reset = 0h]	564
21.8.43	SAPHTBCTL/SAPH_ATBCTL Register (Offset = 7Ah) [reset = 0h]	565
21.8.44	SAPHATIMLO/SAPH_AATIMLO Register (Offset = 7Ch) [reset = 0h]	566
21.8.45	SAPHATIMHI/SAPH_AATIMHI Register (Offset = 7Eh) [reset = 0h]	567
22	Sigma-Delta High Speed (SDHS)	568
22.1	Introduction	569
22.2	SDHS Functional Operation	569
22.2.1	Input Multiplexer	570
22.2.2	Third-Order Modulator	570
22.2.3	Digital Output	571
22.2.4	Data Transfer Controller (DTC) and Internal Data Buffer	578
22.2.5	PGA Gain Control	580
22.2.6	SDHS Power and Conversion Control	582
22.2.7	TRIGEN Bit and SDHS_LOCK Bit	584
22.2.8	AUTOSSDIS (Auto Conversion Start Disable) Bit	588
22.2.9	INTDLY (Interrupt Delay) bits	589
22.2.10	Total Sample Size	590
22.2.11	Window Comparator	591
22.2.12	Conditions to Stop Data Conversion	592
22.3	Interrupts	594
22.3.1	IIDX, Interrupt Vector Generator	594
22.4	Debug Mode	594
22.5	SDHS Registers	595
22.5.1	SDHSIIDX Register (Offset = 0h) [reset = 0h]	596
22.5.2	SDHSMIS Register (Offset = 2h) [reset = 0h]	597
22.5.3	SDHSRIS Register (Offset = 4h) [reset = 0h]	598
22.5.4	SDHSIMSC Register (Offset = 6h) [reset = 0h]	600
22.5.5	SDHSICR Register (Offset = 8h) [reset = 0h]	601
22.5.6	SDHSISR Register (Offset = Ah) [reset = 0h]	602
22.5.7	SDHSDESCLO Register (Offset = Ch) [reset = 110h]	603
22.5.8	SDHSDESCHI Register (Offset = Eh) [reset = BB10h]	604
22.5.9	SDHSCTL0 Register (Offset = 10h) [reset = 8001h]	605
22.5.10	SDHSCTL1 Register (Offset = 12h) [reset = 0h]	607
22.5.11	SDHSCTL2 Register (Offset = 14h) [reset = 0h]	608
22.5.12	SDHSCTL3 Register (Offset = 16h) [reset = 0h]	609
22.5.13	SDHSCTL4 Register (Offset = 18h) [reset = 0h]	610
22.5.14	SDHSCTL5 Register (Offset = 1Ah) [reset = 0h]	611
22.5.15	SDHSCTL6 Register (Offset = 1Ch) [reset = 19h]	613
22.5.16	SDHSCTL7 Register (Offset = 1Eh) [reset = Fh]	614
22.5.17	SDHSDT Register (Offset = 22h) [reset = 0h]	615
22.5.18	SDHSWINHITH Register (Offset = 24h) [reset = 0h]	616
22.5.19	SDHSWINLOTH Register (Offset = 26h) [reset = 0h]	617
22.5.20	SDHSDTCDA Register (Offset = 28h) [reset = 0h]	618
23	Metering Test Interface (MTIF)	619
23.1	MTIF Introduction	620
23.2	MTIF Operation	621
23.2.1	MTIF and RTC_C	621
23.2.2	Initialization of the MTIF	622
23.2.3	Setting the Pulse Rate	622
23.2.4	Reading Pulse Counter	623
23.2.5	Synchronizing Pulse Generator Timing to Application	623
23.2.6	Various Resets During MTIF Operation	623

23.2.7	PUC Reset During Register Access.....	623
23.2.8	Enabling the Pulse Generator and the Pulse Counter	624
23.3	MTIF Block Diagram.....	624
23.3.1	Test Interface Input	624
23.4	MTIF Registers.....	625
23.4.1	MTIFPGCNF Register (Offset = 0h) [reset = 6970h].....	626
23.4.2	MTIFPGKVAL Register (Offset = 2h) [reset = 6900h]	627
23.4.3	MTIFPGCTL Register (Offset = 4h) [reset = 6900h]	628
23.4.4	MTIFPGSR Register (Offset = 6h) [reset = 0h]	629
23.4.5	MTIFPCCNF Register (Offset = 8h) [reset = 9600h]	630
23.4.6	MTIFPCR Register (Offset = Ah) [reset = 0h].....	631
23.4.7	MTIFPCCTL Register (Offset = Ch) [reset = 0h]	632
23.4.8	MTIFPCSR Register (Offset = Eh) [reset = 0h].....	633
23.4.9	MTIFTPCTL Register (Offset = 10h) [reset = F00h]	634
24	Watchdog Timer (WDT_A).....	635
24.1	WDT_A Introduction	636
24.2	WDT_A Operation	638
24.2.1	Watchdog Timer Counter (WDTCNT).....	638
24.2.2	Watchdog Mode	638
24.2.3	Interval Timer Mode	638
24.2.4	Watchdog Timer Interrupts	638
24.2.5	Fail-Safe Features	639
24.2.6	Operation in Low-Power Modes	639
24.3	WDT_A Registers.....	640
24.3.1	WDTCTL Register	641
25	Timer_A.....	642
25.1	Timer_A Introduction	643
25.2	Timer_A Operation	645
25.2.1	16-Bit Timer Counter	645
25.2.2	Starting the Timer.....	645
25.2.3	Timer Mode Control	646
25.2.4	Capture/Compare Blocks	649
25.2.5	Output Unit	651
25.2.6	Timer_A Interrupts.....	655
25.3	Timer_A Registers.....	657
25.3.1	TAxCTL Register	658
25.3.2	TAxR Register.....	659
25.3.3	TAxCCCTLn Register	660
25.3.4	TAxCCRN Register	662
25.3.5	TAxIV Register	662
25.3.6	TAxEX0 Register	663
26	Timer_B.....	664
26.1	Timer_B Introduction	665
26.1.1	Similarities and Differences From Timer_A	665
26.2	Timer_B Operation	667
26.2.1	16-Bit Timer Counter	667
26.2.2	Starting the Timer.....	667
26.2.3	Timer Mode Control	668
26.2.4	Capture/Compare Blocks	671
26.2.5	Output Unit	674
26.2.6	Timer_B Interrupts.....	678
26.3	Timer_B Registers.....	680
26.3.1	TBxCTL Register	681

26.3.2	TBxR Register.....	683
26.3.3	TBxCCTLn Register	684
26.3.4	TBxCCRn Register	686
26.3.5	TBxIV Register	687
26.3.6	TBxEX0 Register	688
27	Real-Time Clock (RTC) Overview	689
27.1	RTC Overview.....	689
28	Real-Time Clock B (RTC_B)	690
28.1	Real-Time Clock RTC_B Introduction	691
28.2	RTC_B Operation.....	693
28.2.1	Real-Time Clock and Prescale Dividers	693
28.2.2	Real-Time Clock Alarm Function	693
28.2.3	Reading or Writing Real-Time Clock Registers.....	694
28.2.4	Real-Time Clock Interrupts	694
28.2.5	Real-Time Clock Calibration	696
28.2.6	Real-Time Clock Operation in LPM3.5 Low-Power Mode.....	697
28.3	RTC_B Registers	698
28.3.1	RTCCTL0 Register	700
28.3.2	RTCCTL1 Register	701
28.3.3	RTCCTL2 Register	702
28.3.4	RTCCTL3 Register	702
28.3.5	RTCSEC Register – Hexadecimal Format	703
28.3.6	RTCSEC Register – BCD Format	703
28.3.7	RTCMIN Register – Hexadecimal Format.....	704
28.3.8	RTCMIN Register – BCD Format.....	704
28.3.9	RTCHOUR Register – Hexadecimal Format.....	705
28.3.10	RTCHOUR Register – BCD Format	705
28.3.11	RTCDOW Register.....	706
28.3.12	RTCDAY Register – Hexadecimal Format.....	706
28.3.13	RTCDAY Register – BCD Format.....	706
28.3.14	RTCMON Register – Hexadecimal Format	707
28.3.15	RTCMON Register – BCD Format.....	707
28.3.16	RTCYEAR Register – Hexadecimal Format	708
28.3.17	RTCYEAR Register – BCD Format	708
28.3.18	RTCAMIN Register – Hexadecimal Format	709
28.3.19	RTCAMIN Register – BCD Format	709
28.3.20	RTCAHOUR Register – Hexadecimal Format	710
28.3.21	RTCAHOUR Register – BCD Format	710
28.3.22	RTCADOW Register	711
28.3.23	RTCADAY Register – Hexadecimal Format	712
28.3.24	RTCADAY Register – BCD Format.....	712
28.3.25	RTCPS0CTL Register	713
28.3.26	RTCPS1CTL Register	714
28.3.27	RTCPS0 Register	715
28.3.28	RTCPS1 Register	715
28.3.29	RTCIV Register.....	716
28.3.30	BIN2BCD Register	717
28.3.31	BCD2BIN Register	717
29	Real-Time Clock C (RTC_C)	718
29.1	Real-Time Clock (RTC_C) Introduction	719
29.2	RTC_C Operation.....	721
29.2.1	Calendar Mode.....	721

29.2.2	Real-Time Clock and Prescale Dividers	721
29.2.3	Real-Time Clock Alarm Function	721
29.2.4	Real-Time Clock Protection	722
29.2.5	Reading or Writing Real-Time Clock Registers	723
29.2.6	Real-Time Clock Interrupts	723
29.2.7	Real-Time Clock Calibration for Crystal Offset Error.....	725
29.2.8	Real-Time Clock Compensation for Crystal Temperature Drift.....	726
29.2.9	Real-Time Clock Operation in LPM3.5 Low-Power Mode.....	728
29.3	RTC_C Operation - Device-Dependent Features	729
29.3.1	Counter Mode	729
29.3.2	Real-Time Clock Event/Tamper Detection With Time Stamp.....	732
29.4	RTC_C Registers	734
29.4.1	RTCCTL0_L Register	737
29.4.2	RTCCTL0_H Register.....	738
29.4.3	RTCCTL1 Register	739
29.4.4	RTCCTL3 Register	740
29.4.5	RTCOCAL Register	740
29.4.6	RTCTCMP Register	741
29.4.7	RTCNT1 Register.....	742
29.4.8	RTCNT2 Register.....	742
29.4.9	RTCNT3 Register.....	742
29.4.10	RTCNT4 Register	742
29.4.11	RTCSEC Register – Calendar Mode With Hexadecimal Format	743
29.4.12	RTCSEC Register – Calendar Mode With BCD Format	743
29.4.13	RTCMIN Register – Calendar Mode With Hexadecimal Format.....	744
29.4.14	RTCMIN Register – Calendar Mode With BCD Format.....	744
29.4.15	RTCHOUR Register – Calendar Mode With Hexadecimal Format	745
29.4.16	RTCHOUR Register – Calendar Mode With BCD Format.....	745
29.4.17	RTCDOW Register – Calendar Mode	746
29.4.18	RTCDAY Register – Calendar Mode With Hexadecimal Format	746
29.4.19	RTCDAY Register – Calendar Mode With BCD Format	746
29.4.20	RTCMON Register – Calendar Mode With Hexadecimal Format	747
29.4.21	RTCMON Register – Calendar Mode With BCD Format	747
29.4.22	RTCYEAR Register – Calendar Mode With Hexadecimal Format	748
29.4.23	RTCYEAR Register – Calendar Mode With BCD Format	748
29.4.24	RTCAMIN Register – Calendar Mode With Hexadecimal Format	749
29.4.25	RTCAMIN Register – Calendar Mode With BCD Format	749
29.4.26	RTCAHOUR Register.....	750
29.4.27	RTCAHOUR Register – Calendar Mode With BCD Format.....	750
29.4.28	RTCADOW Register – Calendar Mode	751
29.4.29	RTCADAY Register – Calendar Mode With Hexadecimal Format	752
29.4.30	RTCADAY Register – Calendar Mode With BCD Format	752
29.4.31	RTCPS0CTL Register	753
29.4.32	RTCPS1CTL Register	754
29.4.33	RTCPS0 Register	756
29.4.34	RTCPS1 Register	756
29.4.35	RTCIV Register.....	757
29.4.36	BIN2BCD Register	758
29.4.37	BCD2BIN Register	758
29.4.38	RTCSECBAXx Register – Hexadecimal Format	759
29.4.39	RTCSECBAXx Register – BCD Format.....	759
29.4.40	RTCMINBAKx Register – Hexadecimal Format.....	760
29.4.41	RTCMINBAKx Register – BCD Format	760

29.4.42	RTCHOURBAKx Register – Hexadecimal Format.....	761
29.4.43	RTCHOURBAKx Register – BCD Format	761
29.4.44	RTCDAYBAKx Register – Hexadecimal Format	762
29.4.45	RTCDAYBAKx Register – BCD Format	762
29.4.46	RTCMONBAKx Register – Hexadecimal Format	763
29.4.47	RTCMONBAKx Register – BCD Format	763
29.4.48	RTCYEARBAKx Register – Hexadecimal Format	764
29.4.49	RTCYEARBAKx Register – BCD Format	764
29.4.50	RTCTCCTL0 Register	765
29.4.51	RTCTCCTL1 Register	765
29.4.52	RTCCAPxCTL Register	766
30	Enhanced Universal Serial Communication Interface (eUSCI) – UART Mode.....	767
30.1	Enhanced Universal Serial Communication Interface A (eUSCI_A) Overview	768
30.2	eUSCI_A Introduction – UART Mode	768
30.3	eUSCI_A Operation – UART Mode	770
30.3.1	eUSCI_A Initialization and Reset	770
30.3.2	Character Format	770
30.3.3	Asynchronous Communication Format	770
30.3.4	Automatic Baud-Rate Detection	773
30.3.5	IrDA Encoding and Decoding	774
30.3.6	Automatic Error Detection	775
30.3.7	eUSCI_A Receive Enable	776
30.3.8	eUSCI_A Transmit Enable	776
30.3.9	UART Baud-Rate Generation	777
30.3.10	Setting a Baud Rate	779
30.3.11	Transmit Bit Timing - Error calculation	780
30.3.12	Receive Bit Timing – Error Calculation	780
30.3.13	Typical Baud Rates and Errors	781
30.3.14	Using the eUSCI_A Module in UART Mode With Low-Power Modes	783
30.3.15	eUSCI_A Interrupts in UART Mode	784
30.3.16	DMA Operation	785
30.4	eUSCI_A UART Registers.....	786
30.4.1	UCAxCTLW0 Register	787
30.4.2	UCAxCTLW1 Register	788
30.4.3	UCAxBRW Register	789
30.4.4	UCAxMCTLW Register	789
30.4.5	UCAxSTATW Register	790
30.4.6	UCAxRXBUF Register	791
30.4.7	UCAxTXBUF Register	791
30.4.8	UCAxABCTL Register.....	792
30.4.9	UCAxIRCTL Register.....	793
30.4.10	UCAxIE Register	794
30.4.11	UCAxIFG Register	795
30.4.12	UCAxIV Register	796
31	Enhanced Universal Serial Communication Interface (eUSCI) – SPI Mode	797
31.1	Enhanced Universal Serial Communication Interfaces (eUSCI_A, eUSCI_B) Overview	798
31.2	eUSCI Introduction – SPI Mode	798
31.3	eUSCI Operation – SPI Mode.....	800
31.3.1	eUSCI Initialization and Reset	800
31.3.2	Character Format	801
31.3.3	Master Mode	801
31.3.4	Slave Mode	802
31.3.5	SPI Enable.....	803

31.3.6	Serial Clock Control	803
31.3.7	Using the SPI Mode With Low-Power Modes.....	804
31.3.8	eUSCI Interrupts in SPI Mode	804
31.4	eUSCI_A SPI Registers.....	806
31.4.1	UCAxCTLW0 Register	807
31.4.2	UCAxBRW Register	808
31.4.3	UCAxSTATW Register	809
31.4.4	UCAxRXBUF Register	810
31.4.5	UCAxTXBUF Register	811
31.4.6	UCAxIE Register	812
31.4.7	UCAxIFG Register	813
31.4.8	UCAxIV Register	814
31.5	eUSCI_B SPI Registers.....	815
31.5.1	UCBxCTLW0 Register	816
31.5.2	UCBxBRW Register	817
31.5.3	UCBxSTATW Register	817
31.5.4	UCBxRXBUF Register	818
31.5.5	UCBxTXBUF Register	818
31.5.6	UCBxIE Register	819
31.5.7	UCBxIFG Register	819
31.5.8	UCBxIV Register	820
32	Enhanced Universal Serial Communication Interface (eUSCI) – I²C Mode	821
32.1	Enhanced Universal Serial Communication Interface B (eUSCI_B) Overview	822
32.2	eUSCI_B Introduction – I ² C Mode	822
32.3	eUSCI_B Operation – I ² C Mode	823
32.3.1	eUSCI_B Initialization and Reset.....	824
32.3.2	I ² C Serial Data	824
32.3.3	I ² C Addressing Modes	825
32.3.4	I ² C Quick Setup	826
32.3.5	I ² C Module Operating Modes	827
32.3.6	Glitch Filtering	837
32.3.7	I ² C Clock Generation and Synchronization.....	837
32.3.8	Byte Counter	839
32.3.9	Multiple Slave Addresses.....	839
32.3.10	Using the eUSCI_B Module in I ² C Mode With Low-Power Modes	840
32.3.11	eUSCI_B Interrupts in I ² C Mode	840
32.4	eUSCI_B I2C Registers.....	844
32.4.1	UCBxCTLW0 Register	845
32.4.2	UCBxCTLW1 Register	847
32.4.3	UCBxBRW Register	849
32.4.4	UCBxSTATW.....	849
32.4.5	UCBxTBCNT Register	850
32.4.6	UCBxRXBUF Register	851
32.4.7	UCBxTXBUF	851
32.4.8	UCBxI2COA0 Register.....	852
32.4.9	UCBxI2COA1 Register.....	853
32.4.10	UCBxI2COA2 Register	853
32.4.11	UCBxI2COA3 Register	854
32.4.12	UCBxADDRX Register	854
32.4.13	UCBxADDMASK Register	855
32.4.14	UCBxI2CSA Register	855
32.4.15	UCBxIE Register	856
32.4.16	UCBxIFG Register	858

	32.4.17 UCBxIV Register	860
33	REF_A	861
33.1	REF_A Introduction	862
33.2	Principle of Operation	863
33.2.1	Low-Power Operation	863
33.2.2	Reference System Requests	863
33.3	REF_A Registers	865
33.3.1	REFCTL0 Register (offset = 00h) [reset = 0000h]	866
34	ADC12_B	868
34.1	ADC12_B Introduction	869
34.2	ADC12_B Operation	871
34.2.1	12-Bit ADC Core	871
34.2.2	ADC12_B Inputs and Multiplexer	872
34.2.3	Voltage References	872
34.2.4	Auto Power Down	873
34.2.5	Sample Frequency Mode Selection	873
34.2.6	Sample and Conversion Timing	873
34.2.7	Conversion Memory	876
34.2.8	ADC12_B Conversion Modes	877
34.2.9	Operation in LPM3 and LPM4	882
34.2.10	Window Comparator	882
34.2.11	Using the Integrated Temperature Sensor	883
34.2.12	ADC12_B Grounding and Noise Considerations	884
34.2.13	ADC12_B Calibration	885
34.2.14	ADC12_B Interrupts	885
34.3	ADC12_B Registers	887
34.3.1	ADC12CTL0 Register (offset = 00h) [reset = 0000h]	893
34.3.2	ADC12CTL1 Register (offset = 02h) [reset = 0000h]	895
34.3.3	ADC12CTL2 Register (offset = 04h) [reset = 0020h]	897
34.3.4	ADC12CTL3 Register (offset = 06h) [reset = 0000h]	898
34.3.5	ADC12MEMx Register (x = 0 to 31)	899
34.3.6	ADC12MCTLx Register (x = 0 to 31)	900
34.3.7	ADC12HI Register (offset = 0Ah) [reset = 0FFFh]	902
34.3.8	ADC12LO Register (offset = 08h) [reset = 0000h]	902
34.3.9	ADC12IER0 Register (offset = 12h) [reset = 0000h]	903
34.3.10	ADC12IER1 Register (offset = 14h) [reset = 0000h]	905
34.3.11	ADC12IER2 Register (offset = 16h) [reset = 0000h]	907
34.3.12	ADC12IFGR0 Register (offset = 0Ch) [reset = 0000h]	908
34.3.13	ADC12IFGR1 Register (offset = 0Eh) [reset = 0000h]	910
34.3.14	ADC12IFGR2 Register (offset = 10h) [reset = 0000h]	912
34.3.15	ADC12IV Register (offset = 18h) [reset = 0000h]	913
35	Comparator E (COMP_E) Module	915
35.1	COMP_E Introduction	916
35.2	COMP_E Operation	917
35.2.1	Comparator	917
35.2.2	Analog Input Switches	917
35.2.3	Port Logic	917
35.2.4	Input Short Switch	917
35.2.5	Output Filter	918
35.2.6	Reference Voltage Generator	919
35.2.7	Port Disable Register (CEPD)	920
35.2.8	Comparator_E Interrupts	920
35.2.9	Comparator_E Used to Measure Resistive Elements	920

35.3	COMP_E Registers.....	923
35.3.1	CECTL0 Register (offset = 00h) [reset = 0000h]	924
35.3.2	CECTL1 Register (offset = 02h) [reset = 0000h]	925
35.3.3	CECTL2 Register (offset = 04h) [reset = 0000h]	926
35.3.4	CECTL3 Register (offset = 06h) [reset = 0000h]	927
35.3.5	CEINT Register (offset = 0Ch) [reset = 0000h]	929
35.3.6	CEIV Register (offset = 0Eh) [reset = 0000h]	930
36	LCD_C Controller.....	931
36.1	LCD_C Introduction.....	932
36.2	LCD_C Operation.....	934
36.2.1	LCD Memory	934
36.2.2	LCD Timing Generation.....	935
36.2.3	Blanking the LCD	936
36.2.4	LCD Blinking.....	936
36.2.5	LCD Voltage And Bias Generation	937
36.2.6	LCD Outputs.....	940
36.2.7	LCD Interrupts.....	941
36.2.8	Static Mode	943
36.2.9	2-Mux Mode	944
36.2.10	3-Mux Mode.....	945
36.2.11	4-Mux Mode.....	946
36.2.12	6-Mux Mode.....	947
36.2.13	8-Mux Mode.....	948
36.3	LCD_C Registers	950
36.3.1	LDCCTL0 Register	955
36.3.2	LDCCTL1 Register	957
36.3.3	LDCBLKCTL Register.....	958
36.3.4	LDCMEMCTL Register.....	959
36.3.5	LDCVCTL Register	960
36.3.6	LDCPCTL0 Register.....	962
36.3.7	LDCPCTL1 Register.....	962
36.3.8	LDCPCTL2 Register.....	963
36.3.9	LDCPCTL3 Register.....	963
36.3.10	LDCCPCTL Register.....	964
36.3.11	LDCIV Register	964
37	Extended Scan Interface (ESI)	965
37.1	ESI Introduction	966
37.2	ESI Operation	967
37.2.1	ESI Analog Front End	967
37.2.2	ESI Timing State Machine.....	974
37.2.3	ESI Pre-Processing and State Storage	979
37.2.4	TimerA Output Stage.....	980
37.2.5	ESI Processing State Machine.....	981
37.2.6	ESI Debug Register	985
37.2.7	ESI Interrupts.....	985
37.2.8	Overview of ESI Applications	986
37.3	ESI Registers.....	993
37.3.1	ESIDEBUG1 Register	994
37.3.2	ESIDEBUG2 Register	994
37.3.3	ESIDEBUG3 Register	994
37.3.4	ESIDEBUG4 Register	995
37.3.5	ESIDEBUG5 Register	995
37.3.6	ESICNT0 Register	996

37.3.7	ESICNT1 Register	996
37.3.8	ESICNT2 Register	997
37.3.9	ESICNT3 Register	997
37.3.10	ESIIV Register	998
37.3.11	ESIINT1 Register	999
37.3.12	ESIINT2 Register	1001
37.3.13	ESIAFE Register	1003
37.3.14	ESIPPU Register	1005
37.3.15	ESITSM Register	1006
37.3.16	ESIPSM Register	1008
37.3.17	ESIOSC Register	1009
37.3.18	ESICTL Register	1010
37.3.19	ESITHR1 Register	1012
37.3.20	ESITHR2 Register	1012
37.3.21	ESIDAC1Rx Register (x = 0 to 7)	1013
37.3.22	ESIDAC2Rx Register (x = 0 to 7)	1013
37.3.23	ESITSMx Register (x = 0 to 31)	1014
37.3.24	Extended Scan Interface Processing State Machine Table Entry (ESI Memory)	1016
38	Embedded Emulation Module (EEM).....	1017
38.1	Embedded Emulation Module (EEM) Introduction	1018
38.2	EEM Building Blocks.....	1020
38.2.1	Triggers	1020
38.2.2	Trigger Sequencer	1020
38.2.3	State Storage (Internal Trace Buffer)	1020
38.2.4	Cycle Counter.....	1020
38.2.5	EnergyTrace++ Technology	1021
38.2.6	Clock Control	1021
38.2.7	Debug Modes	1021
38.3	EEM Configurations.....	1021
	Revision History	1023

List of Figures

1-1.	BOR, POR, and PUC Reset Circuit.....	49
1-2.	Interrupt Priority.....	50
1-3.	Interrupt Processing.....	52
1-4.	Return From Interrupt.....	53
1-5.	Operation Modes	57
1-6.	Devices Descriptor Table.....	66
1-7.	SFRIE1 Register	73
1-8.	SFRIFG1 Register.....	74
1-9.	SFRRPCR Register	75
1-10.	SYSCTL Register	77
1-11.	SYSJMBC Register	78
1-12.	SYSJMBI0 Register	79
1-13.	SYSJMBI1 Register	79
1-14.	SYSJMBO0 Register.....	80
1-15.	SYSJMBO1 Register.....	80
1-16.	SYSUNIV Register	81
1-17.	SYSSNIV Register	81
1-18.	SYSRSTIV Register.....	82
2-1.	PMM Block Diagram	84
2-2.	Voltage Failure and Resulting PMM Actions	85
2-3.	PMM Action at Device Power-Up	86
2-4.	PMMCTL0 Register	89
2-5.	PMMCTL1 Register	90
2-6.	PMMIFG Register	91
2-7.	PM5CTL0 Register	92
3-1.	Clock System Block Diagram.....	95
3-2.	Module Request Clock System.....	99
3-3.	Oscillator Fault Logic	101
3-4.	Switch MCLK From DCOCLK to LFXCLK	102
3-5.	CTL0 Register	104
3-6.	CTL1 Register	105
3-7.	CTL2 Register	106
3-8.	CTL3 Register	107
3-9.	CTL4 Register	108
3-10.	CTL5 Register	110
3-11.	CTL6 Register	111
4-1.	MSP430X CPU Block Diagram	114
4-2.	PC Storage on the Stack for Interrupts	115
4-3.	Program Counter.....	116
4-4.	PC Storage on the Stack for CALLA	116
4-5.	Stack Pointer	117
4-6.	Stack Usage	117
4-7.	PUSHX.A Format on the Stack	117
4-8.	PUSH SP, POP SP Sequence.....	117
4-9.	SR Bits	118
4-10.	Register-Byte and Byte-Register Operation.....	120
4-11.	Register-Word Operation	120

4-12.	Word-Register Operation	121
4-13.	Register – Address-Word Operation	121
4-14.	Address-Word – Register Operation	122
4-15.	Indexed Mode in Lower 64KB	124
4-16.	Indexed Mode in Upper Memory	125
4-17.	Overflow and Underflow for Indexed Mode	126
4-18.	Example for Indexed Mode	127
4-19.	Symbolic Mode Running in Lower 64KB	130
4-20.	Symbolic Mode Running in Upper Memory	131
4-21.	Overflow and Underflow for Symbolic Mode	132
4-22.	MSP430 Double-Operand Instruction Format	141
4-23.	MSP430 Single-Operand Instructions	142
4-24.	Format of Conditional Jump Instructions	143
4-25.	Extension Word for Register Modes	146
4-26.	Extension Word for Non-Register Modes	146
4-27.	Example for Extended Register or Register Instruction	147
4-28.	Example for Extended Immediate or Indexed Instruction	148
4-29.	Extended Format I Instruction Formats	149
4-30.	20-Bit Addresses in Memory	149
4-31.	Extended Format II Instruction Format	150
4-32.	PUSHM and POPM Instruction Format	151
4-33.	RRCM, RRAM, RRUM, and RLAM Instruction Format	151
4-34.	BRA Instruction Format	151
4-35.	CALLA Instruction Format	151
4-36.	Decrement Overlap	177
4-37.	Stack After a RET Instruction	196
4-38.	Destination Operand—Arithmetic Shift Left	198
4-39.	Destination Operand—Carry Left Shift	199
4-40.	Rotate Right Arithmetically RRA.B and RRA.W	200
4-41.	Rotate Right Through Carry RRC.B and RRC.W	201
4-42.	Swap Bytes in Memory	208
4-43.	Swap Bytes in a Register	208
4-44.	Rotate Left Arithmetically—RLAM[.W] and RLAM.A	235
4-45.	Destination Operand-Arithmetic Shift Left	236
4-46.	Destination Operand-Carry Left Shift	237
4-47.	Rotate Right Arithmetically RRAM[.W] and RRAM.A	238
4-48.	Rotate Right Arithmetically RRAX(.B,.A) – Register Mode	240
4-49.	Rotate Right Arithmetically RRAX(.B,.A) – Non-Register Mode	240
4-50.	Rotate Right Through Carry RRCM[.W] and RRCM.A	242
4-51.	Rotate Right Through Carry RRCX(.B,.A) – Register Mode	244
4-52.	Rotate Right Through Carry RRCX(.B,.A) – Non-Register Mode	244
4-53.	Rotate Right Unsigned RRUM[.W] and RRUM.A	245
4-54.	Rotate Right Unsigned RRUX(.B,.A) – Register Mode	246
4-55.	Swap Bytes SWPBX.A Register Mode	250
4-56.	Swap Bytes SWPBX.A In Memory	250
4-57.	Swap Bytes SWPBX[.W] Register Mode	251
4-58.	Swap Bytes SWPBX[.W] In Memory	251
4-59.	Sign Extend SCTX.A	252
4-60.	Sign Extend SCTX[.W]	252

5-1.	MPY32 Block Diagram	272
5-2.	Q15 Format Representation.....	277
5-3.	Q14 Format Representation.....	277
5-4.	Saturation Flow Chart	279
5-5.	Multiplication Flow Chart.....	281
5-6.	MPY32CTL0 Register	287
7-1.	FRAM Controller Block Diagram.....	290
7-2.	FRAM Power Control Diagram.....	293
7-3.	FRCTL0 Register	295
7-4.	GCCTL0 Register	296
7-5.	GCCTL1 Register	297
8-1.	FRCTL_A Block Diagram	299
8-2.	FRAM Power Control Diagram.....	303
8-3.	FRCTL0 Register	305
8-4.	GCCTL0 Register	307
8-5.	GCCTL1 Register	309
9-1.	Memory Protection Unit Overview	312
9-2.	Segment Border Register	313
9-3.	Example of Segment Border Register Fixed Bits When FRAM Size = 128KB	313
9-4.	Example of Segment Border Register Fixed Bits When FRAM Size = 256KB	313
9-5.	Segmentation of Main Memory	314
9-6.	IP Encapsulation Access Rights Equivalent Schematic	315
9-7.	MPUCTL0 Register	322
9-8.	MPUCTL1 Register	323
9-9.	MPUSEGB2 Register	324
9-10.	MPUSEGB1 Register	325
9-11.	MPUSAM Register.....	326
9-12.	MPUIPC0 Register	328
9-13.	MPUIPSEGB2 Register	329
9-14.	MPUIPSEGB1 Register	330
10-1.	RAM Power Mode Transitions Into and Out of LPM3 or LPM4.....	332
10-2.	CTL0 Register	335
10-3.	CTL1 Register	337
11-1.	DMA Controller Block Diagram	340
11-2.	DMA Addressing Modes	341
11-3.	DMA Single Transfer State Diagram	343
11-4.	DMA Block Transfer State Diagram	345
11-5.	DMA Burst-Block Transfer State Diagram.....	347
11-6.	DMACTL0 Register	355
11-7.	DMACTL1 Register	356
11-8.	DMACTL2 Register	357
11-9.	DMACTL3 Register	358
11-10.	DMACTL4 Register	359
11-11.	DMAxCTL Register	360
11-12.	DMAxSA Register	362
11-13.	DMAxDA Register	363
11-14.	DMAxSZ Register.....	364
11-15.	DMAIV Register	365
12-1.	PxIV Register	388

12-2.	PxIN Register.....	389
12-3.	PxOUT Register.....	389
12-4.	PxDIR Register.....	389
12-5.	PxREN Register.....	390
12-6.	PxSEL0 Register.....	390
12-7.	PxSEL1 Register.....	390
12-8.	PxSELC Register	391
12-9.	PxIES Register	391
12-10.	PxIE Register	391
12-11.	PxIFG Register	392
13-1.	Capacitive Touch I/O Principle	394
13-2.	Capacitive Touch I/O Block Diagram.....	395
13-3.	CAPTIOxCTL Register	397
14-1.	AES Accelerator Block Diagram	399
14-2.	AES State Array Input and Output	400
14-3.	AES Encryption Process for 128-Bit Key	403
14-4.	AES Decryption Process Using AESOPx = 01 for 128-Bit Key	404
14-5.	AES Decryption Process Using AESOPx = 10 and 11 for 128-bit Key.....	405
14-6.	ECB Encryption	408
14-7.	ECB Decryption	409
14-8.	CBC Encryption	410
14-9.	CBC Decryption	411
14-10.	OFB Encryption	413
14-11.	OFB Decryption	414
14-12.	CFB Encryption	415
14-13.	CFB Decryption	416
14-14.	AESACTL0 Register.....	418
14-15.	AESACTL1 Register.....	420
14-16.	AESASTAT Register	421
14-17.	AESAKEY Register	422
14-18.	AESADIN Register.....	423
14-19.	AESADOUT Register.....	424
14-20.	AESAXDIN Register.....	425
14-21.	AESAXIN Register.....	426
15-1.	LFSR Implementation of CRC-CCITT Standard, Bit 0 is the MSB of the Result.....	428
15-2.	Implementation of CRC-CCITT Using the CRCDI and CRCINIRES Registers	430
15-3.	CRCDI Register	433
15-4.	CRCDIRB Register	433
15-5.	CRCINIRES Register.....	434
15-6.	CRCRESR Register	434
16-1.	LFSR Implementation of CRC-CCITT as Defined in Standard (Bit 0 is MSB).....	436
16-2.	LFSR Implementation of CRC32-ISO3309 as Defined in Standard (Bit 0 is MSB)	436
16-3.	CRC32DIW0 Register	440
16-4.	CRC32DIW1 Register	440
16-5.	CRC32DIRBW0 Register	441
16-6.	CRC32DIRBW1 Register	441
16-7.	CRC32INIRESW0 Register.....	442
16-8.	CRC32INIRESW1 Register.....	442
16-9.	CRC32RESRW0 Register	443

16-10. CRC32RESRW1 Register	443
16-11. CRC16DIW0 Register	444
16-12. CRC16DIRBW0 Register	444
16-13. CRC16NIRESW0 Register.....	445
16-14. CRC16RESRW0 Register	445
17-1. LEA System Block Diagram	447
18-1. USS and USS_A Block Diagram	452
18-2. USS and USS_A Submodule Connections	453
19-1. USS/USS_A Block Diagram.....	459
19-2. UUPS Block Diagram	460
19-3. USS Power State Control Flow	462
19-4. USS Power Control	463
19-5. UUPSIDX Register.....	468
19-6. UUPSMIS Register	469
19-7. UUPSRIS Register	470
19-8. UUPSIMSC Register	471
19-9. UUPSICR Register	472
19-10. UUPSISR Register	473
19-11. UUPSDESCLO Register.....	474
19-12. UUPSDESCHI Register.....	475
19-13. UUPSCTL Register.....	476
20-1. USS or USS_A Block Diagram	479
20-2. HSPLL Block Diagram	479
20-3. HSPLLIIDX Register.....	484
20-4. HSPLLMIS Register	485
20-5. HSPLLRIS Register	486
20-6. HSPLLMISC Register	487
20-7. HSPLLICR Register	488
20-8. HSPLISR Register	489
20-9. HSPLDESCLO Register.....	490
20-10. HSPLDESCHI Register.....	491
20-11. HSPLLCTL Register.....	492
20-12. HSPLLUSSXTLCTL Register	494
21-1. USS or USS_A Block Diagram	497
21-2. PPG or PPG_A Block Diagram	497
21-3. PPG or PPG_A Internal State Diagrams for Single Tone	498
21-4. PPG Single Tone Generation With SAPHPGC.PPOL = 0 (Starts With High Polarity)	499
21-5. PPG Single Tone Generation With SAPHPGC.PPOL = 1 (Starts With Low Polarity)	499
21-6. PPG_A State Diagram for Dual Tone	500
21-7. PPG_A Dual Tone Generation With SAPHPGC.PPOL = 1 (Starts With Low Polarity).....	500
21-8. PPG_A State Diagram for Trill Tone	501
21-9. PPG_A Trill Tone Generation With SAPHPGC.PPOL = 1 (Starts With Low Polarity).....	501
21-10. PPG_A Software Flow Chart for Multi Tone	502
21-11. PPG_A Multi Tone Generation With SAPHPGC.PPOL = 1 (Starts With Low Polarity)	502
21-12. PHY Output Pins.....	504
21-13. SAPH or SAPH_A Analog Input Signal Chain	506
21-14. Before Excitation.....	507
21-15. Excitation.....	508
21-16. Before Reception	509

21-17. Reception	510
21-18. ASQ Block Diagram	511
21-19. Auto Mode and Register Mode Example	514
21-20. Ultra-Low-Power Bias Mode Example	515
21-21. SAPHIIDX/SAPH_AIIDX Register	518
21-22. SAPHMIS/SAPH_AMIS Register	519
21-23. SAPHRIS/SAPH_ARIS Register	520
21-24. SAPHIMSC/SAPH_AIMSC Register	521
21-25. SAPHICR/SAPH_AICR Register	522
21-26. SAPHISR/SAPH_AISR Register	523
21-27. SAPHDESCLO/SAPH_ADESCLO Register	524
21-28. SAPHDESCHI/SAPH_ADESCHI Register	525
21-29. SAPHKEY/SAPH_AKEY Register	526
21-30. SAPHOCTL0/SAPH_AOCTL0 Register	527
21-31. SAPHOCTL1/SAPH_AOCTL1 Register	528
21-32. SAPHOSEL/SAPH_AOSEL Register	529
21-33. SAPHCH0PUT/SAPH_ACH0PUT Register	530
21-34. SAPHCH0PDT/SAPH_ACH0PDT Register	531
21-35. SAPHCH0TT/SAPH_ACH0TT Register	532
21-36. SAPHCH1PUT/SAPH_ACH1PUT Register	533
21-37. SAPHCH1PDT/SAPH_ACH1PDT Register	534
21-38. SAPHCH1TT/SAPH_ACH1TT Register	535
21-39. SAPHMCNF/SAPH_AMCNF Register	536
21-40. SAPHTACTL/SAPH_ATACTL Register	537
21-41. SAPHICTL0/SAPH_AICTL0 Register	538
21-42. SAPHBCTL/SAPH_ABCTL Register	540
21-43. SAPHPGC/SAPH_APGC Register	542
21-44. SAPHPGLPER/SAPH_APGLPER Register	543
21-45. SAPHPGHPER/SAPH_APGHPER Register	544
21-46. SAPHPGCTL/SAPH_APGCTL Register	545
21-47. SAPHPPGTRIG/SAPH_APPGTRIG Register	547
21-48. SAPH_AXPCTL Register	548
21-49. SAPH_AXPGLPER Register	549
21-50. SAPH_AXPGLPER Register	550
21-51. SAPHASCTL0/SAPH_AASCTL0 Register	551
21-52. SAPHASCTL1/SAPH_AASCTL1 Register	553
21-53. SAPHASQTRIG/SAPH_AASQTRIG Register	555
21-54. SAPHAPOL/SAPH_AAPOL Register	556
21-55. SAPHAPLEV/SAPH_AAPLEV Register	557
21-56. SAPHAPHIZ/SAPH_AAPHIZ Register	558
21-57. SAPHATM_A/SAPH_AATM_A Register	559
21-58. SAPHATM_B/SAPH_AATM_B Register	560
21-59. SAPHATM_C/SAPH_AATM_C Register	561
21-60. SAPHATM_D/SAPH_AATM_D Register	562
21-61. SAPHATM_E/SAPH_AATM_E Register	563
21-62. SAPHATM_F/SAPH_AATM_F Register	564
21-63. SAPHTBCTL/SAPH_ATBCTL Register	565
21-64. SAPHATIMLO/SAPH_AATIMLO Register	566
21-65. SAPHATIMHI/SAPH_AATIMHI Register	567

22-1. USS Block Diagram	569
22-2. SDHS Block Diagram	570
22-3. Sigma-Delta Principle	571
22-4. CIC ⁷ Filter Structure	572
22-5. SDHS Filter Frequency Response, SDHSCTL1.OSR = 10	572
22-6. SDHS Filter Frequency Response Within f_s , SDHSCTL1.OSR = 10	573
22-7. Digital Filter Block Diagram.....	573
22-8. SDHS Filter Frequency Response, SDHSCTL1.OSR = 20	574
22-9. SDHS Filter Frequency Response Within f_s , SDHSCTL1.OSR = 20	574
22-10. SDHS Filter Frequency Response within f_s , SDHSCTL1.OSR = 40	575
22-11. SDHS Filter Frequency Response within f_s , SDHSCTL1.OSR = 80	575
22-12. SDHS Filter Frequency Response within f_s , SDHSCTL1.OSR = 160	576
22-13. Bits Selection From Filter to the Data Register (SDHSCTL0.DALGN = 0).....	577
22-14. Bits Selection From Filter to the Data Register (SDHSCTL0.DALGN = 1).....	578
22-15. Data Output Path	580
22-16. SDHS Power and Conversion Trigger Source	582
22-17. SDHS Operation in Register Mode (SDHSCTL0.TRGSRC = 0)	583
22-18. SDHS Operation as Part of USS Measurement (SDHSCTL0.TRGSRC = 1)	584
22-19. Example Using SDHSCTL3.TRIGEN Bit (SDHSCTL0.AUTOSSDIS = 0).....	586
22-20. Example Using SDHSCTL3.TRIGEN bit (SDHSCTL0.AUTOSSDIS = 1).....	587
22-21. Conversion Start and Stop When SDHSCTL0.AUTOSSDIS = 0.....	588
22-22. Conversion Start and Stop When SDHSCTL0.AUTOSSDIS = 1.....	589
22-23. First Interrupt Position With SDHSCTL0.INTDLY = 2.....	589
22-24. SDHSCTL0.AUTOSSDIS = 0, SDHSCTL2.SMPCTLOFF = 0, SDHSCTL0.INTDLY = 0, Total Sample Size is Controlled by SDHSCTL2.SMPSZ	590
22-25. SDHSCTL0.AUTOSSDIS = 1, SDHSCTL2.SMPCTLOFF = 0, SDHSCTL0.INTDLY = 0, Total Sample Size is Controlled by SDHSCTL2.SMPSZ.....	591
22-26. SDHSCTL0.AUTOSSDIS = 1, SDHSCTL2.SMPCTLOFF = 0, SDHSCTL0.INTDLY = m, Total Sample Size is Controlled by SDHSCTL2.SMPSZ.....	591
22-27. SDHSIIDX Register.....	596
22-28. SDHSMIS Register	597
22-29. SDHSRIS Register	598
22-30. SDHSIMSC Register	600
22-31. SDHSICR Register	601
22-32. SDHSISR Register	602
22-33. SDHSDESCLO Register.....	603
22-34. SDHSDESCHI Register.....	604
22-35. SDHSCTL0 Register	605
22-36. SDHSCTL1 Register	607
22-37. SDHSCTL2 Register	608
22-38. SDHSCTL3 Register	609
22-39. SDHSCTL4 Register	610
22-40. SDHSCTL5 Register	611
22-41. SDHSCTL6 Register	613
22-42. SDHSCTL7 Register	614
22-43. SDHSDT Register	615
22-44. SDHSWINHITH Register	616
22-45. SDHSWINLOTH Register	617
22-46. SDHSDTCDA Register.....	618
23-1. MTIF Use Case	620

23-2.	MTIF Pulse Diagram	621
23-3.	MTIF Internal Interfaces.....	621
23-4.	MTIF Block Diagram.....	624
23-5.	MTIFPGCNF Register.....	626
23-6.	MTIFPGKVAL Register	627
23-7.	MTIFPGCTL Register	628
23-8.	MTIFPGSR Register	629
23-9.	MTIFPCCNF Register	630
23-10.	MTIFPCR Register	631
23-11.	MTIFPCCTL Register	632
23-12.	MTIFPCSR Register.....	633
23-13.	MTIFTPCTL Register.....	634
24-1.	Watchdog Timer Block Diagram	637
24-2.	WDTCTL Register	641
25-1.	Timer_A Block Diagram	644
25-2.	Up Mode	646
25-3.	Up Mode Flag Setting	646
25-4.	Continuous Mode	647
25-5.	Continuous Mode Flag Setting.....	647
25-6.	Continuous Mode Time Intervals	647
25-7.	Up/Down Mode.....	648
25-8.	Up/Down Mode Flag Setting	648
25-9.	Output Unit in Up/Down Mode	649
25-10.	Capture Signal (SCS = 1).....	650
25-11.	Capture Cycle	650
25-12.	Output Example – Timer in Up Mode	652
25-13.	Output Example – Timer in Continuous Mode	653
25-14.	Output Example – Timer in Up/Down Mode	654
25-15.	Capture/Compare Interrupt Flag	655
25-16.	TAxCTL Register.....	658
25-17.	TAxR Register	659
25-18.	TAxCCtLn Register	660
25-19.	TAxCCRn Register	662
25-20.	TAxIV Register	662
25-21.	TAxEX0 Register.....	663
26-1.	Timer_B Block Diagram.....	666
26-2.	Up Mode	668
26-3.	Up Mode Flag Setting	668
26-4.	Continuous Mode	669
26-5.	Continuous Mode Flag Setting.....	669
26-6.	Continuous Mode Time Intervals	669
26-7.	Up/Down Mode.....	670
26-8.	Up/Down Mode Flag Setting	670
26-9.	Output Unit in Up/Down Mode	671
26-10.	Capture Signal (SCS = 1).....	672
26-11.	Capture Cycle	672
26-12.	Output Example – Timer in Up Mode	675
26-13.	Output Example – Timer in Continuous Mode	676
26-14.	Output Example – Timer in Up/Down Mode	677

26-15. Capture/Compare TBxCCR0 Interrupt Flag	678
26-16. TBxCTL Register.....	681
26-17. TBxR Register	683
26-18. TBxCCTLn Register	684
26-19. TBxCCRn Register	686
26-20. TBxIV Register	687
26-21. TBxEX0 Register.....	688
28-1. RTC_B Block Diagram	692
28-2. RTCCTL0 Register	700
28-3. RTCCTL1 Register	701
28-4. RTCCTL2 Register	702
28-5. RTCCTL3 Register	702
28-6. RTCSEC Register	703
28-7. RTCSEC Register	703
28-8. RTCMIN Register	704
28-9. RTCMIN Register	704
28-10. RTCHOUR Register	705
28-11. RTCHOUR Register	705
28-12. RTCDOW Register	706
28-13. RTCDAY Register	706
28-14. RTCDAY Register	706
28-15. RTCMON Register.....	707
28-16. RTCMON Register.....	707
28-17. RTCYEAR Register.....	708
28-18. RTCYEAR Register.....	708
28-19. RTCAMIN Register	709
28-20. RTCAMIN Register	709
28-21. RTCAHOUR Register	710
28-22. RTCAHOUR Register	710
28-23. RTCADOW Register	711
28-24. RTCADAY Register.....	712
28-25. RTCADAY Register.....	712
28-26. RTCPS0CTL Register	713
28-27. RTCPS1CTL Register.....	714
28-28. RTCPS0 Register	715
28-29. RTCPS1 Register	715
28-30. RTCIV Register	716
28-31. BIN2BCD Register.....	717
28-32. BCD2BIN Register.....	717
29-1. RTC_C Block Diagram (RTCMODE = 1)	720
29-2. RTC_C Offset Error Calibration and Temperature Compensation Scheme	727
29-3. RTC_C Functional Block Diagram in Counter Mode (RTCMODE = 0)	729
29-4. RTCCTL0_L Register	737
29-5. RTCCTL0_H Register	738
29-6. RTCCTL1 Register	739
29-7. RTCCTL3 Register	740
29-8. RTCOCAL Register.....	740
29-9. RTCTCMP Register	741
29-10. RTCNT1 Register.....	742

29-11. RTCNT2 Register	742
29-12. RTCNT3 Register	742
29-13. RTCNT4 Register	742
29-14. RTCSEC Register	743
29-15. RTCSEC Register	743
29-16. RTCMIN Register	744
29-17. RTCMIN Register	744
29-18. RTCHOUR Register	745
29-19. RTCHOUR Register	745
29-20. RTCDOW Register	746
29-21. RTCDAY Register	746
29-22. RTCDAY Register	746
29-23. RTCMON Register.....	747
29-24. RTCMON Register.....	747
29-25. RTCYEAR Register.....	748
29-26. RTCYEAR Register.....	748
29-27. RTCAMIN Register	749
29-28. RTCAMIN Register	749
29-29. RTCAHOUR Register	750
29-30. RTCAHOUR Register	750
29-31. RTCADOW Register	751
29-32. RTCADAY Register.....	752
29-33. RTCADAY Register.....	752
29-34. RTCPS0CTL Register.....	753
29-35. RTCPS1CTL Register.....	754
29-36. RTCPS0 Register	756
29-37. RTCPS1 Register	756
29-38. RTCIV Register	757
29-39. BIN2BCD Register.....	758
29-40. BCD2BIN Register.....	758
29-41. RTCSECBAX Register.....	759
29-42. RTCSECBAX Register.....	759
29-43. RTCMINBAKx Register	760
29-44. RTCMINBAKx Register	760
29-45. RTCHOURBAKx Register	761
29-46. RTCHOURBAKx Register	761
29-47. RTCDAYBAKx Register.....	762
29-48. RTCDAYBAKx Register.....	762
29-49. RTCMONBAKx Register	763
29-50. RTCMONBAKx Register.....	763
29-51. RTCYEARBAKx Register	764
29-52. RTCYEARBAKx Register	764
29-53. RTCTCCTL0 Register.....	765
29-54. RTCTCCTL1 Register	765
29-55. RTCCAPxCTL Register	766
30-1. eUSCI_Ax Block Diagram – UART Mode (UCSYNC = 0).....	769
30-2. Character Format	770
30-3. Idle-Line Format.....	771
30-4. Address-Bit Multiprocessor Format.....	772

30-5. Auto Baud-Rate Detection – Break/Synch Sequence	773
30-6. Auto Baud-Rate Detection – Synch Field.....	773
30-7. UART vs IrDA Data Format	774
30-8. Glitch Suppression, eUSCI_A Receive Not Started.....	776
30-9. Glitch Suppression, eUSCI_A Activated	776
30-10. BITCLK Baud-Rate Timing With UCOS16 = 0.....	777
30-11. Receive Error	781
30-12. UCxCTLW0 Register	787
30-13. UCxCTLW1 Register	788
30-14. UCxBRW Register	789
30-15. UCxMCTLW Register	789
30-16. UCxSTATW Register	790
30-17. UCxRXBUF Register	791
30-18. UCxTXBUF Register.....	791
30-19. UCxABCTL Register.....	792
30-20. UCxIRCTL Register.....	793
30-21. UCxIE Register.....	794
30-22. UCxIFG Register	795
30-23. UCxIV Register.....	796
31-1. eUSCI Block Diagram – SPI Mode	799
31-2. eUSCI Master and External Slave (UCSTEM = 0)	801
31-3. eUSCI Slave and External Master.....	802
31-4. eUSCI SPI Timing With UCMSB = 1	804
31-5. UCxCTLW0 Register	807
31-6. UCxBRW Register	808
31-7. UCxSTATW Register	809
31-8. UCxRXBUF Register	810
31-9. UCxTXBUF Register.....	811
31-10. UCxIE Register	812
31-11. UCxIFG Register	813
31-12. UCxIV Register	814
31-13. UCxBCTLW0 Register	816
31-14. UCxBRW Register	817
31-15. UCxBSTATW Register	817
31-16. UCxBRXBUF Register	818
31-17. UCxBTXBUF Register.....	818
31-18. UCxBIE Register	819
31-19. UCxBIFG Register.....	819
31-20. UCxBIV Register	820
32-1. eUSCI_B Block Diagram – I ² C Mode	823
32-2. I ² C Bus Connection Diagram.....	824
32-3. I ² C Module Data Transfer	825
32-4. Bit Transfer on I ² C Bus.....	825
32-5. I ² C Module 7-Bit Addressing Format	825
32-6. I ² C Module 10-Bit Addressing Format.....	826
32-7. I ² C Module Addressing Format With Repeated START Condition	826
32-8. I ² C Time-Line Legend	828
32-9. I ² C Slave Transmitter Mode	829
32-10. I ² C Slave Receiver Mode	830

32-11. I ² C Slave 10-Bit Addressing Mode	831
32-12. I ² C Master Transmitter Mode.....	833
32-13. I ² C Master Receiver Mode.....	835
32-14. I ² C Master 10-Bit Addressing Mode	836
32-15. Arbitration Procedure Between Two Master Transmitters.....	836
32-16. Synchronization of Two I ² C Clock Generators During Arbitration	837
32-17. UCBxCTLW0 Register	845
32-18. UCBxCTLW1 Register	847
32-19. UCBxBRW Register	849
32-20. UCBxSTATW Register	849
32-21. UCBxTBCNT Register	850
32-22. UCBxRXBUF Register	851
32-23. UCBxTXBUF Register.....	851
32-24. UCBxI2COA0 Register.....	852
32-25. UCBxI2COA1 Register.....	853
32-26. UCBxI2COA2 Register.....	853
32-27. UCBxI2COA3 Register.....	854
32-28. UCBxADDRX Register	854
32-29. UCBxADDMASK Register	855
32-30. UCBxI2CSA Register.....	855
32-31. UCBxIE Register.....	856
32-32. UCBxIFG Register	858
32-33. UCBxIV Register.....	860
33-1. REF_A Block Diagram	862
33-2. REFCTL0 Register	866
34-1. ADC12_B Block Diagram	870
34-2. Analog Multiplexer T-Switch.....	872
34-3. Extended Sample Mode Without Internal Reference in 12-Bit Mode	874
34-4. Extended Sample Mode With Internal Reference in 12-Bit Mode.....	874
34-5. Pulse Sample Mode First Conversion or Where ADC12MSC = 0 in 12-Bit Mode	875
34-6. Pulse Sample Mode Subsequent Conversions in 12-Bit Mode.....	875
34-7. Analog Input Equivalent Circuit	875
34-8. Single-Channel Single-Conversion Mode, ADC12ISSH = 0	878
34-9. Sequence-of-Channels Mode, ADC12ISSH = 0	879
34-10. Repeat-Single-Channel Mode, ADC12ISSH = 0	880
34-11. Repeat-Sequence-of-Channels Mode, ADC12ISSH = 0	881
34-12. Typical Temperature Sensor Transfer Function	883
34-13. ADC12_B Grounding and Noise Considerations	884
34-14. ADC12CTL0 Register	893
34-15. ADC12CTL1 Register	895
34-16. ADC12CTL2 Register	897
34-17. ADC12CTL3 Register	898
34-18. ADC12MEMx Register	899
34-19. ADC12MCTLx Register	900
34-20. ADC12HI Register	902
34-21. ADC12LO Register	902
34-22. ADC12IER0 Register.....	903
34-23. ADC12IER1 Register.....	905
34-24. ADC12IER2 Register.....	907

34-25. ADC12IFGR0 Register	908
34-26. ADC12IFGR1 Register	910
34-27. ADC12IFGR2 Register	912
34-28. ADC12IV Register	913
35-1. Comparator_E Block Diagram	916
35-2. Comparator_E Sample-And-Hold	918
35-3. RC-Filter Response at the Output of the Comparator	919
35-4. Reference Generator Block Diagram	919
35-5. Transfer Characteristic and Power Dissipation in a CMOS Inverter and Buffer	920
35-6. Temperature Measurement System	921
35-7. Timing for Temperature Measurement Systems	921
35-8. CECTL0 Register	924
35-9. CECTL1 Register	925
35-10. CECTL2 Register	926
35-11. CECTL3 Register	927
35-12. CEINT Register	929
35-13. CEIV Register	930
36-1. LCD Controller Block Diagram	933
36-2. LCD Memory for Static and 2-Mux to 4-Mux Mode - Example for 160 Segments	934
36-3. LCD Memory for 5-Mux to 8-Mux Mode - Example for 160 Segments	935
36-4. Bias Generation	938
36-5. Example Static Waveforms	943
36-6. Example 2-Mux Waveforms	944
36-7. Example 3-Mux Waveforms	945
36-8. Example 4-Mux Waveforms	946
36-9. Example 6-Mux Waveforms	947
36-10. Example 8-Mux, 1/3 Bias Waveforms (LCDLP = 0)	948
36-11. Example 8-Mux, 1/3 Bias Low-Power Waveforms (LCDLP = 1)	949
36-12. LCDCCTL0 Register	955
36-13. LCDCCTL1 Register	957
36-14. LCDCBLKCTL Register	958
36-15. LCDCMEMCTL Register	959
36-16. LCDCVCTL Register	960
36-17. LCDCPCTL0 Register	962
36-18. LCDCPCTL1 Register	962
36-19. LCDCPCTL2 Register	963
36-20. LCDCPCTL3 Register	963
36-21. LCDCCPCTL Register	964
36-22. LCDCIV Register	964
37-1. ESI Block Diagram	966
37-2. ESI Analog Front End AFE1 Block Diagram	968
37-3. ESI Analog Front End AFE2 Block Diagram	969
37-4. Excitation and Sample-And-Hold Circuitry	970
37-5. Analog Input Equivalent Circuit	971
37-6. Analog Front-End Output Timing	972
37-7. Analog Hysteresis With DAC Registers	973
37-8. Timing State Machine Block Diagram	975
37-9. Test Cycle Insertion	978
37-10. Timing State Machine Example	979

37-11. Pre-Processing Unit	980
37-12. Timer_A Output Stage of the Analog Front End.....	980
37-13. ESI Processing State Machine Block Diagram	981
37-14. Simplest PSM State Diagram (ESIV2SEL=1).....	984
37-15. LC Sensor Oscillations	986
37-16. Sensor Connections For The Oscillation Test	987
37-17. LC Sensor Connections For The Envelope Test	988
37-18. LC Sensor Connections For the Envelope Test	989
37-19. Resistive Sensor Connections	990
37-20. Sensor Position and Quadrature Signals (S1=PPUS1, S2=PPUS2)	991
37-21. Quadrature Decoding State Diagram	991
37-22. ESIDEBUG1 Register	994
37-23. ESIDEBUG2 Register	994
37-24. ESIDEBUG3 Register	994
37-25. ESIDEBUG4 Register	995
37-26. ESIDEBUG5 Register	995
37-27. ESICNT0 Register	996
37-28. ESICNT1 Register	996
37-29. ESICNT2 Register	997
37-30. ESICNT3 Register	997
37-31. ESIIV Register.....	998
37-32. ESIINT1 Register	999
37-33. ESIINT2 Register.....	1001
37-34. ESIAFE Register	1003
37-35. ESIPPU Register	1005
37-36. ESITSM Register	1006
37-37. ESIPSM Register	1008
37-38. ESIOSC Register.....	1009
37-39. ESICTL Register	1010
37-40. ESITHR1 Register.....	1012
37-41. ESITHR2 Register.....	1012
37-42. ESIDAC1Rx Register	1013
37-43. ESIDAC2Rx Register	1013
37-44. ESITSMx Register.....	1014
37-45. Extended Scan Interface Processing State Machine Table Entry Register	1016
38-1. Large Implementation of EEM	1019

List of Tables

1-1.	Interrupt Sources, Flags, and Vectors	53
1-2.	Operation Modes	58
1-3.	Requested vs Actual LPM.....	58
1-4.	Connection of Unused Pins	62
1-5.	Tag Values	67
1-6.	REF Calibration Tags	68
1-7.	ADC Calibration Tags.....	69
1-8.	Random Number Tags	70
1-9.	BSL Configuration Tags	70
1-10.	BSL_COM_IF Values	70
1-11.	BSL_CIF_CONFIG Values	71
1-12.	SFR Registers	72
1-13.	SFRIE1 Register Description	73
1-14.	SFRIFG1 Register Description	74
1-15.	SFRRPCR Register Description.....	75
1-16.	SYS Registers	76
1-17.	SYSCTL Register Description	77
1-18.	SYSJMBC Register Description	78
1-19.	SYSJMBI0 Register Description.....	79
1-20.	SYSJMBI1 Register Description.....	79
1-21.	SYSJMBO0 Register Description	80
1-22.	SYSJMBO1 Register Description	80
1-23.	SYSUNIV Register Description.....	81
1-24.	SYSSNIV Register Description	81
1-25.	SYSRSTIV Register Description	82
2-1.	PMM Registers	88
2-2.	PMMCTL0 Register Description.....	89
2-3.	PMMCTL1 Register Description.....	90
2-4.	PMMIFG Register Description.....	91
2-5.	PM5CTL0 Register Description	92
3-1.	HFFREQ Settings	97
3-2.	System Clocks, Power Modes, and Clock Requests	100
3-3.	MEMORYMAP Registers	103
3-4.	CTL0 Register Field Descriptions.....	104
3-5.	CTL1 Register Field Descriptions.....	105
3-6.	CTL2 Register Field Descriptions.....	106
3-7.	CTL3 Register Field Descriptions.....	107
3-8.	CTL4 Register Field Descriptions.....	108
3-9.	CTL5 Register Field Descriptions.....	110
3-10.	CTL6 Register Field Descriptions.....	111
4-1.	SR Bit Description	118
4-2.	Values of Constant Generators CG1, CG2.....	119
4-3.	Source and Destination Addressing.....	122
4-4.	MSP430 Double-Operand Instructions.....	142
4-5.	MSP430 Single-Operand Instructions.....	142
4-6.	Conditional Jump Instructions	143
4-7.	Emulated Instructions	143

4-8.	Interrupt, Return, and Reset Cycles and Length	144
4-9.	MSP430 Format II Instruction Cycles and Length.....	144
4-10.	MSP430 Format I Instructions Cycles and Length	145
4-11.	Description of the Extension Word Bits for Register Mode.....	146
4-12.	Description of Extension Word Bits for Non-Register Modes	147
4-13.	Extended Double-Operand Instructions.....	148
4-14.	Extended Single-Operand Instructions.....	150
4-15.	Extended Emulated Instructions	152
4-16.	Address Instructions, Operate on 20-Bit Register Data	153
4-17.	MSP430X Format II Instruction Cycles and Length	154
4-18.	MSP430X Format I Instruction Cycles and Length.....	155
4-19.	Address Instruction Cycles and Length	156
4-20.	Instruction Map of MSP430X	157
5-1.	Result Availability (MPYFRAC = 0, MPYSAT = 0)	273
5-2.	OP1 Registers.....	274
5-3.	OP2 Registers.....	274
5-4.	SUMEXT and MPYC Contents.....	275
5-5.	Result Availability in Fractional Mode (MPYFRAC = 1, MPYSAT = 0)	278
5-6.	Result Availability in Saturation Mode (MPYSAT = 1)	279
5-7.	MPY32 Registers	285
5-8.	Alternative Registers	286
5-9.	MPY32CTL0 Register Description.....	287
6-1.	FRAM Controller Overview	288
7-1.	FRCTL Registers	294
7-2.	FRCTL0 Register Description	295
7-3.	GCCTL0 Register Description	296
7-4.	GCCTL1 Register Description	297
8-1.	FRAM memory Access Speed	301
8-2.	FRAM Power Mode Transition	302
8-3.	FRCTL_A Registers	304
8-4.	FRCTL0 Register Field Descriptions	305
8-5.	GCCTL0 Register Field Descriptions	307
8-6.	GCCTL1 Register Field Descriptions	309
9-1.	Address Comparator Bit Selection	313
9-2.	IP Encapsulation Access Rights	315
9-3.	MPU Border Selection Example 64KB (004000h to 013FFFh).....	316
9-4.	Segment Access Rights.....	317
9-5.	Access Rights to IVT	318
9-6.	IPE Signatures	319
9-7.	IPE_Init_Structure	320
9-8.	MPU Registers	321
9-9.	MPUCTL0 Register Description.....	322
9-10.	MPUCTL1 Register Description.....	323
9-11.	MPUSEGB2 Register Description	324
9-12.	MPUSEGB1 Register Description	325
9-13.	MPUSAM Register Description	326
9-14.	MPUIPC0 Register Description	328
9-15.	MPUIPSEGB2 Register Description.....	329
9-16.	MPUIPSEGB1 Register Description.....	330

10-1. RAMCTL Registers	334
10-2. CTL0 Register Field Descriptions.....	335
10-3. CTL1 Register Field Descriptions.....	337
11-1. DMA Transfer Modes.....	342
11-2. DMA Trigger Operation	349
11-3. Maximum Single-Transfer DMA Cycle Time	350
11-4. DMA Registers	353
11-5. DMACTL0 Register Description.....	355
11-6. DMACTL1 Register Description.....	356
11-7. DMACTL2 Register Description.....	357
11-8. DMACTL3 Register Description.....	358
11-9. DMACTL4 Register Description.....	359
11-10. DMAxCTL Register Description.....	360
11-11. DMAxSA Register Description	362
11-12. DMAxDA Register Description	363
11-13. DMAxSZ Register Description	364
11-14. DMAIV Register Description.....	365
12-1. I/O Configuration.....	368
12-2. I/O Function Selection.....	369
12-3. Digital I/O Registers	374
12-4. PxIV Register Description	388
12-5. PxIN Register Description	389
12-6. PxOUT Register Description	389
12-7. P1DIR Register Description	389
12-8. PxREN Register Description	390
12-9. PxSEL0 Register Description	390
12-10. PxSEL1 Register Description	390
12-11. PxSELC Register Description.....	391
12-12. PxIES Register Description.....	391
12-13. PxIE Register Description	391
12-14. PxIFG Register Description	392
13-1. CapTouch Registers.....	396
13-2. CAPTIOxCTL Register Description.....	397
14-1. AES Operation Modes Overview	400
14-2. 'AES trigger 0-2' Operation When AESCMEN = 1	407
14-3. AES and DMA Configuration for ECB Encryption	408
14-4. AES DMA Configuration for ECB Decryption	409
14-5. AES and DMA Configuration for CBC Encryption.....	410
14-6. AES and DMA Configuration for CBC Decryption	411
14-7. AES and DMA Configuration for OFB Encryption	413
14-8. AES and DMA Configuration for OFB Decryption	414
14-9. AES and DMA Configuration for CFB Encryption	415
14-10. AES and DMA Configuration for CFB Decryption	416
14-11. AES256 Registers	417
14-12. AESACTL0 Register Description	418
14-13. AESACTL1 Register Description	420
14-14. AESASTAT Register Description	421
14-15. AESAKEY Register Description.....	422
14-16. AESADIN Register Description	423

14-17. AESADOUT Register Description	424
14-18. AESAXDIN Register Description	425
14-19. AESAXIN Register Description	426
15-1. CRC Registers	432
15-2. CRCDI Register Description.....	433
15-3. CRCDIRB Register Description.....	433
15-4. CRCNIREs Register Description	434
15-5. CRCRESR Register Description.....	434
16-1. CRC32 Registers	439
16-2. CRC32DIW0 Register Description.....	440
16-3. CRC32DIW1 Register Description.....	440
16-4. CRC32DIRBW0 Register Description.....	441
16-5. CRC32DIRBW1 Register Description.....	441
16-6. CRC32NIRESW0 Register Description	442
16-7. CRC32NIRESW1 Register Description	442
16-8. CRC32RESRW0 Register Description.....	443
16-9. CRC32RESRW1 Register Description.....	443
16-10. CRC16DIL0 Register Description.....	444
16-11. CRC16DIRBW0 Register Description	444
16-12. CRC16NIRESW0 Register Description	445
16-13. CRC16RESRW0 Register Description.....	445
17-1. LEA Command Groups	448
17-2. DSP Library and MSPWare Versions for the LEA.....	449
18-1. Auto Mode and Register Mode	454
18-2. Time Mark Events	454
18-3. USS_PWRREQ Signal Source	455
18-4. Control Signals Among USS Submodules	456
19-1. USS Power State	461
19-2. USS Power States and State Changes.....	462
19-3. Device Power Modes and USS Power States	463
19-4. Internal Control Signals	463
19-5. ASQ Trigger	464
19-6. Power States After Measurement Completion	465
19-7. UUPS Registers.....	467
19-8. UUPSIDX Register Field Descriptions	468
19-9. UUPSMIS Register Field Descriptions.....	469
19-10. UUPSRIS Register Field Descriptions	470
19-11. UUPSIMSC Register Field Descriptions.....	471
19-12. UUPSICR Register Field Descriptions	472
19-13. UUPSISR Register Field Descriptions	473
19-14. UUPSDESCLO Register Field Descriptions.....	474
19-15. UUPSDESCHI Register Field Descriptions	475
19-16. UUPSCTL Register Field Descriptions.....	476
20-1. HSPLL Registers.....	483
20-2. HSPLLIIDX Register Field Descriptions	484
20-3. HSPLLMIS Register Field Descriptions	485
20-4. HSPLLRIS Register Field Descriptions	486
20-5. HSPLLMISC Register Field Descriptions.....	487
20-6. HSPLLICR Register Field Descriptions	488

20-7. HSPLLISR Register Field Descriptions	489
20-8. HSPLLESCLO Register Field Descriptions	490
20-9. HSPLLESCHI Register Field Descriptions	491
20-10. HSPLLCTL Register Field Descriptions	492
20-11. HSPLLUSSXTLCTL Register Field Descriptions	494
21-1. Trim Registers.....	505
21-2. Supply to the Rx Multiplexer	507
21-3. Time Mark Events	512
21-4. Auto Mode and Register Mode	513
21-5. SAPH Registers.....	516
21-6. SAPHIIDX/SAPH_AIIDX Register Field Descriptions	518
21-7. SAPHMIS/SAPH_AMIS Register Field Descriptions.....	519
21-8. SAPHRIS/SAPH_ARIS Register Field Descriptions	520
21-9. SAPHIMSC/SAPH_AIMSC Register Field Descriptions.....	521
21-10. SAPHICR/SAPH_AICR Register Field Descriptions	522
21-11. SAPHISR/SAPH_AISR Register Field Descriptions	523
21-12. SAPHDESCLO/SAPH_ADESCLO Register Field Descriptions	524
21-13. SAPHDESCHI/SAPH_ADESCHI Register Field Descriptions	525
21-14. SAPHKEY/SAPH_AKEY Register Field Descriptions.....	526
21-15. SAPHOCTL0/SAPH_AOCTL0 Register Field Descriptions	527
21-16. SAPHOCTL1/SAPH_AOCTL1 Register Field Descriptions	528
21-17. SAPHOSEL/SAPH_AOSEL Register Field Descriptions	529
21-18. SAPHCH0PUT/SAPH_ACH0PUT Register Field Descriptions.....	530
21-19. SAPHCH0PDT/SAPH_ACH0PDT Register Field Descriptions.....	531
21-20. SAPHCH0TT/SAPH_ACH0TT Register Field Descriptions	532
21-21. SAPHCH1PUT/SAPH_ACH1PUT Register Field Descriptions.....	533
21-22. SAPHCH1PDT/SAPH_ACH1PDT Register Field Descriptions.....	534
21-23. SAPHCH1TT/SAPH_ACH1TT Register Field Descriptions	535
21-24. SAPHMCNF/SAPH_AMCNF Register Field Descriptions.....	536
21-25. SAPHTACTL/SAPH_ATACTL Register Field Descriptions	537
21-26. SAPHICTL0/SAPH_AICTL0 Register Field Descriptions.....	538
21-27. SAPHBCTL/SAPH_ABCTL Register Field Descriptions.....	540
21-28. SAPHPGC/SAPH_APGC Register Field Descriptions.....	542
21-29. SAPHGPLPER/SAPH_APGLPER Register Field Descriptions	543
21-30. SAPHPGHPER/SAPH_APGHPER Register Field Descriptions.....	544
21-31. SAPHPGCTL/SAPH_APGCTL Register Field Descriptions	545
21-32. SAPHPPGTRIG/SAPH_APPGTRIG Register Field Descriptions	547
21-33. SAPH_AXPGCTL Register Field Descriptions.....	548
21-34. SAPH_AXGPLPER Register Field Descriptions.....	549
21-35. SAPH_AXPGHPER Register Field Descriptions	550
21-36. SAPHASCTL0/SAPH_AASCTL0 Register Field Descriptions	551
21-37. SAPHASCTL1/SAPH_AASCTL1 Register Field Descriptions	553
21-38. SAPHASQTRIG/SAPH_AASQTRIG Register Field Descriptions	555
21-39. SAPHAPOL/SAPH_AAPOL Register Field Descriptions	556
21-40. SAPHAPLEV/SAPH_AAPLEV Register Field Descriptions	557
21-41. SAPHAPHIZ/SAPH_AAPHIZ Register Field Descriptions	558
21-42. SAPHATM_A/SAPH_AATM_A Register Field Descriptions	559
21-43. SAPHATM_B/SAPH_AATM_B Register Field Descriptions	560
21-44. SAPHATM_C/SAPH_AATM_C Register Field Descriptions	561

21-45. SAPHATM_D/SAPH_AATM_D Register Field Descriptions	562
21-46. SAPHATM_E/SAPH_AATM_E Register Field Descriptions	563
21-47. SAPHATM_F/SAPH_AATM_F Register Field Descriptions.....	564
21-48. SAPHTBCTL/SAPH_ATBCTL Register Field Descriptions	565
21-49. SAPHATIMLO/SAPH_AATIMLO Register Field Descriptions	566
21-50. SAPHATIMHI /SAPH_AATIMHI Register Field Descriptions	567
22-1. Data Format	576
22-2. PGA Gain Table.....	580
22-3. Control Signals for Power and Conversion.....	582
22-4. USS Auto Mode and Register Mode	582
22-5. SDHSCTL3.TRIGEN Bit and SDHSCTL5.SDHS_LOCK Bit	585
22-6. Timing of the SDHS_LOCK bit.....	585
22-7. Conversion Control Mode.....	588
22-8. Conversion Control Mode.....	590
22-9. SDHS Conversion Stop Conditions	592
22-10. SDHS Response to Conversion Stop Signals When Data Conversion is Not Running.....	593
22-11. SDHS Registers.....	595
22-12. SDHSIIDX Register Field Descriptions	596
22-13. SDHSMIS Register Field Descriptions.....	597
22-14. SDHSRIS Register Field Descriptions	598
22-15. SDHSIMSC Register Field Descriptions.....	600
22-16. SDHSICR Register Field Descriptions	601
22-17. SDHSISR Register Field Descriptions	602
22-18. SDHSDESCLO Register Field Descriptions.....	603
22-19. SDHSDESCHI Register Field Descriptions	604
22-20. SDHSCTL0 Register Field Descriptions	605
22-21. SDHSCTL1 Register Field Descriptions	607
22-22. SDHSCTL2 Register Field Descriptions	608
22-23. SDHSCTL3 Register Field Descriptions	609
22-24. SDHSCTL4 Register Field Descriptions	610
22-25. SDHSCTL5 Register Field Descriptions	611
22-26. SDHSCTL6 Register Field Descriptions	613
22-27. SDHSCTL7 Register Field Descriptions	614
22-28. SDHSDT Register Field Descriptions	615
22-29. SDHSWINHITH Register Field Descriptions	616
22-30. SDHSWINLOTH Register Field Descriptions	617
22-31. SDHSDTCDA Register Field Descriptions	618
23-1. PGFS Values	622
23-2. MTIF Initialization	622
23-3. Setting the Pulse Rate	622
23-4. Reading the Pulse Rate.....	623
23-5. MTIF Registers.....	625
23-6. MTIFPGCNF Register Field Descriptions	626
23-7. MTIFPGKVAL Register Field Descriptions	627
23-8. MTIFPGCTL Register Field Descriptions.....	628
23-9. MTIFPGSR Register Field Descriptions	629
23-10. MTIFPCCNF Register Field Descriptions.....	630
23-11. MTIFPCR Register Field Descriptions	631
23-12. MTIFPCCTL Register Field Descriptions	632

23-13. MTIFPCSR Register Field Descriptions	633
23-14. MTIFTPCTL Register Field Descriptions	634
24-1. WDT_A Registers.....	640
24-2. WDTCTL Register Description	641
25-1. Timer Modes	646
25-2. Output Modes	651
25-3. Timer_A Registers	657
25-4. TAxCTL Register Description	658
25-5. TAxR Register Description	659
25-6. TAxCCCTLn Register Description.....	660
25-7. TAxCCRn Register Description	662
25-8. TAxIV Register Description.....	662
25-9. TAxEX0 Register Description	663
26-1. Timer Modes	668
26-2. TBxCLn Load Events.....	673
26-3. Compare Latch Operating Modes	674
26-4. Output Modes	674
26-5. Timer_B Registers	680
26-6. TBxCTL Register Description	681
26-7. TBxR Register Description	683
26-8. TBxCCCTLn Register Description.....	684
26-9. TBxCCRn Register Description	686
26-10. TBxIV Register Description.....	687
26-11. TBxEX0 Register Description	688
27-1. RTC Overview.....	689
28-1. RTC_B Registers	698
28-2. RTCCTL0 Register Description	700
28-3. RTCCTL1 Register Description	701
28-4. RTCCTL2 Register Description	702
28-5. RTCCTL3 Register Description	702
28-6. RTCSEC Register Description	703
28-7. RTCSEC Register Description	703
28-8. RTCMIN Register Description.....	704
28-9. RTCMIN Register Description.....	704
28-10. RTCHOUR Register Description.....	705
28-11. RTCHOUR Register Description.....	705
28-12. RTCDOW Register Description	706
28-13. RTCDAY Register Description	706
28-14. RTCDAY Register Description	706
28-15. RTCMON Register Description	707
28-16. RTCMON Register Description	707
28-17. RTCYEAR Register Description	708
28-18. RTCYEAR Register Description	708
28-19. RTCAMIN Register Description	709
28-20. RTCAMIN Register Description	709
28-21. RTCAHOUR Register Description.....	710
28-22. RTCAHOUR Register Description.....	710
28-23. RTCADOW Register Description	711
28-24. RTCADAY Register Description	712

28-25. RTCADAY Register Description	712
28-26. RTCPS0CTL Register Description	713
28-27. RTCPS1CTL Register Description	714
28-28. RTCPS0 Register Description	715
28-29. RTCPS1 Register Description	715
28-30. RTCIV Register Description	716
28-31. BIN2BCD Register Description	717
28-32. BCD2BIN Register Description	717
29-1. RTCCAPx Pin Configuration	733
29-2. RTC_C Registers	734
29-3. RTC_C Event and Tamper Detection Registers.....	736
29-4. RTC_C Real-Time Clock Counter Mode Aliases	736
29-5. RTCCTL0_L Register Description	737
29-6. RTCCTL0_H Register Description.....	738
29-7. RTCCTL1 Register Description	739
29-8. RTCCTL3 Register Description	740
29-9. RTCOCAL Register Description	740
29-10. RTCTCMP Register Description	741
29-11. RTCNT1 Register Description	742
29-12. RTCNT2 Register Description	742
29-13. RTCNT3 Register Description	742
29-14. RTCNT4 Register Description	742
29-15. RTCSEC Register Description	743
29-16. RTCSEC Register Description	743
29-17. RTCMIN Register Description.....	744
29-18. RTCMIN Register Description.....	744
29-19. RTCHOUR Register Description.....	745
29-20. RTCHOUR Register Description.....	745
29-21. RTCDOW Register Description	746
29-22. RTCDAY Register Description	746
29-23. RTCDAY Register Description	746
29-24. RTCMON Register Description	747
29-25. RTCMON Register Description	747
29-26. RTCYEAR Register Description	748
29-27. RTCYEAR Register Description	748
29-28. RTCAMIN Register Description.....	749
29-29. RTCAMIN Register Description	749
29-30. RTCAHOUR Register Description.....	750
29-31. RTCAHOUR Register Description.....	750
29-32. RTCADOW Register Description	751
29-33. RTCADAY Register Description	752
29-34. RTCADAY Register Description	752
29-35. RTCPS0CTL Register Description	753
29-36. RTCPS1CTL Register Description	754
29-37. RTCPS0 Register Description	756
29-38. RTCPS1 Register Description	756
29-39. RTCIV Register Description	757
29-40. BIN2BCD Register Description	758
29-41. BCD2BIN Register Description	758

29-42. RTCSECBAXx Register Description	759
29-43. RTCSECBAXx Register Description	759
29-44. RTCMINBAKx Register Description	760
29-45. RTCMINBAKx Register Description	760
29-46. RTCHOURBAKx Register Description	761
29-47. RTCHOURBAKx Register Description	761
29-48. RTCDAYBAKx Register Description	762
29-49. RTCDAYBAKx Register Description	762
29-50. RTCMONBAKx Register Description.....	763
29-51. RTCMONBAKx Register Description.....	763
29-52. RTCYEARBAKx Register Description	764
29-53. RTCYEARBAKx Register Description	764
29-54. RTCTCCTL0 Register Description	765
29-55. RTCTCCTL1 Register Description	765
29-56. RTCCAPxCTL Register Description.....	766
30-1. Receive Error Conditions	775
30-2. Modulation Pattern Examples	777
30-3. BITCLK16 Modulation Pattern	778
30-4. UCBRSx Settings for Fractional Portion of $N = f_{BRCLK}/\text{Baud Rate}$	779
30-5. Recommended Settings for Typical Crystals and Baud Rates	782
30-6. UART State Change Interrupt Flags	784
30-7. eUSCI_A UART Registers.....	786
30-8. UCAXCTLW0 Register Description	787
30-9. UCAXCTLW1 Register Description	788
30-10. UCAXBRW Register Description.....	789
30-11. UCAXMCTLW Register Description	789
30-12. UCAXSTATW Register Description.....	790
30-13. UCAXRXBUF Register Description	791
30-14. UCAXTXBUF Register Description	791
30-15. UCAXABCTL Register Description	792
30-16. UCAXIRCTL Register Description	793
30-17. UCAXIE Register Description.....	794
30-18. UCAXIFG Register Description.....	795
30-19. UCAXIV Register Description.....	796
31-1. UCxSTE Operation	800
31-2. eUSCI_A SPI Registers.....	806
31-3. UCAXCTLW0 Register Description	807
31-4. UCAXBRW Register Description	808
31-5. UCAXSTATW Register Description.....	809
31-6. UCAXRXBUF Register Description	810
31-7. UCAXTXBUF Register Description	811
31-8. UCAXIE Register Description.....	812
31-9. UCAXIFG Register Description.....	813
31-10. UCAXIV Register Description.....	814
31-11. eUSCI_B SPI Registers.....	815
31-12. UCBxCTLW0 Register Description	816
31-13. UCBxBRW Register Description.....	817
31-14. UCBxSTATW Register Description.....	817
31-15. UCBxRXBUF Register Description	818

31-16. UCBxTXBUF Register Description	818
31-17. UCBxIE Register Description.....	819
31-18. UCBxIFG Register Description.....	819
31-19. UCBxIV Register Description.....	820
32-1. Glitch Filter Length Selection Bits	837
32-2. I ² C State Change Interrupt Flags	842
32-3. eUSCI_B Registers	844
32-4. UCBxCTLW0 Register Description	845
32-5. UCBxCTLW1 Register Description	847
32-6. UCBxBRW Register Description.....	849
32-7. UCBxSTATW Register Description.....	849
32-8. UCBxTBCNT Register Description	850
32-9. UCBxRXBUF Register Description	851
32-10. UCBxTXBUF Register Description	851
32-11. UCBxI2COA0 Register Description	852
32-12. UCBxI2COA1 Register Description	853
32-13. UCBxI2COA2 Register Description	853
32-14. UCBxI2COA3 Register Description	854
32-15. UCBxADDRX Register Description.....	854
32-16. UCBxADDMASK Register Description.....	855
32-17. UCBxI2CSA Register Description	855
32-18. UCBxIE Register Description.....	856
32-19. UCBxIFG Register Description.....	858
32-20. UCBxIV Register Description.....	860
33-1. REF_A Registers	865
33-2. REFCTL0 Register Description	866
34-1. ADC12_B Conversion Result Formats.....	876
34-2. Conversion Mode Summary.....	877
34-3. ADC12_B Registers	887
34-4. ADC12CTL0 Register Description.....	893
34-5. ADC12CTL1 Register Description.....	895
34-6. ADC12CTL2 Register Description.....	897
34-7. ADC12CTL3 Register Description.....	898
34-8. ADC12MEMx Register Description.....	899
34-9. ADC12MCTLx Register Description.....	900
34-10. ADC12HI Register Description.....	902
34-11. ADC12LO Register Description	902
34-12. ADC12IER0 Register Description	903
34-13. ADC12IER1 Register Description	905
34-14. ADC12IER2 Register Description	907
34-15. ADC12IFGR0 Register Description	908
34-16. ADC12IFGR1 Register Description	910
34-17. ADC12IFGR2 Register Description	912
34-18. ADC12IV Register Description.....	913
35-1. COMP_E Registers.....	923
35-2. CECTL0 Register Description.....	924
35-3. CECTL1 Register Description.....	925
35-4. CECTL2 Register Description.....	926
35-5. CECTL3 Register Description.....	927

35-6. CEINT Register Description	929
35-7. CEIV Register Description	930
36-1. Differences Between LCD_B and LCD_C	932
36-2. Bias Voltages and external Pins	939
36-3. LCD Voltage and Biasing Characteristics	940
36-4. LCD_C Control Registers	950
36-5. LCD_C Memory Registers for Static and 2-Mux to 4-Mux Modes	951
36-6. LCD Blinking Memory Registers for Static and 2-Mux to 4-Mux Modes	952
36-7. LCD Memory Registers for 5-Mux to 8-Mux	953
36-8. LCDCTL0 Register Description	955
36-9. LCDCTL1 Register Description	957
36-10. LCDBLKCTL Register Description	958
36-11. LCDMEMCTL Register Description	959
36-12. LCDVCTL Register Description	960
36-13. LCDPCTL0 Register Description	962
36-14. LCDPCTL1 Register Description	962
36-15. LCDPCTL2 Register Description	963
36-16. LCDPCTL3 Register Description	963
36-17. LCDCCPCTL Register Description	964
36-18. LCDCIV Register Description	964
37-1. ESICAX and ESISH Input Selection	971
37-2. Selected Output Bits	972
37-3. Selected DAC Registers	973
37-4. DAC Register Select When TESTDX=1	974
37-5. TSM State Duration	977
37-6. TSM Example Register Values	978
37-7. ESI Interrupts	985
37-8. Quadrature Decoding PSM Table	992
37-9. ESI Registers	993
37-10. ESIDEBUG1 Register Description	994
37-11. ESIDEBUG2 Register Description	994
37-12. ESIDEBUG3 Register Description	994
37-13. ESIDEBUG4 Register Description	995
37-14. ESIDEBUG5 Register Description	995
37-15. ESICNT0 Register Description	996
37-16. ESICNT1 Register Description	996
37-17. ESICNT2 Register Description	997
37-18. ESICNT3 Register Description	997
37-19. ESIIV Register Description	998
37-20. ESIINT1 Register Description	999
37-21. ESIINT2 Register Description	1001
37-22. ESIAFE Register Description	1003
37-23. ESIPPU Register Description	1005
37-24. ESITSM Register Description	1006
37-25. TSM Start Trigger ACLK Divider	1007
37-26. ESIPSM Register Description	1008
37-27. ESIOSC Register Description	1009
37-28. ESICTL Register Description	1010
37-29. ESITHR1 Register Description	1012

37-30. ESITHR2 Register Description	1012
37-31. ESIDAC1Rx Register Description.....	1013
37-32. ESIDAC2Rx Register Description.....	1013
37-33. ESITSMx Register Description	1014
37-34. Extended Scan Interface Processing State Machine Table Entry Description	1016
38-1. EEM Configurations.....	1022

Read This First

About This Manual

This manual describes the modules and peripherals of the MSP430FR58xx, MSP430FR59xx, and MSP430FR6xx family of devices. Each description presents the module or peripheral in a general sense. Not all features and functions of all modules or peripherals may be present on all devices. In addition, modules or peripherals may differ in their exact implementation between device families, or may not be fully implemented on an individual device or device family.

Pin functions, internal signal connections, and operational parameters differ from device to device. Consult the device-specific data sheet for these details.

Related Documentation From Texas Instruments

For related documentation, visit the MSP430™ web site at <http://www.ti.com/msp430>

Notational Conventions

Program examples are shown in a special typeface; for example:

```
MOV    #255, R10
XOR    @R5, R6
```

Glossary

Abbreviation	Description
ACLK	Auxiliary clock
ADC	Analog-to-digital converter
BOR	Brownout reset
BSL	Bootloader; see www.ti.com/msp430 for application reports
CPU	Central processing unit
DAC	Digital-to-analog converter
DCO	Digitally controlled oscillator
dst	Destination
FLL	Frequency locked loop
GIE Modes	General interrupt enable
INT(N/2)	Integer portion of N/2
I/O	Input/output
ISR	Interrupt service routine
LSB	Least-significant bit
LSD	Least-significant digit
LPM	Low-power mode; also named PM for power mode
MAB	Memory address bus
MCLK	Master clock
MDB	Memory data bus
MSB	Most-significant bit
MSD	Most-significant digit
NMI	(Non)-Maskable interrupt; also split to UNMI (user NMI) and SNMI (system NMI)
PC	Program counter

Abbreviation	Description
PM	Power mode
POR	Power-on reset
PUC	Power-up clear
RAM	Random access memory
SCG	System clock generator
SFR	Special function register
SMCLK	Subsystem master clock
SNMI	System NMI
SP	Stack pointer
SR	Status register
src	Source
TOS	Top of stack
UNMI	User NMI
WDT	Watchdog timer
z16	16-bit address space

Register Bit Conventions

Each register is shown with a key indicating the accessibility of the each individual bit and the initial condition:

Register Bit Accessibility and Initial Condition

Key	Bit Accessibility
rw	Read/write
r	Read only
r0	Read as 0
r1	Read as 1
w	Write only
w0	Write as 0
w1	Write as 1
(w)	No register bit implemented; writing a 1 results in a pulse. The register bit always reads as 0.
h0	Cleared by hardware
h1	Set by hardware
-0,-1	Condition after PUC
-(0),-(1)	Condition after POR
-[0],[-1]	Condition after BOR
-{0},{-1}	Condition after brownout

Trademarks

MSP430, EnergyTrace are trademarks of Texas Instruments.
IAR Embedded Workbench is a trademark of IAR.

System Resets, Interrupts, and Operating Modes, System Control Module (SYS)

The system control module (SYS) is available on all devices. The basic features of SYS are:

- Brownout reset (BOR) and power on reset (POR) handling
- Power up clear (PUC) handling
- (Non)maskable interrupt (SNMI or UNMI) event source selection and management
- User data-exchange mechanism through the JTAG mailbox (JMB)
- Bootloader (BSL) entry mechanism
- Configuration management (device descriptors)
- Interrupt vector generators for reset and NMIs

Topic	Page
1.1 System Control Module (SYS) Introduction	48
1.2 System Reset and Initialization	48
1.3 Interrupts	50
1.4 Operating Modes.....	56
1.5 Principles for Low-Power Applications	61
1.6 Connection of Unused Pins	62
1.7 Reset Pin ($\overline{RST}$/NMI) Configuration.....	62
1.8 Configuring JTAG Pins	62
1.9 Vacant Memory Space	63
1.10 Boot Code	63
1.11 Bootloader (BSL).....	63
1.12 JTAG Mailbox (JMB) System	63
1.13 JTAG and SBW Lock Mechanism Using the Electronic Fuse	64
1.14 Device Descriptor Table	65
1.15 SFR Registers.....	72
1.16 SYS Registers.....	76

1.1 System Control Module (SYS) Introduction

SYS is responsible for the interaction between various modules throughout the system. The functions that SYS provides for are not inherent to the modules themselves. Address decoding, bus arbitration, interrupt event consolidation, and reset generation are some examples of the many functions that SYS provides.

1.2 System Reset and Initialization

The system reset circuitry is shown in [Figure 1-1](#) and sources a brownout reset (BOR), a power-on reset (POR), and a power-up clear (PUC). Different events trigger these reset signals and different initial conditions exist depending on which signal was generated.

A BOR is a device reset. A BOR is generated only by the following events:

- Powering up the device
- Low signal on the $\overline{\text{RST}}/\text{NMI}$ pin when configured in the reset mode
- Wake-up event from LPMx.5 (that is, LPM3.5 or LPM4.5) mode
- SVS_H low condition, when enabled (see the [PMM and SVS](#) chapter for details)
- Software BOR event (see the [PMM and SVS](#) chapter for details)

A POR is always generated when a BOR is generated, but a BOR is not generated by a POR. The following events trigger a POR:

- BOR signal
- Software POR event (see the [PMM and SVS](#) chapter for details)

A PUC is always generated when a POR is generated, but a POR is not generated by a PUC. The following events trigger a PUC:

- POR signal
- Watchdog timer expiration when watchdog mode only (see the [WDT_A](#) chapter for details)
- Watchdog timer password violation (see the [WDT_A](#) chapter for details)
- FRAM memory password violation (see the [FRAM Controller](#) chapter for details)
- Power Management Module password violation (see the [PMM and SVS](#) chapter for details)
- Memory Protection Unit password violation (see the [MPU](#) chapter for details)
- Memory segment violation (see the [MPU](#) chapter for details)
- Clock System password violation (see the [Clock System](#) chapter for details)
- Fetch from peripheral area
- Uncorrectable FRAM bit error detection

NOTE: The number and type of resets available may vary from device to device. See the device-specific data sheet for all reset sources available.

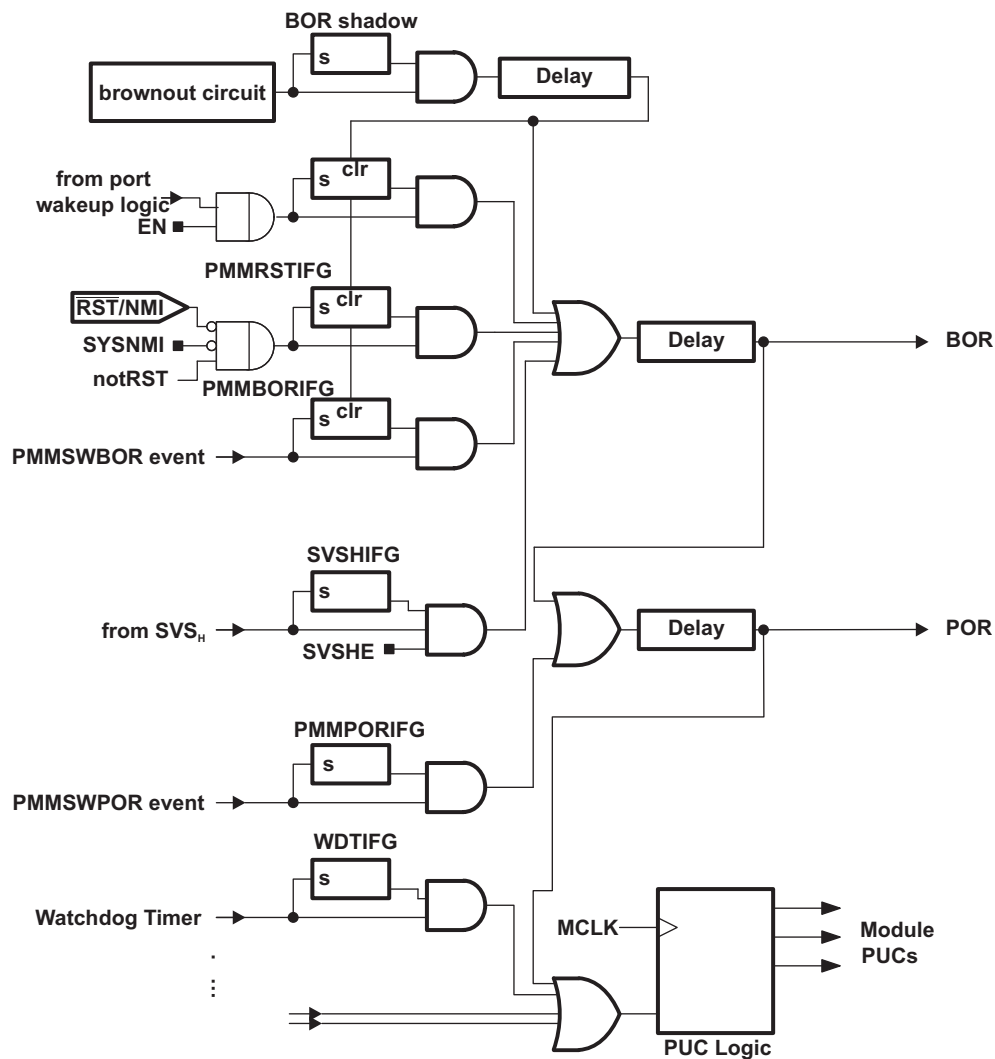


Figure 1-1. BOR, POR, and PUC Reset Circuit

1.2.1 Device Initial Conditions After System Reset

After a BOR, the initial device conditions are:

- The $\overline{\text{RST/NMI}}$ pin is configured in the reset mode. See [Section 1.7](#) for details on configuring the $\overline{\text{RST/NMI}}$ pin.
- I/O pins are switched to input mode as described in the [Digital I/O](#) chapter.
- Other peripheral modules and registers are initialized as described in their respective chapters.
- Status register (SR) is reset.
- The watchdog timer powers up active in watchdog mode.
- Program counter (PC) is loaded with the boot code address and boot code execution begins at that address. See [Section 1.10](#) for more information regarding the boot code. Upon completion of the boot code, the PC is loaded with the address contained at the SYSRSTIV reset location (0FFFEh).

After a system reset, user software must initialize the device for the application requirements. The following must occur:

- Initialize the stack pointer (SP), typically to the top of RAM when available, otherwise FRAM location.
- Initialize the watchdog to the requirements of the application.
- Configure peripheral modules to the requirements of the application.

NOTE: A device that is unprogrammed or blank is defined as having its reset vector value, residing at memory address FFFEh, equal to FFFFh. Upon system reset of a blank device, the device automatically enters operating mode LPM4. See [Section 1.4](#) for information on operating modes and [Section 1.3.6](#) for details on interrupt vectors.

1.3 Interrupts

The interrupt priorities are fixed and defined by the arrangement of the modules in the connection chain as shown in [Figure 1-2](#). Interrupt priorities determine what interrupt is taken when more than one interrupt is pending simultaneously.

There are three types of interrupts:

- System reset
- (Non)maskable
- Maskable

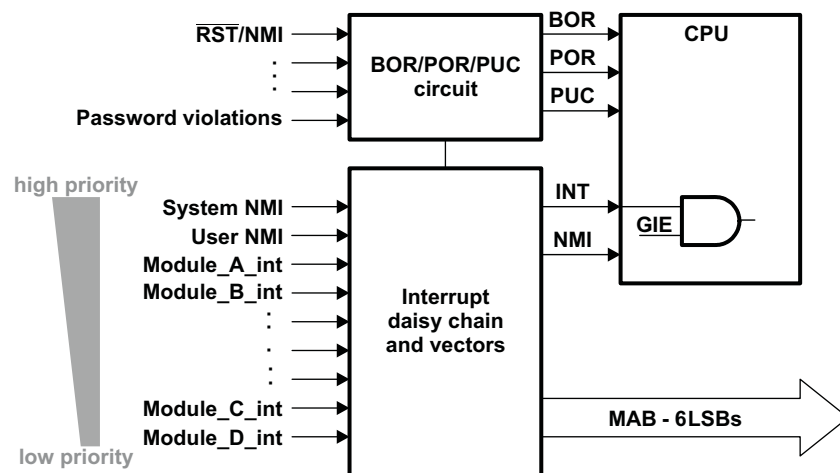


Figure 1-2. Interrupt Priority

NOTE: The types of interrupt sources available and their respective priorities change from device to device. See the device-specific data sheet for all interrupt sources and their priorities.

1.3.1 (Non)Maskable Interrupts (NMIs)

In general, NMIs are not masked by the general interrupt enable (GIE) bit. Two levels of NMIs are supported — system NMI (SNMI) and user NMI (UNMI). The NMI sources are enabled by individual interrupt enable bits. When an NMI interrupt is accepted, other NMIs of that level are automatically disabled to prevent nesting of consecutive NMIs of the same level. Program execution begins at the address stored in the NMI vector as shown in [Section 1.3.6](#). To allow software backward compatibility to users of earlier MSP430 families, the software may, but does not need to, reenables NMI sources. The block diagram for NMI sources is shown in [Section 1.3](#).

A UNMI interrupt can be generated by following sources:

- An edge on the $\overline{\text{RST}}/\text{NMI}$ pin when configured in NMI mode
- An oscillator fault occurs

A SNMI interrupt can be generated by following sources:

- FRAM errors (see the [FRAM Controller](#) chapter for details)
- Vacant memory access
- JTAG mailbox (JMB) event

NOTE: The number and types of NMI sources may vary from device to device. See the device-specific data sheet for all NMI sources available.

1.3.2 SNMI Timing

Consecutive SNMIs that occur at a higher rate than they can be handled (interrupt storm) allow the main program to execute one instruction after the SNMI handler is finished with a RETI instruction, before the SNMI handler is executed again. Consecutive SNMIs are not interrupted by UNMIs in this case. This avoids a blocking behavior on high SNMI rates.

1.3.3 Maskable Interrupts

Maskable interrupts are caused by peripherals with interrupt capability. Each maskable interrupt source can be disabled individually by an interrupt enable bit, or all maskable interrupts can be disabled by the general interrupt enable (GIE) bit in the status register (SR).

Each individual peripheral interrupt is discussed in its respective module chapter in this manual.

1.3.4 Interrupt Processing

When an interrupt is requested from a peripheral and the peripheral interrupt enable bit and GIE bit are set, the interrupt service routine is requested. Only the individual enable bit must be set for (non)-maskable interrupts (NMI) to be requested.

1.3.4.1 Interrupt Acceptance

The interrupt latency is six cycles, starting with the acceptance of an interrupt request, and lasting until the start of execution of the first instruction of the interrupt service routine, as shown in [Figure 1-3](#). The interrupt logic executes the following:

1. Any currently executing instruction is completed.
2. The PC, which points to the next instruction, is pushed onto the stack.
3. The SR is pushed onto the stack.
4. The interrupt with the highest priority is selected if multiple interrupts occurred during the last instruction and are pending for service.
5. The interrupt request flag resets automatically on single-source flags. Multiple source flags remain set for servicing by software.
6. All bits of SR are cleared except SCG0, thereby terminating any low-power mode. Because the GIE bit is cleared, further interrupts are disabled.
7. The content of the interrupt vector is loaded into the PC; the program continues with the interrupt service routine at that address.

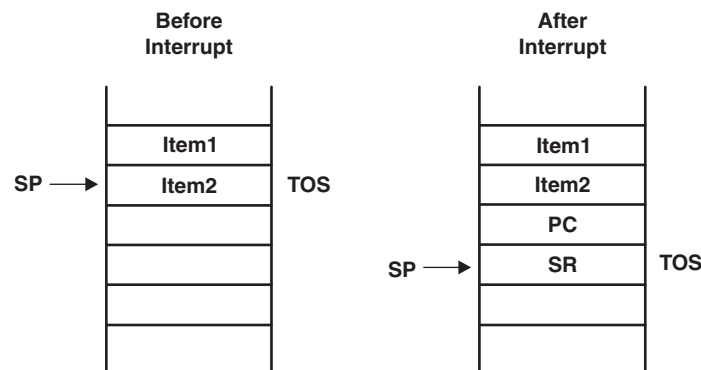


Figure 1-3. Interrupt Processing

NOTE: Enable and Disable Interrupt

Due to the pipelined CPU architecture, setting the general interrupt enable (GIE) requires special care.

- The instruction immediately after the enable interrupts instruction (EINT) is always executed, even if an interrupt service request is pending.
- Include at least one instruction between the clear of an interrupt enable or interrupt flag and the EINT instruction. For example: Insert a NOP instruction in front of the EINT instruction.
- Include at least one instruction between DINT and the start of a code sequence that requires protection from interrupts. For example: Insert a NOP instruction after the DINT.
- Never clear the general interrupt enable (GIE) immediately after setting it. Insert at least one instruction in between such sequence.

The rules above apply to all instructions that set or clear the general interrupt enable bit. Not following these rules might result in unexpected CPU execution.

1.3.4.2 Return From Interrupt

The interrupt handling routine terminates with the instruction:

```
RETI //return from an interrupt service routine
```

The return from the interrupt takes five cycles to execute the following actions and is illustrated in [Figure 1-4](#).

1. The SR with all previous settings pops from the stack. All previous settings of GIE, CPUOFF, and so on are now in effect, regardless of the settings used during the interrupt service routine.
2. The PC pops from the stack and begins execution where it was interrupted.

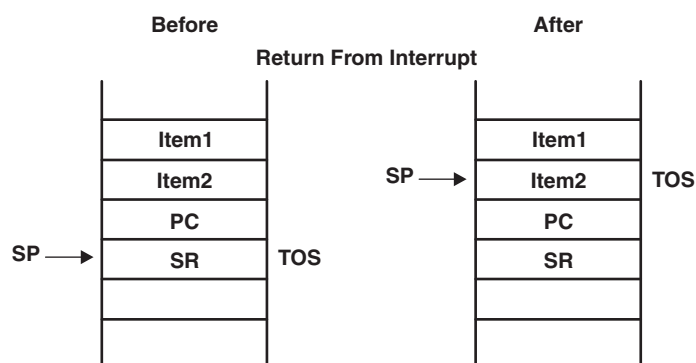


Figure 1-4. Return From Interrupt

1.3.5 Interrupt Nesting

Interrupt nesting is enabled if the GIE bit is set inside an interrupt service routine. When interrupt nesting is enabled, any interrupt occurring during an interrupt service routine interrupts the routine, regardless of the interrupt priorities.

1.3.6 Interrupt Vectors

The interrupt vectors are located in the address range 0FFFFh to 0FF80h, for a maximum of 64 interrupt sources. A vector is programmed by the user and points to the start location of the corresponding interrupt service routine. [Table 1-1](#) is an example of the interrupt vectors available. See the device-specific data sheet for the complete interrupt vector list.

Table 1-1. Interrupt Sources, Flags, and Vectors

Interrupt Source	Interrupt Flag	System Interrupt	Word Address	Priority
Reset: power up, external reset watchdog, FRAM password	... WDTIFG FRCTLPW	... Reset	0FFFEh	... Highest
System NMI: JTAG Mailbox	JMBINIFG, JMBOUTIFG	(Non)maskable	0FFFCh	...
User NMI: NMI oscillator fault	... NMIFG OFIFG	... (Non)maskable (Non)maskable	... 0FFFAh	
Device specific			0FFF8h	...
...			...	...
Watchdog timer	WDTIFG	Maskable	...	...
...			...	...
Device specific			...	...
Reserved		Maskable	...	Lowest

Some interrupt enable bits and interrupt flags, as well as control bits for the $\overline{\text{RST}}$ /NMI pin, are located in the special function registers (SFR). The SFR are located in the peripheral address range and are byte and word accessible. See the device-specific data sheet for the SFR configuration.

1.3.6.1 Alternate Interrupt Vectors

On devices that contain RAM, it is possible to use the RAM as an alternate location for the interrupt vector locations. Setting the SYSRIVECT bit to '1' in SYSCTL causes the interrupt vectors to be remapped to the top of RAM. The total RAM size varies depending on the device configurations and could include one or multiple RAM sections. The alternate location is always the highest address of the entire RAM space available in the device. Note that the SYSRIVECT bit is automatically cleared on a BOR, so the default reset vector location (0FFFEh) will be used after a BOR before setting the SYSRIVECT bit to '1'.

1.3.7 SYS Interrupt Vector Generators

SYS collects all system NMI (SNMI) sources, user NMI (UNMI) sources, and BOR, POR, or PUC (reset) sources of all the other modules. They are combined into three interrupt vectors. The interrupt vector registers SYSRSTIV, SYSSNIV, SYSUNIV are used to determine which flags requested an interrupt or a reset. The interrupt with the highest priority of a group, when enabled, generates a number in the corresponding SYSRSTIV, SYSSNIV, SYSUNIV register. This number can be directly added to the program counter, causing a branch to the appropriate portion of the interrupt service routine. Disabled interrupts do not affect the SYSRSTIV, SYSSNIV, SYSUNIV values. Reading SYSRSTIV, SYSSNIV, SYSUNIV register automatically resets the highest pending interrupt flag of that register. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. Writing to the SYSRSTIV, SYSSNIV, SYSUNIV register automatically resets all pending interrupt flags of the group.

1.3.7.1 SYSSNIV Software Example

The following software example shows the recommended use of SYSSNIV. The SYSSNIV value is added to the PC to automatically jump to the appropriate routine. For SYSRSTIV and SYSUNIV, a similar software approach can be used. The following is an example for a generic device. Vectors can change in priority for a given device. The device-specific data sheet should be referenced for the vector locations. All vectors should be coded symbolically to allow for easy portability of code.

```

SNI_ISR:  ADD    &SYSSNIV,PC    ; Add offset to jump table
          RETI                    ; Vector 0: No interrupt
          JMP    DBD_ISR        ; Vector 2: DBDIFG
          JMP    ACCTIM_ISR     ; Vector 4: ACCTIMIFG
          JMP    RSVD1_ISR     ; Vector 6: Reserved for future usage.
          JMP    RSVD2_ISR     ; Vector 8: Reserved for future usage.
          JMP    RSVD3_ISR     ; Vector 10: Reserved for future usage.
          JMP    RSVD4_ISR     ; Vector 12: Reserved for future usage.
          JMP    ACCV_ISR      ; Vector 14: ACCVIFG
          JMP    VMA_ISR       ; Vector 16: VMAIFG
          JMP    JMBI_ISR      ; Vector 18: JMBINIFG
          JMP    JMBO_ISR      ; Vector 20: JMBOUTIFG
          JMP    SBD_ISR       ; Vector 22: SBDIFG

DBD_ISR:  ; Vector 2: DBDIFG
          ...                    ; Task_2 starts here
          RETI                    ; Return
ACCTIM_ISR: ; Vector 4
          ...                    ; Task_4 starts here
          RETI                    ; Return
RSVD1_ISR: ; Vector 6
          ...                    ; Task_6 starts here
          RETI                    ; Return
RSVD2_ISR: ; Vector 8
          ...                    ; Task_8 starts here
          RETI                    ; Return
RSVD3_ISR: ; Vector 10
          ...                    ; Task_10 starts here
          RETI                    ; Return
RSVD4_ISR: ; Vector 12
          ...                    ; Task_12 starts here
          RETI                    ; Return
ACCV_ISR: ; Vector 14
          ...                    ; Task_14 starts here
          RETI                    ; Return
VMA_ISR:  ; Vector 16
          ...                    ; Task_16 starts here
          RETI                    ; Return
JMBI_ISR: ; Vector 18
          ...                    ; Task_18 starts here
JMBO_ISR: ; Vector 20
          ...                    ; Task_20 starts here
          RETI                    ; Return
SBD_ISR:  ; Vector 22
          ...                    ; Task_22 starts here
          RETI                    ; Return

```

1.4 Operating Modes

The MSP430 family is designed for ultralow-power applications and uses different operating modes shown in [Figure 1-5](#).

The operating modes take into account three different needs:

- Ultra-low power
- Speed and data throughput
- Minimization of individual peripheral current consumption

The low-power modes LPM0 through LPM4 are configured with the CPUOFF, OSCOFF, SCG0, and SCG1 bits in the SR. The advantage of including the CPUOFF, OSCOFF, SCG0, and SCG1 mode-control bits in the SR is that the present operating mode is saved onto the stack during an interrupt service routine. Program flow returns to the previous operating mode if the saved SR value is not altered during the interrupt service routine. Program flow can be returned to a different operating mode by manipulating the saved SR value on the stack inside of the interrupt service routine. When setting any of the mode-control bits, the selected operating mode takes effect immediately. Peripherals operating with any disabled clock are disabled until the clock becomes active. Peripherals may also be disabled with their individual control register settings. All I/O port pins, RAM, and registers are unchanged. Wakeup from LPM0 through LPM4 is possible through all enabled interrupts.

When LPMx.5 (LPM3.5 or LPM4.5) is entered, the voltage regulator of the Power Management Module (PMM) is disabled. All RAM and register contents are lost. Although the I/O register contents are lost, the I/O pin states are locked upon LPMx.5 entry. See the [Digital I/O](#) chapter for further details. Wakeup from LPM4.5 is possible through a power sequence, a $\overline{\text{RST}}$ event, or from specific I/O. Wakeup from LPM3.5 is possible through a power sequence, a $\overline{\text{RST}}$ event, RTC event, or from specific I/O.

NOTE: The TEST/SBWTCK pin is used to enable the connection of external development tools with the device through Spy-Bi-Wire or JTAG debug protocols. The connection is usually enabled when the TEST/SBWTCK is high. When the connection is enabled the device enters a debug mode. In the debug mode the entry and wake-up times to and from low power modes may be different compared to normal operation. Pay careful attention to the real-time behavior when using low power modes with the device connected to a development tool!

See the [EEM](#) chapter for further details.



Table 1-2. Operation Modes

SCG1 ⁽¹⁾	SCG0	OSCOFF ⁽¹⁾	CPUOFF ⁽¹⁾	Mode	CPU and Clocks Status ⁽²⁾
0	0	0	0	Active	CPU, MCLK are active. ACLK is active. SMCLK optionally active (SMCLKOFF = 0). DCO is enabled if sources ACLK, MCLK, or SMCLK (SMCLKOFF = 0). DCO bias is enabled if DCO is enabled or DCO sources MCLK or SMCLK (SMCLKOFF = 0).
0	0	0	1	LPM0	CPU, MCLK are disabled. ACLK is active. SMCLK optionally active (SMCLKOFF = 0). DCO is enabled if sources ACLK or SMCLK (SMCLKOFF = 0). DCO bias is enabled if DCO is enabled or DCO sources MCLK or SMCLK (SMCLKOFF = 0).
0	1	0	1	LPM1	CPU, MCLK are disabled. ACLK is active. SMCLK optionally active (SMCLKOFF = 0). DCO is enabled if sources ACLK or SMCLK (SMCLKOFF = 0). DCO bias is enabled if DCO is enabled or DCO sources MCLK or SMCLK (SMCLKOFF = 0).
1	0	0	1	LPM2	CPU, MCLK are disabled. ACLK is active. SMCLK is disabled.
1	1	0	1	LPM3	CPU, MCLK are disabled. ACLK is active. SMCLK is disabled.
1	1	1	1	LPM4	CPU and all clocks are disabled.
1	1	1	1	LPM3.5	When PMMREGOFF = 1, regulator is disabled. No memory retention. In this mode, RTC operation is possible when configured properly. See the <i>RTC</i> module for further details.
1	1	1	1	LPM4.5	When PMMREGOFF = 1, regulator is disabled. No memory retention. In this mode, all clock sources are disabled; that is, no RTC operation is possible.

⁽¹⁾ This bit is automatically reset when exiting low-power modes. See [Section 1.4.2](#) for details.

⁽²⁾ The low-power modes and, hence, the system clocks can be affected by the clock request system. See the [Clock System](#) chapter for details.

1.4.1 Low-Power Modes and Clock Requests

A peripheral module requests its clock sources automatically from the clock system (CS) module if it is required for its proper operation, regardless of the current power mode of operation. Refer to the "Operation From Low-Power Modes, Requested by Peripheral Modules" section in the [Clock System](#) chapter.

Because of the clock request mechanism the system might not reach the low-power modes requested by the bits set in the CPU's status register SR as listed in [Table 1-3](#).

Table 1-3. Requested vs Actual LPM

Requested LPM (SR Bits according to Table 1-2)	Actual LPM...		
	If No Clock Requested	If Only ACLK Requested	If SMCLK Requested
LPM0	LPM0	LPM0	LPM0
LPM1	LPM1	LPM1	LPM1
LPM2	LPM2	LPM2	LPM0
LPM3	LPM3	LPM3	LPM1
LPM4	LPM4	LPM3	LPM1

1.4.2 Entering and Exiting Low-Power Modes LPM0 Through LPM4

An enabled interrupt event wakes the device from low-power operating modes LPM0 through LPM4. The program flow for exiting LPM0 through LPM4 is:

- Enter interrupt service routine
 - The PC and SR are stored on the stack.
 - The CPUOFF, SCG1, and OSCOFF bits are automatically reset.
- Options for returning from the interrupt service routine
 - The original SR is popped from the stack, restoring the previous operating mode.
 - The SR bits stored on the stack can be modified within the interrupt service routine returning to a different operating mode when the RETI instruction is executed.

```
; Enter LPM0 Example
  BIS    #GIE+CPUOFF, SR                ; Enter LPM0
; ...                                     ; Program stops here
;
; Exit LPM0 Interrupt Service Routine
  BIC    #CPUOFF, 0(SP)                  ; Exit LPM0 on RETI
  RETI

; Enter LPM3 Example
  BIS    #GIE+CPUOFF+SCG1+SCG0, SR      ; Enter LPM3
; ...                                     ; Program stops here
;
; Exit LPM3 Interrupt Service Routine
  BIC    #CPUOFF+SCG1+SCG0, 0(SP)        ; Exit LPM3 on RETI
  RETI

; Enter LPM4 Example
  BIS    #GIE+CPUOFF+OSCOFF+SCG1+SCG0, SR ; Enter LPM4
; ...                                     ; Program stops here
;
; Exit LPM4 Interrupt Service Routine
  BIC    #CPUOFF+OSCOFF+SCG1+SCG0, 0(SP) ; Exit LPM4 on RETI
  RETI
```

1.4.3 Low-Power Modes LPM3.5 and LPM4.5 (LPMx.5)

The low-power modes LPM3.5 and LPM4.5 (LPMx.5⁽¹⁾) give the lowest power consumption on a device. In LPMx.5, the core LDO of the device is switched off. This has the following effects:

- Most of the modules are powered down.
 - In LPM3.5, only modules powered by the RTC LDO continue to operate. At least an RTC module is connected to the RTC LDO. Refer to the device's data sheet for other modules (if any) that are connected to the RTC LDO.
 - In LPM4.5 the RTC LDO and the connected modules are switched off.
- The register content of all modules and the CPU is lost.
- The SRAM content is lost.
- A wake-up from LPMx.5 causes a complete reset of the core.
- The application must initialize the complete device after a wake-up from LPMx.5.

The wake-up time from LPMx.5 is much longer than the wake-up time from any other power mode (see the device-specific data sheet). This is because the core domain must power up and the device internal initialization must be done. In addition, the application must be initialized again. Therefore, use LPMx.5 only when the application is in LPMx.5 for a long time.

⁽¹⁾ The abbreviation "LPMx.5" is used in this document to indicate both LPM3.5 and LPM4.5.

Compute Through Power Loss (CTPL) is a utility API set that leverages FRAM to enable ease of use with LPMx.5 low-power modes and provides a powerful shutdown mode that allows an application to save and restore critical system components when a power loss is detected. Visit [FRAM embedded software utilities for MSP ultra-low-power microcontrollers](#) for details.

1.4.3.1 Enter LPMx.5

Do the following steps to enter LPMx.5:

1. Store any information that must be available after wakeup from LPMx.5 in FRAM.
2. For LPM4.5 set all ports to general-purpose I/Os (PxSEL0 = 00h and PxSEL1 = 00h). For LPM3.5 if the LF crystal oscillator is used do not change the settings for the I/Os shared with the LF-crystal-oscillator. These pins must be configured as LFXIN and LFXOUT. Set all other port pins to general-purpose I/Os with PxSEL0 and PxSEL1 bits equal to 0.
3. Set the port pin direction and output bits as necessary for the application.
4. To enable a wakeup from an I/O do the following:
 - a. Select the wakeup edge (PxIES)
 - b. Clear the interrupt flag (PxIFG)
 - c. Set the interrupt enable bit (PxIE)
5. For LPM3.5 the modules that stay active must be enabled. For example, the RTC must be enabled if necessary. Only modules connected to the RTC LDO can stay active.
6. For LPM3.5 if necessary enable any interrupt sources from these modules as wakeup sources. Refer to the corresponding module chapter.
7. Disable the watchdog timer WDT if it is enabled and in watchdog mode. If the WDT is enabled and in watchdog mode, the device does not enter LPMx.5.
8. Clear the GIE bit:


```
BIC #GIE, SR
```
9. Do the following steps to set the PMMREGOFF bit in the PMMCTL0 register:
 - a. Write the correct PMM password to get write access to the PMM control registers.


```
MOV.B #PMMPW_H, &PMMCTL0_H
```
 - b. Set PMMREGOFF bit in the PMMCTL0 register.


```
BIS.B #PMMREGOFF, &PMMCTL0_L
```
 - c. Optionally, disable the SVS during LPMx.5 by clearing the SVSHE bit in PMMCTL0.


```
BIC.B #SVSHE, &PMMCTL0_L
```
 - d. Write an incorrect PMM password to disable the write access to the PMM control registers.


```
MOV.B #000h, &PMMCTL0_H
```
10. Enter LPMx.5 with the following instruction:


```
BIS #CPU0FF+OSCOFF+SCG0+SCG1, SR
```

After this process, the device enters LPM3.5 if modules connected to the RTC LDO are enabled, and it enters LPM4.5 if none of the modules connected to the RTC LDO are enabled.

1.4.3.2 Exit and Wake up From LPM3.5

The following conditions cause an exit from LPM3.5:

- A wake-up event on an I/O if configured and enabled. The interrupt flag of the corresponding port pin is set (PxIFG). The PMMLPM5IFG bit is set.
- A wake-up event from a module connected to the RTC LDO if enabled. The corresponding interrupt flag in the module is set. The PMMLPM5IFG bit is set.
- A wake-up signal from the RST pin.
- A power cycle. Either the SVSHIFG or none of the PMMIFGs is set.

Any exit from LPM3.5 causes a BOR. The program execution starts at the address the reset vector points to. PMMLPM5IFG = 1 indicates a wakeup from LPM3.5, or the System Reset Vector Word register (SYSRSTIV) can be used to decode the reset condition (see the device data sheet).

After the wakeup from LPM3.5, the state of the I/Os and the modules connected to the RTC LDO are locked and remain unchanged until the application clears the LOCKLPM5 bit in the PM5CTL0 register.

Do the following steps after a wakeup from LPM3.5:

1. Initialize the registers of the modules connected to the RTC LDO exactly the same way as they were configured before the device entered LPM3.5 but do not enable the interrupts.
2. Initialize the port registers exactly the same way as they were configured before the device entered LPM3.5 but do not enable port interrupts.
3. If the LF-crystal-oscillator was used in LPM3.5 the corresponding I/Os must be configured as LFXIN and LFXOUT. The LF-crystal-oscillator must be enabled in the clock system (see the clock system CS chapter).
4. Clear the LOCKLPM5 bit in the PM5CTL0 register.
5. Enable port interrupts as necessary.
6. Enable module interrupts.
7. After enabling the port and module interrupts the wake-up interrupt will be serviced as a normal interrupt.

1.4.3.3 Exit and Wake up From LPM4.5

The following conditions will cause an exit from LPM4.5:

- A wakeup event on an I/O if configured and enabled. The interrupt flag of the corresponding port pin is set (PxIFG). The PMMLPM5IFG bit is set.
- A wakeup from the RST pin.
- A power-cycle. Either the SVSHIFG or none of the PMMIFGs is set.

Any exit from LPM4.5 causes a BOR. The program execution starts at the address the reset vector points to. PMMLPM5IFG = 1 indicates a wakeup from LPM4.5, or the System Reset Vector Word register (SYSRSTIV) can be used to decode the reset condition (see the device data sheet).

After the wake-up from LPM4.5 the state of the I/Os are locked and remain unchanged until the application clears the LOCKLPM5 bit in the PM5CTL0 register.

Do the following steps after a wakeup from LPM4.5:

1. Initialize the port registers exactly the same way as they were configured before the device entered LPM4.5, but do not enable port interrupts.
2. Clear the LOCKLPM5 bit in the PM5CTL0 register.
3. Enable port interrupts as necessary.
4. After enabling the port interrupts the wake-up interrupt will be serviced as a normal interrupt.

If a crystal oscillator is needed after a wakeup from LPM4.5, configure the corresponding pins and start the oscillator after clearing the LOCKLPM5 bit.

1.5 Principles for Low-Power Applications

Often, the most important factor for reducing power consumption is using the device clock system to maximize the time in LPM3 or LPM4 modes whenever possible.

- Use interrupts to wake the processor and control program flow.
- Peripherals should be switched on only when needed.
- Use low-power integrated peripheral modules in place of software driven functions. For example, Timer_A and Timer_B can automatically generate PWM and capture external timing with no CPU resources.
- Calculated branching and fast table lookups should be used in place of flag polling and long software calculations.

- Avoid frequent subroutine and function calls due to overhead.
- For longer software routines, single-cycle CPU registers should be used.

If the application has low duty cycle and slow response time events, maximizing time in LPMx.5 can further reduce power consumption significantly.

1.6 Connection of Unused Pins

The correct termination of all unused pins is listed in [Table 1-4](#).

Table 1-4. Connection of Unused Pins⁽¹⁾

Pin	Potential	Comment
AVCC	DV _{CC}	
AVSS	DV _{SS}	
Px.0 to Px.7	Open	Switched to port function, output direction (PxDIR.n = 1)
$\overline{\text{RST}}/\text{NMI}$	DV _{CC} or V _{CC}	47-k Ω pullup or internal pullup selected with 2.2-nF (10-nF ⁽²⁾) pulldown
PJ.0/TDO PJ.1/TDI PJ.2/TMS PJ.3/TCK	Open	The JTAG pins are shared with general-purpose I/O function (PJ.x). If not being used, these should be switched to port function, output direction. When used as JTAG pins, these pins should remain open.
TEST	Open	This pin always has an internal pulldown enabled.

⁽¹⁾ Any unused pin with a secondary function that is shared with general-purpose I/O should follow the Px.0 to Px.7 unused pin connection guidelines.

⁽²⁾ The pulldown capacitor should not exceed 2.2 nF when using devices in Spy-Bi-Wire mode or in 4-wire JTAG mode with TI tools like FET interfaces or GANG programmers. If JTAG or Spy-Bi-Wire access is not needed, up to a 10-nF pulldown capacitor may be used.

1.7 Reset Pin ($\overline{\text{RST}}/\text{NMI}$) Configuration

The reset pin can be configured as a reset function (default) or as an NMI function through the Special Function Register (SFR), SFRRPCR. Setting SYSNMI causes the $\overline{\text{RST}}/\text{NMI}$ pin to be configured as an external NMI source. The external NMI is edge sensitive and its edge is selectable by SYSNMIIES. Setting the NMIIE enables the interrupt of the external NMI. Upon an external NMI event, the NMIIFG is set.

The $\overline{\text{RST}}/\text{NMI}$ pin can have either a pullup or pulldown present or not. SYSRSTUP selects either pullup or pulldown, and SYSRSTRE causes the pullup or pulldown to be enabled or not. If the $\overline{\text{RST}}/\text{NMI}$ pin is unused, it is required to have either the internal pullup selected and enabled or an external resistor connected to the $\overline{\text{RST}}/\text{NMI}$ pin as shown in [Table 1-4](#).

There is a digital filter that suppresses short pulses on the reset pin to avoid unintended resets of the device. The minimum reset pulse duration is specified in the device data sheet. The filter is active only if the pin is configured in its reset function. The filter is disabled if the pin is used as an external NMI source.

1.8 Configuring JTAG Pins

The JTAG pins are shared with general-purpose I/O pins. After a BOR, the SYSJTAGPIN bit in the SYSCTL register is cleared. With SYSJTAGPIN cleared, the pins with JTAG functionality are configured as general-purpose I/O. In this case only a special sequences on the TEST and $\overline{\text{RST}}/\text{NMI}$ pins enables the JTAG functionality. As long as the TEST pin is pulled to DVCC, the pins remain in their JTAG functionality. If the TEST pin is released to DVSS, the shared JTAG pins revert to general-purpose I/Os.

If SYSJTAGPIN = 1, the JTAG pins are permanently configured to 4-wire JTAG mode and remain in this mode until another BOR occurs. Use this feature early in your software if the MSP430 is part of a JTAG chain. Note, that this also disables the Spy-Bi-Wire mode.

The SYSJTAGPIN is a write only once function. Clearing it by software is not possible.

1.9 Vacant Memory Space

Vacant memory is nonexistent memory space. Accesses to vacant memory space generate a system (non)maskable interrupt (SNMI) when enabled (VMAIE = 1). Reads from vacant memory results in the value 3FFFh. In the case of a fetch, this is taken as JMP \$. Fetch accesses from vacant peripheral space result in a PUC. After the boot code is executed, the boot code memory behaves like vacant memory space and causes an NMI on access.

1.10 Boot Code

The boot code loads factory stored calibration values of the oscillator and reference voltages. In addition, it checks for a bootloader (BSL) entry sequence. The boot code is always executed after a BOR.

1.11 Bootloader (BSL)

The BSL is software that is executed after start-up when a certain BSL entry condition is applied. The BSL lets the user communicate with the embedded memory in the microcontroller during the prototyping phase, final production, and in service. All memory mapped resources, the programmable memory, the data memory (RAM), and the peripherals, can be modified by the BSL as required.

A basic BSL program is provided by TI and resides in ROM at memory space 01000h through 017FFh. The BSL supports the commonly used UART protocol with RS232 interfacing, allowing flexible use of both hardware and software. Depending on the device, additional BSL communication interfaces are supported. For details of the available and configured BSL communication interfaces, see [Section 1.14.3.5](#).

To use the BSL, a specific BSL entry sequence must be applied to the $\overline{\text{RST}}$ /NMI and TEST pins. A correct entry sequence causes SYSBSLIND to be set. An added sequence of commands initiates the desired function. A bootloader session can be exited by continuing operation at a defined user program address or by applying the standard reset sequence. Access to the device memory through the BSL is protected against misuse by a user-defined password.

Two BSL signatures, BSL Signature 1 (memory location 0FF84h) and BSL Signature 2 (memory location 0FF86h) reside in FRAM and can be used to control the behavior of the BSL. Writing 05555h to BSL Signature 1 or BSL Signature 2 disables the BSL function and any access to the BSL memory space causes a vacant memory access as described in [Section 1.9](#). Most BSL commands require the BSL to be unlocked by a user-defined password. An incorrect password erases the device memory as a security feature. Writing 0AAAAh to both BSL Signature 1 and BSL Signature 2 disables this security feature. This causes a password error to be returned by the BSL, but the device memory is not erased. In this case, unlimited password attempts are possible.

For more details, see the [MSP430FR57xx](#), [MSP430FR58xx](#), [MSP430FR59xx](#), [MSP430FR68xx](#), and [MSP430FR69xx Bootloader \(BSL\) User's Guide](#).

Some JTAG commands are still possible after the device is secured, including the BYPASS command (see IEEE Std 1149-2001) and the JMB_EXCHANGE command, which allows access to the JTAG Mailbox System (see [Section 1.12](#) for details).

1.12 JTAG Mailbox (JMB) System

The SYS module provides the capability to exchange user data through the regular JTAG test/debug interface. The idea behind the JMB is to have a direct interface to the CPU during debugging, programming, and test that is identical for all devices of this family and uses only few or no user application resources. The JTAG interface was chosen because it is available on all devices and is a dedicated resource for debugging, programming, and test.

Applications of the JMB are:

- Providing entry password for device lock or unlock protection
- Run-time data exchange (RTDX)

1.12.1 JMB Configuration

The JMB supports two transfer modes: 16-bit and 32-bit. Setting JMBMODE enables 32-bit transfer mode. Clearing JMBMODE enables 16-bit transfer mode.

1.12.2 JMBOUT0 and JMBOUT1 Outgoing Mailbox

Two 16-bit registers are available for outgoing messages to the JTAG port. JMBOUT0 is only used when using 16-bit transfer mode (JMBMODE = 0). JMBOUT1 is used in addition to JMBOUT0 when using 32-bit transfer mode (JMBMODE = 1). When the application wishes to send a message to the JTAG port, it writes data to JMBOUT0 for 16-bit mode, or JMBOUT0 and JMBOUT1 for 32-bit mode.

JMBOUT0FG and JMBOUT1FG are read only flags that indicate the status of JMBOUT0 and JMBOUT1, respectively. When JMBOUT0FG is set, JMBOUT0 has been read by the JTAG port and is ready to receive new data. When JMBOUT0FG is reset, the JMBOUT0 is not ready to receive new data. JMBOUT1FG behaves similarly.

1.12.3 JMBIN0 and JMBIN1 Incoming Mailbox

Two 16-bit registers are available for incoming messages from the JTAG port. Only JMBIN0 is used when in 16-bit transfer mode (JMBMODE = 0). JMBIN1 is used in addition to JMBIN0 when using 32-bit transfer mode (JMBMODE = 1). When the JTAG port wishes to send a message to the application, it writes data to JMBIN0 for 16-bit mode, or JMBIN0 and JMBIN1 for 32-bit mode.

JMBIN0FG and JMBIN1FG are flags that indicate the status of JMBIN0 and JMBIN1, respectively. When JMBIN0FG is set, JMBIN0 has data that is available for reading. When JMBIN0FG is reset, no new data is available in JMBIN0. JMBIN1FG behaves similarly.

JMBIN0FG and JMBIN1FG can be configured to clear automatically by clearing JMBCLR0OFF and JMBCLR1OFF, respectively. Otherwise, these flags must be cleared by software.

1.12.4 JMB NMI Usage

The JMB handshake mechanism can be configured to use interrupts to avoid unnecessary polling if desired. In 16-bit mode, JMBOUTIFG is set when JMBOUT0 has been read by the JTAG port and is ready to receive data. In 32-bit mode, JMBOUTIFG is set when both JMBOUT0 and JMBOUT1 has been read by the JTAG port and are ready to receive data. If JMBOUTIE is set, these events cause a system NMI. In 16-bit mode, JMBOUTIFG is cleared automatically when data is written to JMBOUT0. In 32-bit mode, JMBOUTIFG is cleared automatically when data is written to both JMBOUT0 and JMBOUT1. In addition, the JMBOUTIFG can be cleared when reading SYSSNIV. Clearing JMBOUTIE disables the NMI interrupt.

In 16-bit mode, JMBINIFG is set when JMBIN0 is available for reading. In 32-bit mode, JMBINIFG is set when both JMBIN0 and JMBIN1 are available for reading. If JMBOUTIE is set, these events cause a system NMI. In 16-bit mode, JMBINIFG is cleared automatically when JMBIN0 is read. In 32-bit mode, JMBINIFG is cleared automatically when both JMBIN0 and JMBIN1 are read. In addition, the JMBINIFG can be cleared when reading SYSSNIV. Clearing JMBINIE disables the NMI interrupt.

1.13 JTAG and SBW Lock Mechanism Using the Electronic Fuse

A device can be protected from unauthorized access by restricting accessibility of JTAG commands that can be transferred to the device by the JTAG and SBW interface. This is achieved by programming the electronic fuse. When the device is protected, the JTAG and SBW interface still remains functional, but JTAG commands that give direct access into the device are completely disabled. There are two ways to lock the device. Both of these require the programming of two signatures that reside in FRAM. JTAG Signature 1 (memory location 0FF80h) and JTAG Signature 2 (memory location 0FF82h) control the behavior of the device locking mechanism.

NOTE: When a device has been protected, Texas Instruments cannot access the device for a customer return. Access is only possible if a BSL is provided with its corresponding key or an unlock mechanism is provided by the customer.

1.13.1 JTAG and SBW Lock Without Password

A device can be locked by writing 05555h to both JTAG Signature 1 and JTAG Signature 2. In this case, the JTAG and SBW interfaces grant access to a limited JTAG command set that restricts accessibility into the device. The only way to unlock the device in this case is to use the BSL to overwrite the JTAG signatures with anything other than 05555h or 0AAAAh. Some JTAG commands are still possible after the device is secured, including the BYPASS command (see IEEE1149-2001 Standard) and the JMB_EXCHANGE command, which allows access to the JTAG Mailbox System (see [Section 1.12](#) for details).

NOTE: Signatures that have been entered do not take effect until the next BOR event has occurred, at which time the signatures are checked.

1.13.2 JTAG and SBW Lock With Password

A device can also be locked by writing 0AAAAh to JTAG Signature 1 and writing JTAG Signature 2 with any value except 05555h. In this case, JTAG and SBW interfaces grant access to a limited JTAG command set that restricts accessibility into the device as in [Section 1.13.1](#), but an additional mechanism is available that can unlock the device with a user-defined password. In this case, JTAG Signature 2 represents a user-defined length in words of the user defined password. For example, a password length of four words would require writing 0004h to JTAG Signature 2. The starting location of the password is fixed at location 0FF88h. As an example, for a password of length 4, the password memory locations would reside at 0FF88h, 0FF8Ah, 0FF8Ch, and 0FF8Eh.

The password is not checked after each BOR; it is checked only if a specific signature is present in the JTAG incoming mailbox. If the JTAG incoming mailbox contains 0A55Ah and 01E1Eh in JMBIN0 and JMBIN1, respectively, the device is expecting a password to be applied. The entered password is compared to the password that is stored in the device password memory locations. If they match, the device unlocks the JTAG and SBW to the complete JTAG command set until the next BOR event occurs.

NOTE: Memory locations 0FF80h through 0FFFFh may also be used for interrupt vector address locations (see the device-specific data sheet). Therefore, if using the password mechanism for JTAG and SBW lock, which uses address locations 0FF88h and higher, these locations may also have interrupt vector addresses assigned to them. Therefore, the same values assigned for any interrupt vector addresses must also be used as password values.

NOTE: Signatures that have been entered do not take effect until the next BOR event has occurred, at which time the signatures are checked. For example, entering a correct password that grants entry into the device followed by an incorrect password without a BOR sequence may still grant access to the device.

1.14 Device Descriptor Table

Each device provides a data structure in memory that allows an unambiguous identification of the device. The validity of the device descriptor can be verified by cyclic redundancy check (CRC). [Figure 1-6](#) shows the logical order and structure of the device descriptor table. The complete device descriptor table and its contents can be found in the device-specific data sheet.

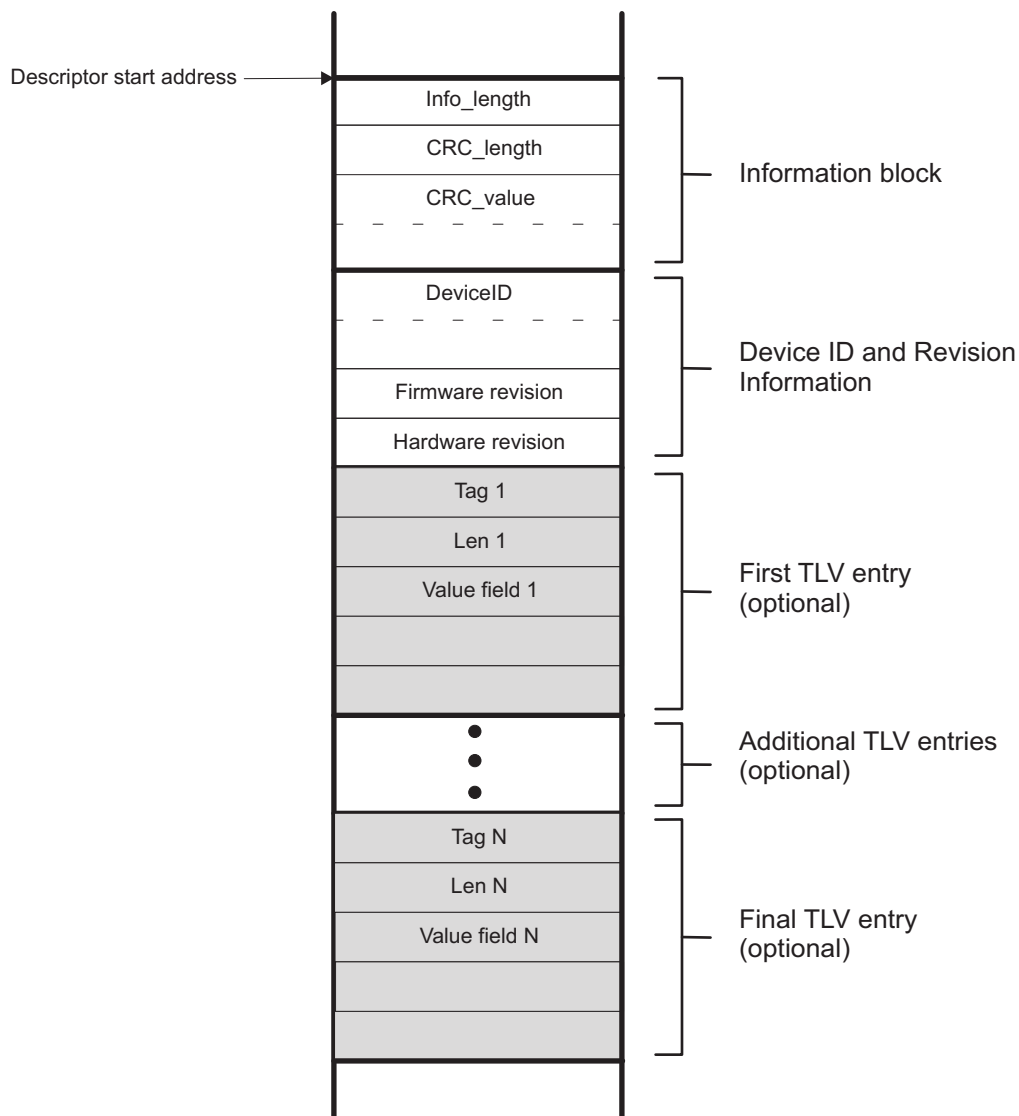


Figure 1-6. Devices Descriptor Table

1.14.1 Identifying Device Type

The value read at address 00FF0h identifies the family branch of the device. All values starting with 80h indicate a hierarchical structure consisting of the information block and a TLV tag-length-value (TLV) structure containing the various descriptors. Any other value than 80h read at address location 00FF0h indicates the device is of an older family and contains a flat descriptor beginning at location 0FF0h. The information block (see [Figure 1-6](#)) contains the device ID, die revisions, firmware revisions, and other manufacturer and tool related information.

The length of the descriptors represented by Info_length is computed as shown in [Equation 1](#):

$$\text{Length} = 2^{\text{Info\_length}} \text{ in 32-bit words} \quad (1)$$

For example, if Info_length = 5, then the length of the descriptors equals 128 bytes.

1.14.2 TLV Descriptors

The TLV descriptors follow the information block. Because the information block is always a fixed length, the start location of the TLV descriptors is fixed for a given device family. For the MSP430FR57xx family, this location is 01A08h. See the device-specific data sheet for the complete TLV structure and what descriptors are available.

The TLV descriptors are unique to their respective TLV block and are always followed by the descriptor block length.

Each TLV descriptor contains a tag field that identifies the descriptor type. [Table 1-5](#) shows the currently supported tags.

Table 1-5. Tag Values

Short Name	Value	Description
LDTAG	01h	Legacy descriptor (1xx, 2xx, 4xx families)
PDTAG	02h	Peripheral discovery descriptor
Reserved	03h	Reserved for future use
Reserved	04h	Reserved for future use
BLANK	05h	Blank descriptor
Reserved	06h	Reserved for future use
Reserved	07h	Reserved for future use
Reserved	08h	Unique Die Record
Reserved	09h-0Fh	Reserved for future use
Reserved	10h	Reserved
ADC12CAL	11h	ADC12 calibration (see Section 1.14.3.2 and Section 1.14.3.3)
REFCAL	12h	REF calibration (see Section 1.14.3.1)
ADC10CAL	13h	ADC10 calibration (see Section 1.14.3.2 and Section 1.14.3.3)
Reserved	14h	Reserved for future use
RANDTAG	15h	Random Number Seed (see Section 1.14.3.4)
Reserved	16h-1Bh	Reserved for future use
BSLTAG	1Ch	BSL Configuration
Reserved	1Dh-FDh	Reserved for future use
TAGEXT	FEh	Tag extender

Each tag field is unique to its respective descriptor and is always followed by a length field. The length field is one byte if the tag value is 01h through 0FDh and represents the length of the descriptor in bytes. If the tag value equals 0FEh (TAGEXT), the next byte extends the tag values, and the following two bytes represent the length of the descriptor in bytes. In this way, a user can search through the TLV descriptor table for a particular tag value, using a routine similar to the following pseudo code:

```
// Identify the descriptor ID (d_ID_value) for the TLV descriptor of interest:
descriptor_address = TLV_START address;

while ( value at descriptor_address != d_ID_value && descriptor_address != TLV_TAGEND &&
descriptor_address < TLV_END)
{
    // Point to next descriptor
    descriptor_address = descriptor_address + (length of the current TLV block) + 2;
}

if (value at descriptor_address == d_ID_value) {
    // Appropriate TLV descriptor has been found!
    Return length of descriptor & descriptor_address as the location of the TLV descriptor
} else {
    // No TLV descriptor found with a matching d_ID_value
    Return a failing condition
}
```

1.14.3 Calibration Values

The TLV structure contains calibration values that can be used to improve the measurement capability of various functions. The calibration values available on a given device are shown in the TLV structure of the device-specific data sheet.

1.14.3.1 REF Calibration

Table 1-6 shows the REF calibration tags.

Table 1-6. REF Calibration Tags

REF Calibration	TAG	12h
	Length	06h
	Low Byte	CAL_ADC_12VREF_FACTOR
	High Byte	
	Low Byte	CAL_ADC_20VREF_FACTOR
	High Byte	
	Low Byte	CAL_ADC_25VREF_FACTOR
	High Byte	

The calibration data for the REF module consists of three words, one word for each reference voltage available (1.2 V, 2.0 V, and 2.5 V). The reference voltages are measured at room temperature. The measured values are normalized by 1.2 V, 2.0 V, or 2.5 V before being stored into the TLV structure, as shown in Equation 2:

$$\begin{aligned}
 \text{CAL\_ADC\_12VREF\_FACTOR} &= \frac{V_{\text{REF+}}}{1.2\text{V}} \times 2^{15} \\
 \text{CAL\_ADC\_20VREF\_FACTOR} &= \frac{V_{\text{REF+}}}{2.0\text{V}} \times 2^{15} \\
 \text{CAL\_ADC\_25VREF\_FACTOR} &= \frac{V_{\text{REF+}}}{2.5\text{V}} \times 2^{15}
 \end{aligned} \tag{2}$$

In this way, a conversion result is corrected by multiplying it with the CAL_12VREF_FACTOR (or CAL_20VREF_FACTOR, CAL_25VREF_FACTOR) and dividing the result by 2^{15} as shown in Equation 3 for each of the respective reference voltages:

$$\begin{aligned}
 \text{ADC}(\text{corrected}) &= \text{ADC}(\text{raw}) \times \text{CAL\_ADC12VREF\_FACTOR} \times \frac{1}{2^{15}} \\
 \text{ADC}(\text{corrected}) &= \text{ADC}(\text{raw}) \times \text{CAL\_ADC20VREF\_FACTOR} \times \frac{1}{2^{15}} \\
 \text{ADC}(\text{corrected}) &= \text{ADC}(\text{raw}) \times \text{CAL\_ADC25VREF\_FACTOR} \times \frac{1}{2^{15}}
 \end{aligned} \tag{3}$$

In the following example, the integrated 1.2-V reference voltage is used during a conversion.

- Conversion result: 0x0100 = 256 decimal
- Reference voltage calibration factor (CAL_12VREF_FACTOR) : 0x7BBB

The following steps show how the ADC conversion result can be corrected:

- Multiply the conversion result by 2 (this step simplifies the final division): 0x0100 x 0x0002 = 0x0200
- Multiply the result by CAL_12VREF_FACTOR: 0x200 x 0x7FEE = 0x00F7_7600
- Divide the result by 2^{16} : 0x00F7_7600 / 0x0001_0000 = 0x0000_00F7 = 247 decimal

1.14.3.2 ADC Offset and Gain Calibration

Table 1-7 shows the ADC calibration tags.

Table 1-7. ADC Calibration Tags

ADC Calibration	TAG	ADC10: 13h ADC12: 11h
	Length	10h
	Low Byte	CAL_ADC_GAIN_FACTOR
	High Byte	
	Low Byte	CAL_ADC_OFFSET
	High Byte	
	Low Byte	CAL_ADC_12T30
	High Byte	
	Low Byte	CAL_ADC_12T85
	High Byte	
	Low Byte	CAL_ADC_20T30
	High Byte	
	Low Byte	CAL_ADC_20T85
	High Byte	
	Low Byte	CAL_ADC_25T30
	High Byte	
	Low Byte	CAL_ADC_25T85
	High Byte	

The offset of the ADC at room temperature is determined and stored as a two's-complement number in the TLV structure. The offset error correction is done by adding the CAL_ADC_OFFSET to the conversion result.

$$ADC(offset\_corrected) = ADC(raw) + CAL\_ADC\_OFFSET \quad (4)$$

The gain of the ADC at room temperature with an external reference voltage of 2.5 V is calculated by [Equation 5](#):

$$CAL\_ADC\_GAIN\_FACTOR = \frac{1}{GAIN} \times 2^{15} \quad (5)$$

The conversion result is gain corrected by multiplying it with the CAL_ADC_GAIN_FACTOR and dividing the result by 2^{15} :

$$ADC(gain\_corrected) = ADC(raw) \times CAL\_ADC\_GAIN\_FACTOR \times \frac{1}{2^{15}} \quad (6)$$

If both gain and offset are corrected, the gain correction is done first:

$$ADC(gain\_corrected) = ADC(raw) \times CAL\_ADC\_GAIN\_FACTOR \times \frac{1}{2^{15}}$$

$$ADC(final) = ADC(gain\_corrected) + CAL\_ADC\_OFFSET \quad (7)$$

1.14.3.3 Temperature Sensor Calibration

The temperature sensor calibration data is part of the ADC tag as shown in [Table 1-7](#).

The temperature sensor is calibrated using the internal voltage references. Each reference voltage (1.2 V, 2.0 V, or 2.5 V) contains a measured value for two temperatures ($30^{\circ}\text{C} \pm 3^{\circ}\text{C}$ and $85^{\circ}\text{C} \pm 3^{\circ}\text{C}$) and are stored in the TLV structure. The characteristic equation of the temperature sensor voltage, in millivolts is:

$$V_{SENSE} = TC_{SENSOR} \times Temp + V_{SENSOR} \quad (8)$$

The temperature coefficient, TC_{SENSOR} , in $\text{mV}/^{\circ}\text{C}$, represents the slope of the equation. V_{SENSOR} , in mV, represents the y-intercept of the equation. Temp, in $^{\circ}\text{C}$, is the temperature of interest.

The temperature (Temp, °C) can be computed as follows for each of the reference voltages used in the ADC measurement:

$$\begin{aligned} \text{Temp } [^{\circ}\text{C}] &= (\text{ADC}(\text{raw}) - \text{CAL\_ADC\_12T30}) \times \left(\frac{85 - 30}{\text{CAL\_ADC\_12T85} - \text{CAL\_ADC\_12T30}} \right) + 30 \\ \text{Temp } [^{\circ}\text{C}] &= (\text{ADC}(\text{raw}) - \text{CAL\_ADC\_20T30}) \times \left(\frac{85 - 30}{\text{CAL\_ADC\_20T85} - \text{CAL\_ADC\_20T30}} \right) + 30 \\ \text{Temp } [^{\circ}\text{C}] &= (\text{ADC}(\text{raw}) - \text{CAL\_ADC\_25T30}) \times \left(\frac{85 - 30}{\text{CAL\_ADC\_25T85} - \text{CAL\_ADC\_25T30}} \right) + 30 \end{aligned} \quad (9)$$

1.14.3.4 Random Number Seed

Table 1-8 shows the tags used for the random number seed.

Table 1-8. Random Number Tags

Random Number	TAG	15h
	Length	10h
	16 bytes	128-bit random number seed

The random number stored as a seed for a deterministic random number generator is programmed during test of the device. It is generated on the test system using a cryptographic random number generator.

1.14.3.5 BSL Configuration

Table 1-9 shows the tags used for the BSL configuration. The BSL configuration stores the communication interface selection and corresponding communication interface settings. The Tag is optional for devices only providing the basic UART BSL interface. The TAG length field is variable and determined by the length of the configuration option field BSL_CIF_CONFIG. The BSL configuration cannot be changed by the user.

Table 1-9. BSL Configuration Tags

BSL Configuration	TAG	1Ch
	Length	Depends on the BSL_COM_IF value (actual: 02h for UART or I2C)
	Low Byte	BSL_COM_IF
	High Byte	BSL_CIF_CONFIG[0]
	Low Byte	BSL_CIF_CONFIG[1] (optional)
	High Byte	BSL_CIF_CONFIG[2] (optional)
	Low Byte	BSL_CIF_CONFIG[3] (optional)
	High Byte	BSL_CIF_CONFIG[4] (optional)
	⋮	⋮
	⋮	⋮
	High Byte	BSL_CIF_CONFIG[n] (optional)

Table 1-10. BSL_COM_IF Values

BSL_COM_IF	Description	Length
00h	UART interface selected	02h
01h	I2C interface selected	02h
02h to FFh	Reserved for future communication interface	reserved

Table 1-10 shows the defined value for the BSL_COM_IF field. Depending on the selected communication interface, the subsequent bytes in the BSL config tag are interpreted to configure the communication interface. The interpretation is shown in Table 1-11. Unused bytes in BSL_CIF_CONFIG are defined as 00h.

Table 1-11. BSL_CIF_CONFIG Values

BSL_CIF_CONFIG_IF[n]	UART [BSL_COM_IF == 00h]	I2C [BSL_COM_IF == 01h]
0	00h	I2C address (valid values: 0 to 7Fh)
1 to FFh	N/A	N/A

Table 1-11 shows the defined configuration options for the given BSL communication interface.

1.15 SFR Registers

The SFRs are listed in [Table 1-12](#). The base address for the SFRs is 00100h. Many of the bits inside the SFRs are described in other chapters throughout this user's guide. These bits are marked with a note and a reference. See the specific chapter of the respective module for details.

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 1-12. SFR Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	SFRIE1	Interrupt Enable	Read/write	Word	0000h	Section 1.15.1
00h	SFRIE1_L (IE1)		Read/write	Byte	00h	
01h	SFRIE1_H (IE2)		Read/write	Byte	00h	
02h	SFRIFG1	Interrupt Flag	Read/write	Word	0082h	Section 1.15.2
02h	SFRIFG1_L (IFG1)		Read/write	Byte	82h	
03h	SFRIFG1_H (IFG2)		Read/write	Byte	00h	
04h	SFRRPCR	Reset Pin Control	Read/write	Word	001Ch	Section 1.15.3
04h	SFRRPCR_L		Read/write	Byte	1Ch	
05h	SFRRPCR_H		Read/write	Byte	00h	

1.15.1 SFRIE1 Register

Interrupt Enable Register

Figure 1-7. SFRIE1 Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
JMBOUTIE	JMBINIE	Reserved	NMIIE	VMAIE	Reserved	OFIE ⁽¹⁾	WDTIE ⁽²⁾
rw-0	rw-0	r-0	rw-0	rw-0	r0	rw-0	rw-0

⁽¹⁾ See the [Clock System](#) chapter for details.

⁽²⁾ See the [WDT_A](#) chapter for details.

Table 1-13. SFRIE1 Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved. Always reads as 0.
7	JMBOUTIE	RW	0h	JTAG mailbox output interrupt enable 0b = Interrupts disabled 1b = Interrupts enabled
6	JMBINIE	RW	0h	JTAG mailbox input interrupt enable 0b = Interrupts disabled 1b = Interrupts enabled
5	Reserved	R	0h	Reserved. Always reads as 0.
4	NMIIE	RW	0h	NMI pin interrupt enable 0b = Interrupts disabled 1b = Interrupts enabled
3	VMAIE	RW	0h	Vacant memory access interrupt enable 0b = Interrupts disabled 1b = Interrupts enabled
2	Reserved	R	0h	Reserved. Always reads as 0.
1	OFIE	RW	0h	Oscillator fault interrupt enable 0b = Interrupts disabled 1b = Interrupts enabled
0	WDTIE	RW	0h	Watchdog timer interrupt enable. This bit enables the WDTIFG interrupt for interval timer mode. It is not necessary to set this bit for watchdog mode. Because other bits in SFRIE1 may be used for other modules, it is recommended to set or clear this bit using BIS.B or BIC.B instructions, rather than MOV.B or CLR.B instruction. 0b = Interrupts disabled 1b = Interrupts enabled

1.15.2 SFRIFG1 Register

Interrupt Flag Register

Figure 1-8. SFRIFG1 Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
JMBOUTIFG	JMBINIFG	Reserved	NMIIFG	VMAIFG	Reserved	OFIFG ⁽¹⁾	WDTIFG ⁽²⁾
rw-(1)	rw-(0)	r0	rw-0	rw-0	r0	rw-(1)	rw-0

⁽¹⁾ See the [Clock System](#) chapter for details.

⁽²⁾ See the [WDT_A](#) chapter for details.

Table 1-14. SFRIFG1 Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved. Always reads as 0.
7	JMBOUTIFG	RW	1h	JTAG mailbox output interrupt flag 0b = No interrupt pending. When in 16-bit mode (JMBMODE = 0), this bit is cleared automatically when JMBO0 has been written with a new message to the JTAG module by the CPU. When in 32-bit mode (JMBMODE = 1), this bit is cleared automatically when both JMBO0 and JMBO1 have been written with new messages to the JTAG module by the CPU. This bit is also cleared when the associated vector in SYSUNIV has been read. 1b = Interrupt pending, JMBO registers are ready for new messages. In 16-bit mode (JMBMODE = 0), JMBO0 has been received by the JTAG module and is ready for a new message from the CPU. In 32-bit mode (JMBMODE = 1), JMBO0 and JMBO1 have been received by the JTAG module and are ready for new messages from the CPU.
6	JMBINIFG	RW	0h	JTAG mailbox input interrupt flag 0b = No interrupt pending. When in 16-bit mode (JMBMODE = 0), this bit is cleared automatically when JMBIO is read by the CPU. When in 32-bit mode (JMBMODE = 1), this bit is cleared automatically when both JMBIO and JMBI1 have been read by the CPU. This bit is also cleared when the associated vector in SYSUNIV has been read 1b = Interrupt pending, a message is waiting in the JMBIN registers. In 16-bit mode (JMBMODE = 0) when JMBIO has been written by the JTAG module. In 32-bit mode (JMBMODE = 1) when JMBIO and JMBI1 have been written by the JTAG module.
5	Reserved	R	0h	Reserved. Always reads as 0.
4	NMIIFG	RW	0h	NMI pin interrupt flag 0b = No interrupt pending 1b = Interrupt pending
3	VMAIFG	RW	0h	Vacant memory access interrupt flag 0b = No interrupt pending 1b = Interrupt pending
2	Reserved	R	0h	Reserved. Always reads as 0.
1	OFIFG	RW	1h	Oscillator fault interrupt flag 0b = No interrupt pending 1b = Interrupt pending
0	WDTIFG	RW	0h	Watchdog timer interrupt flag. In watchdog mode, WDTIFG clears itself upon a watchdog timeout event. The SYSRSTIV can be read to determine if the reset was caused by a watchdog timeout event. In interval mode, WDTIFG is reset automatically by servicing the interrupt, or can be reset by software. Because other bits in SFRIFG1 may be used for other modules, it is recommended to set or clear WDTIFG by using BIS.B or BIC.B instructions, rather than MOV.B or CLR.B instructions. 0b = No interrupt pending 1b = Interrupt pending

1.15.3 SFRRPCR Register

Reset Pin Control Register

Figure 1-9. SFRRPCR Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved			Reserved	SYSRSTRE	SYSRSTUP	SYSNMIIES	SYSNMI
r0	r0	r0	r(w ⁽¹⁾)-1	rw-1	rw-1	rw-0	rw-0

⁽¹⁾ On some devices this bit can be written, but it must always be written as 1.

Table 1-15. SFRRPCR Register Description

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Always reads as 0.
4	Reserved	R(W ⁽¹⁾)	1h	Reserved. Must be written as 1.
3	SYSRSTRE	RW	1h	Reset pin resistor enable 0b = Pullup or pulldown resistor at the RST/NMI pin is disabled. 1b = Pullup or pulldown resistor at the RST/NMI pin is enabled.
2	SYSRSTUP	RW	1h	Reset resistor pin pullup or pulldown 0b = Pulldown is selected. 1b = Pullup is selected.
1	SYSNMIIES	RW	0h	NMI edge select. This bit selects the interrupt edge for the NMI when SYSNMI = 1. Modifying this bit can trigger an NMI. Modify this bit when SYSNMI = 0 to avoid triggering an accidental NMI. 0b = NMI on rising edge 1b = NMI on falling edge
0	SYSNMI	RW	0h	NMI select. This bit selects the function for the RST/NMI pin. 0b = Reset function 1b = NMI function

⁽¹⁾ On some devices this bit can be written, but it must always be written as 1.

1.16 SYS Registers

The SYS configuration registers are listed in [Table 1-16](#) and the base address is 00180h. A detailed description of each register and its bits is also provided. Each register starts at a word boundary. Either word or byte data can be written to the SYS configuration registers.

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 1-16. SYS Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	SYSCTL	System Control	Read/write	Word	0000h	Section 1.16.1
00h	SYSCTL_L		Read/write	Byte	00h	
01h	SYSCTL_H		Read/write	Byte	00h	
06h	SYSJMBBC	JTAG Mailbox Control	Read/write	Word	000Ch	Section 1.16.2
06h	SYSJMBBC_L		Read/write	Byte	0Ch	
07h	SYSJMBBC_H		Read/write	Byte	00h	
08h	SYSJMBI0	JTAG Mailbox Input 0	Read/write	Word	0000h	Section 1.16.3
08h	SYSJMBI0_L		Read/write	Byte	00h	
09h	SYSJMBI0_H		Read/write	Byte	00h	
0Ah	SYSJMBI1	JTAG Mailbox Input 1	Read/write	Word	0000h	Section 1.16.4
0Ah	SYSJMBI1_L		Read/write	Byte	00h	
0Bh	SYSJMBI1_H		Read/write	Byte	00h	
0Ch	SYSJMBO0	JTAG Mailbox Output 0	Read/write	Word	0000h	
0Ch	SYSJMBO0_L		Read/write	Byte	00h	
0Dh	SYSJMBO0_H		Read/write	Byte	00h	
0Eh	SYSJMBO1	JTAG Mailbox Output 1	Read/write	Word	0000h	Section 1.16.6
0Eh	SYSJMBO1_L		Read/write	Byte	00h	
0Fh	SYSJMBO1_H		Read/write	Byte	00h	
1Ah	SYSUNIV	User NMI Vector Generator	Read	Word	0000h	Section 1.16.7
1Ch	SYSSNIV	System NMI Vector Generator	Read	Word	0000h	Section 1.16.8
1Eh	SYSRSTIV	Reset Vector Generator	Read	Word	0002h	Section 1.16.9

1.16.1 SYSTCTL Register

SYS Control Register

Figure 1-10. SYSTCTL Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved	SYSJTAGPIN		SYSBSLIND	Reserved	SYSMMPE	Reserved	SYSRIVECT
r0	r0	rw-[0]		r0	rw-[0]		rw-[0]

Table 1-17. SYSTCTL Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved. Always reads as 0.
7-6	Reserved	R	0h	Reserved. Always reads as 0.
5	SYSJTAGPIN	RW	0h	Dedicated JTAG pins enable. Setting this bit disables the shared functionality of the JTAG pins and permanently enables the JTAG function. This bit can only be set once. Once it is set it remains set until a BOR occurs. 0b = Shared JTAG pins (JTAG mode selectable using SBW sequence) 1b = Dedicated JTAG pins (explicit 4-wire JTAG mode selection)
4	SYSBSLIND	R	0h	BSL entry indication. This bit indicates a BSL entry sequence detected on the Spy-Bi-Wire pins. 0b = No BSL entry sequence detected 1b = BSL entry sequence detected
3	Reserved	R	0h	Reserved. Always reads as 0.
2	SYSMMPE	RW	0h	PMM access protect. This controls the accessibility of the PMM control registers. Once set to 1, it only can be cleared by a BOR. 0b = Access from anywhere in memory 1b = Access only from the BSL segments
1	Reserved	R	0h	Reserved. Always reads as 0.
0	SYSRIVECT	RW	0h	RAM-based interrupt vectors 0b = Interrupt vectors generated with end address TOP of lower 64K FRAM FFFFh 1b = Interrupt vectors generated with end address TOP of RAM, when RAM available. Note: On devices that contain RAM, it is possible to use the RAM as an alternate location for the interrupt vector locations. Setting the SYSRIVECT bit to '1' in SYSTCTL causes the interrupt vectors to be remapped to the top of RAM. The total RAM size varies depending on the device configurations and could include one or multiple RAM sections. The alternate location is always the highest address of the entire RAM space available in the device. Note that the SYSRIVECT bit is automatically cleared on a BOR, so the default reset vector location (0FFFFh) will be used after a BOR before setting the SYSRIVECT bit to '1'. On devices with LEA, the highest RAM address may be part of the LEA shared RAM. Care must be taken to avoid address conflicts if LEA is used in this case.

1.16.2 SYSJMBC Register

JTAG Mailbox Control Register

Figure 1-11. SYSJMBC Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
JMBCLR1OFF	JMBCLR0OFF	Reserved	JMBMODE	JMBOUT1FG	JMBOUT0FG	JMBIN1FG	JMBIN0FG
rw-(0)	rw-(0)	r0	rw-0	r-(1)	r-(1)	rw-(0)	rw-(0)

Table 1-18. SYSJMBC Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved. Always reads as 0.
7	JMBCLR1OFF	RW	0h	Incoming JTAG Mailbox 1 flag auto-clear disable 0b = JMBIN1FG cleared on read of JMB1IN register 1b = JMBIN1FG cleared by software
6	JMBCLR0OFF	RW	0h	Incoming JTAG Mailbox 0 flag auto-clear disable 0b = JMBIN0FG cleared on read of JMB0IN register 1b = JMBIN0FG cleared by software
5	Reserved	R	0h	Reserved. Always reads as 0.
4	JMBMODE	RW	0h	This bit defines the operation mode of JMB for JMBI0, JMBI1, JMBO0, and JMBO1. Before switching this bit, pad and flush out any partial content to avoid data drops. 0b = 16-bit transfers using JMBO0 and JMBI0 only 1b = 32-bit transfers using JMBO0 with JMBO1 and JMBI0 with JMBI1
3	JMBOUT1FG	R	1h	Outgoing JTAG Mailbox 1 flag. This bit is cleared automatically when a message is written to the upper byte of JMBO1 or as word access (by the CPU, DMA,...) and is set after the message was read through JTAG. 0b = JMBO1 is not ready to receive new data. 1b = JMBO1 is ready to receive new data.
2	JMBOUT0FG	R	1h	Outgoing JTAG Mailbox 0 flag. This bit is cleared automatically when a message is written to the upper byte of JMBO0 or as word access (by the CPU, DMA,...) and is set after the message was read through JTAG. 0b = JMBO0 is not ready to receive new data. 1b = JMBO0 is ready to receive new data.
1	JMBIN1FG	RW	0h	Incoming JTAG Mailbox 1 flag. This bit is set when a new message (provided through JTAG) is available in JMBI1. This flag is cleared automatically on read of JMBI1 when JMBCLR1OFF = 0 (auto clear mode). On JMBCLR1OFF = 1, JMBIN1FG needs to be cleared by software. 0b = JMBI1 has no new data. 1b = JMBI1 has new data available.
0	JMBIN0FG	RW	0h	Incoming JTAG Mailbox 0 flag. This bit is set when a new message (provided through JTAG) is available in JMBI0. This flag is cleared automatically on read of JMBI0 when JMBCLR0OFF = 0 (auto clear mode). On JMBCLR0OFF = 1, JMBIN0FG needs to be cleared by software. 0b = JMBI0 has no new data. 1b = JMBI0 has new data available.

1.16.3 SYSJMBI0 Register

JTAG Mailbox Input 0 Register

Figure 1-12. SYSJMBI0 Register

15	14	13	12	11	10	9	8
MSGHI							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
MSGLO							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 1-19. SYSJMBI0 Register Description

Bit	Field	Type	Reset	Description
15-8	MSGHI	RW	0h	JTAG mailbox incoming message high byte
7-0	MSGLO	RW	0h	JTAG mailbox incoming message low byte

1.16.4 SYSJMBI1 Register

JTAG Mailbox Input 1 Register

Figure 1-13. SYSJMBI1 Register

15	14	13	12	11	10	9	8
MSGHI							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
MSGLO							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 1-20. SYSJMBI1 Register Description

Bit	Field	Type	Reset	Description
15-8	MSGHI	RW	0h	JTAG mailbox incoming message high byte
7-0	MSGLO	RW	0h	JTAG mailbox incoming message low byte

1.16.5 SYSJMBO0 Register

JTAG Mailbox Output 0 Register

Figure 1-14. SYSJMBO0 Register

15	14	13	12	11	10	9	8
MSGHI							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
MSGLO							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 1-21. SYSJMBO0 Register Description

Bit	Field	Type	Reset	Description
15-8	MSGHI	RW	0h	JTAG mailbox outgoing message high byte
7-0	MSGLO	RW	0h	JTAG mailbox outgoing message low byte

1.16.6 SYSJMBO1 Register

JTAG Mailbox Output 1 Register

Figure 1-15. SYSJMBO1 Register

15	14	13	12	11	10	9	8
MSGHI							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
MSGLO							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 1-22. SYSJMBO1 Register Description

Bit	Field	Type	Reset	Description
15-8	MSGHI	RW	0h	JTAG mailbox outgoing message high byte
7-0	MSGLO	RW	0h	JTAG mailbox outgoing message low byte

1.16.7 SYSUNIV Register

User NMI Vector Register

Figure 1-16. SYSUNIV Register

15	14	13	12	11	10	9	8
SYSUNIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
SYSUNIV							
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Table 1-23. SYSUNIV Register Description

Bit	Field	Type	Reset	Description
15-0	SYSUNIV	R	0h	User NMI vector. Generates a value that can be used as address offset for fast interrupt service routine handling. Writing to this register clears all pending user NMI flags. See the device-specific data sheet for a list of values.

1.16.8 SYSSNIV Register

System NMI Vector Register

Figure 1-17. SYSSNIV Register

15	14	13	12	11	10	9	8
SYSSNIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
SYSSNIV							
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Table 1-24. SYSSNIV Register Description

Bit	Field	Type	Reset	Description
15-0	SYSSNIV	R	0h	System NMI vector. Generates a value that can be used as address offset for fast interrupt service routine handling. Writing to this register clears all pending system NMI flags. See the device-specific data sheet for a list of values.

1.16.9 SYSRSTIV Register

Reset Interrupt Vector Register

Figure 1-18. SYSRSTIV Register

15	14	13	12	11	10	9	8
SYSRSTIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
SYSRSTIV							
r0	r0	r ⁽¹⁾	r ⁽¹⁾	r ⁽¹⁾	r ⁽¹⁾	r ⁽¹⁾	r0

⁽¹⁾ Reset value depends on reset source.

Table 1-25. SYSRSTIV Register Description

Bit	Field	Type	Reset	Description
15-0	SYSRSTIV	R	02h-03Eh ⁽¹⁾	Reset interrupt vector. Generates a value that can be used as address offset for fast interrupt service routine handling to identify the last cause of a reset (BOR, POR, PUC) . Writing to this register clears all pending reset source flags. See the device-specific data sheet for a list of values.

⁽¹⁾ Reset value depends on reset source.

Power Management Module (PMM) and Supply Voltage Supervisor (SVS)

This chapter describes the operation of the Power Management Module (PMM) and Supply Voltage Supervisor (SVS). The PMM is family specific.

Topic	Page
2.1 Power Management Module (PMM) Introduction	84
2.2 PMM Operation	85
2.3 PMM Registers.....	88

2.1 Power Management Module (PMM) Introduction

PMM features include:

- Wide supply voltage (DV_{CC}) range: 1.8 V to 3.6 V
- Generation of voltage for the device core (V_{CORE})
- Supply voltage supervisor (SVS) for DV_{CC}
- Brownout reset (BOR)
- Software accessible power-fail indicators
- I/O protection during power-fail condition

The PMM manages all functions related to the power supply and its supervision for the device. Its primary functions are first to generate a supply voltage for the core logic, and second, provide several mechanisms for the supervision of the voltage applied to the device (DV_{CC}).

The PMM uses an integrated low-dropout voltage regulator (LDO) to produce a secondary core voltage (V_{CORE}) from the primary one applied to the device (DV_{CC}). In general, V_{CORE} supplies the CPU, memories, and the digital modules, while DV_{CC} supplies the I/Os and analog modules. The V_{CORE} output is maintained using a dedicated voltage reference. The input or primary side of the regulator is referred to in this chapter as its high side. The output or secondary side is referred to in this chapter as its low side.

Figure 2-1 shows the block diagram of the PMM.

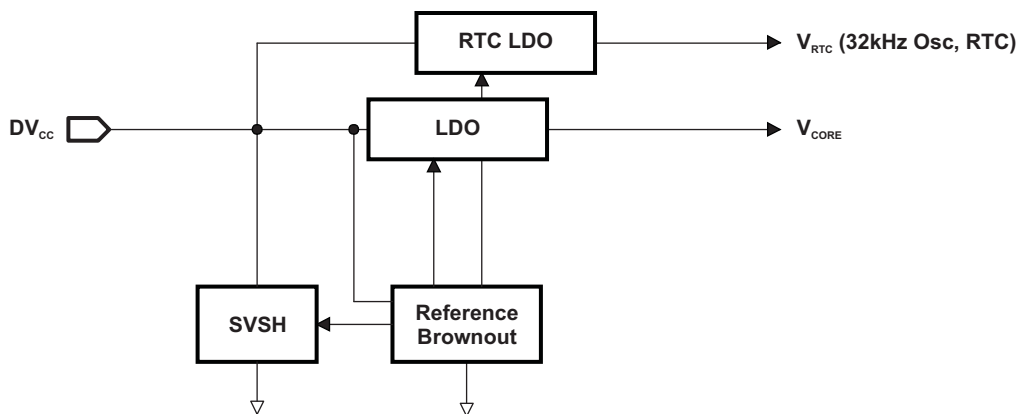


Figure 2-1. PMM Block Diagram

2.2 PMM Operation

2.2.1 V_{CORE} and the Regulator

DV_{CC} can be powered from a wide input voltage range, but the core logic of the device must be kept at a voltage lower than what this range allows. For this reason, a regulator (LDO) has been integrated into the PMM. The regulator derives the necessary core voltage (V_{CORE}) from DV_{CC} .

The regulator supports different load settings to optimize power. The hardware controls the load settings automatically, according to the following criteria:

- Selected and active power modes
- Selected and active clocks
- Clock frequencies according to Clock System (CS) settings
- JTAG is active

In addition to the main LDO, an ultra-low-power regulator (RTC LDO) provides a regulated voltage to the real-time clock module (including the 32-kHz crystal oscillator) and other ultra-low-power modules that remain active during LPM3.5 when the main LDO is off.

2.2.2 Supply Voltage Supervisor

The high-side supervisor (SVSH) oversees DV_{CC} . It is activate in all power modes by default. To disable the SVSH in LPM3, LPM4, LPM3.5, and LPM4.5, set SVSHE = 0.

2.2.2.1 SVS Thresholds

As [Figure 2-2](#) shows, there is hysteresis built into the supervision thresholds, such that the thresholds in force depend on whether the voltage rail is going up or down.

The behavior of the SVS according to these thresholds is best portrayed graphically. [Figure 2-2](#) shows how the supervisors respond to various supply failure conditions.

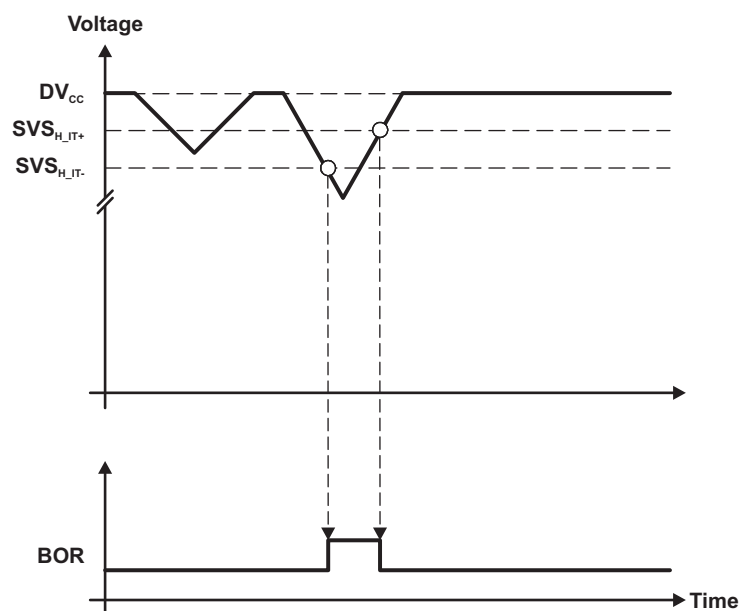


Figure 2-2. Voltage Failure and Resulting PMM Actions

2.2.3 Supply Voltage Supervisor - Power-Up

When the device is powering up, the SVSH function is enabled by default. Initially, DV_{CC} is low, and therefore the PMM holds the device in BOR reset. When the SVSH level is met, after a short delay the BOR reset is released. Figure 2-3 shows this process.

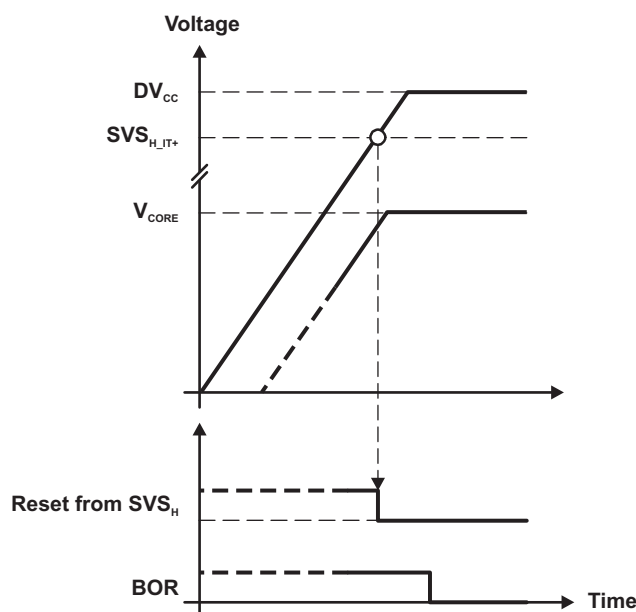


Figure 2-3. PMM Action at Device Power-Up

2.2.4 LPM3.5 and LPM4.5

LPM3.5 and LPM4.5 are additional low-power modes in which the core voltage regulator of the PMM is completely disabled, providing additional power savings. Because there is no power supplied to V_{CORE} during LPMx.5, the CPU and all digital modules including RAM are unpowered. This essentially disables the entire device and thus the contents of the registers and RAM are lost. Any essential values should be stored to FRAM before entering LPMx.5.

To enable LPMx.5 the PMMREGOFF bit in the PMMCTL0 register must be set.

The LOCKLPM5 bit in the PM5CTL0 register locks the I/O configuration and other LPMx.5 relevant configurations after a wakeup from LPMx.5 until all the registers are configured again.

LPM3.5 and LPM4.5 can be configured with active SVS ($SVSHE = 1$) or with SVS disabled ($SVSHE = 0$). Disabling the SVS results in lower power consumption, whereas enabling it provides the ability to detect supply drops and getting a "wake-up" due to the supply drop below the SVS threshold. Note, the "wake-up" due to a supply failure would not be flagged as a LPMx.5 wake-up but as a SVS reset event. In LPM4.5 enabling the SVS results additionally in an about 4 times faster start-up time than with disabled SVS.

Refer to Section 1.4.3 for complete descriptions and uses of LPMx.5.

NOTE: In watchdog mode, the WDT_A prevents LPMx.5. Refer to Section 24.2.5.

2.2.5 Brownout Reset (BOR)

The primary function of the brownout reset (BOR) circuit occurs when the device is powering up. It is functional very early in the power-up ramp, generating a BOR that initializes the system. It also functions when no SVS is enabled and a brownout condition occurs. It sustains this reset until the input power is sufficient for the logic, for proper reset of the system.

In an application, it may be desired to cause a BOR through software. Setting PMMSWBOR causes a software-driven BOR. PMMBORIFG is set accordingly. Note that a BOR also initiates a POR and PUC. PMMBORIFG can be cleared by software or by reading SYSRSTIV. Similarly, it is possible to cause a POR through software by setting PMMSWPOR. PMMPORIFG is set accordingly. A POR also initiates a PUC. PMMPORIFG can be cleared by software or by reading SYSRSTIV. Both PMMSWBOR and PMMSWPOR are self clearing. See the SYS module for complete descriptions of BOR, POR, and PUC resets.

2.2.6 $\overline{\text{RST}}/\text{NMI}$

The external $\overline{\text{RST}}/\text{NMI}$ terminal is pulled low on a BOR reset condition. The $\overline{\text{RST}}/\text{NMI}$ can be used as reset source for the rest of the application.

2.2.7 PMM Interrupts

Interrupt flags generated by the PMM are routed to the system NMI interrupt vector generator register, SYSSNIV. When the PMM causes a reset, a value is generated in the system reset interrupt vector generator register, SYSRSTIV, corresponding to the source of the reset. These registers are defined within the SYS module. More information on the relationship between the PMM and SYS modules is available in the SYS chapter.

2.2.8 Port I/O Control

The PMM provides a means of ensuring that I/O pins cannot behave in uncontrolled fashion during an undervoltage event. During these times, outputs are disabled, both normal drive and the weak pullup or pulldown function. If the CPU is functioning normally, and then an undervoltage event occurs, any pin configured as an input has its PxIN register value locked in at the point the event occurs, until voltage is restored. During the undervoltage event, external voltage changes on the pin are not registered internally. This helps prevent erratic behavior from occurring.

2.3 PMM Registers

The PMM registers are listed in [Table 2-1](#). The base address of the PMM module can be found in the device-specific data sheet. The address offset of each PMM register is given in [Table 2-1](#).

The password defined in the PMMCTL0 register controls access to all PMM registers except PM5CTL0. PM5CTL0 can be accessed without a password. After the correct password is written, the write access is enabled (this includes byte access to the PMMCTL0 lower byte). The write access is disabled by writing a wrong password in byte mode to the PMMCTL0 upper byte. Word accesses to PMMCTL0 with a wrong password triggers a PUC. A write access to a register other than PMMCTL0 while write access is not enabled causes a PUC.

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 2-1. PMM Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	PMMCTL0	PMM control register 0	Read/write	Word	9640h	Section 2.3.1
00h	PMMCTL0_L		Read/write	Byte	40h	
01h	PMMCTL0_H		Read/write	Byte	96h	
02h	PMMCTL1	PMM control register 1	Read/write ⁽¹⁾	Word	9600h	Section 2.3.2
02h	PMMCTL1_L		Read ⁽¹⁾	Byte	00h	
03h	PMMCTL1_H		Read ⁽¹⁾	Byte	96h	
0Ah	PMMIFG	PMM interrupt flag register	Read/write	Word	0000h	Section 2.3.3
0Ah	PMMIFG_L		Read/write	Byte	00h	
0Bh	PMMIFG_H		Read/write	Byte	00h	
10h	PM5CTL0	Power mode 5 control register 0	Read/write	Word	0001h	Section 2.3.4
10h	PM5CTL0_L		Read/write	Byte	01h	
11h	PM5CTL0_H		Read/write	Byte	00h	

⁽¹⁾ PMMCTL1 can be written as word only.

2.3.1 PMMCTL0 Register (offset = 00h) [reset = 9640h]

Power Management Module Control Register 0

Figure 2-4. PMMCTL0 Register

15	14	13	12	11	10	9	8
PMPW							
rw-1	rw-0	rw-0	rw-1	rw-0	rw-1	rw-1	rw-0
7	6	5	4	3	2	1	0
Reserved	SVSHE	Reserved	PMMREGOFF	PMMSWPOR	PMMSWBOR	Reserved	
rw-[0]	rw-[1]	r0	rw-[0]	rw-(0)	rw-[0]	r0	r0

Table 2-2. PMMCTL0 Register Description

Bit	Field	Type	Reset	Description
15-8	PMPW	RW	96h	PMM password. Always reads as 096h. Must be written with 0A5h to unlock the PMM registers.
7	Reserved	RW	0h	Reserved. Must be written with 0.
6	SVSHE	RW	1h	High-side SVS enable. 0b = High-side SVS (SVSH) is disabled in LPM2, LPM3, LPM4, LPM3.5, and LPM4.5. SVSH is always enabled in active mode, LPM0, and LPM1. 1b = SVSH is always enabled.
5	Reserved	R	0h	Reserved. Always reads as 0.
4	PMMREGOFF	RW	0h	Regulator off 0b = Regulator remains on when going into LPM3 or LPM4 1b = Regulator is turned off when going to LPM3 or LPM4. System enters LPM3.5 or LPM4.5, respectively.
3	PMMSWPOR	RW	0h	Software POR. Setting this bit to 1 triggers a POR. This bit is self clearing. 0b = Normal operation 1b = Set to 1 to trigger a POR
2	PMMSWBOR	RW	0h	Software brownout reset. Setting this bit to 1 triggers a BOR. This bit is self clearing. 0b = Normal operation 1b = Set to 1 to trigger a BOR
1-0	Reserved	R	0h	Reserved. Always reads as 0.

2.3.2 PMMCTL1 Register (offset = 02h) [reset = 9600h]

Power Management Module Control Register 1

Figure 2-5. PMMCTL1 Register

15	14	13	12	11	10	9	8
Reserved							
rw-1	rw-0	rw-0	rw-1	rw-0	rw-1	rw-1	rw-0
7	6	5	4	3	2	1	0
Reserved							
rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	r0

Table 2-3. PMMCTL1 Register Description

Bit	Field	Type	Reset	Description
15-0	Reserved	R	9600h	Reserved. Always reads as 9600h.

2.3.3 PMMIFG Register (offset = 0Ah) [reset = 0000h]

Power Management Module Interrupt Flag Register

Figure 2-6. PMMIFG Register

15	14	13	12	11	10	9	8
PMMLPM5IFG	Reserved	SVSHIFG	Reserved		PMPORIFG	PMMRSTIFG	PMMBORIFG
rw-{0}	r0	rw-{0}	r0	r0	rw-{0}	rw-{0}	rw-{0}
7	6	5	4	3	2	1	0
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0

Table 2-4. PMMIFG Register Description

Bit	Field	Type	Reset	Description
15	PMMLPM5IFG	RW	0h	LPMx.5 flag. This bit has a specific reset conditions. This bit is only set if the system was in LPMx.5 before. The bit is cleared by software or by reading the reset vector word SYSRSTIV. A power failure on the DVCC domain triggered by the high-side SVS (if enabled) or the brownout clears the bit. 0b = Reset not due to wake-up from LPMx.5 1b = Reset due to wake-up from LPMx.5
14	Reserved	R	0h	Reserved. Always reads as 0.
13	SVSHIFG	RW	0h	High-side SVS interrupt flag. This bit has a specific reset conditions. The SVSHIFG interrupt flag is only set if the SVSH is the reset source; that is, if DVCC dropped below the high-side SVS levels but remained above the brownout levels. The bit is cleared by software or by reading the reset vector word SYSRSTIV. 0b = Reset not due to SVSH 1b = Reset due to SVSH
12-11	Reserved	R	0h	Reserved. Always reads as 0.
10	PMPORIFG	RW	0h	PMM software POR interrupt flag. This bit has a specific reset conditions. This interrupt flag is only set if a software POR (PMMSWPOR) is triggered. The bit is cleared by software or by reading the reset vector word SYSRSTIV. 0b = Reset not due to PMMSWPOR 1b = Reset due to PMMSWPOR
9	PMMRSTIFG	RW	0h	PMM reset pin interrupt flag. This bit has a specific reset conditions. This interrupt flag is only set if the RST/NMI pin is the reset source. The bit is cleared by software or by reading the reset vector word SYSRSTIV. 0b = Reset not due to reset pin 1b = Reset due to reset pin
8	PMMBORIFG	RW	0h	PMM software brownout reset interrupt flag. This bit has a specific reset conditions. This interrupt flag is only set if a software BOR (PMMSWBOR) is triggered. The bit is cleared by software or by reading the reset vector word SYSRSTIV. 0b = Reset not due to PMMSWBOR 1b = Reset due to PMMSWBOR
7-0	Reserved	R	0h	Reserved. Always reads as 0.

2.3.4 PM5CTL0 Register (offset = 10h) [reset = 0001h]

Power Mode 5 Control Register 0

Figure 2-7. PM5CTL0 Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved							LOCKLPM5
r0	r0	r0	r0	r0	r0	r0	rw-{1}

Table 2-5. PM5CTL0 Register Description

Bit	Field	Type	Reset	Description
15-1	Reserved	R	0h	Reserved. Always reads as 0.
0	LOCKLPM5	RW	1h	Locks I/O pin and other LPMx.5 relevant (for example, RTC) configurations upon exit from LPMx.5. This bit is set by hardware and must be cleared by software. It cannot be set by software. After a power cycle I/O pins are locked in high-impedance state with input Schmitt triggers disabled until LOCKLPM5 is cleared by the user software. After a wake-up from LPMx.5 I/O pins and other LPMx.5 relevant (for example, RTC) configurations are locked in their states configured before LPMx.5 entry until LOCKLPM5 is cleared by the user software. 0b = I/O pin and LPMx.5 configurations unlocked. 1b = I/O pin and LPMx.5 configuration remains locked.

Clock System (CS) Module

This chapter describes the operation of the clock system, which is implemented in all devices.

Topic	Page
3.1 Clock System Introduction	94
3.2 Clock System Operation	96
3.3 MemoryMap Registers.....	103

3.1 Clock System Introduction

The clock system module supports low system cost and low power consumption. Using three system clock signals, the user can select the best balance of performance and power consumption. The clock module can be configured to operate without any external components, with one or two external crystals, or with resonators, under full software control.

The clock system module includes the following clock sources:

- **LFXTCLK:** Low-frequency oscillator that can be used either with low-frequency 32768-Hz watch crystals, standard crystals, resonators, or external clock sources in the 50 kHz or below range. When in bypass mode, LFXTCLK can be driven with an external square wave signal.
- **VLOCLK:** Internal very-low-power low-frequency oscillator with 10-kHz typical frequency
- **DCOCLK:** Internal digitally controlled oscillator (DCO) with selectable frequencies
- **MODCLK:** Internal low-power oscillator with 5-MHz typical frequency. LFMODCLK is MODCLK divided by 128.
- **HFXTCLK:** High-frequency oscillator that can be used with standard crystals or resonators in the 4-MHz to 24-MHz range. When in bypass mode, HFXTCLK can be driven with an external square wave signal.

Four system clock signals are available from the clock module:

- **ACLK:** Auxiliary clock. The ACLK is software selectable as LFXTCLK, VLOCLK, or LFMODCLK. ACLK can be divided by 1, 2, 4, 8, 16, or 32. ACLK is software selectable by individual peripheral modules.
- **MCLK:** Master clock. MCLK is software selectable as LFXTCLK, VLOCLK, LFMODCLK, DCOCLK, MODCLK, or HFXTCLK. MCLK can be divided by 1, 2, 4, 8, 16, or 32. MCLK is used by the CPU and system.
- **SMCLK:** Subsystem master clock. SMCLK is software selectable as LFXTCLK, VLOCLK, LFMODCLK, DCOCLK, MODCLK, or HFXTCLK. SMCLK is software selectable by individual peripheral modules.
- **MODCLK:** Module clock. MODCLK may also be used by various peripheral modules and is sourced by MODOSC.
- **VLOCLK:** VLO clock. VLOCLK may also be used directly by various peripheral modules and is sourced by VLO.

NOTE: Not all devices contain both LFXT and HFXT clock sources. See the device-specific data sheet for availability.

Figure 3-1 shows the block diagram of the clock system module.



Figure 3-1. Clock System Block Diagram

3.2 Clock System Operation

After PUC, the CS module default configuration is:

- LFXT is selected as the oscillator source for LFXTCCLK. LFXTCCLK is selected for ACLK (SELAX = 0) and ACLK is undivided (DIVAX = 0).
- DCOCLK is selected for MCLK and SMCLK (SELMx = SELSx = 3) and each are divided by 8 (DIVMx = DIVSx = 3).
- LFXIN and LFXOUT pins are set to general-purpose I/Os and LFXT remains disabled until the I/O ports are configured for LFXT operation.
- HFXIN and HFXOUT pins are set to general-purpose I/Os and HFXT is disabled.

As previously stated, LFXT is selected by default, but LFXT is disabled. The crystal pins (LFXIN, LFXOUT) are shared with general-purpose I/Os. To enable LFXT, the PSEL bits associated with the crystal pins must be set. When a 32768-Hz crystal is used for LFXTCCLK, the fault control logic immediately causes ACLK to be sourced by LFMODCLK, and MCLK and SMCLK to be sourced by MODCLK, because LFXT is not stable immediately (see [Section 3.2.8](#)).

Status register control bits (SCG0, SCG1, OSCOFF, and CPUOFF) configure the MCU operating modes and enable or disable portions of the clock system module (see the *System Resets, Interrupts, and Operating Modes* chapter). Registers CSCTL0 to CSCTL6 configure the CS module.

The CS module can be configured or reconfigured by software at any time during program execution. The CS control registers are password protected to prevent inadvertent access.

3.2.1 CS Module Features for Low-Power Applications

Conflicting requirements typically exist in battery-powered applications:

- Low clock frequency for energy conservation and time keeping
- High clock frequency for fast response times and fast burst processing capabilities
- Clock stability over operating temperature and supply voltage
- Low-cost applications with less-constrained clock accuracy requirements

The CS module addresses these conflicting requirements by allowing the user to select from the three available clock signals: ACLK, MCLK, and SMCLK. A flexible clock distribution and divider system is provided to fine tune the individual clock requirements.

3.2.2 LFXT Oscillator

The LFXT oscillator supports ultra-low-current consumption using a 32768-Hz watch crystal. A watch crystal connects to LFXIN and LFXOUT and requires external capacitors on both terminals. These capacitors should be sized according to the crystal or resonator specifications. Different crystal or resonator ranges are supported by LFXT by choosing the proper LFXTDRIIVE settings.

The LFXT pins are shared with general-purpose I/O ports. At power up, the LFXT clock defaults to "on" and is the source for ACLK. However, at power-up the LFXT pins default to general-purpose I/O mode, therefore, the LFXT clock remains disabled until the pins associated with LFXT are configured for LFXT operation. The configuration of the shared I/O is determined by the PSEL bit associated with LFXIN and the LFXTBYPASS bit. Setting the PSEL bit causes the LFXIN and LFXOUT ports to be configured for LFXT operation. If LFXTBYPASS is also set, LFXT is configured for bypass mode of operation, and the oscillator that is associated with LFXT is powered down. In bypass mode of operation, LFXIN can accept an external square-wave clock input signal, and LFXOUT is configured as a general-purpose I/O. The PSEL bit associated with LFXOUT is a don't care.

If the PSEL bit associated with LFXIN is cleared, both LFXIN and LFXOUT ports are configured as general-purpose I/Os, and LFXT is disabled.

LFXT is enabled under any of the following conditions:

- LFXT is a source for ACLK (SELAX = 0) and in active mode (AM) through LPM3 (OSCOFF = 0)
- LFXT is a source for MCLK (SELMx = 0) and in active mode (AM) (CPUOFF = 0)
- LFXT is a source for SMCLK (SELSx = 0) and in active mode (AM) through LPM1 (SMCLKOFF = 0)

- LFXTOFF = 0. LFXT enabled in active mode (AM) through LPM4.
- LFXT is selected as the source for RTC, RTC is enabled (RTCHOLD = 0), and LPMx.5 is entered.

NOTE: If LFXT is disabled when entering into a low-power mode, it is not fully enabled and stable upon exit from the low-power mode, because its enable time is much longer than the wake-up time. If the application requires or desires to keep LFXT enabled during a low-power mode, the LFXTOFF bit can be cleared before entering the low-power mode. This causes LFXT to remain enabled.

3.2.3 HFXT Oscillator

The HFXT high-frequency oscillator can be used with standard crystals or resonators in the 4 MHz to 24 MHz range. The HFXTDRIVE bits select the drive capability of HFXT. HFXTDRIVE bits can be used to provide optimal settings for a given crystal characteristic. HFXT sources HFXTCLK. The HFFREQ bits must be set for the appropriate frequency range of operation as show in [Table 3-1](#) in crystal or bypass modes of operation.

Table 3-1. HFFREQ Settings

HFXT Frequency Range	HFFREQ[1:0]
0 to 4 MHz	00
> 4 MHz to 8 MHz	01
> 8 MHz to 16 MHz	10
> 16 MHz to 24 MHz	11

NOTE: The HFXT HFFREQ bit settings are also used to control the Power Management Module and must match the intended frequency of operation for proper functioning of the device as listed in [Table 3-1](#). In addition, these bits should be configured properly before use of HFXT in either crystal or bypass modes of operation.

The HFXT pins are shared with general-purpose I/O ports. At power up, the default operation is HFXT crystal operation. However, HFXT remains disabled until the ports shared with HFXT are configured for HFXT operation. The configuration of the shared I/O is determined by the PSEL bit associated with HFXIN and the HFXTBYPASS bit. Setting the PSEL bit causes the HFXIN and HFXOUT ports to be configured for HFXT operation. If HFXTBYPASS is also set, HFXT is configured for bypass mode of operation, and the oscillator associated with HFXT is powered down. In bypass mode of operation, HFXIN can accept an external square-wave clock input signal, and HFXOUT is configured as a general-purpose I/O. The PSEL bit that is associated with HFXOUT is a don't care.

If the PSEL bit associated with HFXIN is cleared, both HFXIN and HFXOUT ports are configured as general-purpose I/Os, and HFXT is disabled.

HFXT is enabled under any of the following conditions:

- HFXT is a source for MCLK (SELMx = 5) and in active mode (AM) (CPUOFF = 0)
- HFXT is a source for SMCLK (SELSx = 5) and in active mode (AM) through LPM1 (SMCLKOFF = 0)
- LFXTOFF = 0. HFXT enabled in active mode (AM) through LPM4.

NOTE: If HFXT is disabled when entering into a low-power mode, it is not fully enabled and stable upon exit from the low-power mode, because its enable time is much longer than the wake-up time. If the application requires or desires to keep HFXT enabled during a low-power mode, the HFXTOFF bit can be cleared before entering the low-power mode. This causes HFXT to remain enabled.

3.2.4 Internal Very-Low-Power Low-Frequency Oscillator (VLO)

The internal VLO provides a typical frequency of 10 kHz (see the device-specific data sheet for parameters) without requiring a crystal. The VLO provides for a low-cost ultra-low-power clock source for applications that do not require an accurate time base. To conserve power, VLO is powered down when not needed and enabled only when required.

VLO is enabled under any of the following conditions:

- VLO is a source for ACLK (SEL_{AX} = 1) and in active mode (AM) through LPM3 (OSCOFF = 0)
- VLO is a source for MCLK (SEL_{MX} = 1) and in active mode (AM) (CPUOFF = 0)
- VLO is a source for SMCLK (SEL_{SX} = 1) and in active mode (AM) through LPM1 (SMCLKOFF = 0)
- VLOFF = 0. VLO enabled in active mode (AM) through LPM4.
- VLO is selected as the source for RTC, RTC is enabled (RTCHOLD = 0), and LPMx.5 is entered.

3.2.5 Module Oscillator (MODOSC)

The CS module also supports an internal oscillator, MODOSC, that can be used by ACLK, MCLK, or SMCLK, as well as by other modules in the system. It is also used as a fail-safe clock source as described in [Section 3.2.8](#). The MODOSC sources MODCLK and LFMODCLK.

To conserve power, MODOSC is powered down when not needed and is enabled only when required. When the MODOSC source is required, the respective module requests it. MODOSC is enabled based on unconditional and conditional requests. Setting MODOSCREQEN enables conditional requests. Unconditional requests are always enabled. It is not necessary to set MODOSCREQEN for modules that utilize unconditional requests; for example, ADC or fail-safe.

MODOSC is enabled under any of the following conditions:

- LFMODCLK is a source for ACLK (SEL_{AX} = 2) and in active mode (AM) through LPM3 (OSCOFF = 0)
- LFMODCLK or MODCLK is a source for MCLK (SEL_{MX} = 2, 4) and in active mode (AM) (CPUOFF = 0)
- LFMODCLK or MODCLK is a source for SMCLK (SEL_{SX} = 2, 4) and in active mode (AM) through LPM1 (SMCLKOFF = 0)
- During a fault detection (when fault detection enabled) and LFXT or HFXT is active.
- Unconditional requests from modules that require MODCLK; for example, ADC conversion clock when selected as the source.
- LFMODCLK is used as the fail-safe source for the WDT in watchdog mode. Should the selected source for ACLK or SMCLK not be available, the WDT automatically switches to LFMODCLK as its clock source.

The ADC12 may optionally use MODOSC as a clock source for its conversion clock. The user chooses the ADC12OSC as the conversion clock source. During a conversion, the ADC12 module issues an unconditional request for the ADC12OSC clock source. Upon doing so, the MODOSC source is enabled, if not already enabled from other modules' previous requests.

3.2.6 Digitally Controlled Oscillator (DCO)

The DCO is an integrated digitally controlled oscillator. The DCO has three frequency settings determined by the DCOFSEL bits. Each frequency is trimmed at the factory. The DCO can be used as a source for MCLK or SMCLK. See the device-specific data sheet for DCO characteristics.

The DCO frequency can be changed at any time, but care should be taken to ensure no other system clock frequency constraints are exceeded with the new frequency selection. Any change in the DCOFSEL or DCORSEL bits causes the DCOCLK to be held for four clock cycles before releasing the new value into the system. This allows for the DCO to settle properly.

3.2.7 Operation From Low-Power Modes, Requested by Peripheral Modules

A peripheral module requests its clock sources automatically from the CS module if required for its proper operation, regardless of the current power mode of operation, as shown in Figure 3-2.

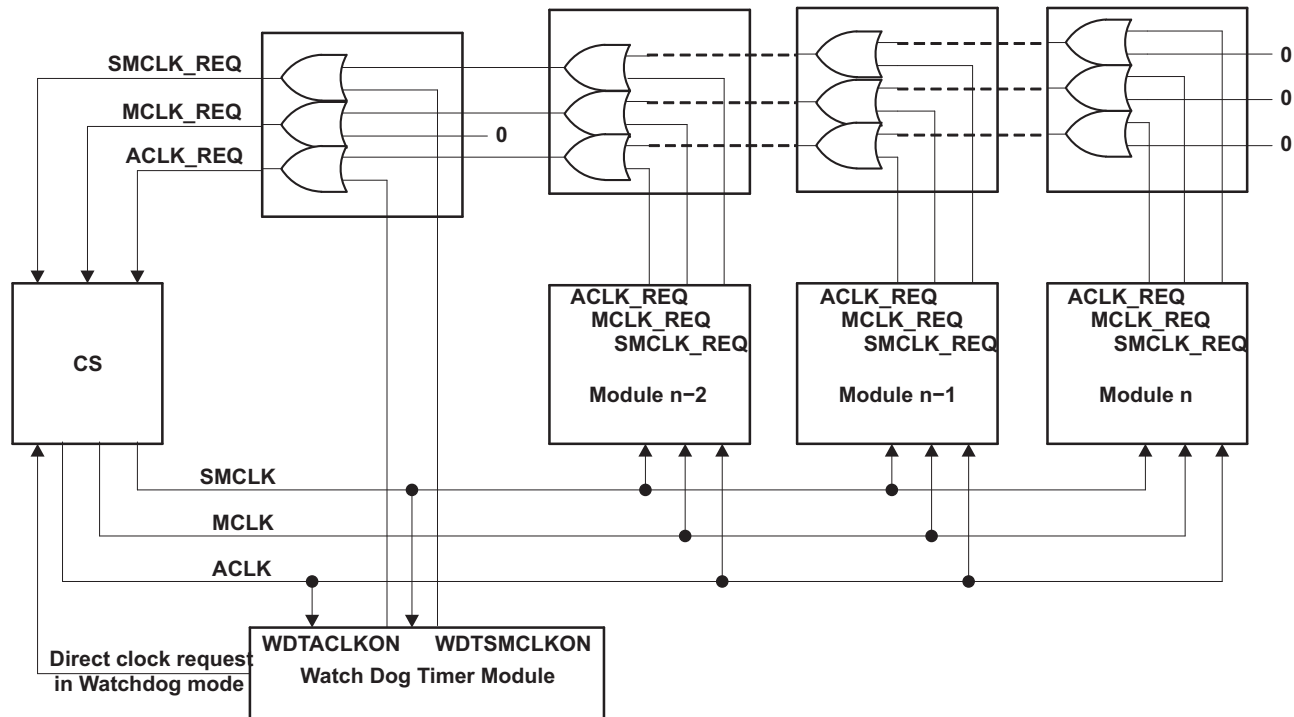


Figure 3-2. Module Request Clock System

A peripheral module asserts one of three possible clock request signals based on its control bits: ACLK_REQ, MCLK_REQ, or SMCLK_REQ. These request signals are based on the configuration and clock selection of the respective module. For example, if a timer selects ACLK as its clock source and the timer is enabled, the timer generates an ACLK_REQ signal to the CS system. The CS, in turn, enables ACLK regardless of the power mode settings.

Any clock request from a peripheral module causes its respective clock off signal to be overridden, but does not change the setting of the clock off control bit. For example, a peripheral module may require ACLK that is currently disabled by the OSCOFF bit (OSCOFF = 1). The module can request ACLK by generating an ACLK_REQ. This causes the OSCOFF bit to have no effect, thereby allowing ACLK to be available to the requesting peripheral module. The OSCOFF bit remains at its current setting (OSCOFF = 1).

If the requested source is not active, the software NMI handler must take care of the required actions. For the previous example, if ACLK was sourced by LFXT and LFXT was not enabled, an oscillator fault condition occurs, and the software must handle the event. The watchdog, due to its security requirement, actively selects the LFMODCLK source if the originally selected clock source is not available.

Due to the clock request feature, care must be taken in the application when entering low-power modes to save power. Although the device enters the selected low-power mode, a clock request causes more current consumption than the specified values in the data sheet. By default, the clock request feature is enabled. The feature can be disabled for each system clock by clearing ACLKREQEN, MCLKREQEN, or SMCLKREQEN for the respective clocks. This does not disable fail-safe clock requests; for example, those of the watchdog timer or the clock system itself.

The function of the ACLKREQEN, MCLKREQEN, and SMCLKREQEN bits are dependent upon which power mode is selected; that is, they do not have an effect across all power modes. For example, ACLKREQEN is used to enable or disable ACLK requests. It is effective only in LPM4, because ACLK is always active in all other modes (AM, LPM0, LPM1, LPM2, LPM3). SMCLKREQEN is used to enable or disable SMCLK requests. When SMCLKOFF = 0 and in AM, LPM0, or LPM1, it is a don't care, because SMCLK is always on in these cases. For SMCLKOFF = 0 and in LPM2, LPM3, and LPM4, SMCLKREQEN can be used to enable or disable SMCLK requests, because SMCLK is normally off in these modes. When SMCLKOFF = 1, SMCLKREQEN can be used to enable or disable SMCLK requests, because SMCLK is normally off in all power modes under this condition. This is summarized in [Table 3-2](#).

Table 3-2. System Clocks, Power Modes, and Clock Requests

Mode	System Clocks							
	MCLK		ACLK		SMCLK			
					SMCLKOFF = 0		SMCLKOFF = 1	
	MCLKREQEN = 0 and Clock Requested	MCLKREQEN = 1 and Clock Requested	ACLKREQEN = 0 and Clock Requested	ACLKREQEN = 1 and Clock Requested	SMCLKREQEN = 0 and Clock Requested	SMCLKREQEN = 1 and Clock Requested	SMCLKREQEN = 0 and Clock Requested	SMCLKREQEN = 1 and Clock Requested
AM	Active	Active	Active	Active	Active	Active	Disabled	Active
LPM0	Disabled	Active	Active	Active	Active	Active	Disabled	Active
LPM1	Disabled	Active	Active	Active	Active	Active	Disabled	Active
LPM2	Disabled	Active	Active	Active	Disabled	Active	Disabled	Active
LPM3	Disabled	Active	Active	Active	Disabled	Active	Disabled	Active
LPM4	Disabled	Active	Disabled	Active	Disabled	Active	Disabled	Active
LPM3.5	Disabled	Disabled	Disabled ⁽¹⁾	Disabled	Disabled	Disabled	Disabled	Disabled
LPM4.5	Disabled	Disabled	Disabled	Disabled	Disabled	Disabled	Disabled	Disabled

⁽¹⁾ LFXCLK is available directly as the clock source to the RTC module.

3.2.8 CS Module Fail-Safe Operation

The CS module incorporates an oscillator-fault fail-safe feature. This feature detects an oscillator fault for LFXT and HFXT as shown in [Figure 3-3](#). The available fault conditions are:

- Low-frequency oscillator fault (LFXTOFFG) for LFXT
- High-frequency oscillator fault (HFXTTOFFG) for HFXT
- External clock signal faults for all bypass modes; that is, LFXTBYPASS = 1 or HFXTBYPASS = 1

The crystal oscillator fault bits LFXTOFFG and HFXTTOFFG are set if the corresponding crystal oscillator is turned on and not operating properly. Once set, the fault bits remain set until reset in software, even if the fault condition no longer exists. If the user clears the fault bits and the fault condition still exists, the fault bits are automatically set; otherwise, they remain cleared.

The OFIFG oscillator-fault interrupt flag is set and latched at POR or when any oscillator fault (LFXTOFFG or HFXTTOFFG) is detected. When OFIFG is set and OFIE is set, the OFIFG requests a user NMI. When the interrupt is granted, the OFIE is not reset automatically as it is in previous MSP430 families. It is no longer required to reset the OFIE. NMI entry and exit circuitry removes this requirement. The OFIFG flag must be cleared by software. The source of the fault can be identified by checking the individual fault bits.

If LFXT is sourcing any system clock (ACLK, MCLK, or SMCLK) and a fault is detected, the system clock is automatically switched to LFMODCLK for its clock source. The LFXT fault logic works in all power modes, including LPM3.5. Similarly, if HFXT is sourcing MCLK or SMCLK, and a fault is detected, the system clock is automatically switched to MODCLK for its clock source. By default, the HFXT fault logic works in all power modes except LPM3.5 or LPM4.5, because high-frequency operation in these modes is not supported. The fail-safe logic does not change the respective SELA, SELM, and SELS bit settings. The fail-safe mechanism behaves the same in normal and bypass modes. Reconfigure the CS settings and follow the instructions in [Section 1.4.3](#) after wakeup from LPM3.5 or LPM4.5, because all CS registers are reset to default values.

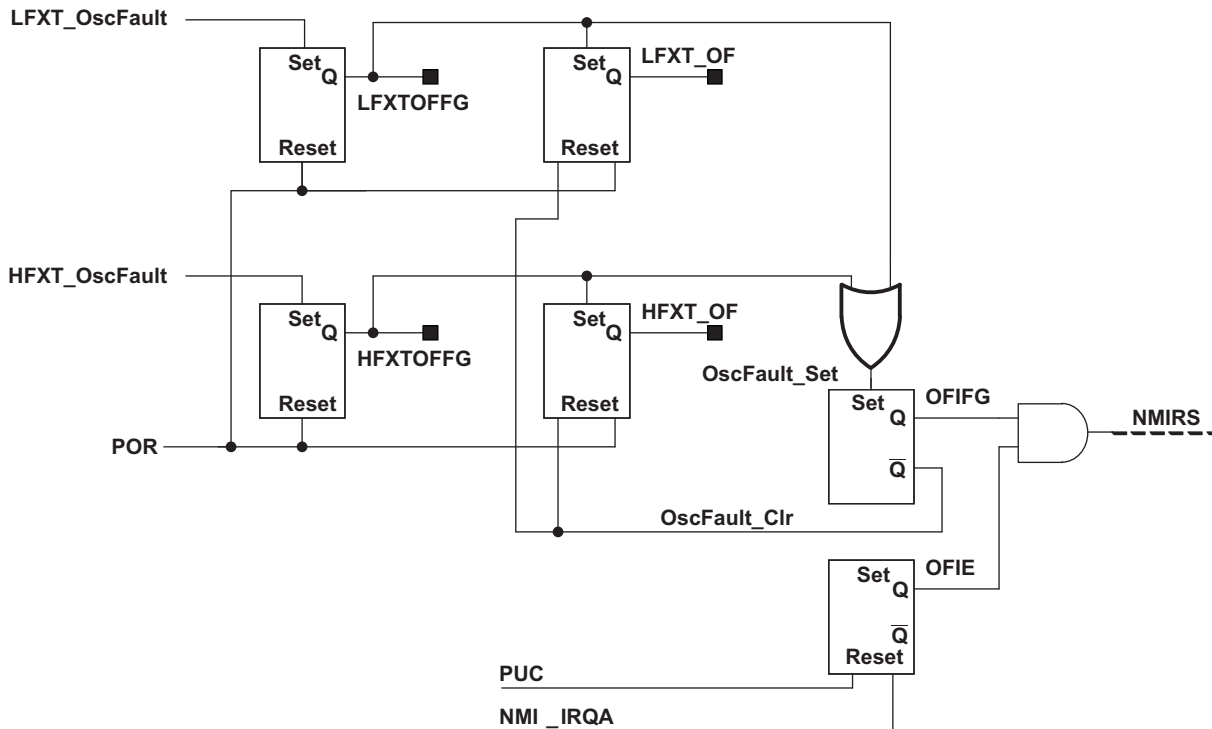


Figure 3-3. Oscillator Fault Logic

NOTE: Fault conditions

LFXT_OscFault: When the fault detection logic is enabled (ENLFXTD = 1), this signal is set after the LFXT oscillator has stopped operation and is cleared after operation resumes. The fault condition causes LFXTOFFG to be set and remain set. If the user clears LFXTOFFG and the fault condition still exists, LFXTOFFG remains set.

HFXT_OscFault: When the fault detection logic is enabled (ENHFXTD = 1), this signal is set after the HFXT oscillator has stopped operation and is cleared after operation resumes. The fault condition causes HFXTTOFFG to be set and remain set. If the user clears HFXTTOFFG and the fault condition still exists, HFXTTOFFG remains set.

NOTE: Fault logic

As long as a fault condition still exists, the OFIFG remains set. The application must take special care when clearing the OFIFG signal. If no fault condition remains when the OFIFG signal is cleared, the clock logic switches back to the original user settings before the fault condition.

NOTE: The LFXT startup includes a counter that ensures that 1024 valid clock cycles have passed before LFXT_OscFault signal is cleared. A valid cycle is any cycle that meets the frequency requirement ($f_{Fault,LF}$) as outlined in the device specific data sheet. Any crystal fault restarts the counter. It is recommended that the counter always be enabled, however the counter can be disabled by clearing ENSTFCNT1.

Similarly, HFXT startup also includes a counter that ensures that 1024 valid clock cycles have passed before HFXT_OscFault signal is cleared. This counter can be disabled by clearing ENSTFCNT2.

The disabling of the counters is valid for bypass and normal modes of operation.

3.2.9 Synchronization of Clock Signals

When switching ACLK, MCLK, or SMCLK from one clock source to the other, the switch is synchronized to avoid critical race conditions (see [Figure 3-4](#)):

- The current clock cycle continues until the next rising edge.
- The clock remains high until the next rising edge of the new clock.
- The new clock source is selected and continues with a full high period.

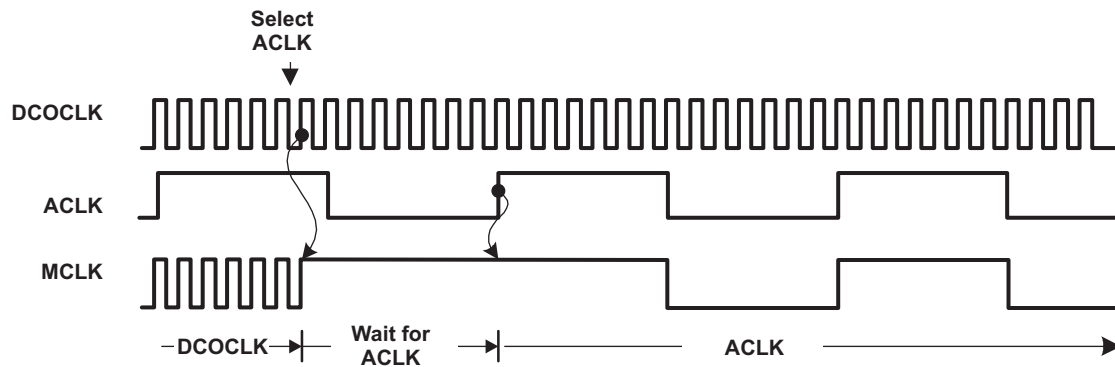


Figure 3-4. Switch MCLK From DCOCLK to LFXTCLK

3.3 MemoryMap Registers

[Table 3-3](#) lists the MemoryMap registers. All register offset addresses not listed in [Table 3-3](#) should be considered as reserved locations and the register contents should not be modified.

Table 3-3. MEMORYMAP Registers

Offset	Acronym	Register Name	Section
0h	CTL0	Clock System Control 0	Section 3.3.1
2h	CTL1	Clock System Control 1	Section 3.3.2
4h	CTL2	Clock System Control 2	Section 3.3.3
6h	CTL3	Clock System Control 3	Section 3.3.4
8h	CTL4	Clock System Control 4	Section 3.3.5
Ah	CTL5	Clock System Control 5	Section 3.3.6
Ch	CTL6	Clock System Control 6	Section 3.3.7

3.3.1 CTL0 Register (Offset = 0h) [reset = 9600h]

CTL0 is shown in [Figure 3-5](#) and described in [Table 3-4](#).

Return to the [Summary Table](#).

Clock System Control 0 Register

Figure 3-5. CTL0 Register

15	14	13	12	11	10	9	8
KEY							
R/W-96h							
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

Table 3-4. CTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	KEY	R/W	96h	CSKEY password. Must always be written with A5h; a PUC is generated if any other value is written. Always reads as 96h. After the correct password is written, all CS registers are available for writing. A5h (W) = 0xA5
7-0	RESERVED	R	0h	Reserved. Always reads as 0.

3.3.2 CTL1 Register (Offset = 2h) [reset = Ch]

CTL1 is shown in [Figure 3-6](#) and described in [Table 3-5](#).

Return to the [Summary Table](#).

Clock System Control 1 Register

Figure 3-6. CTL1 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	DCORSEL	RESERVED		DCOFSEL		RESERVED	
R-0h	R/W-0h	R-0h		R/W-6h		R-0h	

Table 3-5. CTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved. Always reads as 0.
6	DCORSEL	R/W	0h	DCO range select. For high speed devices, this bit can be written by the user. For low speed devices, it is always reset. See description of DCOFSEL bit for details.
5-4	RESERVED	R	0h	Reserved. Always reads as 0.
3-1	DCOFSEL	R/W	6h	DCO frequency select. Selects frequency settings for the DCO. Values shown below are approximate. Please refer to the device specific datasheet. 0h (R/W) = If DCORSEL = 0: 1 MHz; If DCORSEL = 1: 1 MHz 1h (R/W) = If DCORSEL = 0: 2.67 MHz; If DCORSEL = 1: 5.33 MHz 2h (R/W) = If DCORSEL = 0: 3.5 MHz; If DCORSEL = 1: 7 MHz 3h (R/W) = If DCORSEL = 0: 4 MHz; If DCORSEL = 1: 8 MHz 4h (R/W) = If DCORSEL = 0: 5.33 MHz; If DCORSEL = 1: 16 MHz 5h (R/W) = If DCORSEL = 0: 7 MHz; If DCORSEL = 1: 21 MHz 6h (R/W) = If DCORSEL = 0: 8 MHz; If DCORSEL = 1: 24 MHz 7h (R/W) = If DCORSEL = 0: Reserved. Defaults to 8. It is not recommended to use this setting; If DCORSEL = 1: Reserved. Defaults to 24. It is not recommended to use this setting
0	RESERVED	R	0h	Reserved. Always reads as 0.

3.3.3 CTL2 Register (Offset = 4h) [reset = 33h]

CTL2 is shown in [Figure 3-7](#) and described in [Table 3-6](#).

Return to the [Summary Table](#).

Clock System Control 2 Register

Figure 3-7. CTL2 Register

15	14	13	12	11	10	9	8
RESERVED						SELA	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		SELS		RESERVED		SELM	
R-0h		R/W-3h		R-0h		R/W-3h	

Table 3-6. CTL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0h	Reserved. Always reads as 0.
10-8	SELA	R/W	0h	Selects the ACLK source 0h (R/W) = LFXTCLK : LFXTCLK when LFXT available, otherwise VLOCLK. 1h (R/W) = VLOCLK : VLOCLK 2h (R/W) = LFMODCLK : LFMODCLK
7	RESERVED	R	0h	Reserved. Always reads as 0.
6-4	SELS	R/W	3h	Selects the SMCLK source 0h (R/W) = LFXTCLK : LFXTCLK when LFXT available, otherwise VLOCLK. 1h (R/W) = VLOCLK : VLOCLK 2h (R/W) = LFMODCLK : LFMODCLK 3h (R/W) = DCOCLK : DCOCLK 4h (R/W) = MODCLK : MODCLK 5h (R/W) = HFXTCLK : HFXTCLK when HFXT available, otherwise DCOCLK.
3	RESERVED	R	0h	Reserved. Always reads as 0.
2-0	SELM	R/W	3h	Selects the MCLK source 0h (R/W) = LFXTCLK : LFXTCLK when LFXT available, otherwise VLOCLK 1h (R/W) = VLOCLK : VLOCLK 2h (R/W) = LFMODCLK : LFMODCLK 3h (R/W) = DCOCLK : DCOCLK 4h (R/W) = MODCLK : MODCLK 5h (R/W) = HFXTCLK : HFXTCLK when HFXT available, otherwise DCOCLK

3.3.4 CTL3 Register (Offset = 6h) [reset = 33h]

CTL3 is shown in [Figure 3-8](#) and described in [Table 3-7](#).

Return to the [Summary Table](#).

Clock System Control 3 Register

Figure 3-8. CTL3 Register

15	14	13	12	11	10	9	8
RESERVED						DIVA	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		DIVS		RESERVED		DIVM	
R-0h		R/W-3h		R-0h		R/W-3h	

Table 3-7. CTL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0h	Reserved. Always reads as 0.
10-8	DIVA	R/W	0h	ACLK source divider. Divides the frequency of the ACLK clock source. 0h (R/W) = 1 : /1 1h (R/W) = 2 : /2 2h (R/W) = 4 : /4 3h (R/W) = 8 : /8 4h (R/W) = 16 : /16 5h (R/W) = 32 : /32
7	RESERVED	R	0h	Reserved. Always reads as 0.
6-4	DIVS	R/W	3h	SMCLK source divider. Divides the frequency of the SMCLK clock source. 0h (R/W) = 1 : /1 1h (R/W) = 2 : /2 2h (R/W) = 4 : /4 3h (R/W) = 8 : /8 4h (R/W) = 16 : /16 5h (R/W) = 32 : /32
3	RESERVED	R	0h	Reserved. Always reads as 0.
2-0	DIVM	R/W	3h	MCLK source divider. Divides the frequency of the MCLK clock source. 0h (R/W) = 1 : /1 1h (R/W) = 2 : /2 2h (R/W) = 4 : /4 3h (R/W) = 8 : /8 4h (R/W) = 16 : /16 5h (R/W) = 32 : /32

3.3.5 CTL4 Register (Offset = 8h) [reset = CDC9h]

CTL4 is shown in [Figure 3-9](#) and described in [Table 3-8](#).

Return to the [Summary Table](#).

Clock System Control 4 Register

Figure 3-9. CTL4 Register

15	14	13	12	11	10	9	8
HFXTDRIVE		RESERVED	HFXTBYPASS		HFFREQ	RESERVED	HFXTOFF
R/W-3h		R-0h	R/W-0h		R/W-3h	R-0h	R/W-1h
7	6	5	4	3	2	1	0
LFXTDRIVE		RESERVED	LFXTBYPASS	VLOOFF	RESERVED	SMCLKOFF	LFXTOFF
R/W-3h		R-0h	R/W-0h	R/W-1h	R-0h	R/W-0h	R/W-1h

Table 3-8. CTL4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	HFXTDRIVE	R/W	3h	The HFXT oscillator current can be adjusted to its drive needs. This in combination with the HFFREQ bits can be used for optimizing crystal power based on crystal characteristics. 0h (R/W) = Lowest current consumption 1h (R/W) = Increased drive strength HFXT oscillator 2h (R/W) = Increased drive strength HFXT oscillator 3h (R/W) = Maximum drive strength HFXT oscillator
13	RESERVED	R	0h	Reserved. Always reads as 0.
12	HFXTBYPASS	R/W	0h	HFXT bypass select 0h (R/W) = HFXT sourced from external crystal 1h (R/W) = HFXT sourced from external clock signal
11-10	HFFREQ	R/W	3h	The HFXT frequency selection. These bits must be set to the appropriate frequency for crystal or bypass modes of operation. 0h (R/W) = 0 to 4 MHz 1h (R/W) = Greater than 4 MHz to 8 MHz 2h (R/W) = Greater than 8 MHz to 16 MHz 3h (R/W) = Greater than 16 MHz to 24 MHz
9	RESERVED	R	0h	Reserved. Always reads as 0.
8	HFXTOFF	R/W	1h	Turns off the HFXT oscillator 0h (R/W) = HFXT is on if HFXT is selected via the port selection and HFXT is not in bypass mode of operation 1h (R/W) = HFXT is off if it is not used as a source for ACLK, MCLK, or SMCLK
7-6	LFXTDRIVE	R/W	3h	The LFXT oscillator current can be adjusted to its drive needs. 0h (R/W) = Lowest drive strength and current consumption LFXT oscillator 1h (R/W) = Increased drive strength LFXT oscillator 2h (R/W) = Increased drive strength LFXT oscillator 3h (R/W) = Maximum drive strength and maximum current consumption LFXT oscillator
5	RESERVED	R	0h	Reserved. Always reads as 0.
4	LFXTBYPASS	R/W	0h	LFXT bypass select 0h (R/W) = LFXT sourced from external crystal 1h (R/W) = LFXT sourced from external clock signal
3	VLOOFF	R/W	1h	VLO off. This bit turns off the VLO. 0h (R/W) = VLO is on 1h (R/W) = VLO is off if it is not used as a source for ACLK, MCLK, or SMCLK or if not used as a source for the RTC in LPM3.5

Table 3-8. CTL4 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	RESERVED	R	0h	Reserved. Always reads as 0.
1	SMCLKOFF	R/W	0h	SMCLK off. This bit turns off the SMCLK. 0h (R/W) = SMCLK on 1h (R/W) = SMCLK off
0	LFXTOFF	R/W	1h	LFXT off. This bit turns off the LFXT. 0h (R/W) = LFXT is on if LFXT is selected via the port selection and LFXT is not in bypass mode of operation 1h (R/W) = LFXT is off if it is not used as a source for ACLK, MCLK, or SMCLK

3.3.6 CTL5 Register (Offset = Ah) [reset = 00C5h]

CTL5 is shown in [Figure 3-10](#) and described in [Table 3-9](#).

Return to the [Summary Table](#).

Clock System Control 5 Register

Figure 3-10. CTL5 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
ENSTFCNT2	ENSTFCNT1	RESERVED			SWDONE	HFXTOFFG	LFXTOFFG
R/W-1h	R/W-1h	R-0h			R-1h	R/W-0h	R/W-1h

Table 3-9. CTL5 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved. Always reads as 0.
7	ENSTFCNT2	R/W	1h	Enable start counter for HFXT when available. 0h (R/W) = DISABLE : Startup fault counter disabled. Counter is cleared. 1h (R/W) = ENABLE : Startup fault counter enabled
6	ENSTFCNT1	R/W	1h	Enable start counter for LFXT. 0h (R/W) = DISABLE : Startup fault counter disabled. Counter is cleared. 1h (R/W) = ENABLE : Startup fault counter enabled
5-3	RESERVED	R	0h	Reserved. Always reads as 0.
2	SWDONE	R	1h	Clock switch done. This bit indicates a clock switch is done. A clock switch happens when changing the clock system configuration, including any write access of CSCTL1/CSCTL2/CSCTL3 registers, or any fail-safe condition happens. When clock switch happens, this bit is reset automatically, and is set again after the switching is done. Only available in CS_A module. 0h (R) = Clock switch is in progress 1h (R) = Clock switch is done
1	HFXTOFFG	R/W	0h	HFXT oscillator fault flag. If this bit is set, the OFIFG flag is also set. HFXTOFFG is set if a HFXT fault condition exists. HFXTOFFG can be cleared via software. If the HFXT fault condition still remains, HFXTOFFG is set. 0h (R/W) = No fault condition occurred after the last reset 1h (R/W) = HFXT fault; an HFXT fault occurred after the last reset
0	LFXTOFFG	R/W	1h	LFXT oscillator fault flag. If this bit is set, the OFIFG flag is also set. LFXTOFFG is set if a LFXT fault condition exists. LFXTOFFG can be cleared via software. If the LFXT fault condition still remains, LFXTOFFG is set. 0h (R/W) = No fault condition occurred after the last reset 1h (R/W) = LFXT fault; an LFXT fault occurred after the last reset

3.3.7 CTL6 Register (Offset = Ch) [reset = 7h]

CTL6 is shown in [Figure 3-11](#) and described in [Table 3-10](#).

Return to the [Summary Table](#).

Clock System Control 6 Register

Figure 3-11. CTL6 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				MODCLKREQE N	SMCLKREQEN	MCLKREQEN	ACLKREQEN
R-0h				R/W-0h	R/W-1h	R/W-1h	R/W-1h

Table 3-10. CTL6 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved. Always reads as 0.
3	MODCLKREQEN	R/W	0h	MODCLK clock request enable. Setting this enables conditional module requests for MODCLK. 0h (R/W) = DISABLE : MODCLK conditional requests are disabled 1h (R/W) = ENABLE : MODCLK conditional requests are enabled
2	SMCLKREQEN	R/W	1h	SMCLK clock request enable. Setting this enables conditional module requests for SMCLK. 0h (R/W) = DISABLE : SMCLK conditional requests are disabled 1h (R/W) = ENABLE : SMCLK conditional requests are enabled
1	MCLKREQEN	R/W	1h	MCLK clock request enable. Setting this enables conditional module requests for MCLK. 0h (R/W) = DISABLE : MCLK conditional requests are disabled 1h (R/W) = ENABLE : MCLK conditional requests are enabled
0	ACLKREQEN	R/W	1h	ACLK clock request enable. Setting this enables conditional module requests for ACLK. 0h (R/W) = DISABLE : ACLK conditional requests are disabled 1h (R/W) = ENABLE : ACLK conditional requests are enabled

CPUX

This chapter describes the extended MSP430X 16-bit RISC CPU (CPUX) with 1MB memory access, its addressing modes, and instruction set.

NOTE: The MSP430X CPIX implemented on this device family, formally called CPVXV2, has in some cases, slightly different cycle counts from the MSP430X CPIX implemented on the F2xx and F4xx families.

Topic	Page
4.1 MSP430X CPU (CPUX) Introduction	113
4.2 Interrupts	115
4.3 CPU Registers	116
4.4 Addressing Modes	122
4.5 MSP430 and MSP430X Instructions	141
4.6 Instruction Set Description	157

4.1 MSP430X CPU (CPUX) Introduction

The MSP430X CPU incorporates features specifically designed for modern programming techniques, such as calculated branching, table processing, and the use of high-level languages such as C. The MSP430X CPU can address a 1MB address range without paging. The MSP430X CPU is completely backward compatible with the MSP430 CPU.

The MSP430X CPU features include:

- RISC architecture
- Orthogonal architecture
- Full register access including program counter (PC), status register (SR), and stack pointer (SP)
- Single-cycle register operations
- Large register file reduces fetches to memory.
- 20-bit address bus allows direct access and branching throughout the entire memory range without paging.
- 16-bit data bus allows direct manipulation of word-wide arguments.
- Constant generator provides the 6 most often used immediate values and reduces code size.
- Direct memory-to-memory transfers without intermediate register holding
- Byte, word, and 20-bit address-word addressing

[Figure 4-1](#) shows the block diagram of the MSP430X CPU.

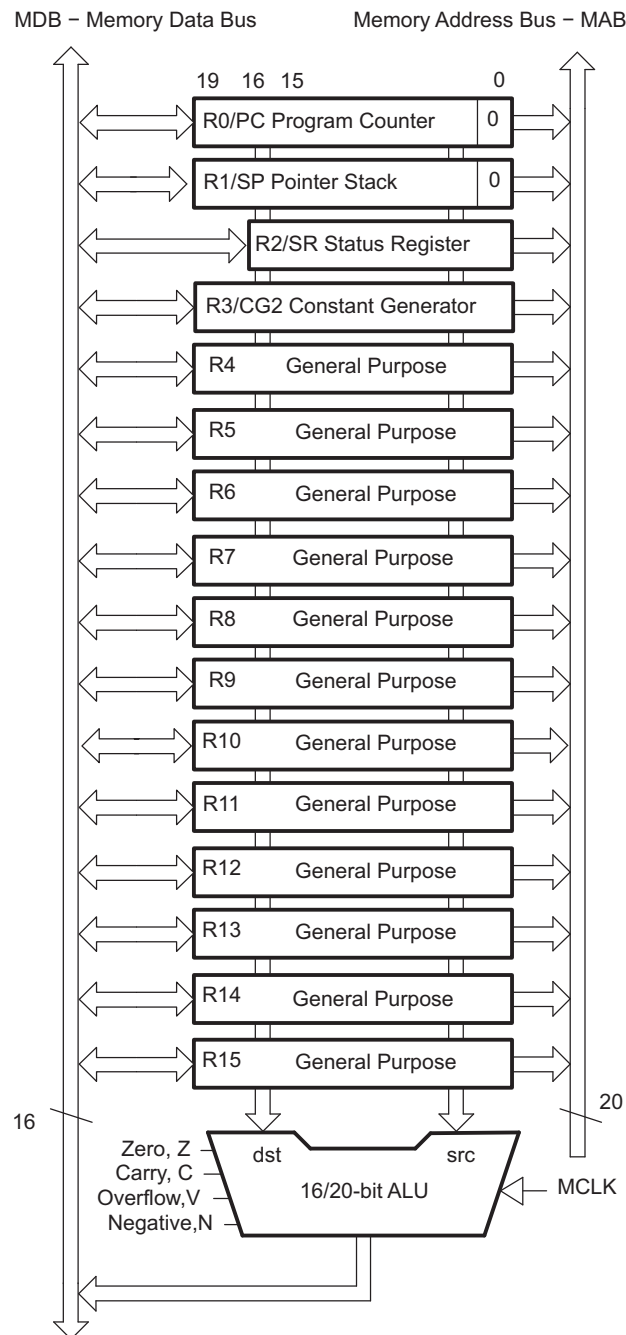


Figure 4-1. MSP430X CPU Block Diagram

4.2 Interrupts

The MSP430X has the following interrupt structure:

- Vectored interrupts with no polling necessary
- Interrupt vectors are located downward from address 0FFFEh.

The interrupt vectors contain 16-bit addresses that point into the lower 64KB memory. This means all interrupt handlers must start in the lower 64KB memory.

During an interrupt, the program counter (PC) and the status register (SR) are pushed onto the stack as shown in Figure 4-2. The MSP430X architecture stores the complete 20-bit PC value efficiently by appending the PC bits 19:16 to the stored SR value automatically on the stack. When the RETI instruction is executed, the full 20-bit PC is restored making return from interrupt to any address in the memory range possible.

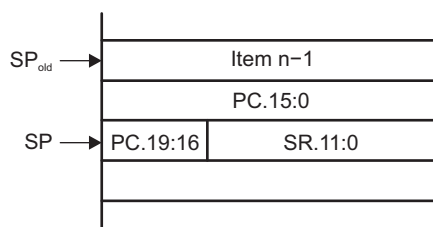


Figure 4-2. PC Storage on the Stack for Interrupts

4.3 CPU Registers

The CPU incorporates 16 registers (R0 through R15). Registers R0, R1, R2, and R3 have dedicated functions. Registers R4 through R15 are working registers for general use.

4.3.1 Program Counter (PC)

The 20-bit program counter (PC, also called R0) points to the next instruction to be executed. Each instruction uses an even number of bytes (2, 4, 6, or 8 bytes), and the PC is incremented accordingly. Instruction accesses are performed on word boundaries, and the PC is aligned to even addresses. Figure 4-3 shows the PC.



Figure 4-3. Program Counter

The PC can be addressed with all instructions and addressing modes. A few examples:

```
MOV.W #LABEL,PC ; Branch to address LABEL (lower 64KB)
```

```
MOVA #LABEL,PC ; Branch to address LABEL (1MB memory)
```

```
MOV.W LABEL,PC ; Branch to address in word LABEL
                ; (lower 64KB)
```

```
MOV.W @R14,PC ; Branch indirect to address in
                ; R14 (lower 64KB)
```

```
ADDA #4,PC ; Skip 2 words (1MB memory)
```

The BR and CALL instructions reset the upper 4 PC bits to 0. Only addresses in the lower 64KB address range can be reached with the BR or CALL instruction. When branching or calling, addresses beyond the lower 64KB range can only be reached using the BRA or CALLA instructions. Also, any instruction to directly modify the PC does so according to the used addressing mode. For example, MOV.W #value,PC clears the upper 4 bits of the PC, because it is a .W instruction.

The PC is automatically stored on the stack with CALL (or CALLA) instructions and during an interrupt service routine. Figure 4-4 shows the storage of the PC with the return address after a CALLA instruction. A CALL instruction stores only bits 15:0 of the PC.

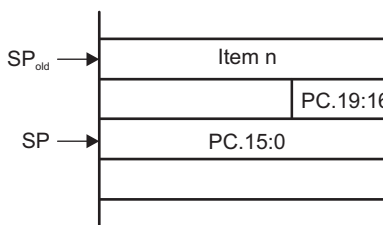


Figure 4-4. PC Storage on the Stack for CALLA

The RETA instruction restores bits 19:0 of the PC and adds 4 to the stack pointer (SP). The RET instruction restores bits 15:0 to the PC and adds 2 to the SP.

4.3.2 Stack Pointer (SP)

The 20-bit stack pointer (SP, also called R1) is used by the CPU to store the return addresses of subroutine calls and interrupts. It uses a predecrement, postincrement scheme. In addition, the SP can be used by software with all instructions and addressing modes. Figure 4-5 shows the SP. The SP is initialized into RAM by the user, and is always aligned to even addresses.

Figure 4-6 shows the stack usage. Figure 4-7 shows the stack usage when 20-bit address words are pushed.

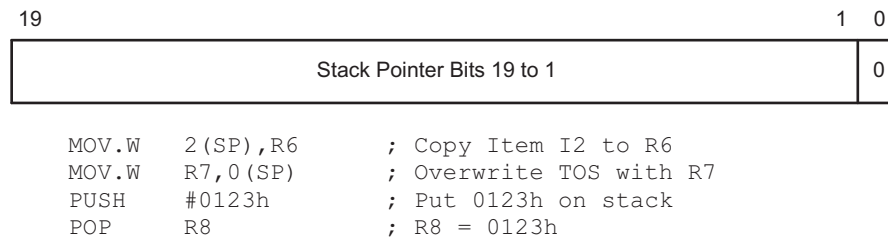


Figure 4-5. Stack Pointer

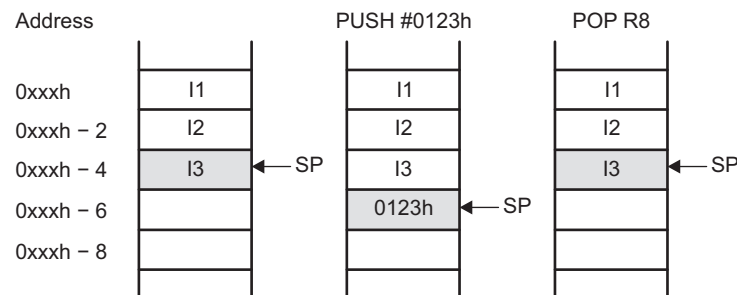


Figure 4-6. Stack Usage

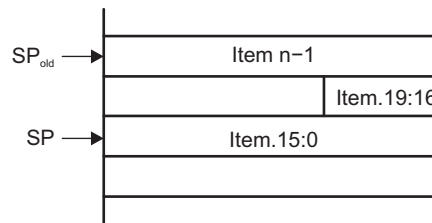


Figure 4-7. PUSHX.A Format on the Stack

Figure 4-8 shows the special cases of using the SP as an argument to the PUSH and POP instructions.

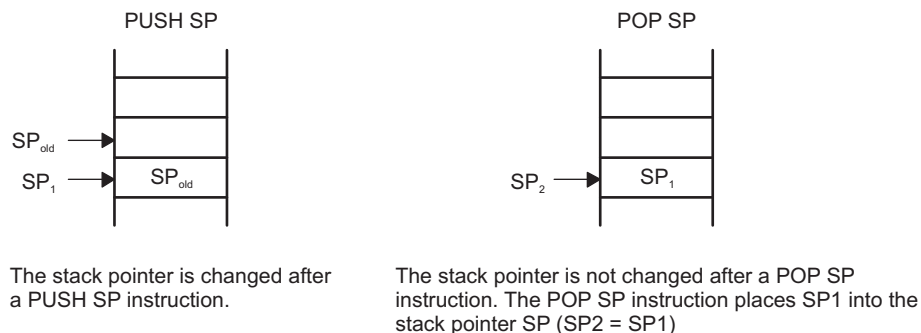


Figure 4-8. PUSH SP, POP SP Sequence

4.3.3 Status Register (SR)

The 16-bit status register (SR, also called R2), used as a source or destination register, can only be used in register mode addressed with word instructions. The remaining combinations of addressing modes are used to support the constant generator. Figure 4-9 shows the SR bits. Do not write 20-bit values to the SR. Unpredictable operation can result.

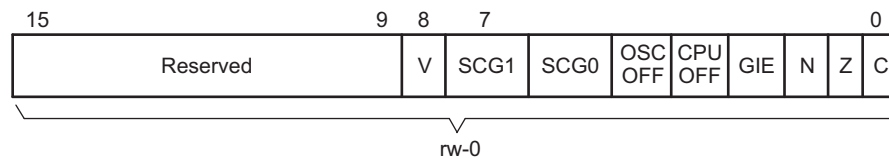

Figure 4-9. SR Bits

Table 4-1 describes the SR bits.

Table 4-1. SR Bit Description

Bit	Description
Reserved	Reserved
V	Overflow. This bit is set when the result of an arithmetic operation overflows the signed-variable range.
	ADD(.B), ADDX(.B, .A), ADDC(.B), ADDCX(.B.A), ADDA Set when: positive + positive = negative negative + negative = positive otherwise reset
SCG1	SUB(.B), SUBX(.B, .A), SUBC(.B), SUBCX(.B, .A), SUBA, CMP(.B), CMPX(.B, .A), CMPA Set when: positive – negative = negative negative – positive = positive otherwise reset
SCG0	System clock generator 0. This bit may be used to enable or disable functions in the clock system depending on the device family; for example, FLL enable or disable.
OSCOFF	Oscillator off. When this bit is set, it turns off the LFXT1 crystal oscillator when LFXT1CLK is not used for MCLK or SMCLK. In FRAM devices, CPUOFF must be 1 to disable the crystal oscillator.
CPUOFF	CPU off. When this bit is set, it turns off the CPU and requests a low-power mode according to the settings of bits OSCOFF, SCG0, and SCG1.
GIE	General interrupt enable. When this bit is set, it enables maskable interrupts. When it is reset, all maskable interrupts are disabled.
N	Negative. This bit is set when the result of an operation is negative and cleared when the result is positive.
Z	Zero. This bit is set when the result of an operation is 0 and cleared when the result is not 0.
C	Carry. This bit is set when the result of an operation produced a carry and cleared when no carry occurred.

NOTE: Bit manipulations of the SR should be done by the following instructions: MOV, BIS, and BIC.

4.3.4 Constant Generator Registers (CG1 and CG2)

Six commonly-used constants are generated with the constant generator registers R2 (CG1) and R3 (CG2), without requiring an additional 16-bit word of program code. The constants are selected with the source register addressing modes (As), as described in [Table 4-2](#).

Table 4-2. Values of Constant Generators CG1, CG2

Register	As	Constant	Remarks
R2	00	–	Register mode
R2	01	(0)	Absolute address mode
R2	10	00004h	+4, bit processing
R2	11	00008h	+8, bit processing
R3	00	00000h	0, word processing
R3	01	00001h	+1
R3	10	00002h	+2, bit processing
R3	11	FFh, FFFFh, FFFFFh	–1, word processing

The constant generator advantages are:

- No special instructions required
- No additional code word for the 6 constants
- No code memory access required to retrieve the constant

The assembler uses the constant generator automatically if 1 of the 6 constants is used as an immediate source operand. Registers R2 and R3, used in the constant mode, cannot be addressed explicitly; they act as source-only registers.

4.3.4.1 Constant Generator – Expanded Instruction Set

The RISC instruction set of the MSP430 has only 27 instructions. However, the constant generator allows the MSP430 assembler to support 24 additional emulated instructions. For example, the single-operand instruction:

```
CLR dst
```

is emulated by the double-operand instruction with the same length:

```
MOV R3, dst
```

where the #0 is replaced by the assembler, and R3 is used with As = 00.

```
INC dst
```

is replaced by:

```
ADD #1, dst
```

4.3.5 General-Purpose Registers (R4 to R15)

The 12 CPU registers (R4 to R15) contain 8-bit, 16-bit, or 20-bit values. Any byte-write to a CPU register clears bits 19:8. Any word-write to a register clears bits 19:16. The only exception is the SXT instruction. The SXT instruction extends the sign through the complete 20-bit register.

Figure 4-10 through Figure 4-14 show the handling of byte, word, and address-word data. Note the reset of the leading most significant bits (MSBs) if a register is the destination of a byte or word instruction.

Figure 4-10 shows byte handling (8-bit data, .B suffix). The handling is shown for a source register and a destination memory byte and for a source memory byte and a destination register.

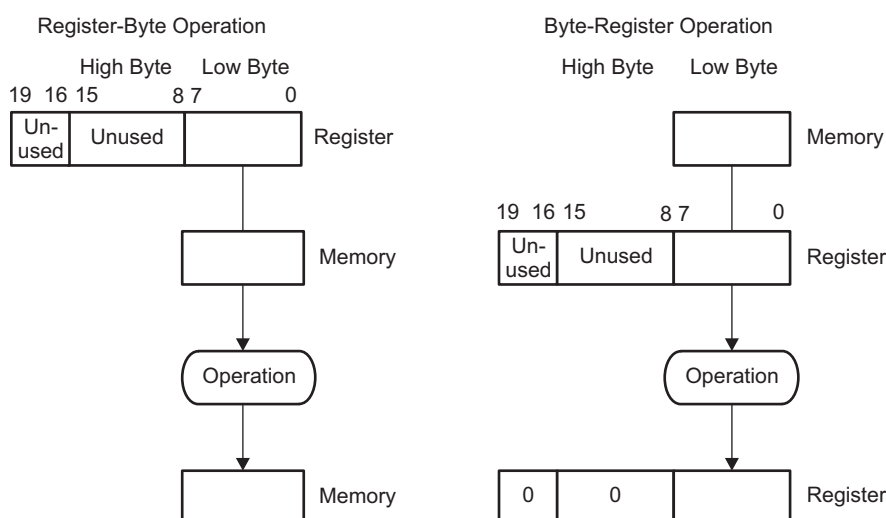


Figure 4-10. Register-Byte and Byte-Register Operation

Figure 4-11 and Figure 4-12 show 16-bit word handling (.W suffix). The handling is shown for a source register and a destination memory word and for a source memory word and a destination register.

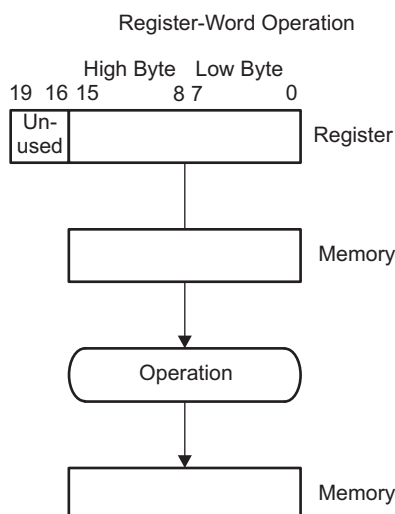


Figure 4-11. Register-Word Operation

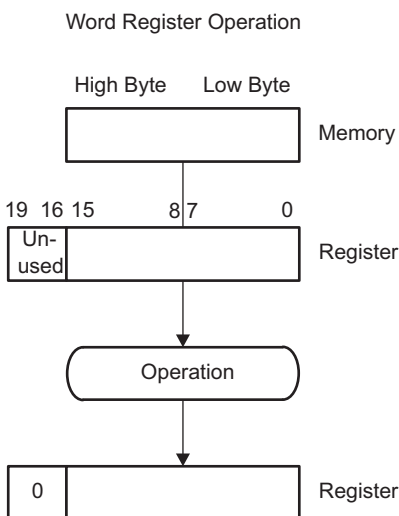


Figure 4-12. Word-Register Operation

Figure 4-13 and Figure 4-14 show 20-bit address-word handling (.A suffix). The handling is shown for a source register and a destination memory address-word and for a source memory address-word and a destination register.

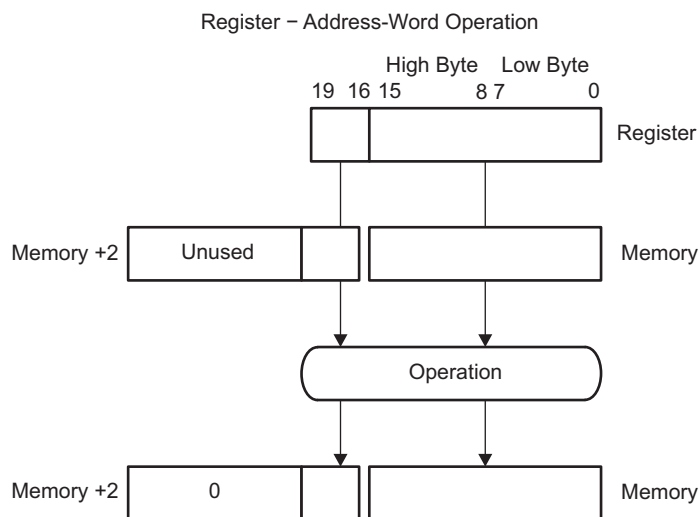


Figure 4-13. Register – Address-Word Operation

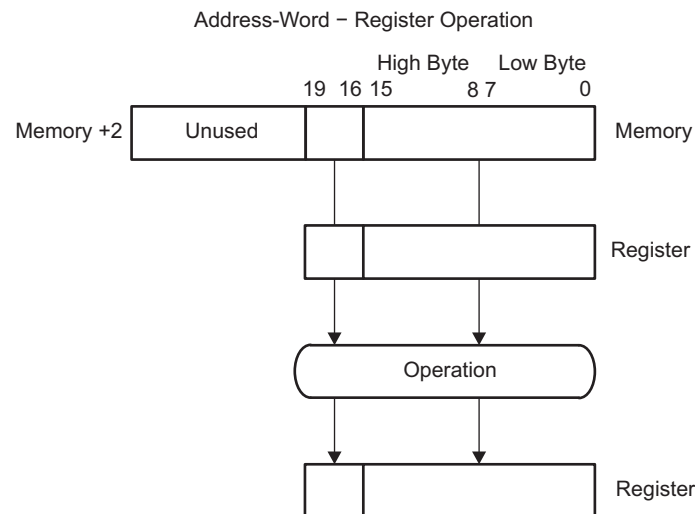


Figure 4-14. Address-Word – Register Operation

4.4 Addressing Modes

7 addressing modes for the source operand and 4 addressing modes for the destination operand use 16-bit or 20-bit addresses (see [Table 4-3](#)). The MSP430 and MSP430X instructions are usable throughout the entire 1MB memory range.

Table 4-3. Source and Destination Addressing

As, Ad	Addressing Mode	Syntax	Description
00, 0	Register	Rn	Register contents are operand.
01, 1	Indexed	X(Rn)	(Rn + X) points to the operand. X is stored in the next word, or stored in combination of the preceding extension word and the next word.
01, 1	Symbolic	ADDR	(PC + X) points to the operand. X is stored in the next word, or stored in combination of the preceding extension word and the next word. Indexed mode X(PC) is used.
01, 1	Absolute	&ADDR	The word following the instruction contains the absolute address. X is stored in the next word, or stored in combination of the preceding extension word and the next word. Indexed mode X(SR) is used.
10, –	Indirect register	@Rn	Rn is used as a pointer to the operand.
11, –	Indirect autoincrement	@Rn+	Rn is used as a pointer to the operand. Rn is incremented afterwards by 1 for .B instructions, by 2 for .W instructions, and by 4 for .A instructions.
11, –	Immediate	#N	N is stored in the next word, or stored in combination of the preceding extension word and the next word. Indirect autoincrement mode @PC+ is used.

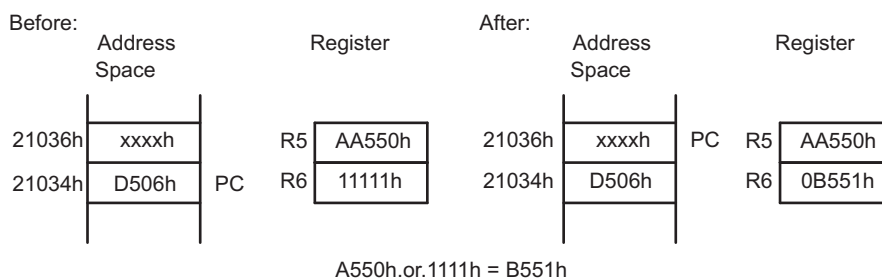
The 7 addressing modes are explained in detail in the following sections. Most of the examples show the same addressing mode for the source and destination, but any valid combination of source and destination addressing modes is possible in an instruction.

NOTE: Use of Labels EDE, TONI, TOM, and LEO

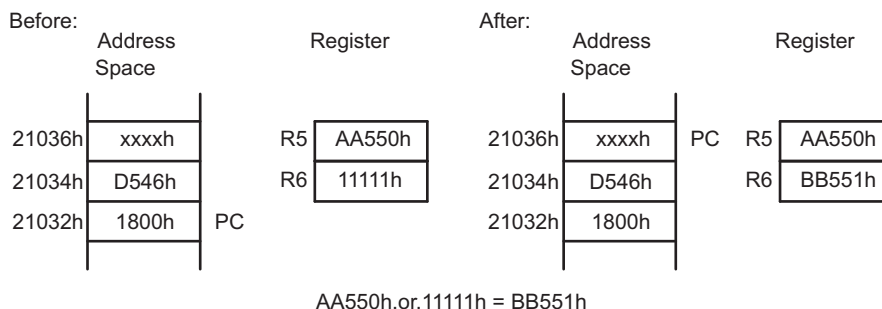
Throughout MSP430 documentation, EDE, TONI, TOM, and LEO are used as generic labels. They are only labels and have no special meaning.

4.4.1 Register Mode

Operation:	The operand is the 8-, 16-, or 20-bit content of the used CPU register.
Length:	1, 2, or 3 words
Comment:	Valid for source and destination
Byte operation:	Byte operation reads only the 8 least significant bits (LSBs) of the source register Rsrc and writes the result to the 8 LSBs of the destination register Rdst. The bits Rdst.19:8 are cleared. The register Rsrc is not modified.
Word operation:	Word operation reads the 16 LSBs of the source register Rsrc and writes the result to the 16 LSBs of the destination register Rdst. The bits Rdst.19:16 are cleared. The register Rsrc is not modified.
Address-word operation:	Address-word operation reads the 20 bits of the source register Rsrc and writes the result to the 20 bits of the destination register Rdst. The register Rsrc is not modified
SXT exception:	The SXT instruction is the only exception for register operation. The sign of the low byte in bit 7 is extended to the bits Rdst.19:8.
Example:	BIS.W R5,R6 ; This instruction logically ORs the 16-bit data contained in R5 with the 16-bit contents of R6. R6.19:16 is cleared.



Example:	BISX.A R5,R6 ; This instruction logically ORs the 20-bit data contained in R5 with the 20-bit contents of R6. The extension word contains the A/L bit for 20-bit data. The instruction word uses byte mode with bits A/L:B/W = 01. The result of the instruction is:



4.4.2 Indexed Mode

The indexed mode calculates the address of the operand by adding the signed index to a CPU register. The Indexed mode has 4 addressing possibilities:

- MSP430 instruction with indexed mode in lower 64KB memory (see [Section 4.4.2.1](#))
- MSP430 instruction with indexed mode addressing memory above the lower 64KB memory (see [Section 4.4.2.2](#))
- MSP430X instruction with indexed mode (see [Section 4.4.2.3](#))
- MSP430X address instructions with indexed mode (see [Section 4.4.2.4](#))

4.4.2.1 MSP430 Instruction With Indexed Mode in Lower 64KB Memory

If the CPU register Rn points to an address in the lower 64KB of the memory range, the calculated memory address bits 19:16 are cleared after the addition of the CPU register Rn and the signed 16-bit index. This means the calculated memory address is always located in the lower 64KB and does not overflow or underflow out of the lower 64KB memory space. The RAM and the peripheral registers can be accessed this way and existing MSP430 software is usable without modifications as shown in [Figure 4-15](#).

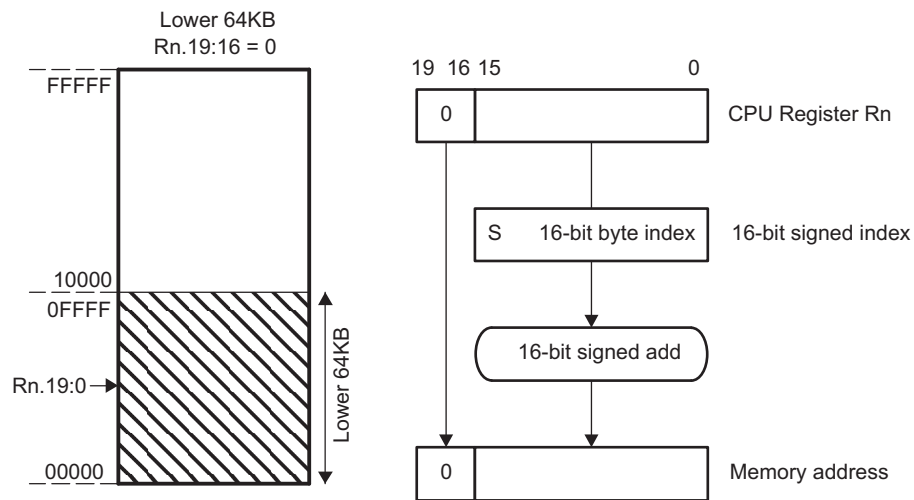
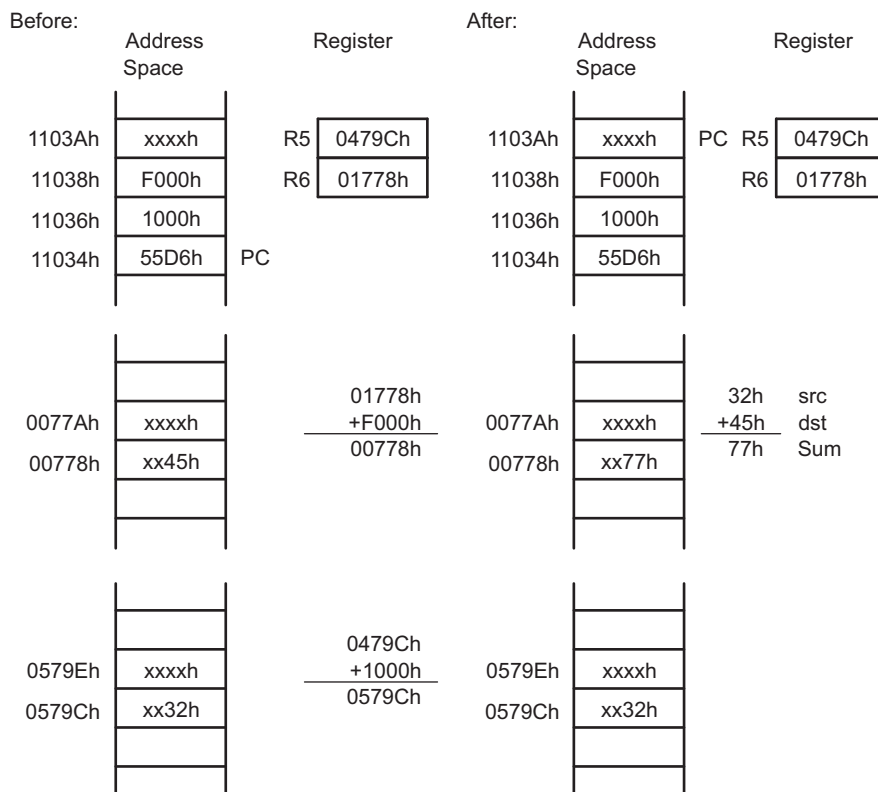


Figure 4-15. Indexed Mode in Lower 64KB

Length:	2 or 3 words
Operation:	The signed 16-bit index is located in the next word after the instruction and is added to the CPU register Rn. The resulting bits 19:16 are cleared giving a truncated 16-bit memory address, which points to an operand address in the range 00000h to 0FFFFh. The operand is the content of the addressed memory location.
Comment:	Valid for source and destination. The assembler calculates the register index and inserts it.
Example:	ADD.B 1000h(R5), 0F000h(R6); This instruction adds the 8-bit data contained in source byte 1000h(R5) and the destination byte 0F000h(R6) and places the result into the destination byte. Source and destination bytes are both located in the lower 64KB due to the cleared bits 19:16 of registers R5 and R6.
Source:	The byte pointed to by R5 + 1000h results in address 0479Ch + 1000h = 0579Ch after truncation to a 16-bit address.
Destination:	The byte pointed to by R6 + F000h results in address 01778h + F000h = 00778h after truncation to a 16-bit address.



4.4.2.2 MSP430 Instruction With Indexed Mode in Upper Memory

If the CPU register Rn points to an address above the lower 64KB memory, the Rn bits 19:16 are used for the address calculation of the operand. The operand may be located in memory in the range $Rn \pm 32KB$, because the index, X, is a signed 16-bit value. In this case, the address of the operand can overflow or underflow into the lower 64KB memory space (see [Figure 4-16](#) and [Figure 4-17](#)).

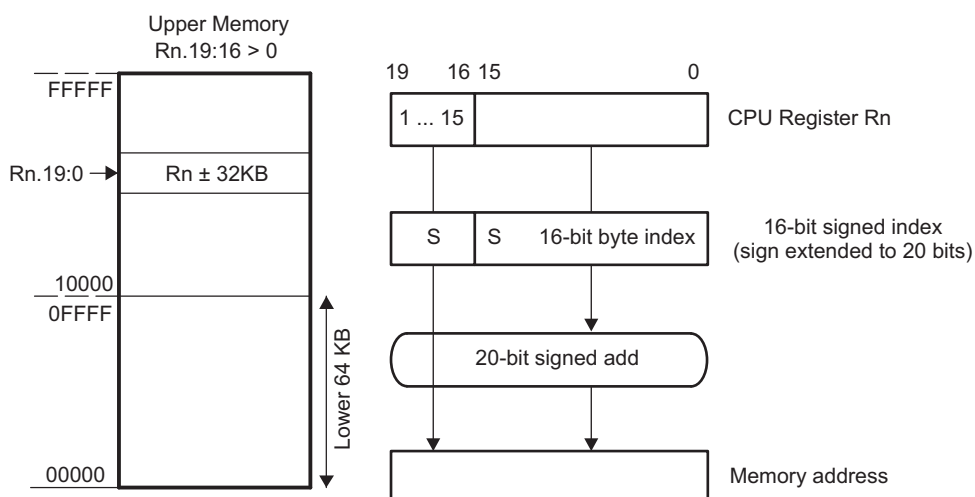


Figure 4-16. Indexed Mode in Upper Memory

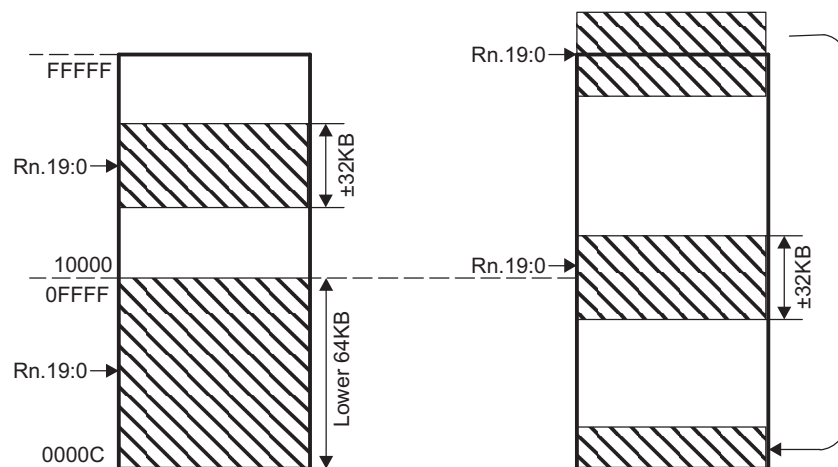


Figure 4-17. Overflow and Underflow for Indexed Mode

Length:	2 or 3 words
Operation:	The sign-extended 16-bit index in the next word after the instruction is added to the 20 bits of the CPU register Rn. This delivers a 20-bit address, which points to an address in the range 0 to FFFFFh. The operand is the content of the addressed memory location.
Comment:	Valid for source and destination. The assembler calculates the register index and inserts it.
Example:	ADD.W 8346h(R5),2100h(R6) ; This instruction adds the 16-bit data contained in the source and the destination addresses and places the 16-bit result into the destination. Source and destination operand can be located in the entire address range.
Source:	The word pointed to by R5 + 8346h. The negative index 8346h is sign extended, which results in address 23456h + F8346h = 1B79Ch.
Destination:	The word pointed to by R6 + 2100h results in address 15678h + 2100h = 17778h.

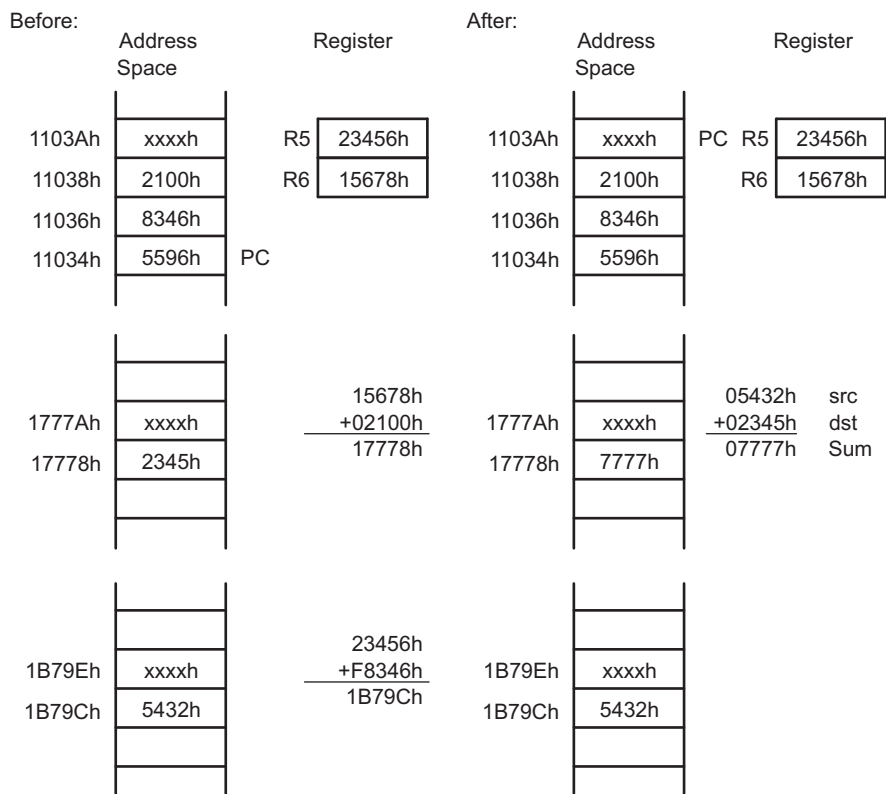


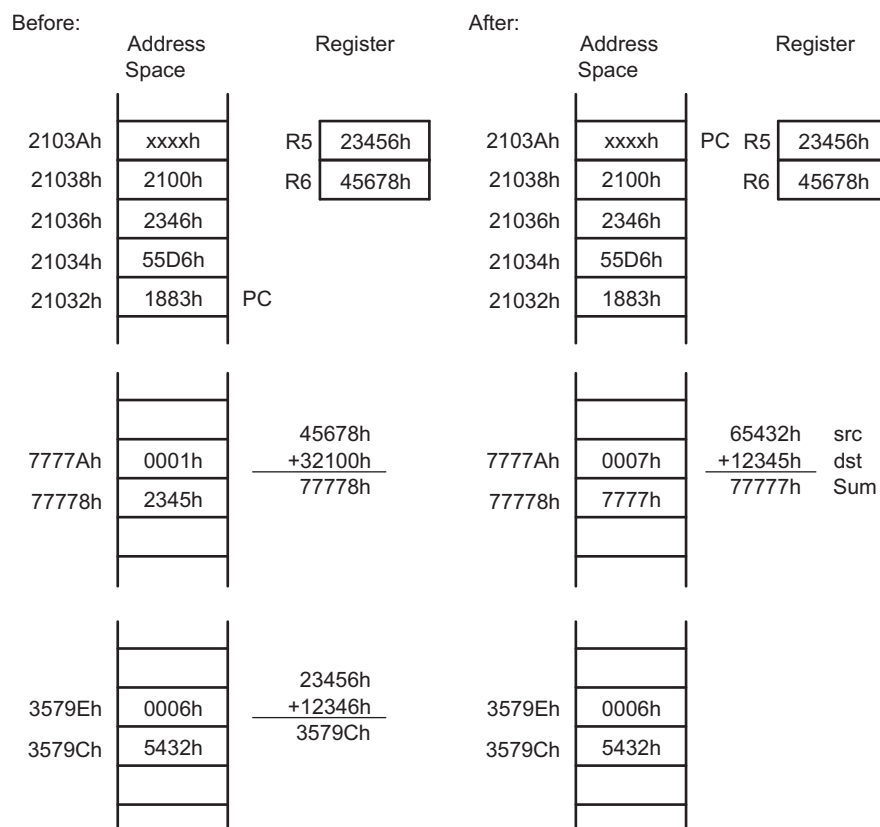
Figure 4-18. Example for Indexed Mode

4.4.2.3 MSP430X Instruction With Indexed Mode

When using an MSP430X instruction with indexed mode, the operand can be located anywhere in the range of $Rn + 19$ bits.

Length:	3 or 4 words
Operation:	The operand address is the sum of the 20-bit CPU register content and the 20-bit index. The 4 MSBs of the index are contained in the extension word; the 16 LSBs are contained in the word following the instruction. The CPU register is not modified
Comment:	Valid for source and destination. The assembler calculates the register index and inserts it.
Example:	ADDX.A 12346h(R5), 32100h(R6) ; This instruction adds the 20-bit data contained in the source and the destination addresses and places the result into the destination.
Source:	2 words pointed to by R5 + 12346h which results in address 23456h + 12346h = 3579Ch.
Destination:	2 words pointed to by R6 + 32100h which results in address 45678h + 32100h = 77778h.

The extension word contains the MSBs of the source index and of the destination index and the A/L bit for 20-bit data. The instruction word uses byte mode due to the 20-bit data length with bits A/L:B/W = 01.



4.4.2.4 MSP430X Address Instructions With Indexed Mode

When using an MSP430X address instruction with indexed mode, the operand is located in memory in the range $R_n \pm 32\text{KB}$, because the index, X, is a signed 16-bit value.

Length:	2 words
Operation:	The sign-extended 16-bit index in the next word after the instruction is added to the 20 bits of the CPU register R_n . This delivers a 20-bit address, which points to an address in the range 0 to FFFFFh. The operand is the content of the addressed memory location.
Comment:	Valid for source and destination. The assembler calculates the register index and inserts it.
Example:	MOVA 8002h(R5),R6 ; // R5 = 0x100 This instruction loads the 20-bit data contained in the source address into destination register.
Source:	2 words pointed to by $R5 + 8002\text{h}$ and $R5 + 8002\text{h} + 2\text{h}$ which results in address 00100h + F8002h (+2h) = F8102h and F8104h.
Destination:	Register R6

4.4.3 Symbolic Mode

The symbolic mode calculates the address of the operand by adding the signed index to the PC. The symbolic mode has 3 addressing possibilities:

- Symbolic mode in lower 64KB of memory
- MSP430 instruction with symbolic mode addressing memory above the lower 64KB of memory
- MSP430X instruction with symbolic mode

4.4.3.1 Symbolic Mode in Lower 64KB

If the PC points to an address in the lower 64KB of the memory range, the calculated memory address bits 19:16 are cleared after the addition of the PC and the signed 16-bit index. This means the calculated memory address is always located in the lower 64KB and does not overflow or underflow out of the lower 64KB memory space. The RAM and the peripheral registers can be accessed this way and existing MSP430 software is usable without modifications as shown in [Figure 4-19](#).

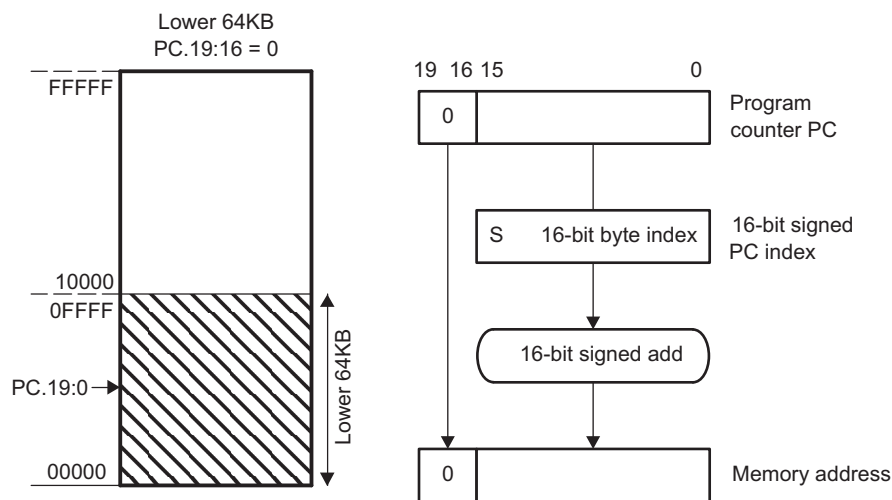
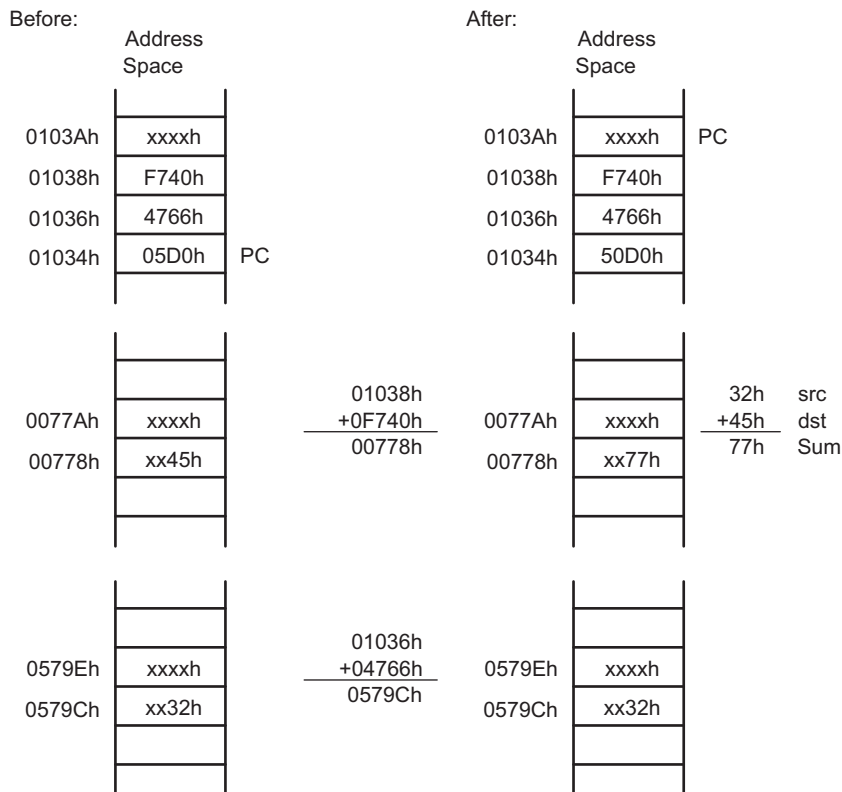


Figure 4-19. Symbolic Mode Running in Lower 64KB

Operation:	The signed 16-bit index in the next word after the instruction is added temporarily to the PC. The resulting bits 19:16 are cleared giving a truncated 16-bit memory address, which points to an operand address in the range 00000h to 0FFFFh. The operand is the content of the addressed memory location.
Length:	2 or 3 words
Comment:	Valid for source and destination. The assembler calculates the PC index and inserts it.
Example:	ADD.B EDE, TONI ; This instruction adds the 8-bit data contained in source byte EDE and destination byte TONI and places the result into the destination byte TONI. Bytes EDE and TONI and the program are located in the lower 64KB.
Source:	Byte EDE located at address 0579Ch, pointed to by PC + 4766h, where the PC index 4766h is the result of 0579Ch – 01036h = 04766h. Address 01036h is the location of the index for this example.
Destination:	Byte TONI located at address 00778h, pointed to by PC + F740h, is the truncated 16-bit result of 00778h – 1038h = FF740h. Address 01038h is the location of the index for this example.



4.4.3.2 MSP430 Instruction With Symbolic Mode in Upper Memory

If the PC points to an address above the lower 64KB memory, the PC bits 19:16 are used for the address calculation of the operand. The operand may be located in memory in the range $PC \pm 32KB$, because the index, X, is a signed 16-bit value. In this case, the address of the operand can overflow or underflow into the lower 64KB memory space as shown in Figure 4-20 and Figure 4-21.

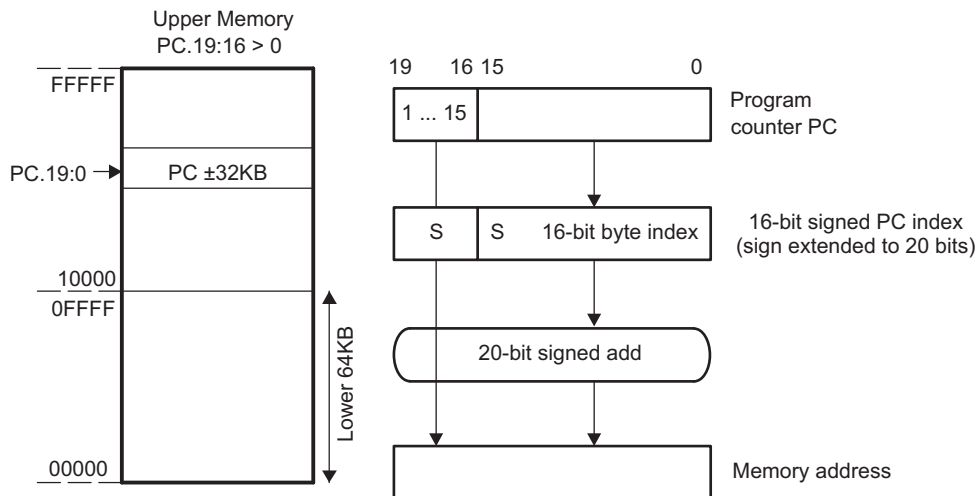


Figure 4-20. Symbolic Mode Running in Upper Memory

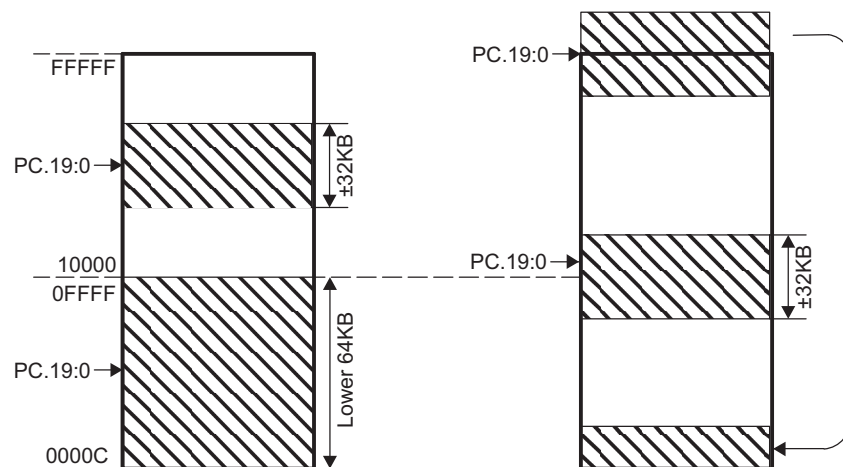
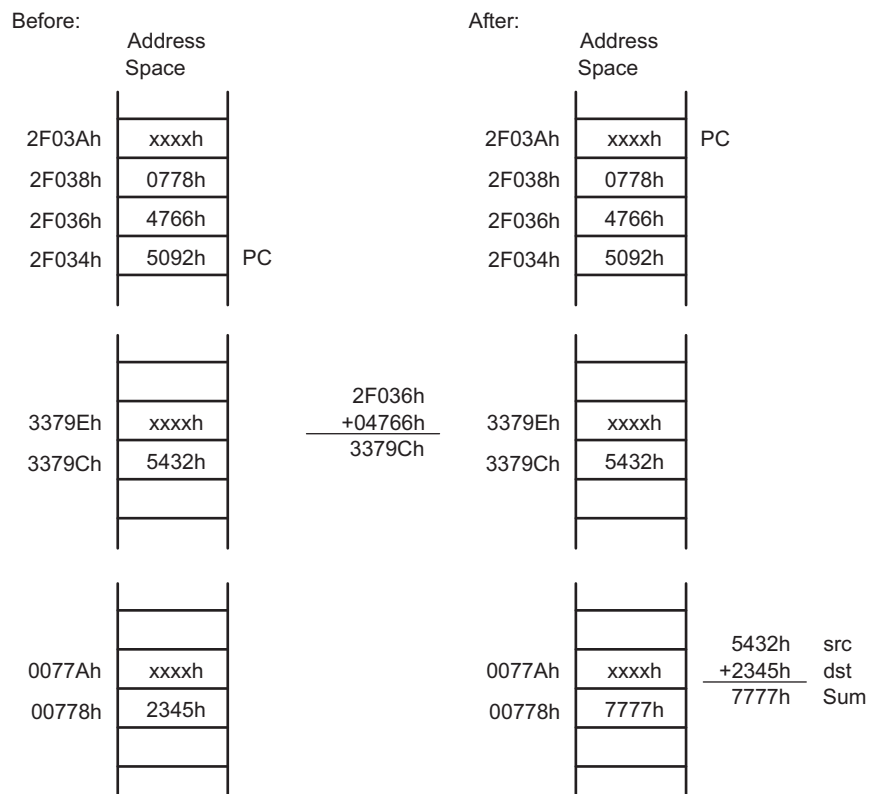


Figure 4-21. Overflow and Underflow for Symbolic Mode

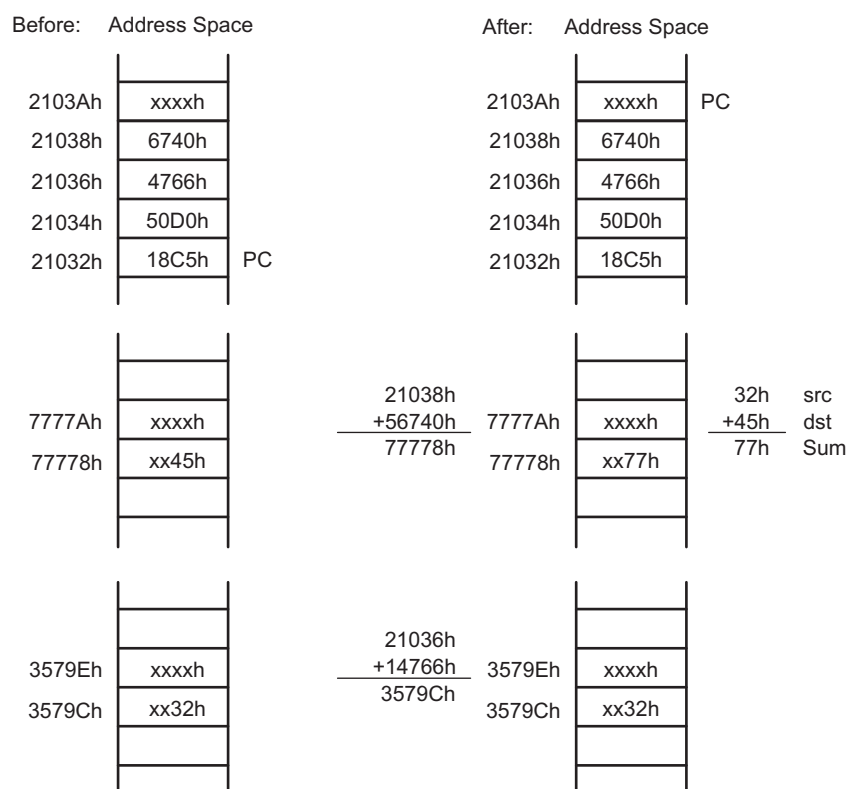
Length:	2 or 3 words
Operation:	The sign-extended 16-bit index in the next word after the instruction is added to the 20 bits of the PC. This delivers a 20-bit address, which points to an address in the range 0 to FFFFFh. The operand is the content of the addressed memory location.
Comment:	Valid for source and destination. The assembler calculates the PC index and inserts it
Example:	ADD.W EDE,&TONI ; This instruction adds the 16-bit data contained in source word EDE and destination word TONI and places the 16-bit result into the destination word TONI. For this example, the instruction is located at address 2F034h.
Source:	Word EDE at address 3379Ch, pointed to by PC + 4766h, which is the 16-bit result of 3379Ch – 2F036h = 04766h. Address 2F036h is the location of the index for this example.
Destination:	Word TONI located at address 00778h pointed to by the absolute address 00778h



4.4.3.3 MSP430X Instruction With Symbolic Mode

When using an MSP430X instruction with symbolic mode, the operand can be located anywhere in the range of PC + 19 bits.

Length:	3 or 4 words
Operation:	The operand address is the sum of the 20-bit PC and the 20-bit index. The 4 MSBs of the index are contained in the extension word; the 16 LSBs are contained in the word following the instruction.
Comment:	Valid for source and destination. The assembler calculates the register index and inserts it.
Example:	ADDX.B EDE, TONI ; This instruction adds the 8-bit data contained in source byte EDE and destination byte TONI and places the result into the destination byte TONI.
Source:	Byte EDE located at address 3579Ch, pointed to by PC + 14766h, is the 20-bit result of 3579Ch – 21036h = 14766h. Address 21036h is the address of the index in this example.
Destination:	Byte TONI located at address 77778h, pointed to by PC + 56740h, is the 20-bit result of 77778h – 21038h = 56740h. Address 21038h is the address of the index in this example.



4.4.4 Absolute Mode

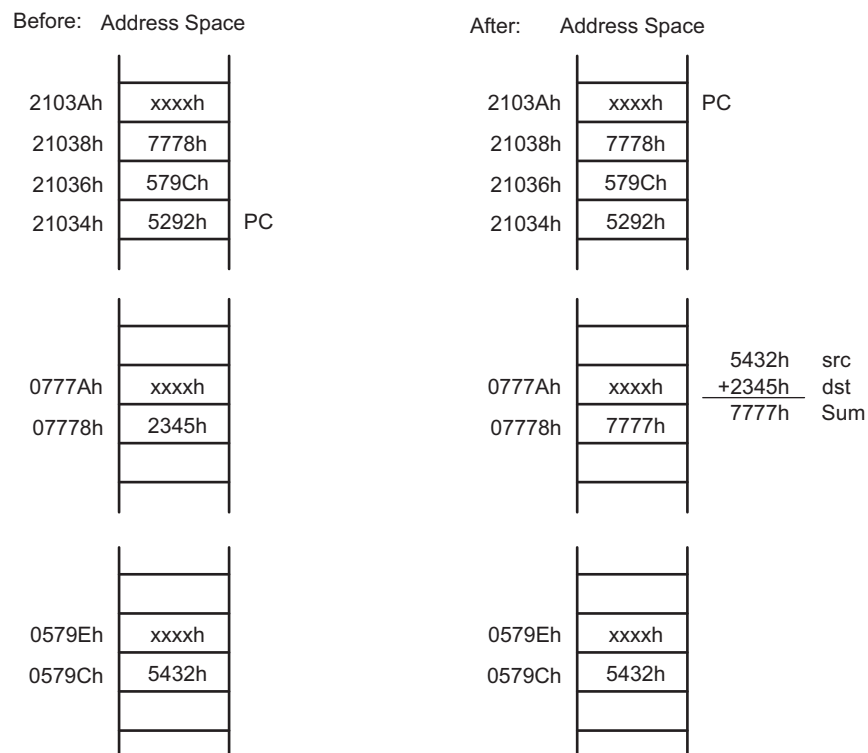
The absolute mode uses the contents of the word following the instruction as the address of the operand. The absolute mode has 2 addressing possibilities:

- Absolute mode in lower 64KB memory
- MSP430X instruction with absolute mode

4.4.4.1 Absolute Mode in Lower 64KB

If an MSP430 instruction is used with absolute addressing mode, the absolute address is a 16-bit value and, therefore, points to an address in the lower 64KB of the memory range. The address is calculated as an index from 0 and is stored in the word following the instruction. The RAM and the peripheral registers can be accessed this way and existing MSP430 software is usable without modifications.

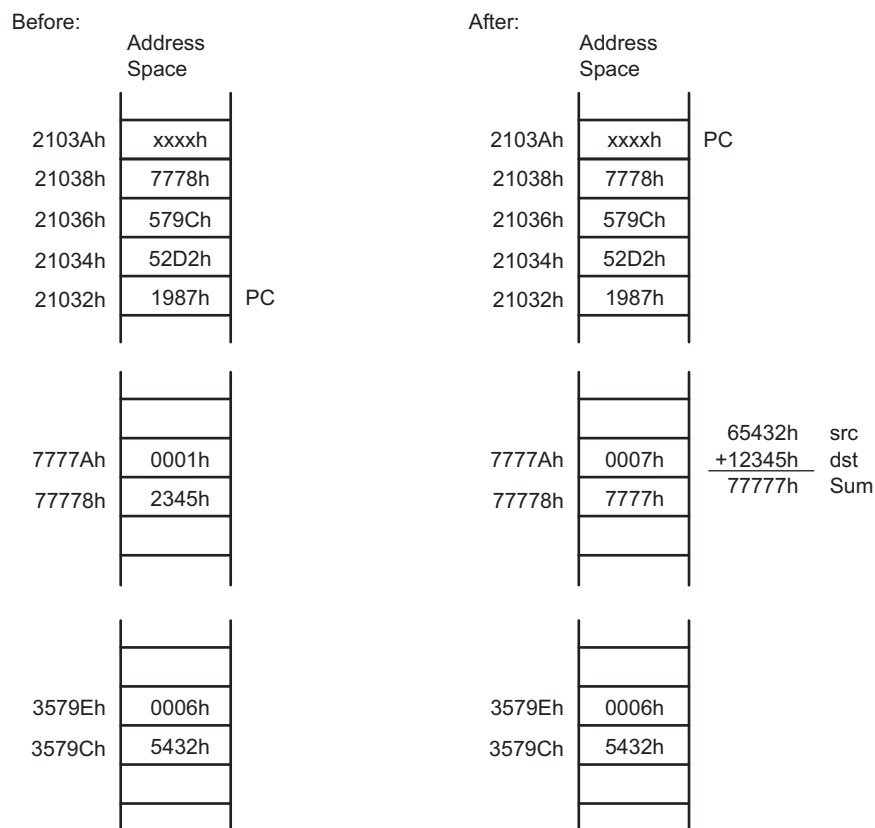
Length: 2 or 3 words
Operation: The operand is the content of the addressed memory location.
Comment: Valid for source and destination. The assembler calculates the index from 0 and inserts it.
Example: `ADD.W &EDE, &TONI ;`
This instruction adds the 16-bit data contained in the absolute source and destination addresses and places the result into the destination.
Source: Word at address EDE
Destination: Word at address TONI



4.4.4.2 MSP430X Instruction With Absolute Mode

If an MSP430X instruction is used with absolute addressing mode, the absolute address is a 20-bit value and, therefore, points to any address in the memory range. The address value is calculated as an index from 0. The 4 MSBs of the index are contained in the extension word, and the 16 LSBs are contained in the word following the instruction.

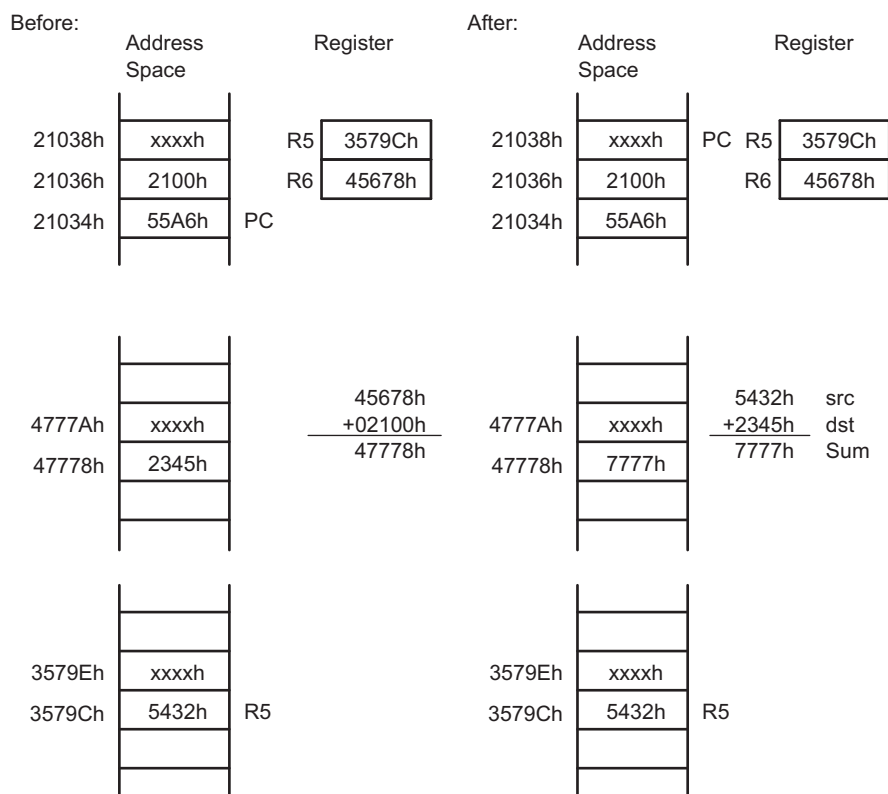
Length: 3 or 4 words
Operation: The operand is the content of the addressed memory location.
Comment: Valid for source and destination. The assembler calculates the index from 0 and inserts it.
Example: `ADDX.A &EDE, &TONI ;`
This instruction adds the 20-bit data contained in the absolute source and destination addresses and places the result into the destination.
Source: 2 words beginning with address EDE
Destination: 2 words beginning with address TONI



4.4.5 Indirect Register Mode

The indirect register mode uses the contents of the CPU register Rsrc as the source operand. The indirect register mode always uses a 20-bit address.

Length:	1, 2, or 3 words
Operation:	The operand is the content the addressed memory location. The source register Rsrc is not modified.
Comment:	Valid only for the source operand. The substitute for the destination operand is 0(Rdst).
Example:	ADDX.W @R5, 2100h(R6) This instruction adds the 2 16-bit operands contained in the source and the destination addresses and places the result into the destination.
Source:	Word pointed to by R5. R5 contains address 3579Ch for this example.
Destination:	Word pointed to by R6 + 2100h, which results in address 45678h + 2100h = 47778h



4.4.6 Indirect Autoincrement Mode

The indirect autoincrement mode uses the contents of the CPU register Rsrc as the source operand. Rsrc is then automatically incremented by 1 for byte instructions, by 2 for word instructions, and by 4 for address-word instructions immediately after accessing the source operand. If the same register is used for source and destination, it contains the incremented address for the destination access. Indirect autoincrement mode always uses 20-bit addresses.

Length: 1, 2, or 3 words

Operation: The operand is the content of the addressed memory location.

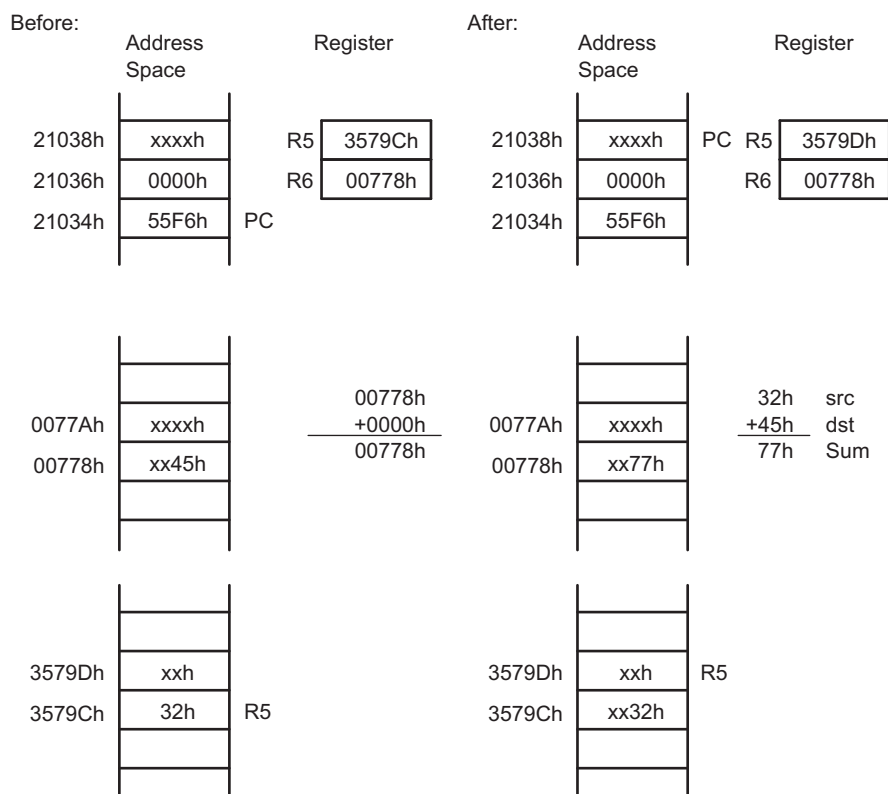
Comment: Valid only for the source operand

Example: `ADD.B @R5+, 0(R6)`

This instruction adds the 8-bit data contained in the source and the destination addresses and places the result into the destination.

Source: Byte pointed to by R5. R5 contains address 3579Ch for this example.

Destination: Byte pointed to by R6 + 0h, which results in address 0778h for this example



4.4.7 Immediate Mode

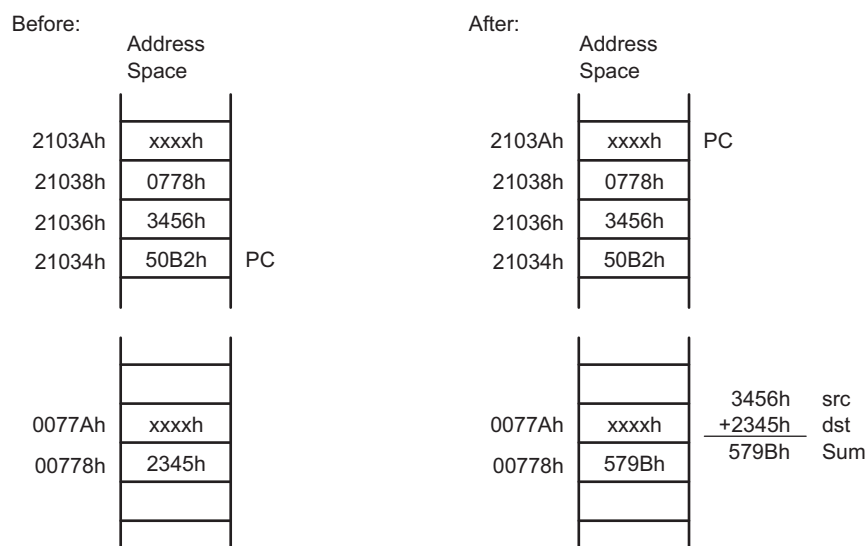
The immediate mode allows accessing constants as operands by including the constant in the memory location following the instruction. The PC is used with the indirect autoincrement mode. The PC points to the immediate value contained in the next word. After the fetching of the immediate operand, the PC is incremented by 2 for byte, word, or address-word instructions. The immediate mode has 2 addressing possibilities:

- 8-bit or 16-bit constants with MSP430 instructions
- 20-bit constants with MSP430X instruction

4.4.7.1 MSP430 Instructions With Immediate Mode

If an MSP430 instruction is used with immediate addressing mode, the constant is an 8- or 16-bit value and is stored in the word following the instruction.

Length:	2 or 3 words. 1 word less if a constant of the constant generator can be used for the immediate operand.
Operation:	The 16-bit immediate source operand is used together with the 16-bit destination operand.
Comment:	Valid only for the source operand
Example:	ADD #3456h, &TONI This instruction adds the 16-bit immediate operand 3456h to the data in the destination address TONI.
Source:	16-bit immediate value 3456h
Destination:	Word at address TONI



4.4.7.2 MSP430X Instructions With Immediate Mode

If an MSP430X instruction is used with immediate addressing mode, the constant is a 20-bit value. The 4 MSBs of the constant are stored in the extension word, and the 16 LSBs of the constant are stored in the word following the instruction.

Length:	3 or 4 words. 1 word less if a constant of the constant generator can be used for the immediate operand.
Operation:	The 20-bit immediate source operand is used together with the 20-bit destination operand.
Comment:	Valid only for the source operand
Example:	ADDX.A #23456h,&TONI ; This instruction adds the 20-bit immediate operand 23456h to the data in the destination address TONI.
Source:	20-bit immediate value 23456h
Destination:	2 words beginning with address TONI

Before:	Address Space	After:	Address Space
2103Ah	xxxxh	2103Ah	xxxxh
21038h	7778h	21038h	7778h
21036h	3456h	21036h	3456h
21034h	50F2h	21034h	50F2h
21032h	1907h	21032h	1907h
7777Ah	0001h	7777Ah	0003h
77778h	2345h	77778h	579Bh

PC

PC

23456h src

+12345h dst

3579Bh Sum

4.5 MSP430 and MSP430X Instructions

MSP430 instructions are the 27 implemented instructions of the MSP430 CPU. These instructions are used throughout the 1MB memory range unless their 16-bit capability is exceeded. The MSP430X instructions are used when the addressing of the operands or the data length exceeds the 16-bit capability of the MSP430 instructions.

There are 3 possibilities when choosing between an MSP430 and MSP430X instruction:

- To use only the MSP430 instructions – The only exceptions are the CALLA and the RETA instruction. This can be done if a few, simple rules are met:
 - Place all constants, variables, arrays, tables, and data in the lower 64KB. This allows the use of MSP430 instructions with 16-bit addressing for all data accesses. No pointers with 20-bit addresses are needed.
 - Place subroutine constants immediately after the subroutine code. This allows the use of the symbolic addressing mode with its 16-bit index to reach addresses within the range of PC + 32KB.
- To use only MSP430X instructions – The disadvantages of this method are the reduced speed due to the additional CPU cycles and the increased program space due to the necessary extension word for any double-operand instruction.
- Use the best fitting instruction where needed.

[Section 4.5.1](#) lists and describes the MSP430 instructions, and [Section 4.5.2](#) lists and describes the MSP430X instructions.

4.5.1 MSP430 Instructions

The MSP430 instructions can be used, regardless if the program resides in the lower 64KB or beyond it. The only exceptions are the instructions CALL and RET, which are limited to the lower 64KB address range. CALLA and RETA instructions have been added to the MSP430X CPU to handle subroutines in the entire address range with no code size overhead.

4.5.1.1 MSP430 Double-Operand (Format I) Instructions

[Figure 4-22](#) shows the format of the MSP430 double-operand instructions. Source and destination words are appended for the indexed, symbolic, absolute, and immediate modes. [Table 4-4](#) lists the 12 MSP430 double-operand instructions.

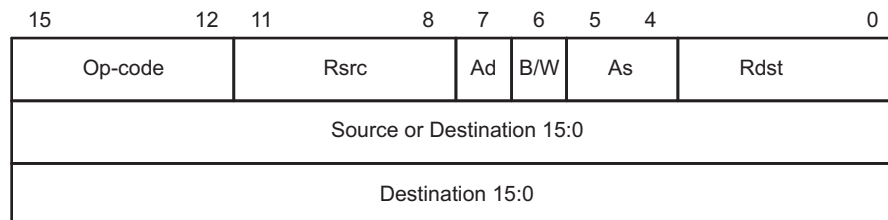


Figure 4-22. MSP430 Double-Operand Instruction Format

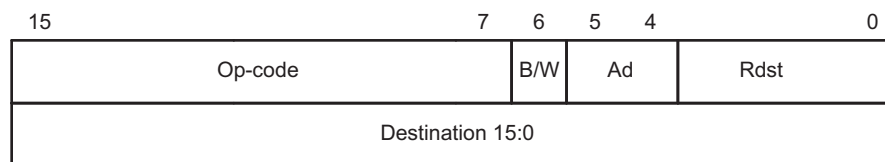
Table 4-4. MSP430 Double-Operand Instructions

Mnemonic	S-Reg, D-Reg	Operation	Status Bits ⁽¹⁾			
			V	N	Z	C
MOV(. B)	src,dst	src → dst	—	—	—	—
ADD(. B)	src,dst	src + dst → dst	*	*	*	*
ADDC(. B)	src,dst	src + dst + C → dst	*	*	*	*
SUB(. B)	src,dst	dst + .not.src + 1 → dst	*	*	*	*
SUBC(. B)	src,dst	dst + .not.src + C → dst	*	*	*	*
CMP(. B)	src,dst	dst - src	*	*	*	*
DADD(. B)	src,dst	src + dst + C → dst (decimally)	*	*	*	*
BIT(. B)	src,dst	src .and. dst	0	*	*	$\bar{Z}$
BIC(. B)	src,dst	.not.src .and. dst → dst	—	—	—	—
BIS(. B)	src,dst	src .or. dst → dst	—	—	—	—
XOR(. B)	src,dst	src .xor. dst → dst	*	*	*	$\bar{Z}$
AND(. B)	src,dst	src .and. dst → dst	0	*	*	Z

(1) * = Status bit is affected.
— = Status bit is not affected.
0 = Status bit is cleared.
1 = Status bit is set.

4.5.1.2 MSP430 Single-Operand (Format II) Instructions

Figure 4-23 shows the format for MSP430 single-operand instructions, except RETI. The destination word is appended for the indexed, symbolic, absolute, and immediate modes. Table 4-5 lists the 7 single-operand instructions.


Figure 4-23. MSP430 Single-Operand Instructions
Table 4-5. MSP430 Single-Operand Instructions

Mnemonic	S-Reg, D-Reg	Operation	Status Bits ⁽¹⁾			
			V	N	Z	C
RRC(. B)	dst	C → MSB → LSB → C	0	*	*	*
RRA(. B)	dst	MSB → MSB → LSB → C	0	*	*	*
PUSH(. B)	src	SP - 2 → SP, src → SP	—	—	—	—
SWPB	dst	bit 15...bit 8 ↔ bit 7...bit 0	—	—	—	—
CALL	dst	Call subroutine in lower 64KB	—	—	—	—
RETI		TOS → SR, SP + 2 → SP	*	*	*	*
		TOS → PC, SP + 2 → SP				
SXT	dst	Register mode: bit 7 → bit 8...bit 19 Other modes: bit 7 → bit 8...bit 15	0	*	*	$\bar{Z}$

(1) * = Status bit is affected.
— = Status bit is not affected.
0 = Status bit is cleared.
1 = Status bit is set.

4.5.1.3 Jump Instructions

Figure 4-24 shows the format for MSP430 and MSP430X jump instructions. The signed 10-bit word offset of the jump instruction is multiplied by 2, sign-extended to a 20-bit address, and added to the 20-bit PC. This allows jumps in a range of –511 to +512 words relative to the PC in the full 20-bit address space. Jumps do not affect the status bits. Table 4-6 lists and describes the 8 jump instructions.

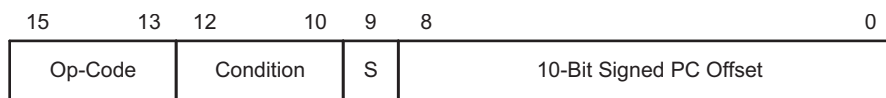


Figure 4-24. Format of Conditional Jump Instructions

Table 4-6. Conditional Jump Instructions

Mnemonic	S-Reg, D-Reg	Operation
JEQ, JZ	Label	Jump to label if zero bit is set
JNE, JNZ	Label	Jump to label if zero bit is reset
JC	Label	Jump to label if carry bit is set
JNC	Label	Jump to label if carry bit is reset
JN	Label	Jump to label if negative bit is set
JGE	Label	Jump to label if (N .XOR. V) = 0
JL	Label	Jump to label if (N .XOR. V) = 1
JMP	Label	Jump to label unconditionally

4.5.1.4 Emulated Instructions

In addition to the MSP430 and MSP430X instructions, emulated instructions are instructions that make code easier to write and read, but do not have op-codes themselves. Instead, they are replaced automatically by the assembler with a core instruction. There is no code or performance penalty for using emulated instructions. The emulated instructions are listed in Table 4-7.

Table 4-7. Emulated Instructions

Instruction	Explanation	Emulation	Status Bits ⁽¹⁾			
			V	N	Z	C
ADC(.B) dst	Add Carry to dst	ADDC(.B) #0, dst	*	*	*	*
BR dst	Branch indirectly dst	MOV dst, PC	–	–	–	–
CLR(.B) dst	Clear dst	MOV(.B) #0, dst	–	–	–	–
CLRC	Clear Carry bit	BIC #1, SR	–	–	–	0
CLRN	Clear Negative bit	BIC #4, SR	–	0	–	–
CLRZ	Clear Zero bit	BIC #2, SR	–	–	0	–
DADC(.B) dst	Add Carry to dst decimally	DADD(.B) #0, dst	*	*	*	*
DEC(.B) dst	Decrement dst by 1	SUB(.B) #1, dst	*	*	*	*
DECD(.B) dst	Decrement dst by 2	SUB(.B) #2, dst	*	*	*	*
DINT	Disable interrupt	BIC #8, SR	–	–	–	–
EINT	Enable interrupt	BIS #8, SR	–	–	–	–
INC(.B) dst	Increment dst by 1	ADD(.B) #1, dst	*	*	*	*
INCD(.B) dst	Increment dst by 2	ADD(.B) #2, dst	*	*	*	*

⁽¹⁾ * = Status bit is affected
– = Status bit is not affected
0 = Status bit is cleared
1 = Status bit is set

Table 4-7. Emulated Instructions (continued)

Instruction	Explanation	Emulation	Status Bits ⁽¹⁾			
			V	N	Z	C
INV(.B) dst	Invert dst	XOR(.B) #-1, dst	*	*	*	*
NOP	No operation	MOV R3, R3	—	—	—	—
POP dst	Pop operand from stack	MOV @SP+, dst	—	—	—	—
RET	Return from subroutine	MOV @SP+, PC	—	—	—	—
RLA(.B) dst	Shift left dst arithmetically	ADD(.B) dst, dst	*	*	*	*
RLC(.B) dst	Shift left dst logically through Carry	ADDC(.B) dst, dst	*	*	*	*
SBC(.B) dst	Subtract Carry from dst	SUBC(.B) #0, dst	*	*	*	*
SETC	Set Carry bit	BIS #1, SR	—	—	—	1
SETN	Set Negative bit	BIS #4, SR	—	1	—	—
SETZ	Set Zero bit	BIS #2, SR	—	—	1	—
TST(.B) dst	Test dst (compare with 0)	CMP(.B) #0, dst	0	*	*	1

4.5.1.5 MSP430 Instruction Execution

The number of CPU clock cycles required for an instruction depends on the instruction format and the addressing modes used – not the instruction itself. The number of clock cycles refers to MCLK.

4.5.1.5.1 Instruction Cycles and Length for Interrupt, Reset, and Subroutines

Table 4-8 lists the length and the CPU cycles for reset, interrupts, and subroutines.

Table 4-8. Interrupt, Return, and Reset Cycles and Length

Action	Execution Time (MCLK Cycles)	Length of Instruction (Words)
Return from interrupt RETI	5	1
Return from subroutine RET	4	1
Interrupt request service (cycles needed before first instruction)	6	—
WDT reset	4	—
Reset (RST/NMI)	4	—

4.5.1.5.2 Format II (Single-Operand) Instruction Cycles and Lengths

Table 4-9 lists the length and the CPU cycles for all addressing modes of the MSP430 single-operand instructions.

Table 4-9. MSP430 Format II Instruction Cycles and Length

Addressing Mode	No. of Cycles			Length of Instruction	Example
	RRA, RRC SWPB, SXT	PUSH	CALL		
Rn	1	3	4	1	SWPB R5
@Rn	3	3	4	1	RRC @R9
@Rn+	3	3	4	1	SWPB @R10+
#N	N/A	3	4	2	CALL #LABEL
X(Rn)	4	4	5	2	CALL 2(R7)
EDE	4	4	5	2	PUSH EDE
&EDE	4	4	6	2	SXT &EDE

4.5.1.5.3 Jump Instructions Cycles and Lengths

All jump instructions require 1 code word and take 2 CPU cycles to execute, regardless of whether the jump is taken or not.

4.5.1.5.4 Format I (Double-Operand) Instruction Cycles and Lengths

Table 4-10 lists the length and CPU cycles for all addressing modes of the MSP430 Format I instructions.

Table 4-10. MSP430 Format I Instructions Cycles and Length

Addressing Mode		No. of Cycles	Length of Instruction	Example
Source	Destination			
Rn	Rm	1	1	MOV R5, R8
	PC	3	1	BR R9
	x(Rm)	4 ⁽¹⁾	2	ADD R5, 4(R6)
	EDE	4 ⁽¹⁾	2	XOR R8, EDE
	&EDE	4 ⁽¹⁾	2	MOV R5, &EDE
@Rn	Rm	2	1	AND @R4, R5
	PC	4	1	BR @R8
	x(Rm)	5 ⁽¹⁾	2	XOR @R5, 8(R6)
	EDE	5 ⁽¹⁾	2	MOV @R5, EDE
	&EDE	5 ⁽¹⁾	2	XOR @R5, &EDE
@Rn+	Rm	2	1	ADD @R5+, R6
	PC	4	1	BR @R9+
	x(Rm)	5 ⁽¹⁾	2	XOR @R5, 8(R6)
	EDE	5 ⁽¹⁾	2	MOV @R9+, EDE
	&EDE	5 ⁽¹⁾	2	MOV @R9+, &EDE
#N	Rm	2	2	MOV #20, R9
	PC	3	2	BR #2AEh
	x(Rm)	5 ⁽¹⁾	3	MOV #0300h, 0(SP)
	EDE	5 ⁽¹⁾	3	ADD #33, EDE
	&EDE	5 ⁽¹⁾	3	ADD #33, &EDE
x(Rn)	Rm	3	2	MOV 2(R5), R7
	PC	5	2	BR 2(R6)
	TONI	6 ⁽¹⁾	3	MOV 4(R7), TONI
	x(Rm)	6 ⁽¹⁾	3	ADD 4(R4), 6(R9)
	&TONI	6 ⁽¹⁾	3	MOV 2(R4), &TONI
EDE	Rm	3	2	AND EDE, R6
	PC	5	2	BR EDE
	TONI	6 ⁽¹⁾	3	CMP EDE, TONI
	x(Rm)	6 ⁽¹⁾	3	MOV EDE, 0(SP)
	&TONI	6 ⁽¹⁾	3	MOV EDE, &TONI
&EDE	Rm	3	2	MOV &EDE, R8
	PC	5	2	BR &EDE
	TONI	6 ⁽¹⁾	3	MOV &EDE, TONI
	x(Rm)	6 ⁽¹⁾	3	MOV &EDE, 0(SP)
	&TONI	6 ⁽¹⁾	3	MOV &EDE, &TONI

⁽¹⁾ MOV, BIT, and CMP instructions execute in 1 fewer cycle.

4.5.2 MSP430X Extended Instructions

The extended MSP430X instructions give the MSP430X CPU full access to its 20-bit address space. Most MSP430X instructions require an additional word of op-code called the extension word. Some extended instructions do not require an additional word and are noted in the instruction description. All addresses, indexes, and immediate numbers have 20-bit values when preceded by the extension word.

There are 2 types of extension words:

- Register or register mode for Format I instructions and register mode for Format II instructions
- Extension word for all other address mode combinations

4.5.2.1 Register Mode Extension Word

The register mode extension word is shown in [Figure 4-25](#) and described in [Table 4-11](#). An example is shown in [Figure 4-27](#).

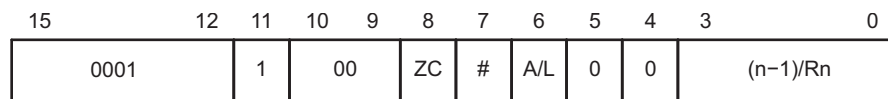


Figure 4-25. Extension Word for Register Modes

Table 4-11. Description of the Extension Word Bits for Register Mode

Bit	Description		
15:11	Extension word op-code. Op-codes 1800h to 1FFFh are extension words.		
10:9	Reserved		
ZC	Zero carry		
	0	The executed instruction uses the status of the carry bit C.	
	1	The executed instruction uses the carry bit as 0. The carry bit is defined by the result of the final operation after instruction execution.	
#	Repetition		
	0	The number of instruction repetitions is set by extension word bits 3:0.	
	1	The number of instruction repetitions is defined by the value of the 4 LSBs of Rn. See description for bits 3:0.	
A/L	Data length extension. Together with the B/W bits of the following MSP430 instruction, the AL bit defines the used data length of the instruction.		
	A/L	B/W	Comment
	0	0	Reserved
	0	1	20-bit address word
	1	0	16-bit word
	1	1	8-bit byte
5:4	Reserved		
3:0	Repetition count		
	# = 0	These 4 bits set the repetition count n. These bits contain n – 1.	
	# = 1	These 4 bits define the CPU register whose bits 3:0 set the number of repetitions. Rn.3:0 contain n – 1.	

4.5.2.2 Non-Register Mode Extension Word

The extension word for non-register modes is shown in [Figure 4-26](#) and described in [Table 4-12](#). An example is shown in [Figure 4-28](#).

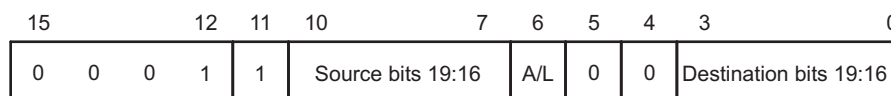


Figure 4-26. Extension Word for Non-Register Modes

Table 4-12. Description of Extension Word Bits for Non-Register Modes

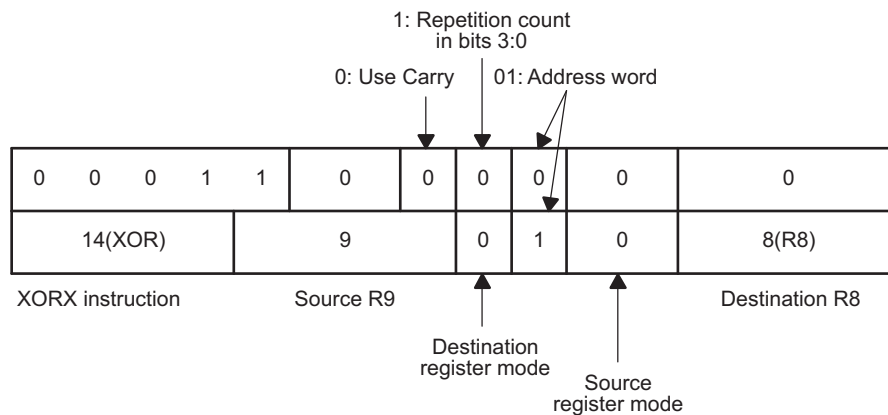
Bit	Description
15:11	Extension word op-code. Op-codes 1800h to 1FFFh are extension words.
Source Bits 19:16	The 4 MSBs of the 20-bit source. Depending on the source addressing mode, these 4 MSBs may belong to an immediate operand, an index, or to an absolute address.
A/L	Data length extension. Together with the B/W bits of the following MSP430 instruction, the AL bit defines the used data length of the instruction.
	A/L B/W Comment
	0 0 Reserved
	0 1 20-bit address word
	1 0 16-bit word
	1 1 8-bit byte
5:4	Reserved
Destination Bits 19:16	The 4 MSBs of the 20-bit destination. Depending on the destination addressing mode, these 4 MSBs may belong to an index or to an absolute address.

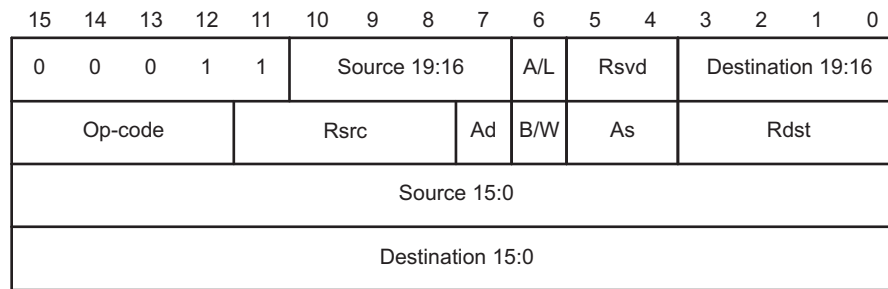
NOTE: B/W and A/L bit settings for SWPBX and SXTX

A/L	B/W	
0	0	SWPBX.A, SXTX.A
0	1	N/A
1	0	SWPB.W, SXTX.W
1	1	N/A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	00	ZC	#	A/L	Rsvd	(n-1)/Rn					
Op-code					Rsrc			Ad	B/W	As	Rdst				

XORX.A R9, R8


Figure 4-27. Example for Extended Register or Register Instruction



XORX.A #12345h, 45678h (R15)

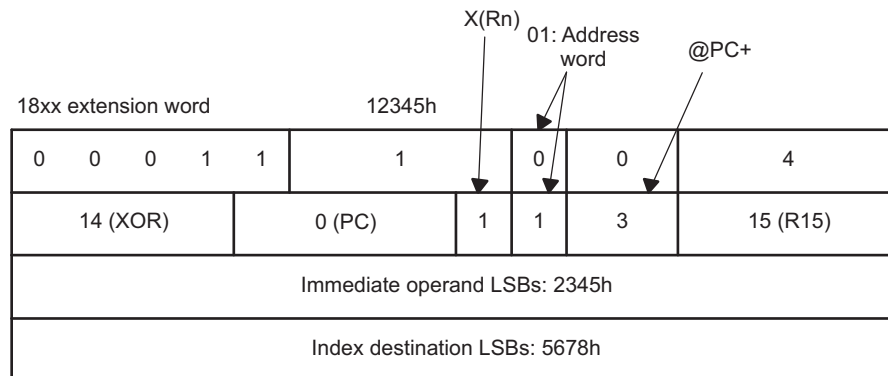


Figure 4-28. Example for Extended Immediate or Indexed Instruction

4.5.2.3 Extended Double-Operand (Format I) Instructions

All 12 double-operand instructions have extended versions as listed in [Table 4-13](#).

Table 4-13. Extended Double-Operand Instructions

Mnemonic	Operands	Operation	Status Bits ⁽¹⁾			
			V	N	Z	C
MOVX(.B, .A)	src,dst	src → dst	—	—	—	—
ADDX(.B, .A)	src,dst	src + dst → dst	*	*	*	*
ADDCX(.B, .A)	src,dst	src + dst + C → dst	*	*	*	*
SUBX(.B, .A)	src,dst	dst + .not.src + 1 → dst	*	*	*	*
SUBCX(.B, .A)	src,dst	dst + .not.src + C → dst	*	*	*	*
CMPX(.B, .A)	src,dst	dst – src	*	*	*	*
DADDX(.B, .A)	src,dst	src + dst + C → dst (decimal)	*	*	*	*
BITX(.B, .A)	src,dst	src .and. dst	0	*	*	Z
BICX(.B, .A)	src,dst	.not.src .and. dst → dst	—	—	—	—
BISX(.B, .A)	src,dst	src .or. dst → dst	—	—	—	—
XORX(.B, .A)	src,dst	src .xor. dst → dst	*	*	*	Z
ANDX(.B, .A)	src,dst	src .and. dst → dst	0	*	*	Z

⁽¹⁾ * = Status bit is affected
 — = Status bit is not affected
 0 = Status bit is cleared
 1 = Status bit is set

Figure 4-29 shows the possible addressing combinations for the extension word for Format I instructions.

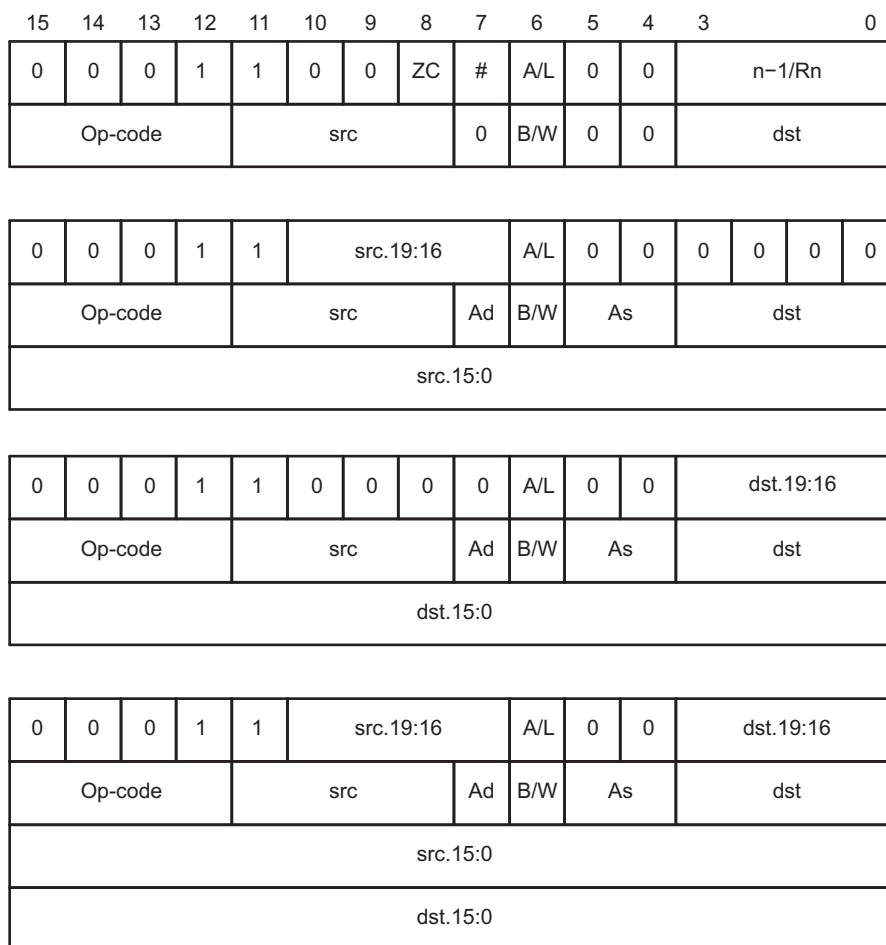


Figure 4-29. Extended Format I Instruction Formats

If the 20-bit address of a source or destination operand is located in memory, not in a CPU register, then 2 words are used for this operand (see Figure 4-30).

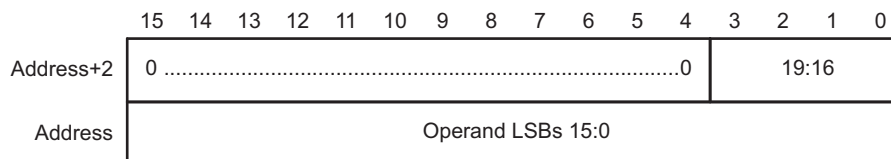


Figure 4-30. 20-Bit Addresses in Memory

4.5.2.4 Extended Single-Operand (Format II) Instructions

Table 4-14 lists the extended MSP430X Format II instructions.

Table 4-14. Extended Single-Operand Instructions

Mnemonic	Operands	Operation	Status Bits ⁽¹⁾				
			n	V	N	Z	C
CALLA	dst	Call indirect to subroutine (20-bit address)		–	–	–	–
POPM .A	#n,Rdst	Pop n 20-bit registers from stack	1 to 16	–	–	–	–
POPM .W	#n,Rdst	Pop n 16-bit registers from stack	1 to 16	–	–	–	–
PUSHM .A	#n,Rsrc	Push n 20-bit registers to stack	1 to 16	–	–	–	–
PUSHM .W	#n,Rsrc	Push n 16-bit registers to stack	1 to 16	–	–	–	–
PUSHX(.B, .A)	src	Push 8-, 16-, or 20-bit source to stack		–	–	–	–
RRCM(.A)	#n,Rdst	Rotate right Rdst n bits through carry (16-, 20-bit register)	1 to 4	0	*	*	*
RRUM(.A)	#n,Rdst	Rotate right Rdst n bits unsigned (16-, 20-bit register)	1 to 4	0	*	*	*
RRAM(.A)	#n,Rdst	Rotate right Rdst n bits arithmetically (16-, 20-bit register)	1 to 4	0	*	*	*
RLAM(.A)	#n,Rdst	Rotate left Rdst n bits arithmetically (16-, 20-bit register)	1 to 4	*	*	*	*
RRCX(.B, .A)	dst	Rotate right dst through carry (8-, 16-, 20-bit data)	1	0	*	*	*
RRUX(.B, .A)	Rdst	Rotate right dst unsigned (8-, 16-, 20-bit)	1	0	*	*	*
RRAX(.B, .A)	dst	Rotate right dst arithmetically	1	0	*	*	*
SWPBX(.A)	dst	Exchange low byte with high byte	1	–	–	–	–
SCTX(.A)	Rdst	Bit7 → bit8 ... bit19	1	0	*	*	Z
SCTX(.A)	dst	Bit7 → bit8 ... MSB	1	0	*	*	Z

(1) * = Status bit is affected
– = Status bit is not affected
0 = Status bit is cleared
1 = Status bit is set

Figure 4-31 shows the addressing mode combinations for Format II instructions.

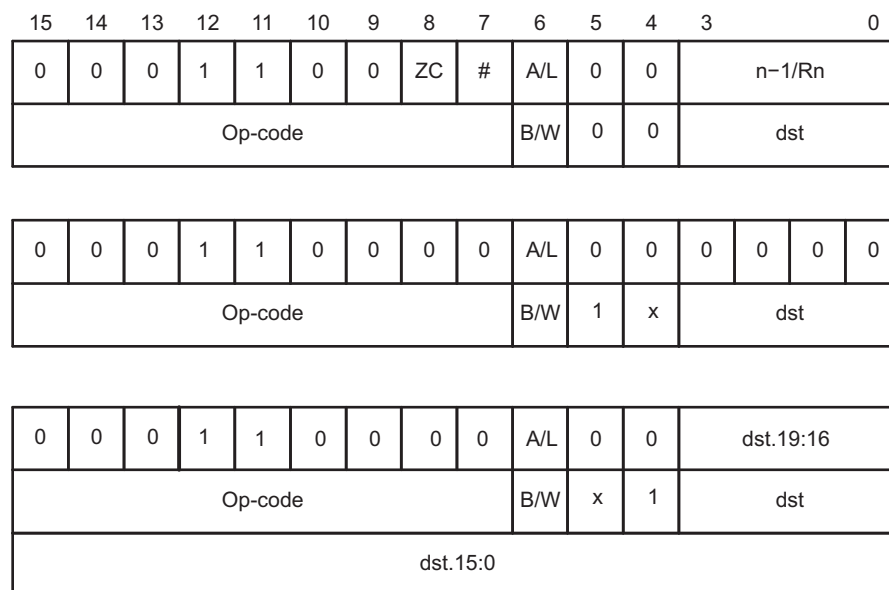


Figure 4-31. Extended Format II Instruction Format

4.5.2.4.1 Extended Format II Instruction Format Exceptions

Exceptions for the Format II instruction formats are shown in [Figure 4-32](#) through [Figure 4-35](#).

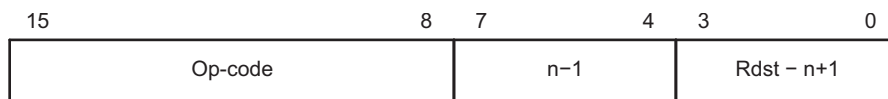


Figure 4-32. PUSHM and POPM Instruction Format

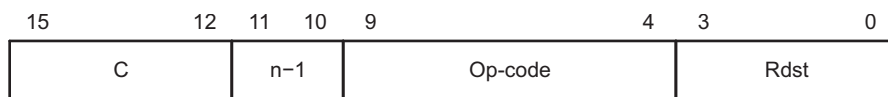


Figure 4-33. RRCM, RRAM, RRUM, and RLAM Instruction Format

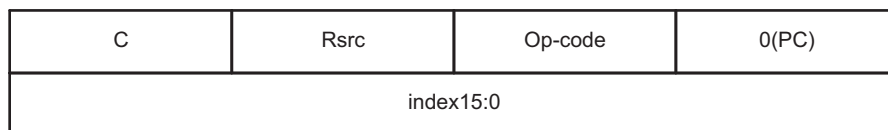
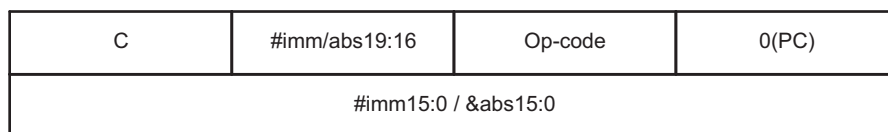
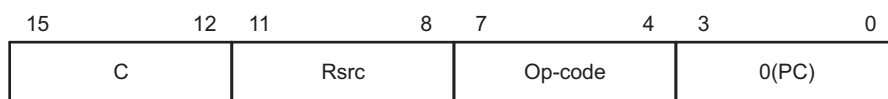


Figure 4-34. BRA Instruction Format

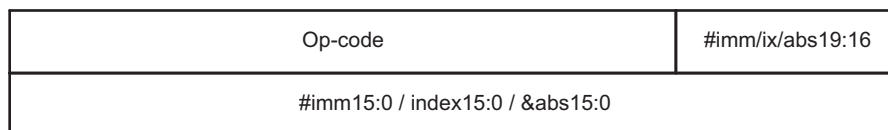
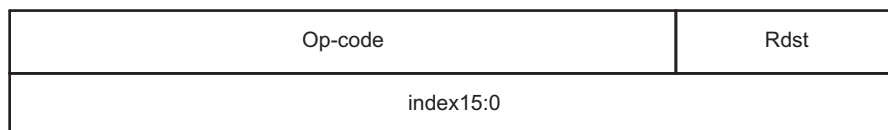
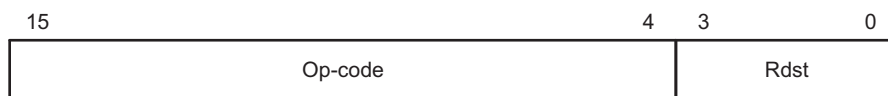


Figure 4-35. CALLA Instruction Format

4.5.2.5 Extended Emulated Instructions

The extended instructions together with the constant generator form the extended emulated instructions. [Table 4-15](#) lists the emulated instructions.

Table 4-15. Extended Emulated Instructions

Instruction	Explanation	Emulation
ADCX(.B, .A) dst	Add Carry to dst	ADDCX(.B, .A) #0, dst
BRA dst	Branch indirect dst	MOVA dst, PC
RETA	Return from subroutine	MOVA @SP+, PC
CLRA Rdst	Clear Rdst	MOV #0, Rdst
CLR(.B, .A) dst	Clear dst	MOVX(.B, .A) #0, dst
DADCX(.B, .A) dst	Add carry to dst decimally	DADDX(.B, .A) #0, dst
DECX(.B, .A) dst	Decrement dst by 1	SUBX(.B, .A) #1, dst
DECDA Rdst	Decrement Rdst by 2	SUBA #2, Rdst
DECDX(.B, .A) dst	Decrement dst by 2	SUBX(.B, .A) #2, dst
INCX(.B, .A) dst	Increment dst by 1	ADDX(.B, .A) #1, dst
INCDA Rdst	Increment Rdst by 2	ADDA #2, Rdst
INCDX(.B, .A) dst	Increment dst by 2	ADDX(.B, .A) #2, dst
INVX(.B, .A) dst	Invert dst	XORX(.B, .A) #-1, dst
RLAX(.B, .A) dst	Shift left dst arithmetically	ADDX(.B, .A) dst, dst
RLCX(.B, .A) dst	Shift left dst logically through carry	ADDCX(.B, .A) dst, dst
SBCX(.B, .A) dst	Subtract carry from dst	SUBCX(.B, .A) #0, dst
TSTA Rdst	Test Rdst (compare with 0)	CMPA #0, Rdst
TSTX(.B, .A) dst	Test dst (compare with 0)	CMPX(.B, .A) #0, dst
POPX dst	Pop to dst	MOVX(.B, .A) @SP+, dst

4.5.2.6 MSP430X Address Instructions

MSP430X address instructions are instructions that support 20-bit operands but have restricted addressing modes. The addressing modes are restricted to the register mode and the immediate mode, except for the MOVA instruction as listed in [Table 4-16](#). Restricting the addressing modes removes the need for the additional extension-word op-code improving code density and execution time. Address instructions should be used any time an MSP430X instruction is needed with the corresponding restricted addressing mode.

Table 4-16. Address Instructions, Operate on 20-Bit Register Data

Mnemonic	Operands	Operation	Status Bits ⁽¹⁾			
			V	N	Z	C
ADDA	Rsrc, Rdst	Add source to destination register	*	*	*	*
	#imm20, Rdst					
MOVA	Rsrc, Rdst	Move source to destination	–	–	–	–
	#imm20, Rdst					
	z16(Rsrc), Rdst					
	EDE, Rdst					
	&abs20, Rdst					
	@Rsrc, Rdst					
	@Rsrc+, Rdst					
	Rsrc, z16(Rdst)					
CMPA	Rsrc, Rdst	Compare source to destination register	*	*	*	*
	#imm20, Rdst					
SUBA	Rsrc, Rdst	Subtract source from destination register	*	*	*	*
	#imm20, Rdst					

- ⁽¹⁾ * = Status bit is affected
– = Status bit is not affected
0 = Status bit is cleared
1 = Status bit is set

4.5.2.7 MSP430X Instruction Execution

The number of CPU clock cycles required for an MSP430X instruction depends on the instruction format and the addressing modes used, not the instruction itself. The number of clock cycles refers to MCLK.

4.5.2.7.1 MSP430X Format II (Single-Operand) Instruction Cycles and Lengths

Table 4-17 lists the length and the CPU cycles for all addressing modes of the MSP430X extended single-operand instructions.

Table 4-17. MSP430X Format II Instruction Cycles and Length

Instruction	Execution Cycles, Length of Instruction (Words)						
	Rn	@Rn	@Rn+	#N	X(Rn)	EDE	&EDE
RRAM	n, 1	—	—	—	—	—	—
RRCM	n, 1	—	—	—	—	—	—
RRUM	n, 1	—	—	—	—	—	—
RLAM	n, 1	—	—	—	—	—	—
PUSHM	2+n, 1	—	—	—	—	—	—
PUSHM.A	2+2n, 1	—	—	—	—	—	—
POPM	2+n, 1	—	—	—	—	—	—
POPM.A	2+2n, 1	—	—	—	—	—	—
CALLA	5, 1	6, 1	6, 1	5, 2	5 ⁽¹⁾ , 2	7, 2	7, 2
RRAX(.B)	1+n, 2	4, 2	4, 2	—	5, 3	5, 3	5, 3
RRAX.A	1+n, 2	6, 2	6, 2	—	7, 3	7, 3	7, 3
RRCX(.B)	1+n, 2	4, 2	4, 2	—	5, 3	5, 3	5, 3
RRCX.A	1+n, 2	6, 2	6, 2	—	7, 3	7, 3	7, 3
PUSHX(.B)	4, 2	4, 2	4, 2	4, 3	5 ⁽¹⁾ , 3	5, 3	5, 3
PUSHX.A	5, 2	6, 2	6, 2	5, 3	7 ⁽¹⁾ , 3	7, 3	7, 3
POPX(.B)	3, 2	—	—	—	5, 3	5, 3	5, 3
POPX.A	4, 2	—	—	—	7, 3	7, 3	7, 3

⁽¹⁾ Add 1 cycle when Rn = SP

4.5.2.7.2 MSP430X Format I (Double-Operand) Instruction Cycles and Lengths

Table 4-18 lists the length and CPU cycles for all addressing modes of the MSP430X extended Format I instructions.

Table 4-18. MSP430X Format I Instruction Cycles and Length

Addressing Mode		No. of Cycles		Length of Instruction	Examples
Source	Destination	.B/.W	.A	.B/.W/.A	
Rn	Rm ⁽¹⁾	2	2	2	BITX.B R5, R8
	PC	4	4	2	ADDX R9, PC
	x(Rm)	5 ⁽²⁾	7 ⁽³⁾	3	ANDX.A R5, 4(R6)
	EDE	5 ⁽²⁾	7 ⁽³⁾	3	XORX R8, EDE
	&EDE	5 ⁽²⁾	7 ⁽³⁾	3	BITX.W R5, &EDE
@Rn	Rm	3	4	2	BITX @R5, R8
	PC	5	6	2	ADDX @R9, PC
	x(Rm)	6 ⁽²⁾	9 ⁽³⁾	3	ANDX.A @R5, 4(R6)
	EDE	6 ⁽²⁾	9 ⁽³⁾	3	XORX @R8, EDE
	&EDE	6 ⁽²⁾	9 ⁽³⁾	3	BITX.B @R5, &EDE
@Rn+	Rm	3	4	2	BITX @R5+, R8
	PC	5	6	2	ADDX.A @R9+, PC
	x(Rm)	6 ⁽²⁾	9 ⁽³⁾	3	ANDX @R5+, 4(R6)
	EDE	6 ⁽²⁾	9 ⁽³⁾	3	XORX.B @R8+, EDE
	&EDE	6 ⁽²⁾	9 ⁽³⁾	3	BITX @R5+, &EDE
#N	Rm	3	3	3	BITX #20, R8
	PC ⁽⁴⁾	4	4	3	ADDX.A #FE000h, PC
	x(Rm)	6 ⁽²⁾	8 ⁽³⁾	4	ANDX #1234, 4(R6)
	EDE	6 ⁽²⁾	8 ⁽³⁾	4	XORX #A5A5h, EDE
	&EDE	6 ⁽²⁾	8 ⁽³⁾	4	BITX.B #12, &EDE
x(Rn)	Rm	4	5	3	BITX 2(R5), R8
	PC ⁽⁴⁾	6	7	3	SUBX.A 2(R6), PC
	TONI	7 ⁽²⁾	10 ⁽³⁾	4	ANDX 4(R7), 4(R6)
	x(Rm)	7 ⁽²⁾	10 ⁽³⁾	4	XORX.B 2(R6), EDE
	&TONI	7 ⁽²⁾	10 ⁽³⁾	4	BITX 8(SP), &EDE
EDE	Rm	4	5	3	BITX.B EDE, R8
	PC ⁽⁴⁾	6	7	3	ADDX.A EDE, PC
	TONI	7 ⁽²⁾	10 ⁽³⁾	4	ANDX EDE, 4(R6)
	x(Rm)	7 ⁽²⁾	10 ⁽³⁾	4	ANDX EDE, TONI
	&TONI	7 ⁽²⁾	10 ⁽³⁾	4	BITX EDE, &TONI
&EDE	Rm	4	5	3	BITX &EDE, R8
	PC ⁽⁴⁾	6	7	3	ADDX.A &EDE, PC
	TONI	7 ⁽²⁾	10 ⁽³⁾	4	ANDX.B &EDE, 4(R6)
	x(Rm)	7 ⁽²⁾	10 ⁽³⁾	4	XORX &EDE, TONI
	&TONI	7 ⁽²⁾	10 ⁽³⁾	4	BITX &EDE, &TONI

⁽¹⁾ Repeat instructions require n + 1 cycles, where n is the number of times the instruction is executed.

⁽²⁾ Reduce the cycle count by 1 for MOV, BIT, and CMP instructions.

⁽³⁾ Reduce the cycle count by 2 for MOV, BIT, and CMP instructions.

⁽⁴⁾ Reduce the cycle count by 1 for MOV, ADD, and SUB instructions.

4.5.2.7.3 MSP430X Address Instruction Cycles and Lengths

Table 4-19 lists the length and the CPU cycles for all addressing modes of the MSP430X address instructions.

Table 4-19. Address Instruction Cycles and Length

Addressing Mode		Execution Time (MCLK Cycles)		Length of Instruction (Words)		Example
Source	Destination	MOVA BRA	CMPA ADDA SUBA	MOVA	CMPA ADDA SUBA	
Rn	Rn	1	1	1	1	CMPA R5, R8
	PC	3	3	1	1	SUBA R9, PC
	x(Rm)	4	–	2	–	MOVA R5, 4(R6)
	EDE	4	–	2	–	MOVA R8, EDE
	&EDE	4	–	2	–	MOVA R5, &EDE
@Rn	Rm	3	–	1	–	MOVA @R5, R8
	PC	5	–	1	–	MOVA @R9, PC
@Rn+	Rm	3	–	1	–	MOVA @R5+, R8
	PC	5	–	1	–	MOVA @R9+, PC
#N	Rm	2	3	2	2	CMPA #20, R8
	PC	3	3	2	2	SUBA #FE000h, PC
x(Rn)	Rm	4	–	2	–	MOVA 2(R5), R8
	PC	6	–	2	–	MOVA 2(R6), PC
EDE	Rm	4	–	2	–	MOVA EDE, R8
	PC	6	–	2	–	MOVA EDE, PC
&EDE	Rm	4	–	2	–	MOVA &EDE, R8
	PC	6	–	2	–	MOVA &EDE, PC

4.6 Instruction Set Description

Table 4-20 shows all available instructions:

Table 4-20. Instruction Map of MSP430X

	000	040	080	0C0	100	140	180	1C0	200	240	280	2C0	300	340	380	3C0
0xxx	MOVA, CMPA, ADDA, SUBA, RRCM, RRAM, RLAM, RRUM															
10xx	RRC	RRC. B	SWP B		RRA	RRA. B	SXT		PUS H	PUS H.B	CALL		RETI	CALL A		
14xx	PUSHM.A, POPM.A, PUSHM.W, POPM.W															
18xx	Extension word for Format I and Format II instructions															
1Cxx																
20xx	JNE, JNZ															
24xx	JEQ, JZ															
28xx	JNC															
2Cxx	JC															
30xx	JN															
34xx	JGE															
38xx	JL															
3Cxx	JMP															
4xxx	MOV, MOV.B															
5xxx	ADD, ADD.B															
6xxx	ADDC, ADDC.B															
7xxx	SUBC, SUBC.B															
8xxx	SUB, SUB.B															
9xxx	CMP, CMP.B															
Axxx	DADD, DADD.B															
Bxxx	BIT, BIT.B															
Cxxx	BIC, BIC.B															
Dxxx	BIS, BIS.B															
Exxx	XOR, XOR.B															
Fxxx	AND, AND.B															

4.6.1 Extended Instruction Binary Descriptions

Detailed MSP430X instruction binary descriptions are shown in the following tables.

Instruction	Instruction Group				src or data.19:16				Instruction Identifier				dst				
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
MOVA	0	0	0	0	src				0	0	0	0	dst				MOVA @Rsrc,Rdst
	0	0	0	0	src				0	0	0	1	dst				MOVA @Rsrc+,Rdst
	0	0	0	0	&abs.19:16				0	0	1	0	dst				MOVA &abs20,Rdst
	&abs.15:0																
	0	0	0	0	src				0	0	1	1	dst				MOVA z16(Rsrc),Rdst
	x.15:0																
	0	0	0	0	src				0	1	1	0	&abs.19:16				MOVA Rsrc,&abs20
	&abs.15:0																
	0	0	0	0	src				0	1	1	1	dst				MOVA Rsrc,z16(Rdst)
	x.15:0																
CMPA	0	0	0	0	imm.19:16				1	0	0	1	dst				CMPA #imm20,Rdst
	imm.15:0																
ADDA	0	0	0	0	imm.19:16				1	0	1	0	dst				ADDA #imm20,Rdst
	imm.15:0																
SUBA	0	0	0	0	imm.19:16				1	0	1	1	dst				SUBA #imm20,Rdst
	imm.15:0																
MOVA	0	0	0	0	src				1	1	0	0	dst				MOVA Rsrc,Rdst
CMPA	0	0	0	0	src				1	1	0	1	dst				CMPA Rsrc,Rdst
ADDA	0	0	0	0	src				1	1	1	0	dst				ADDA Rsrc,Rdst
SUBA	0	0	0	0	src				1	1	1	1	dst				SUBA Rsrc,Rdst

Instruction	Instruction Group				Bit Loc.		Inst. ID		Instruction Identifier				dst				
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RRCM.A	0	0	0	0	n – 1		0	0	0	1	0	0	dst				RRCM.A #n,Rdst
RRAM.A	0	0	0	0	n – 1		0	1	0	1	0	0	dst				RRAM.A #n,Rdst
RLAM.A	0	0	0	0	n – 1		1	0	0	1	0	0	dst				RLAM.A #n,Rdst
RRUM.A	0	0	0	0	n – 1		1	1	0	1	0	0	dst				RRUM.A #n,Rdst
RRCM.W	0	0	0	0	n – 1		0	0	0	1	0	1	dst				RRCM.W #n,Rdst
RRAM.W	0	0	0	0	n – 1		0	1	0	1	0	1	dst				RRAM.W #n,Rdst
RLAM.W	0	0	0	0	n – 1		1	0	0	1	0	1	dst				RLAM.W #n,Rdst
RRUM.W	0	0	0	0	n – 1		1	1	0	1	0	1	dst				RRUM.W #n,Rdst

Instruction	Instruction Identifier												dst					
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
RETI	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0		
CALLA	0	0	0	1	0	0	1	1	0	1	0	0	dst			CALLA Rdst		
	0	0	0	1	0	0	1	1	0	1	0	1	dst			CALLA x(Rdst)		
	x.15:0																	
	0	0	0	1	0	0	1	1	0	1	1	0	dst			CALLA @Rdst		
	0	0	0	1	0	0	1	1	0	1	1	1	dst			CALLA @Rdst+		
	0	0	0	1	0	0	1	1	1	0	0	0	&abs.19:16			CALLA &abs20		
	&abs.15:0																	
	0	0	0	1	0	0	1	1	1	0	0	1	x.19:16			CALLA EDE		
	x.15:0																	
	0	0	0	1	0	0	1	1	1	0	1	1	imm.19:16			CALLA #imm20		
imm.15:0																		
Reserved	0	0	0	1	0	0	1	1	1	0	1	0	x	x	x	x		
Reserved	0	0	0	1	0	0	1	1	1	1	x	x	x	x	x	x		
PUSHM.A	0	0	0	1	0	1	0	0	n − 1			dst			PUSHM.A #n,Rdst			
PUSHM.W	0	0	0	1	0	1	0	1	n − 1			dst			PUSHM.W #n,Rdst			
POPM.A	0	0	0	1	0	1	1	0	n − 1			dst − n + 1			POPM.A #n,Rdst			
POPM.W	0	0	0	1	0	1	1	1	n − 1			dst − n + 1			POPM.W #n,Rdst			

4.6.2 MSP430 Instructions

The MSP430 instructions are listed and described on the following pages.

4.6.2.1 ADC

* ADC[W]	Add carry to destination		
* ADC.B	Add carry to destination		
Syntax	ADC dst or	ADC.W dst	
	ADC.B dst		
Operation	$dst + C \rightarrow dst$		
Emulation	ADDC #0, dst		
	ADDC.B #0, dst		
Description	The carry bit (C) is added to the destination operand. The previous contents of the destination are lost.		
Status Bits	N: Set if result is negative, reset if positive		
	Z: Set if result is zero, reset otherwise		
	C: Set if dst was incremented from 0FFFFh to 0000, reset otherwise		
	Set if dst was incremented from 0FFh to 00, reset otherwise		
	V: Set if an arithmetic overflow occurs, otherwise reset		
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.		
Example	The 16-bit counter pointed to by R13 is added to a 32-bit counter pointed to by R12.		
	ADD	@R13, 0(R12)	; Add LSDs
	ADC	2(R12)	; Add carry to MSD
Example	The 8-bit counter pointed to by R13 is added to a 16-bit counter pointed to by R12.		
	ADD.B	@R13, 0(R12)	; Add LSDs
	ADC.B	1(R12)	; Add carry to MSD

4.6.2.2 ADD

ADD[.W]	Add source word to destination word
ADD.B	Add source byte to destination byte
Syntax	ADD src,dst OR ADD.W src,dst ADD.B src,dst
Operation	src + dst → dst
Description	The source operand is added to the destination operand. The previous content of the destination is lost.
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if there is a carry from the MSB of the result, reset otherwise V: Set if the result of 2 positive operands is negative, or if the result of 2 negative numbers is positive, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	10 is added to the 16-bit counter CNTR located in lower 64 K.

```
ADD.W    #10,&CNTR    ; Add 10 to 16-bit counter
```

Example A table word pointed to by R5 (20-bit address in R5) is added to R6. The jump to label TONI is performed on a carry.

```
ADD.W    @R5,R6        ; Add table word to R6. R6.19:16 = 0
JC       TONI           ; Jump if carry
...      ; No carry
```

Example A table byte pointed to by R5 (20-bit address) is added to R6. The jump to label TONI is performed if no carry occurs. The table pointer is auto-incremented by 1. R6.19:8 = 0

```
ADD.B    @R5+,R6        ; Add byte to R6. R5 + 1. R6: 000xxh
JNC      TONI           ; Jump if no carry
...      ; Carry occurred
```

4.6.2.3 ADDC

ADDC.W]	Add source word and carry to destination word
ADDC.B	Add source byte and carry to destination byte
Syntax	ADDC src,dst or ADDC.W src,dst ADDC.B src,dst
Operation	$\text{src} + \text{dst} + \text{C} \rightarrow \text{dst}$
Description	The source operand and the carry bit C are added to the destination operand. The previous content of the destination is lost.
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if there is a carry from the MSB of the result, reset otherwise V: Set if the result of 2 positive operands is negative, or if the result of 2 negative numbers is positive, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Constant value 15 and the carry of the previous instruction are added to the 16-bit counter CNTR located in lower 64 K.

```
ADDC.W    #15,&CNTR    ; Add 15 + C to 16-bit CNTR
```

Example A table word pointed to by R5 (20-bit address) and the carry C are added to R6. The jump to label TONI is performed on a carry. R6.19:16 = 0

```
ADDC.W    @R5,R6      ; Add table word + C to R6
JC        TONI        ; Jump if carry
...       ; No carry
```

Example A table byte pointed to by R5 (20-bit address) and the carry bit C are added to R6. The jump to label TONI is performed if no carry occurs. The table pointer is auto-incremented by 1. R6.19:8 = 0

```
ADDC.B    @R5+,R6     ; Add table byte + C to R6. R5 + 1
JNC       TONI        ; Jump if no carry
...       ; Carry occurred
```

4.6.2.4 AND

AND[.W]	Logical AND of source word with destination word		
AND.B	Logical AND of source byte with destination byte		
Syntax	AND src,dst or AND.W src,dst AND.B src,dst		
Operation	src .and. dst → dst		
Description	The source operand and the destination operand are logically ANDed. The result is placed into the destination. The source operand is not affected.		
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if the result is not zero, reset otherwise. C = (.not. Z) V: Reset		
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.		
Example	The bits set in R5 (16-bit data) are used as a mask (AA55h) for the word TOM located in the lower 64 K. If the result is zero, a branch is taken to label TONI. R5.19:16 = 0		
	MOV	#AA55h,R5	; Load 16-bit mask to R5
	AND	R5,&TOM	; TOM .and. R5 -> TOM
	JZ	TONI	; Jump if result 0
	...		; Result > 0
	or shorter:		
	AND	#AA55h,&TOM	; TOM .and. AA55h -> TOM
	JZ	TONI	; Jump if result 0
Example	A table byte pointed to by R5 (20-bit address) is logically ANDed with R6. R5 is incremented by 1 after the fetching of the byte. R6.19:8 = 0		
	AND.B	@R5+,R6	; AND table byte with R6. R5 + 1

4.6.2.5 BIC

BIC[.W]	Clear bits set in source word in destination word
BIC.B	Clear bits set in source byte in destination byte
Syntax	BIC src,dst or BIC.W src,dst BIC.B src,dst
Operation	(.not. src) .and. dst → dst
Description	The inverted source operand and the destination operand are logically ANDed. The result is placed into the destination. The source operand is not affected.
Status Bits	N: Not affected Z: Not affected C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The bits 15:14 of R5 (16-bit data) are cleared. R5.19:16 = 0
	<pre>BIC #0C000h,R5 ; Clear R5.19:14 bits</pre>
Example	A table word pointed to by R5 (20-bit address) is used to clear bits in R7. R7.19:16 = 0
	<pre>BIC.W @R5,R7 ; Clear bits in R7 set in @R5</pre>
Example	A table byte pointed to by R5 (20-bit address) is used to clear bits in Port1.
	<pre>BIC.B @R5,&P1OUT ; Clear I/O port P1 bits set in @R5</pre>

4.6.2.6 BIS

BIS[.W]	Set bits set in source word in destination word
BIS.B	Set bits set in source byte in destination byte
Syntax	BIS src,dst or BIS.W src,dst BIS.B src,dst
Operation	src .or. dst → dst
Description	The source operand and the destination operand are logically ORed. The result is placed into the destination. The source operand is not affected.
Status Bits	N: Not affected Z: Not affected C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Bits 15 and 13 of R5 (16-bit data) are set to 1. R5.19:16 = 0
	<pre>BIS #A000h,R5 ; Set R5 bits</pre>
Example	A table word pointed to by R5 (20-bit address) is used to set bits in R7. R7.19:16 = 0
	<pre>BIS.W @R5,R7 ; Set bits in R7</pre>
Example	A table byte pointed to by R5 (20-bit address) is used to set bits in Port1. R5 is incremented by 1 afterwards.
	<pre>BIS.B @R5+,&P1OUT ; Set I/O port P1 bits. R5 + 1</pre>

4.6.2.7 BIT

BIT[.W]	Test bits set in source word in destination word
BIT.B	Test bits set in source byte in destination byte
Syntax	BIT src,dst OR BIT.W src,dst BIT.B src,dst
Operation	src .and. dst
Description	The source operand and the destination operand are logically ANDed. The result affects only the status bits in SR. Register mode: the register bits Rdst.19:16 (.W) resp. Rdst. 19:8 (.B) are not cleared!
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if the result is not zero, reset otherwise. C = (.not. Z) V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Test if 1 (or both) of bits 15 and 14 of R5 (16-bit data) is set. Jump to label TONI if this is the case. R5.19:16 are not affected.

```

BIT    #C000h,R5      ; Test R5.15:14 bits
JNZ    TONI            ; At least 1 bit is set in R5
...                      ; Both bits are reset

```

Example A table word pointed to by R5 (20-bit address) is used to test bits in R7. Jump to label TONI if at least 1 bit is set. R7.19:16 are not affected.

```

BIT.W  @R5,R7          ; Test bits in R7
JC      TONI            ; At least 1 bit is set
...                      ; Both are reset

```

Example A table byte pointed to by R5 (20-bit address) is used to test bits in output Port1. Jump to label TONI if no bit is set. The next table byte is addressed.

```

BIT.B  @R5+,&P1OUT      ; Test I/O port P1 bits. R5 + 1
JNC     TONI            ; No corresponding bit is set
...                      ; At least 1 bit is set

```

4.6.2.8 BR, BRANCH

* BR, BRANCH	Branch to destination in lower 64K address space
Syntax	BR dst
Operation	dst → PC
Emulation	MOV dst,PC
Description	An unconditional branch is taken to an address anywhere in the lower 64K address space. All source addressing modes can be used. The branch instruction is a word instruction.
Status Bits	Status bits are not affected.
Example	Examples for all addressing modes are given.

BR	#EXEC	; Branch to label EXEC or direct branch (for example #0A4h) ; Core instruction MOV @PC+,PC
BR	EXEC	; Branch to the address contained in EXEC ; Core instruction MOV X(PC),PC ; Indirect address
BR	&EXEC	; Branch to the address contained in absolute ; address EXEC ; Core instruction MOV X(0),PC ; Indirect address
BR	R5	; Branch to the address contained in R5 ; Core instruction MOV R5,PC ; Indirect R5
BR	@R5	; Branch to the address contained in the word ; pointed to by R5. ; Core instruction MOV @R5,PC ; Indirect, indirect R5
BR	@R5+	; Branch to the address contained in the word pointed ; to by R5 and increment pointer in R5 afterwards. ; The next time-S/W flow uses R5 pointer-it can ; alter program execution due to access to ; next address in a table pointed to by R5 ; Core instruction MOV @R5,PC ; Indirect, indirect R5 with autoincrement
BR	X(R5)	; Branch to the address contained in the address ; pointed to by R5 + X (for example table with address ; starting at X). X can be an address or a label ; Core instruction MOV X(R5),PC ; Indirect, indirect R5 + X

4.6.2.9 CALL

CALL	Call a subroutine in lower 64 K		
Syntax	CALL dst		
Operation	dst → tmp 16-bit dst is evaluated and stored SP – 2 → SP PC → @SP updated PC with return address to TOS tmp → PC saved 16-bit dst to PC		
Description	A subroutine call is made from an address in the lower 64 K to a subroutine address in the lower 64 K. All 7 source addressing modes can be used. The call instruction is a word instruction. The return is made with the RET instruction.		
Status Bits	Status bits are not affected. PC.19:16 cleared (address in lower 64 K)		
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.		
Examples	Examples for all addressing modes are given. Immediate Mode: Call a subroutine at label EXEC (lower 64 K) or call directly to address.		
CALL	#EXEC		; Start address EXEC
CALL	#0AA04h		; Start address 0AA04h
	Symbolic Mode: Call a subroutine at the 16-bit address contained in address EXEC. EXEC is located at the address (PC + X) where X is within PC ± 32 K.		
CALL	EXEC		; Start address at @EXEC. z16(PC)
	Absolute Mode: Call a subroutine at the 16-bit address contained in absolute address EXEC in the lower 64 K.		
CALL	&EXEC		; Start address at @EXEC
	Register mode: Call a subroutine at the 16-bit address contained in register R5.15:0.		
CALL	R5		; Start address at R5
	Indirect Mode: Call a subroutine at the 16-bit address contained in the word pointed to by register R5 (20-bit address).		
CALL	@R5		; Start address at @R5

4.6.2.10 CLR

* CLR[.W]	Clear destination	
* CLR.B	Clear destination	
Syntax	CLR dst or CLR.B dst	CLR.W dst
Operation	0 → dst	
Emulation	MOV #0, dst MOV.B #0, dst	
Description	The destination operand is cleared.	
Status Bits	Status bits are not affected.	
Example	RAM word TONI is cleared.	
	CLR	TONI ; 0 -> TONI
Example	Register R5 is cleared.	
	CLR	R5
Example	RAM byte TONI is cleared.	
	CLR.B	TONI ; 0 -> TONI

4.6.2.11 CLRC

* CLRC	Clear carry bit
Syntax	CLRC
Operation	$0 \rightarrow C$
Emulation	BIC #1, SR
Description	The carry bit (C) is cleared. The clear carry instruction is a word instruction.
Status Bits	N: Not affected
	Z: Not affected
	C: Cleared
	V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 16-bit decimal counter pointed to by R13 is added to a 32-bit counter pointed to by R12.
<pre> CLRC ; C=0: defines start DADD @R13,0(R12) ; add 16-bit counter to low word of 32-bit counter DADC 2(R12) ; add carry to high word of 32-bit counter </pre>	

4.6.2.12 CLRN

* CLRN	Clear negative bit
Syntax	CLRN
Operation	0 → N or (.NOT.src .AND. dst → dst)
Emulation	BIC #4, SR
Description	The constant 04h is inverted (0FFFBh) and is logically ANDed with the destination operand. The result is placed into the destination. The clear negative bit instruction is a word instruction.
Status Bits	N: Reset to 0 Z: Not affected C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The negative bit in the SR is cleared. This avoids special treatment with negative numbers of the subroutine called.

```

        CLRN
        CALL SUBR
        .....
        .....
SUBR    JN      SUBRET    ; If input is negative: do nothing and return
        .....
        .....
        .....
SUBRET  RET

```

4.6.2.13 CLRZ

* CLRZ	Clear zero bit
Syntax	CLRZ
Operation	$0 \rightarrow Z$ or (.NOT.src .AND. dst $\rightarrow$ dst)
Emulation	BIC #2, SR
Description	The constant 02h is inverted (0FFFDh) and logically ANDed with the destination operand. The result is placed into the destination. The clear zero bit instruction is a word instruction.
Status Bits	N: Not affected Z: Reset to 0 C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The zero bit in the SR is cleared.

CLRZ

Indirect, Auto-Increment mode: Call a subroutine at the 16-bit address contained in the word pointed to by register R5 (20-bit address) and increment the 16-bit address in R5 afterwards by 2. The next time the software uses R5 as a pointer, it can alter the program execution due to access to the next word address in the table pointed to by R5.

CALL @R5+ ; Start address at @R5. R5 + 2

Indexed mode: Call a subroutine at the 16-bit address contained in the 20-bit address pointed to by register (R5 + X); for example, a table with addresses starting at X. The address is within the lower 64KB. X is within $\pm 32\text{KB}$.

CALL X(R5) ; Start address at @(R5+X). z16(R5)

4.6.2.14 CMP

CMP[.W]	Compare source word and destination word	
CMP.B	Compare source byte and destination byte	
Syntax	CMP src,dst OR CMP.W src,dst CMP.B src,dst	
Operation	(.not.src) + 1 + dst or dst – src	
Description	The source operand is subtracted from the destination operand. This is made by adding the 1s complement of the source + 1 to the destination. The result affects only the status bits in SR. Register mode: the register bits Rdst.19:16 (.W) resp. Rdst. 19:8 (.B) are not cleared.	
Status Bits	N: Set if result is negative (src > dst), reset if positive (src ≤ dst) Z: Set if result is zero (src = dst), reset otherwise (src ≠ dst) C: Set if there is a carry from the MSB, reset otherwise V: Set if the subtraction of a negative source operand from a positive destination operand delivers a negative result, or if the subtraction of a positive source operand from a negative destination operand delivers a positive result, reset otherwise (no overflow).	
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.	
Example	Compare word EDE with a 16-bit constant 1800h. Jump to label TONI if EDE equals the constant. The address of EDE is within PC + 32 K.	
	CMP	#01800h,EDE ; Compare word EDE with 1800h
	JEQ	TONI ; EDE contains 1800h
	...	; Not equal
Example	A table word pointed to by (R5 + 10) is compared with R7. Jump to label TONI if R7 contains a lower, signed 16-bit number. R7.19:16 is not cleared. The address of the source operand is a 20-bit address in full memory range.	
	CMP.W	10(R5),R7 ; Compare 2 signed numbers
	JL	TONI ; R7 < 10(R5)
	...	; R7 >= 10(R5)
Example	A table byte pointed to by R5 (20-bit address) is compared to the value in output Port1. Jump to label TONI if values are equal. The next table byte is addressed.	
	CMP.B	@R5+,&P1OUT ; Compare P1 bits with table. R5 + 1
	JEQ	TONI ; Equal contents
	...	; Not equal

4.6.2.15 DADC

* DADC[W]	Add carry decimally to destination
* DADC.B	Add carry decimally to destination
Syntax	DADC dst or DADC.W dst DADC.B dst
Operation	dst + C → dst (decimally)
Emulation	DADD #0, dst DADD.B #0, dst
Description	The carry bit (C) is added decimally to the destination.
Status Bits	N: Set if MSB is 1 Z: Set if dst is 0, reset otherwise C: Set if destination increments from 9999 to 0000, reset otherwise Set if destination increments from 99 to 00, reset otherwise V: Undefined
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 4-digit decimal number contained in R5 is added to an 8-digit decimal number pointed to by R8.

```

CLRC                ; Reset carry
                    ; next instruction's start condition is defined
DADD R5,0(R8)       ; Add LSDs + C
DADC 2(R8)           ; Add carry to MSD

```

Example The 2-digit decimal number contained in R5 is added to a 4-digit decimal number pointed to by R8.

```

CLRC                ; Reset carry
                    ; next instruction's start condition is defined
DADD.B R5,0(R8)     ; Add LSDs + C
DADC 1(R8)           ; Add carry to MSDs

```

4.6.2.16 DADD

* DADD[.W]	Add source word and carry decimally to destination word
* DADD.B	Add source byte and carry decimally to destination byte
Syntax	DADD src,dst or DADD.W src,dst DADD.B src,dst
Operation	src + dst + C → dst (decimally)
Description	The source operand and the destination operand are treated as 2 (.B) or 4 (.W) binary coded decimals (BCD) with positive signs. The source operand and the carry bit C are added decimally to the destination operand. The source operand is not affected. The previous content of the destination is lost. The result is not defined for non-BCD numbers.
Status Bits	N: Set if MSB of result is 1 (word > 7999h, byte > 79h), reset if MSB is 0 Z: Set if result is zero, reset otherwise C: Set if the BCD result is too large (word > 9999h, byte > 99h), reset otherwise V: Undefined
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Decimal 10 is added to the 16-bit BCD counter DECCNTR.

```
DADD    #10h,&DECCNTR    ; Add 10 to 4-digit BCD counter
```

Example	The 8-digit BCD number contained in 16-bit RAM addresses BCD and BCD+2 is added decimally to an 8-digit BCD number contained in R4 and R5 (BCD+2 and R5 contain the MSDs). The carry C is added, and cleared.

```
CLRC                                ; Clear carry
DADD.W    &BCD,R4                   ; Add LSDs. R4.19:16 = 0
DADD.W    &BCD+2,R5                 ; Add MSDs with carry. R5.19:16 = 0
JC        OVERFLOW                 ; Result >9999,9999: go to error routine
...                                ; Result ok
```

Example	The 2-digit BCD number contained in word BCD (16-bit address) is added decimally to a 2-digit BCD number contained in R4. The carry C is added, also. R4.19:8 = 0

```
CLRC                                ; Clear carry
DADD.B    &BCD,R4                   ; Add BCD to R4 decimally.
                                         R4: 0,00ddh
```

4.6.2.17 DEC

* DEC[.W]	Decrement destination
* DEC.B	Decrement destination
Syntax	DEC dst or DEC.W dst DEC.B dst
Operation	$dst - 1 \rightarrow dst$
Emulation	SUB #1, dst SUB.B #1, dst
Description	The destination operand is decremented by 1. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if dst contained 1, reset otherwise C: Reset if dst contained 0, set otherwise V: Set if an arithmetic overflow occurs, otherwise reset. Set if initial value of destination was 08000h, otherwise reset. Set if initial value of destination was 080h, otherwise reset.
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	R10 is decremented by 1.

```

DEC    R10                ; Decrement R10

; Move a block of 255 bytes from memory location starting with EDE to
; memory location starting with TONI. Tables should not overlap: start of
; destination address TONI must not be within the range EDE to EDE+0FEh

MOV    #EDE, R6
MOV    #255, R10
L$1    MOV.B  @R6+, TONI-EDE-1(R6)
DEC    R10
JNZ    L$1

```

Do not transfer tables using the routine above with the overlap shown in [Figure 4-36](#).

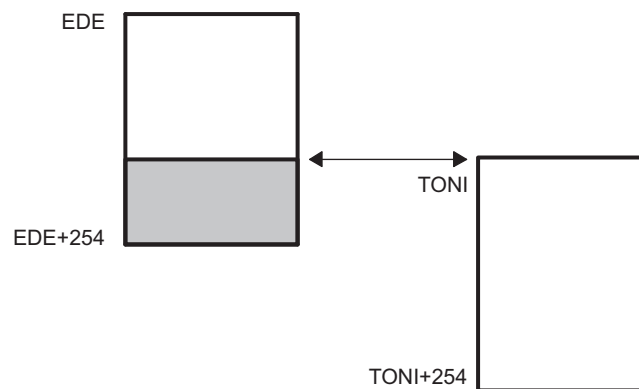


Figure 4-36. Decrement Overlap

4.6.2.18 DECD

* DECD[.W]	Double-decrement destination
* DECD.B	Double-decrement destination
Syntax	DECD dst or DECD.W dst DECD.B dst
Operation	$dst - 2 \rightarrow dst$
Emulation	SUB #2, dst SUB.B #2, dst
Description	The destination operand is decremented by 2. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if dst contained 2, reset otherwise C: Reset if dst contained 0 or 1, set otherwise V: Set if an arithmetic overflow occurs, otherwise reset Set if initial value of destination was 08001 or 08000h, otherwise reset Set if initial value of destination was 081 or 080h, otherwise reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	R10 is decremented by 2.

```
DECD    R10                ; Decrement R10 by 2
```

```
; Move a block of 255 bytes from memory location starting with EDE to
; memory location starting with TONI.
; Tables should not overlap: start of destination address TONI must not
; be within the range EDE to EDE+0FEh
```

```

MOV     #EDE, R6
MOV     #255, R10
L$1     MOV.B   @R6+, TONI - EDE - 2(R6)
        DECD    R10
        JNZ     L$1
```

Example Memory at location LEO is decremented by 2.

```
DECD.B   LEO                ; Decrement MEM(LE0)
```

Decrement status byte STATUS by 2

```
DECD.B   STATUS
```

4.6.2.19 DINT

* DINT	Disable (general) interrupts
Syntax	DINT
Operation	0 → GIE or (0FFF7h .AND. SR → SR / .NOT.src .AND. dst → dst)
Emulation	BIC #8, SR
Description	All interrupts are disabled. The constant 08h is inverted and logically ANDed with the SR. The result is placed into the SR.
Status Bits	Status bits are not affected.
Mode Bits	GIE is reset. OSCOFF and CPUOFF are not affected.
Example	The general interrupt enable (GIE) bit in the SR is cleared to allow a nondisrupted move of a 32-bit counter. This ensures that the counter is not modified during the move by any interrupt.

```

DINT                ; All interrupt events using the GIE bit are disabled
NOP                 ; Required due to pipelined CPU architecture
MOV    COUNTHI,R5   ; Copy counter
MOV    COUNTLO,R6
EINT                ; All interrupt events using the GIE bit are enabled

```

NOTE: Disable interrupt

Due to the pipelined CPU architecture, clearing the general interrupt enable (GIE) requires special care.

- Include at least 1 instruction between DINT and the start of an code sequence that requires protection from interrupts. For example: Insert a NOP instruction after the DINT.
- Never clear the general interrupt enable (GIE) immediately after setting it. Insert at least 1 instruction in between such sequence.

The rules above apply to all instructions that clear the general interrupt enable bit. Not following these rules might result in unexpected CPU execution.

4.6.2.20 EINT

* EINT	Enable (general) interrupts
Syntax	EINT
Operation	1 → GIE or (0008h .OR. SR → SR / .src .OR. dst → dst)
Emulation	BIS #8, SR
Description	All interrupts are enabled. The constant #08h and the SR are logically ORed. The result is placed into the SR.
Status Bits	Status bits are not affected.
Mode Bits	GIE is set. OSCOFF and CPUOFF are not affected.
Example	The general interrupt enable (GIE) bit in the SR is set.

```

PUSH.B    &P1IN
BIC.B     @SP,&P1IFG    ; Reset only accepted flags
NOP                               ; Required due to pipelined CPU architecture
EINT                               ; Preset port 1 interrupt flags stored on stack
                               ; other interrupts are allowed

BIT       #Mask,@SP
JEQ       MaskOK        ; Flags are present identically to mask: jump
.....
MaskOK    BIC       #Mask,@SP
.....
INCD      SP            ; Housekeeping: inverse to PUSH instruction
                               ; at the start of interrupt subroutine. Corrects
                               ; the stack pointer.

RETI

```

NOTE: Enable interrupt

Due to the pipelined CPU architecture, setting the general interrupt enable (GIE) requires special care.

- The instruction immediately after the enable interrupts instruction (EINT) is always executed, even if an interrupt service request is pending.
- Include at least 1 instruction between the clear of an interrupt enable or interrupt flag and the EINT instruction. For example: Insert a NOP instruction in front of the EINT instruction.
- Never clear the general interrupt enable (GIE) immediately after setting it. Insert at least 1 instruction in between such sequence.

The rules above apply to all instructions that set the general interrupt enable bit. Not following these rules might result in unexpected CPU execution.

4.6.2.21 INC

* INC[.W]	Increment destination
* INC.B	Increment destination
Syntax	INC dst or INC.W dst INC.B dst
Operation	dst + 1 → dst
Emulation	ADD #1, dst
Description	The destination operand is incremented by 1. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if dst contained 0FFFFh, reset otherwise Set if dst contained 0FFh, reset otherwise C: Set if dst contained 0FFFFh, reset otherwise Set if dst contained 0FFh, reset otherwise V: Set if dst contained 07FFFh, reset otherwise Set if dst contained 07Fh, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The status byte, STATUS, of a process is incremented. When it is equal to 11, a branch to OVFL is taken.
INC.B	STATUS
CMP.B	#11, STATUS
JEQ	OVFL

4.6.2.22 INCD

* INCD[.W]	Double-increment destination
* INCD.B	Double-increment destination
Syntax	INCD dst or INCD.W dst INCD.B dst
Operation	dst + 2 → dst
Emulation	ADD #2, dst
Description	The destination operand is incremented by 2. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if dst contained 0FFFEh, reset otherwise Set if dst contained 0FEh, reset otherwise C: Set if dst contained 0FFFEh or 0FFFFh, reset otherwise Set if dst contained 0FEh or 0FFh, reset otherwise V: Set if dst contained 07FFEh or 07FFFh, reset otherwise Set if dst contained 07Eh or 07Fh, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The item on the top of the stack (TOS) is removed without using a register.

```

.....
PUSH    R5        ; R5 is the result of a calculation, which is stored
                ; in the system stack
INCD     SP        ; Remove TOS by double-increment from stack
                ; Do not use INCD.B, SP is a word-aligned register
RET

```

Example The byte on the top of the stack is incremented by 2.

```
INCD.B   0(SP)    ; Byte on TOS is increment by 2
```


4.6.2.23 INV

* INV[.W]	Invert destination
* INV.B	Invert destination
Syntax	INV dst or INV.W dst INV.B dst
Operation	.not.dst → dst
Emulation	XOR #0FFFFh, dst XOR.B #0FFh, dst
Description	The destination operand is inverted. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if dst contained 0FFFFh, reset otherwise Set if dst contained 0FFh, reset otherwise C: Set if result is not zero, reset otherwise (= .NOT. Zero) V: Set if initial destination operand was negative, otherwise reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Content of R5 is negated (twos complement).

```

MOV    #00AEh, R5    ; R5 = 000AEh
INV     R5            ; Invert R5, R5 = 0FF51h
INC     R5            ; R5 is now negated, R5 = 0FF52h

```

Example Content of memory byte LEO is negated.

```

MOV.B   #0AEh, LEO    ; MEM(LEO) = 0AEh
INV.B   LEO           ; Invert LEO, MEM(LEO) = 051h
INC.B   LEO           ; MEM(LEO) is negated, MEM(LEO) = 052h

```

4.6.2.24 JC, JHS

JC	Jump if carry
JHS	Jump if higher or same (unsigned)
Syntax	JC label JHS label
Operation	If C = 1: PC + (2 × Offset) → PC If C = 0: execute the following instruction
Description	The carry bit C in the SR is tested. If it is set, the signed 10-bit word offset contained in the instruction is multiplied by 2, sign extended, and added to the 20-bit PC. This means a jump in the range –511 to +512 words relative to the PC in the full memory range. If C is reset, the instruction after the jump is executed. JC is used for the test of the carry bit C. JHS is used for the comparison of unsigned numbers.
Status Bits	Status bits are not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The state of the port 1 pin P1IN.1 bit defines the program flow.

```

BIT.B  #2,&P1IN      ; Port 1, bit 1 set? Bit -> C
JC      Label1        ; Yes, proceed at Label1
...           ; No, continue

```

Example If R5 ≥ R6 (unsigned), the program continues at Label2.

```

CMP      R6,R 5        ; Is R5 >= R6? Info to C
JHS      Label2        ; Yes, C = 1
...           ; No, R5 < R6. Continue

```

Example If R5 ≥ 12345h (unsigned operands), the program continues at Label2.

```

CMPA     #12345h,R5    ; Is R5 >= 12345h? Info to C
JHS      Label2        ; Yes, 12344h < R5 <= F,FFFFh. C = 1
...           ; No, R5 < 12345h. Continue

```

4.6.2.25 JEQ, JZ

JEQ	Jump if equal
JZ	Jump if zero
Syntax	JEQ label JZ label
Operation	If Z = 1: PC + (2 × Offset) → PC If Z = 0: execute following instruction
Description	The zero bit Z in the SR is tested. If it is set, the signed 10-bit word offset contained in the instruction is multiplied by 2, sign extended, and added to the 20-bit PC. This means a jump in the range –511 to +512 words relative to the PC in the full memory range. If Z is reset, the instruction after the jump is executed. JZ is used for the test of the zero bit Z. JEQ is used for the comparison of operands.
Status Bits	Status bits are not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The state of the P2IN.0 bit defines the program flow.

```

BIT.B  #1,&P2IN      ; Port 2, bit 0 reset?
JZ     Label1        ; Yes, proceed at Label1
...           ; No, set, continue

```

Example If R5 = 15000h (20-bit data), the program continues at Label2.

```

CMPA   #15000h,R5    ; Is R5 = 15000h? Info to SR
JEQ    Label2         ; Yes, R5 = 15000h. Z = 1
...           ; No, R5 not equal 15000h. Continue

```

Example R7 (20-bit counter) is incremented. If its content is zero, the program continues at Label4.

```

ADDA   #1,R7         ; Increment R7
JZ     Label4         ; Zero reached: Go to Label4
...           ; R7 not equal 0. Continue here.

```

4.6.2.26 JGE

JGE	Jump if greater or equal (signed)
Syntax	JGE label
Operation	If (N .xor. V) = 0: PC + (2 × Offset) → PC If (N .xor. V) = 1: execute following instruction
Description	The negative bit N and the overflow bit V in the SR are tested. If both bits are set or both are reset, the signed 10-bit word offset contained in the instruction is multiplied by 2, sign extended, and added to the 20-bit PC. This means a jump in the range -511 to +512 words relative to the PC in full memory range. If only 1 bit is set, the instruction after the jump is executed. JGE is used for the comparison of signed operands: also for incorrect results due to overflow, the decision made by the JGE instruction is correct. Note that JGE emulates the nonimplemented JP (jump if positive) instruction if used after the instructions AND, BIT, RRA, SXTX, and TST. These instructions clear the V bit.
Status Bits	Status bits are not affected.
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	If byte EDE (lower 64 K) contains positive data, go to Label1. Software can run in the full memory range.

```

TST.B    &EDE           ; Is EDE positive? V <- 0
JGE      Label1         ; Yes, JGE emulates JP
...      ; No, 80h <= EDE <= FFh

```

Example	If the content of R6 is greater than or equal to the memory pointed to by R7, the program continues a Label5. Signed data. Data and program in full memory range.

```

CMP      @R7,R6         ; Is R6 >= @R7?
JGE      Label5         ; Yes, go to Label5
...      ; No, continue here

```

Example	If R5 ≥ 12345h (signed operands), the program continues at Label2. Program in full memory range.

```

CMPA     #12345h,R5     ; Is R5 >= 12345h?
JGE      Label2         ; Yes, 12344h < R5 <= 7FFFFh
...      ; No, 80000h <= R5 < 12345h

```

4.6.2.27 JL

JL	Jump if less (signed)
Syntax	JL label
Operation	If (N .xor. V) = 1: $PC + (2 \times \text{Offset}) \rightarrow PC$ If (N .xor. V) = 0: execute following instruction
Description	The negative bit N and the overflow bit V in the SR are tested. If only 1 is set, the signed 10-bit word offset contained in the instruction is multiplied by 2, sign extended, and added to the 20-bit PC. This means a jump in the range –511 to +512 words relative to the PC in full memory range. If both bits N and V are set or both are reset, the instruction after the jump is executed. JL is used for the comparison of signed operands: also for incorrect results due to overflow, the decision made by the JL instruction is correct.
Status Bits	Status bits are not affected.
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	If byte EDE contains a smaller, signed operand than byte TONI, continue at Label1. The address EDE is within $PC \pm 32 K$.

```

CMP.B    &TONI,EDE      ; Is EDE < TONI
JL       Label1        ; Yes
...      ; No, TONI <= EDE

```

Example	If the signed content of R6 is less than the memory pointed to by R7 (20-bit address), the program continues at Label5. Data and program in full memory range.

```

CMP      @R7,R6         ; Is R6 < @R7?
JL       Label5        ; Yes, go to Label5
...      ; No, continue here

```

Example	If $R5 < 12345h$ (signed operands), the program continues at Label2. Data and program in full memory range.

```

CMPA     #12345h,R5     ; Is R5 < 12345h?
JL       Label2        ; Yes, 80000h =< R5 < 12345h
...      ; No, 12344h < R5 <= 7FFFFh

```

4.6.2.28 JMP

JMP Jump unconditionally

Syntax JMP label

Operation $PC + (2 \times \text{Offset}) \rightarrow PC$

Description The signed 10-bit word offset contained in the instruction is multiplied by 2, sign extended, and added to the 20-bit PC. This means an unconditional jump in the range –511 to +512 words relative to the PC in the full memory. The JMP instruction may be used as a BR or BRA instruction within its limited range relative to the PC.

Status Bits Status bits are not affected

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example The byte STATUS is set to 10. Then a jump to label MAINLOOP is made. Data in lower 64 K, program in full memory range.

```
MOV.B  #10,&STATUS    ; Set STATUS to 10
JMP     MAINLOOP      ; Go to main loop
```

Example The interrupt vector TAIV of Timer_A3 is read and used for the program flow. Program in full memory range, but interrupt handlers always starts in lower 64 K.

```
ADD     &TAIV,PC       ; Add Timer_A interrupt vector to PC
RETI                    ; No Timer_A interrupt pending
JMP     IHCCR1          ; Timer block 1 caused interrupt
JMP     IHCCR2          ; Timer block 2 caused interrupt
RETI                    ; No legal interrupt, return
```

4.6.2.29 JN

JN	Jump if negative
Syntax	JN label
Operation	If N = 1: $PC + (2 \times \text{Offset}) \rightarrow PC$ If N = 0: execute following instruction
Description	The negative bit N in the SR is tested. If it is set, the signed 10-bit word offset contained in the instruction is multiplied by 2, sign extended, and added to the 20-bit program PC. This means a jump in the range -511 to +512 words relative to the PC in the full memory range. If N is reset, the instruction after the jump is executed.
Status Bits	Status bits are not affected.
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The byte COUNT is tested. If it is negative, program execution continues at Label0. Data in lower 64 K, program in full memory range.

```

TST.B    &COUNT    ; Is byte COUNT negative?
JN       Label0     ; Yes, proceed at Label0
...      ; COUNT >= 0

```

Example R6 is subtracted from R5. If the result is negative, program continues at Label2. Program in full memory range.

```

SUB      R6,R5      ; R5 - R6 -> R5
JN       Label2     ; R5 is negative: R6 > R5 (N = 1)
...      ; R5 >= 0. Continue here.

```

Example R7 (20-bit counter) is decremented. If its content is below zero, the program continues at Label4. Program in full memory range.

```

SUBA     #1,R7      ; Decrement R7
JN       Label4     ; R7 < 0: Go to Label4
...      ; R7 >= 0. Continue here.

```

4.6.2.30 JNC, JLO

JNC	Jump if no carry
JLO	Jump if lower (unsigned)
Syntax	JNC label JLO label
Operation	If C = 0: PC + (2 × Offset) → PC If C = 1: execute following instruction
Description	The carry bit C in the SR is tested. If it is reset, the signed 10-bit word offset contained in the instruction is multiplied by 2, sign extended, and added to the 20-bit PC. This means a jump in the range –511 to +512 words relative to the PC in the full memory range. If C is set, the instruction after the jump is executed. JNC is used for the test of the carry bit C. JLO is used for the comparison of unsigned numbers.
Status Bits	Status bits are not affected.
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	If byte EDE < 15, the program continues at Label2. Unsigned data. Data in lower 64 K, program in full memory range.
	<pre> CMP.B #15,&EDE ; Is EDE < 15? Info to C JLO Label2 ; Yes, EDE < 15. C = 0 ... ; No, EDE >= 15. Continue </pre>
Example	The word TONI is added to R5. If no carry occurs, continue at Label0. The address of TONI is within PC ± 32 K.
	<pre> ADD TONI,R5 ; TONI + R5 -> R5. Carry -> C JNC Label0 ; No carry ... ; Carry = 1: continue here </pre>

4.6.2.31 JNZ, JNE

JNZ Jump if not zero

JNE Jump if not equal

Syntax JNZ label

JNE label

Operation If Z = 0: PC + (2 × Offset) → PC
If Z = 1: execute following instruction

Description The zero bit Z in the SR is tested. If it is reset, the signed 10-bit word offset contained in the instruction is multiplied by 2, sign extended, and added to the 20-bit PC. This means a jump in the range –511 to +512 words relative to the PC in the full memory range. If Z is set, the instruction after the jump is executed.

JNZ is used for the test of the zero bit Z.

JNE is used for the comparison of operands.

Status Bits Status bits are not affected.

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example The byte STATUS is tested. If it is not zero, the program continues at Label3. The address of STATUS is within PC ± 32 K.

```
TST.B STATUS      ; Is STATUS = 0?
JNZ  Label3       ; No, proceed at Label3
...              ; Yes, continue here
```

Example If word EDE ≠ 1500, the program continues at Label2. Data in lower 64 K, program in full memory range.

```
CMP  #1500,&EDE    ; Is EDE = 1500? Info to SR
JNE  Label2        ; No, EDE not equal 1500.
...              ; Yes, R5 = 1500. Continue
```

Example R7 (20-bit counter) is decremented. If its content is not zero, the program continues at Label4. Program in full memory range.

```
SUBA #1,R7         ; Decrement R7
JNZ  Label4        ; Zero not reached: Go to Label4
...              ; Yes, R7 = 0. Continue here.
```

4.6.2.32 MOV

MOV[.W]	Move source word to destination word
MOV.B	Move source byte to destination byte
Syntax	MOV src,dst or MOV.W src,dst MOV.B src,dst
Operation	src → dst
Description	The source operand is copied to the destination. The source operand is not affected.
Status Bits	N: Not affected Z: Not affected C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Move a 16-bit constant 1800h to absolute address-word EDE (lower 64 K)

```
MOV    #01800h,&EDE           ; Move 1800h to EDE
```

Example The contents of table EDE (word data, 16-bit addresses) are copied to table TOM. The length of the tables is 030h words. Both tables reside in the lower 64 K.

```

Loop   MOV    #EDE,R10          ; Prepare pointer (16-bit address)
        MOV    @R10+,TOM-EDE-2(R10) ; R10 points to both tables.
        ; R10+2
        CMP    #EDE+60h,R10      ; End of table reached?
        JLO    Loop             ; Not yet
        ...                     ; Copy completed
```

Example The contents of table EDE (byte data, 16-bit addresses) are copied to table TOM. The length of the tables is 020h bytes. Both tables may reside in full memory range, but must be within R10 ± 32 K.

```

Loop   MOVA    #EDE,R10          ; Prepare pointer (20-bit)
        MOV    #20h,R9           ; Prepare counter
        MOV.B  @R10+,TOM-EDE-1(R10) ; R10 points to both tables.
        ; R10+1
        DEC    R9                ; Decrement counter
        JNZ    Loop             ; Not yet done
        ...                     ; Copy completed
```

4.6.2.33 NOP

* NOP	No operation
Syntax	NOP
Operation	None
Emulation	MOV #0, R3
Description	No operation is performed. The instruction may be used for the elimination of instructions during the software check or for defined waiting times.
Status Bits	Status bits are not affected.

4.6.2.34 POP

* POP[.W]	Pop word from stack to destination
* POP.B	Pop byte from stack to destination
Syntax	POP dst POP.B dst
Operation	@SP → temp SP + 2 → SP temp → dst
Emulation	MOV @SP+,dst or MOV.W @SP+,dst MOV.B @SP+,dst
Description	The stack location pointed to by the SP (TOS) is moved to the destination. The SP is incremented by 2 afterwards.
Status Bits	Status bits are not affected.
Example	The contents of R7 and the SR are restored from the stack.

```
POP    R7        ; Restore R7
POP    SR        ; Restore status register
```

Example The contents of RAM byte LEO is restored from the stack.

```
POP.B  LEO        ; The low byte of the stack is moved to LEO.
```

Example The contents of R7 is restored from the stack.

```
POP.B  R7          ; The low byte of the stack is moved to R7,
                   ; the high byte of R7 is 00h
```

Example The contents of the memory pointed to by R7 and the SR are restored from the stack.

```
POP.B  0(R7)       ; The low byte of the stack is moved to the
                   ; the byte which is pointed to by R7
                   : Example:   R7 = 203h
                   ;           Mem(R7) = low byte of system stack
                   : Example:   R7 = 20Ah
                   ;           Mem(R7) = low byte of system stack
POP     SR          ; Last word on stack moved to the SR
```

NOTE: System stack pointer

The system SP is always incremented by 2, independent of the byte suffix.

4.6.2.35 PUSH

PUSH[.W] Save a word on the stack

PUSH.B Save a byte on the stack

Syntax PUSH dst or PUSH.W dst
PUSH.B dst

Operation SP – 2 → SP
dst → @SP

Description The 20-bit SP is decremented by 2. The operand is then copied to the RAM word addressed by the SP. A pushed byte is stored in the low byte; the high byte is not affected.

Status Bits Status bits are not affected.

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example Save the 2 16-bit registers R9 and R10 on the stack

```
PUSH    R9        ; Save R9 and R10 XXXXh
PUSH    R10       ; YYYYYh
```

Example Save the 2 bytes EDE and TONI on the stack. The addresses EDE and TONI are within PC ± 32 K.

```
PUSH.B  EDE       ; Save EDE   xxXXh
PUSH.B  TONI      ; Save TONI  xxYYh
```

4.6.2.36 RET

* RET	Return from subroutine
Syntax	RET
Operation	@SP → PC.15:0 Saved PC to PC.15:0. PC.19:16 ← 0 SP + 2 → SP
Description	The 16-bit return address (lower 64 K), pushed onto the stack by a CALL instruction is restored to the PC. The program continues at the address following the subroutine call. The 4 MSBs of the PC.19:16 are cleared.
Status Bits	Status bits are not affected. PC.19:16: Cleared
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Call a subroutine SUBR in the lower 64 K and return to the address in the lower 64 K after the CALL.

```

CALL    #SUBR      ; Call subroutine starting at SUBR
...      ; Return by RET to here
SUBR    PUSH    R14  ; Save R14 (16 bit data)
...      ; Subroutine code
POP     R14        ; Restore R14
RET      ; Return to lower 64 K

```

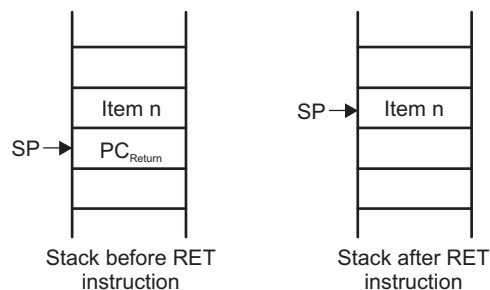


Figure 4-37. Stack After a RET Instruction

4.6.2.37 RETI

RETI	Return from interrupt		
Syntax	RETI		
Operation	@SP → SR.15:0	Restore saved SR with PC.19:16	
	SP + 2 → SP		
	@SP → PC.15:0	Restore saved PC.15:0	
	SP + 2 → SP	Housekeeping	
Description	The SR is restored to the value at the beginning of the interrupt service routine. This includes the 4 MSBs of the PC.19:16. The SP is incremented by 2 afterward.		
	The 20-bit PC is restored from PC.19:16 (from same stack location as the status bits) and PC.15:0. The 20-bit PC is restored to the value at the beginning of the interrupt service routine. The program continues at the address following the last executed instruction when the interrupt was granted. The SP is incremented by 2 afterward. No interrupt flags are modified by this command.		
Status Bits	N:	Restored from stack	
	C:	Restored from stack	
	Z:	Restored from stack	
	V:	Restored from stack	
Mode Bits	OSCOFF, CPUOFF, and GIE are restored from stack.		
Example	Interrupt handler in the lower 64 K. A 20-bit return address is stored on the stack.		
INTRPT	PUSHM.A	#2, R14	; Save R14 and R13 (20-bit data)
	...		; Interrupt handler code
	POPM.A	#2, R14	; Restore R13 and R14 (20-bit data)
	RETI		; Return to 20-bit address in full memory range

4.6.2.38 RLA

*** RLA[.W]** Rotate left arithmetically

*** RLA.B** Rotate left arithmetically

Syntax RLA dst or RLA.W dst
RLA.B dst

Operation $C \leftarrow \text{MSB} \leftarrow \text{MSB}-1 \dots \text{LSB}+1 \leftarrow \text{LSB} \leftarrow 0$

Emulation ADD dst, dst
ADD.B dst, dst

Description The destination operand is shifted left 1 position as shown in Figure 4-38. The MSB is shifted into the carry bit (C) and the LSB is filled with 0. The RLA instruction acts as a signed multiplication by 2.
An overflow occurs if $\text{dst} \geq 04000\text{h}$ and $\text{dst} < 0\text{C}000\text{h}$ before operation is performed; the result has changed sign.

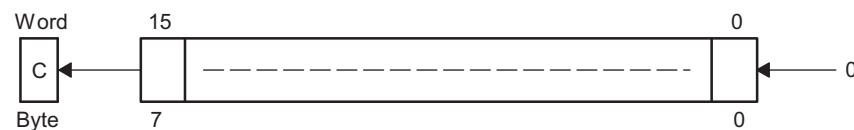


Figure 4-38. Destination Operand—Arithmetic Shift Left

An overflow occurs if $\text{dst} \geq 040\text{h}$ and $\text{dst} < 0\text{C}0\text{h}$ before the operation is performed; the result has changed sign.

Status Bits

- N: Set if result is negative, reset if positive
- Z: Set if result is zero, reset otherwise
- C: Loaded from the MSB
- V: Set if an arithmetic overflow occurs; the initial value is $04000\text{h} \leq \text{dst} < 0\text{C}000\text{h}$, reset otherwise
- Set if an arithmetic overflow occurs; the initial value is $040\text{h} \leq \text{dst} < 0\text{C}0\text{h}$, reset otherwise

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example R7 is multiplied by 2.

```
RLA    R7    ; Shift left R7 (x 2)
```

Example The low byte of R7 is multiplied by 4.

```
RLA.B  R7    ; Shift left low byte of R7 (x 2)
RLA.B  R7    ; Shift left low byte of R7 (x 4)
```

NOTE: RLA substitution

The assembler does not recognize the instructions:

```
RLA    @R5+      RLA.B    @R5+      RLA(.B) @R5
```

They must be substituted by:

```
ADD    @R5+, -2(R5)  ADD.B    @R5+, -1(R5)  ADD(.B) @R5
```


4.6.2.39 RLC

*** RLC[.W]** Rotate left through carry

*** RLC.B** Rotate left through carry

Syntax RLC dst or RLC.W dst
RLC.B dst

Operation $C \leftarrow \text{MSB} \leftarrow \text{MSB}-1 \dots \text{LSB}+1 \leftarrow \text{LSB} \leftarrow C$

Emulation ADDC dst, dst

Description The destination operand is shifted left 1 position as shown in Figure 4-39. The carry bit (C) is shifted into the LSB, and the MSB is shifted into the carry bit (C).

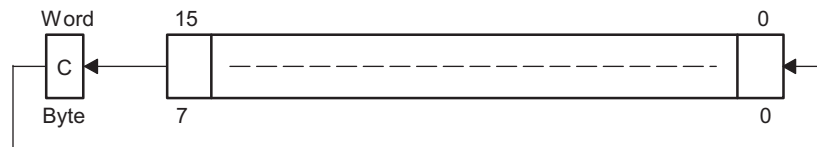


Figure 4-39. Destination Operand—Carry Left Shift

Status Bits

- N: Set if result is negative, reset if positive
- Z: Set if result is zero, reset otherwise
- C: Loaded from the MSB
- V: Set if an arithmetic overflow occurs; the initial value is $04000h \leq \text{dst} < 0C000h$, reset otherwise
Set if an arithmetic overflow occurs; the initial value is $040h \leq \text{dst} < 0C0h$, reset otherwise

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example R5 is shifted left 1 position.

```
RLC    R5          ; (R5 x 2) + C -> R5
```

Example The input P1IN.1 information is shifted into the LSB of R5.

```
BIT.B  #2,&P1IN    ; Information -> Carry
RLC    R5          ; Carry=P0in.1 -> LSB of R5
```

Example The MEM(LE0) content is shifted left 1 position.

```
RLC.B  LE0         ; Mem(LE0) x 2 + C -> Mem(LE0)
```

NOTE: RLA substitution

The assembler does not recognize the instructions:

```
RLC    @R5+        RLC.B  @R5+        RLC(.B) @R5
```

They must be substituted by:

```
ADDC   @R5+, -2(R5)  ADDC.B  @R5+, -1(R5)  ADDC(.B) @R5
```

4.6.2.40 RRA

RRA[.W]	Rotate right arithmetically destination word
RRA.B	Rotate right arithmetically destination byte
Syntax	RRA.B dst or RRA.W dst
Operation	MSB → MSB → MSB-1 → ... LSB+1 → LSB → C
Description	The destination operand is shifted right arithmetically by 1 bit position as shown in Figure 4-40 . The MSB retains its value (sign). RRA operates equal to a signed division by 2. The MSB is retained and shifted into the MSB-1. The LSB+1 is shifted into the LSB. The previous LSB is shifted into the carry bit C.
Status Bits	N: Set if result is negative (MSB = 1), reset otherwise (MSB = 0) Z: Set if result is zero, reset otherwise C: Loaded from the LSB V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The signed 16-bit number in R5 is shifted arithmetically right 1 position.

```
RRA    R5                ; R5/2 -> R5
```

Example The signed RAM byte EDE is shifted arithmetically right 1 position.

```
RRA.B  EDE                ; EDE/2 -> EDE
```

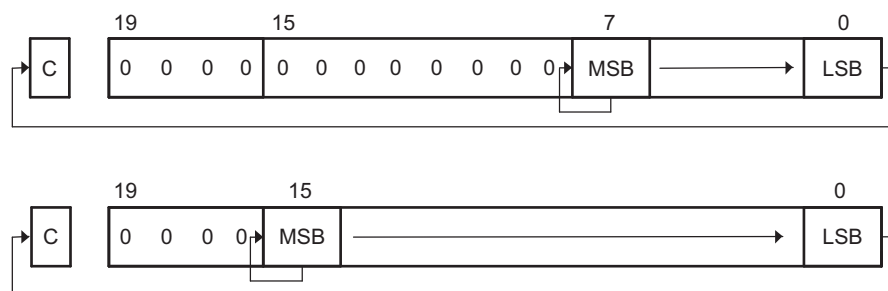


Figure 4-40. Rotate Right Arithmetically RRA.B and RRA.W

4.6.2.41 RRC

RRC[.W]	Rotate right through carry destination word
RRC.B	Rotate right through carry destination byte
Syntax	RRC dst or RRC.W dst RRC.B dst
Operation	$C \rightarrow \text{MSB} \rightarrow \text{MSB}-1 \rightarrow \dots \text{LSB}+1 \rightarrow \text{LSB} \rightarrow C$
Description	The destination operand is shifted right by 1 bit position as shown in Figure 4-41 . The carry bit C is shifted into the MSB and the LSB is shifted into the carry bit C.
Status Bits	N: Set if result is negative (MSB = 1), reset otherwise (MSB = 0) Z: Set if result is zero, reset otherwise C: Loaded from the LSB V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	RAM word EDE is shifted right 1 bit position. The MSB is loaded with 1.

```
SETC      ; Prepare carry for MSB
RRC  EDE  ; EDE = EDE >> 1 + 8000h
```

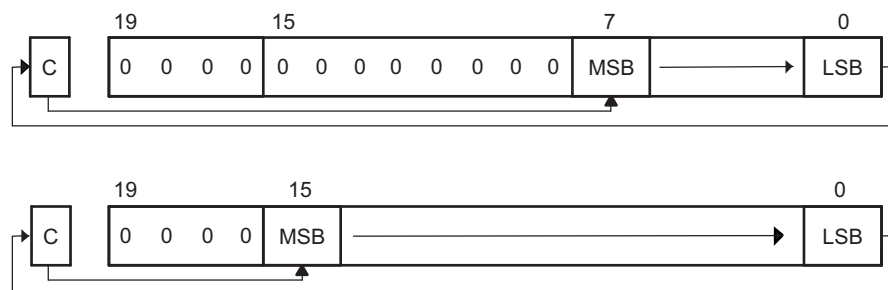


Figure 4-41. Rotate Right Through Carry RRC.B and RRC.W

4.6.2.42 SBC

* SBC[.W]	Subtract borrow (.NOT. carry) from destination
* SBC.B	Subtract borrow (.NOT. carry) from destination
Syntax	SBC dst or SBC.W dst SBC.B dst
Operation	dst + 0FFFFh + C → dst dst + 0FFh + C → dst
Emulation	SUBC #0, dst SUBC.B #0, dst
Description	The carry bit (C) is added to the destination operand minus 1. The previous contents of the destination are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if result is zero, reset otherwise C: Set if there is a carry from the MSB of the result, reset otherwise Set to 1 if no borrow, reset if borrow V: Set if an arithmetic overflow occurs, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 16-bit counter pointed to by R13 is subtracted from a 32-bit counter pointed to by R12.

```

SUB    @R13,0(R12)    ; Subtract LSDs
SBC    2(R12)         ; Subtract carry from MSD

```

Example The 8-bit counter pointed to by R13 is subtracted from a 16-bit counter pointed to by R12.

```

SUB.B   @R13,0(R12)    ; Subtract LSDs
SBC.B   1(R12)         ; Subtract carry from MSD

```

NOTE: Borrow implementation

The borrow is treated as a .NOT. carry:

Borrow	Carry Bit
Yes	0
No	1

4.6.2.43 SETC

* SETC	Set carry bit
Syntax	SETC
Operation	$1 \rightarrow C$
Emulation	BIS #1, SR
Description	The carry bit (C) is set.
Status Bits	N: Not affected
	Z: Not affected
	C: Set
	V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Emulation of the decimal subtraction:
	Subtract R5 from R6 decimally.
	Assume that R5 = 03987h and R6 = 04137h.

```

DSUB  ADD    #06666h,R5    ; Move content R5 from 0-9 to 6-0Fh
                                ; R5 = 03987h + 06666h = 09FEDh
                                ; Invert this (result back to 0-9)
                                ; R5 = .NOT. R5 = 06012h
                                ; Prepare carry = 1
                                ; Emulate subtraction by addition of:
                                ; (010000h - R5 - 1)
                                ; R6 = R6 + R5 + 1
                                ; R6 = 0150h

```

4.6.2.44 SETN

* SETN	Set negative bit
Syntax	SETN
Operation	$1 \rightarrow N$
Emulation	BIS #4, SR
Description	The negative bit (N) is set.
Status Bits	N: Set
	Z: Not affected
	C: Not affected
	V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.

4.6.2.45 SETZ

* SETZ	Set zero bit
Syntax	SETZ
Operation	$1 \rightarrow N$
Emulation	BIS #2, SR
Description	The zero bit (Z) is set.
Status Bits	N: Not affected
	Z: Set
	C: Not affected
	V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.

4.6.2.46 SUB

SUB[.W]	Subtract source word from destination word
SUB.B	Subtract source byte from destination byte
Syntax	SUB src,dst or SUB.W src,dst SUB.B src,dst
Operation	$(\text{not.src}) + 1 + \text{dst} \rightarrow \text{dst}$ or $\text{dst} - \text{src} \rightarrow \text{dst}$
Description	The source operand is subtracted from the destination operand. This is made by adding the 1s complement of the source + 1 to the destination. The source operand is not affected, the result is written to the destination operand.
Status Bits	N: Set if result is negative ($\text{src} > \text{dst}$), reset if positive ($\text{src} \leq \text{dst}$) Z: Set if result is zero ($\text{src} = \text{dst}$), reset otherwise ($\text{src} \neq \text{dst}$) C: Set if there is a carry from the MSB, reset otherwise V: Set if the subtraction of a negative source operand from a positive destination operand delivers a negative result, or if the subtraction of a positive source operand from a negative destination operand delivers a positive result, reset otherwise (no overflow)
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	A 16-bit constant 7654h is subtracted from RAM word EDE.
	<pre>SUB #7654h,&EDE ; Subtract 7654h from EDE</pre>
Example	A table word pointed to by R5 (20-bit address) is subtracted from R7. Afterwards, if R7 contains zero, jump to label TONI. R5 is then auto-incremented by 2. $\text{R7.19:16} = 0$.
	<pre>SUB @R5+,R7 ; Subtract table number from R7. R5 + 2 JZ TONI ; R7 = @R5 (before subtraction) ... ; R7 <> @R5 (before subtraction)</pre>
Example	Byte CNT is subtracted from byte R12 points to. The address of CNT is within $\text{PC} \pm 32\text{K}$. The address R12 points to is in full memory range.
	<pre>SUB.B CNT,0(R12) ; Subtract CNT from @R12</pre>

4.6.2.47 SUBC

SUBC[.W]	Subtract source word with carry from destination word
SUBC.B	Subtract source byte with carry from destination byte
Syntax	SUBC src,dst or SUBC.W src,dst SUBC.B src,dst
Operation	$(\text{not.src}) + C + \text{dst} \rightarrow \text{dst}$ or $\text{dst} - (\text{src} - 1) + C \rightarrow \text{dst}$
Description	The source operand is subtracted from the destination operand. This is done by adding the 1s complement of the source + carry to the destination. The source operand is not affected, the result is written to the destination operand. Used for 32, 48, and 64-bit operands.
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if there is a carry from the MSB, reset otherwise V: Set if the subtraction of a negative source operand from a positive destination operand delivers a negative result, or if the subtraction of a positive source operand from a negative destination operand delivers a positive result, reset otherwise (no overflow)
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	A 16-bit constant 7654h is subtracted from R5 with the carry from the previous instruction. R5.19:16 = 0
<pre>SUBC.W #7654h,R5 ; Subtract 7654h + C from R5</pre>	
Example	A 48-bit number (3 words) pointed to by R5 (20-bit address) is subtracted from a 48-bit counter in RAM, pointed to by R7. R5 points to the next 48-bit number afterwards. The address R7 points to is in full memory range.
<pre>SUB @R5+,0(R7) ; Subtract LSBs. R5 + 2 SUBC @R5+,2(R7) ; Subtract MIDs with C. R5 + 2 SUBC @R5+,4(R7) ; Subtract MSBs with C. R5 + 2</pre>	
Example	Byte CNT is subtracted from the byte, R12 points to. The carry of the previous instruction is used. The address of CNT is in lower 64 K.
<pre>SUBC.B &CNT,0(R12) ; Subtract byte CNT from @R12</pre>	

4.6.2.48 SWPB

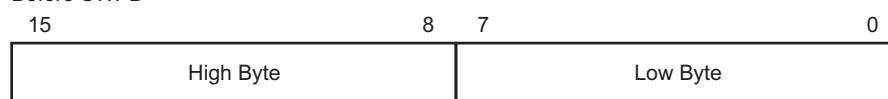
SWPB	Swap bytes
Syntax	SWPB dst
Operation	dst.15:8 ↔ dst.7:0
Description	The high and the low byte of the operand are exchanged. PC.19:16 bits are cleared in register mode.
Status Bits	Status bits are not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Exchange the bytes of RAM word EDE (lower 64 K)

```

MOV    #1234h, &EDE    ; 1234h -> EDE
SWPB   &EDE             ; 3412h -> EDE

```

Before SWPB



After SWPB

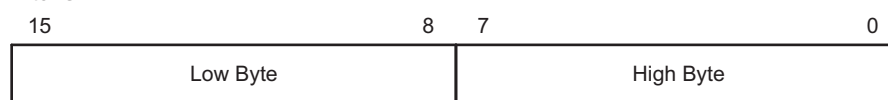
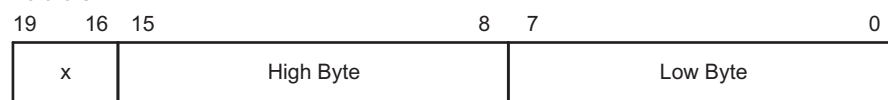


Figure 4-42. Swap Bytes in Memory

Before SWPB



After SWPB

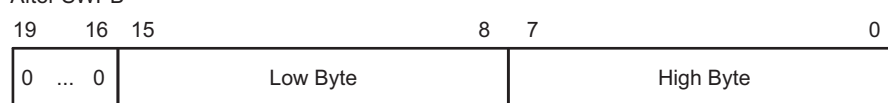


Figure 4-43. Swap Bytes in a Register

4.6.2.49 SXT

SXT	Extend sign	
Syntax	SXT dst	
Operation	dst.7 → dst.15:8, dst.7 → dst.19:8 (register mode)	
Description	Register mode: the sign of the low byte of the operand is extended into the bits Rdst.19:8. Rdst.7 = 0: Rdst.19:8 = 000h afterwards Rdst.7 = 1: Rdst.19:8 = FFFh afterwards Other modes: the sign of the low byte of the operand is extended into the high byte. dst.7 = 0: high byte = 00h afterwards dst.7 = 1: high byte = FFh afterwards	
Status Bits	N: Set if result is negative, reset otherwise Z: Set if result is zero, reset otherwise C: Set if result is not zero, reset otherwise (C = .not.Z) V: Reset	
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.	
Example	The signed 8-bit data in EDE (lower 64 K) is sign extended and added to the 16-bit signed data in R7.	
	MOV.B	&EDE, R5 ; EDE -> R5. 00XXh
	SXT	R5 ; Sign extend low byte to R5.19:8
	ADD	R5, R7 ; Add signed 16-bit values
Example	The signed 8-bit data in EDE (PC +32 K) is sign extended and added to the 20-bit data in R7.	
	MOV.B	EDE, R5 ; EDE -> R5. 00XXh
	SXT	R5 ; Sign extend low byte to R5.19:8
	ADDA	R5, R7 ; Add signed 20-bit values

4.6.2.50 TST

* TST[.W]	Test destination
* TST.B	Test destination
Syntax	TST dst or TST.W dst TST.B dst
Operation	dst + 0FFFFh + 1 dst + 0FFh + 1
Emulation	CMP #0, dst CMP.B #0, dst
Description	The destination operand is compared with zero. The status bits are set according to the result. The destination is not affected.
Status Bits	N: Set if destination is negative, reset if positive Z: Set if destination contains zero, reset otherwise C: Set V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	R7 is tested. If it is negative, continue at R7NEG; if it is positive but not zero, continue at R7POS.

```

TST    R7        ; Test R7
JN     R7NEG     ; R7 is negative
JZ     R7ZERO    ; R7 is zero
R7POS  .....    ; R7 is positive but not zero
R7NEG  .....    ; R7 is negative
R7ZERO .....    ; R7 is zero

```

Example The low byte of R7 is tested. If it is negative, continue at R7NEG; if it is positive but not zero, continue at R7POS.

```

TST.B  R7        ; Test low byte of R7
JN     R7NEG     ; Low byte of R7 is negative
JZ     R7ZERO    ; Low byte of R7 is zero
R7POS  .....    ; Low byte of R7 is positive but not zero
R7NEG  .....    ; Low byte of R7 is negative
R7ZERO .....    ; Low byte of R7 is zero

```

4.6.2.51 XOR

XOR[.W]	Exclusive OR source word with destination word	
XOR.B	Exclusive OR source byte with destination byte	
Syntax	XOR src,dst or XOR.W src,dst XOR.B src,dst	
Operation	src .xor. dst → dst	
Description	The source and destination operands are exclusively ORed. The result is placed into the destination. The source operand is not affected. The previous content of the destination is lost.	
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if result is not zero, reset otherwise (C = .not. Z) V: Set if both operands are negative before execution, reset otherwise	
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.	
Example	Toggle bits in word CNTR (16-bit data) with information (bit = 1) in address-word TONI. Both operands are located in lower 64 K.	
	XOR	&TONI,&CNTR ; Toggle bits in CNTR
Example	A table word pointed to by R5 (20-bit address) is used to toggle bits in R6. R6.19:16 = 0.	
	XOR	@R5,R6 ; Toggle bits in R6
Example	Reset to zero those bits in the low byte of R7 that are different from the bits in byte EDE. R7.19:8 = 0. The address of EDE is within PC ± 32 K.	
	XOR.B	EDE,R7 ; Set different bits to 1 in R7.
	INV.B	R7 ; Invert low byte of R7, high byte is 0h

4.6.3 Extended Instructions

The extended MSP430X instructions give the MSP430X CPU full access to its 20-bit address space. MSP430X instructions require an additional word of op-code called the extension word. All addresses, indexes, and immediate numbers have 20-bit values when preceded by the extension word. The MSP430X extended instructions are listed and described in the following pages.

4.6.3.1 ADCX

* ADCX.A	Add carry to destination address-word
* ADCX.[W]	Add carry to destination word
* ADCX.B	Add carry to destination byte
Syntax	ADCX.A dst ADCX dst or ADCX.W dst ADCX.B dst
Operation	$dst + C \rightarrow dst$
Emulation	ADDCX.A #0, dst ADDCX #0, dst ADDCX.B #0, dst
Description	The carry bit (C) is added to the destination operand. The previous contents of the destination are lost.
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if there is a carry from the MSB of the result, reset otherwise V: Set if the result of 2 positive operands is negative, or if the result of 2 negative numbers is positive, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 40-bit counter, pointed to by R12 and R13, is incremented.
	<pre> INCX.A @R12 ; Increment lower 20 bits ADCX.A @R13 ; Add carry to upper 20 bits </pre>

4.6.3.2 ADDX

ADDX.A	Add source address-word to destination address-word
ADDX.[W]	Add source word to destination word
ADDX.B	Add source byte to destination byte
Syntax	ADDX.A src,dst ADDX src,dst or ADDX.W src,dst ADDX.B src,dst
Operation	src + dst → dst
Description	The source operand is added to the destination operand. The previous contents of the destination are lost. Both operands can be located in the full address space.
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if there is a carry from the MSB of the result, reset otherwise V: Set if the result of 2 positive operands is negative, or if the result of 2 negative numbers is positive, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	10 is added to the 20-bit pointer CNTR located in 2 words CNTR (LSBs) and CNTR+2 (MSBs).

```
ADDX.A    #10,CNTR    ; Add 10 to 20-bit pointer
```

Example A table word (16-bit) pointed to by R5 (20-bit address) is added to R6. The jump to label TONI is performed on a carry.

```
ADDX.W    @R5,R6      ; Add table word to R6
JC         TONI        ; Jump if carry
...       ; No carry
```

Example A table byte pointed to by R5 (20-bit address) is added to R6. The jump to label TONI is performed if no carry occurs. The table pointer is auto-incremented by 1.

```
ADDX.B    @R5+,R6     ; Add table byte to R6. R5 + 1. R6: 000xxh
JNC       TONI        ; Jump if no carry
...       ; Carry occurred
```

Note: Use ADDA for the following 2 cases for better code density and execution.

```
ADDX.A    Rsrc,Rdst
ADDX.A    #imm20,Rdst
```


4.6.3.3 ADDCX

ADDCX.A	Add source address-word and carry to destination address-word
ADDCX.[W]	Add source word and carry to destination word
ADDCX.B	Add source byte and carry to destination byte
Syntax	ADDCX.A src,dst ADDCX src,dst or ADDCX.W src,dst ADDCX.B src,dst
Operation	$\text{src} + \text{dst} + \text{C} \rightarrow \text{dst}$
Description	The source operand and the carry bit C are added to the destination operand. The previous contents of the destination are lost. Both operands may be located in the full address space.
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if there is a carry from the MSB of the result, reset otherwise V: Set if the result of 2 positive operands is negative, or if the result of 2 negative numbers is positive, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Constant 15 and the carry of the previous instruction are added to the 20-bit counter CNTR located in 2 words.

```
ADDCX.A    #15,&CNTR    ; Add 15 + C to 20-bit CNTR
```

Example A table word pointed to by R5 (20-bit address) and the carry C are added to R6. The jump to label TONI is performed on a carry.

```
ADDCX.W    @R5,R6      ; Add table word + C to R6
JC         TONI        ; Jump if carry
...        ; No carry
```

Example A table byte pointed to by R5 (20-bit address) and the carry bit C are added to R6. The jump to label TONI is performed if no carry occurs. The table pointer is auto-incremented by 1.

```
ADDCX.B    @R5+,R6     ; Add table byte + C to R6. R5 + 1
JNC        TONI        ; Jump if no carry
...        ; Carry occurred
```

4.6.3.4 ANDX

ANDX.A	Logical AND of source address-word with destination address-word
ANDX.[W]	Logical AND of source word with destination word
ANDX.B	Logical AND of source byte with destination byte
Syntax	ANDX.A src,dst ANDX src,dst or ANDX.W src,dst ANDX.B src,dst
Operation	src .and. dst → dst
Description	The source operand and the destination operand are logically ANDed. The result is placed into the destination. The source operand is not affected. Both operands may be located in the full address space.
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if the result is not zero, reset otherwise. C = (.not. Z) V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The bits set in R5 (20-bit data) are used as a mask (AAA55h) for the address-word TOM located in 2 words. If the result is zero, a branch is taken to label TONI.

```

MOVA    #AAA55h,R5      ; Load 20-bit mask to R5
ANDX.A  R5,TOM           ; TOM .and. R5 -> TOM
JZ      TONI             ; Jump if result 0
...      ; Result > 0

```

or shorter:

```

ANDX.A  #AAA55h,TOM      ; TOM .and. AAA55h -> TOM
JZ      TONI             ; Jump if result 0

```

Example	A table byte pointed to by R5 (20-bit address) is logically ANDed with R6. R6.19:8 = 0. The table pointer is auto-incremented by 1.

```

ANDX.B  @R5+,R6          ; AND table byte with R6. R5 + 1

```

4.6.3.5 BICX

BICX.A	Clear bits set in source address-word in destination address-word
BICX.[W]	Clear bits set in source word in destination word
BICX.B	Clear bits set in source byte in destination byte
Syntax	BICX.A src,dst BICX src,dst or BICX.W src,dst BICX.B src,dst
Operation	(.not. src) .and. dst → dst
Description	The inverted source operand and the destination operand are logically ANDed. The result is placed into the destination. The source operand is not affected. Both operands may be located in the full address space.
Status Bits	N: Not affected Z: Not affected C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The bits 19:15 of R5 (20-bit data) are cleared.

```
BICX.A    #0F8000h,R5    ; Clear R5.19:15 bits
```

Example A table word pointed to by R5 (20-bit address) is used to clear bits in R7. R7.19:16 = 0.

```
BICX.W    @R5,R7        ; Clear bits in R7
```

Example A table byte pointed to by R5 (20-bit address) is used to clear bits in output Port1.

```
BICX.B    @R5,&P1OUT     ; Clear I/O port P1 bits
```

4.6.3.6 BISX

BISX.A	Set bits set in source address-word in destination address-word
BISX.[W]	Set bits set in source word in destination word
BISX.B	Set bits set in source byte in destination byte
Syntax	BISX.A src,dst BISX src,dst or BISX.W src,dst BISX.B src,dst
Operation	src .or. dst → dst
Description	The source operand and the destination operand are logically ORed. The result is placed into the destination. The source operand is not affected. Both operands may be located in the full address space.
Status Bits	N: Not affected Z: Not affected C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Bits 16 and 15 of R5 (20-bit data) are set to 1.

```
BISX.A    #018000h,R5    ; Set R5.16:15 bits
```

Example A table word pointed to by R5 (20-bit address) is used to set bits in R7.

```
BISX.W    @R5,R7        ; Set bits in R7
```

Example A table byte pointed to by R5 (20-bit address) is used to set bits in output Port1.

```
BISX.B    @R5,&P1OUT     ; Set I/O port P1 bits
```

4.6.3.7 BITX

BITX.A	Test bits set in source address-word in destination address-word
BITX.[W]	Test bits set in source word in destination word
BITX.B	Test bits set in source byte in destination byte
Syntax	BITX.A src,dst BITX src,dst or BITX.W src,dst BITX.B src,dst
Operation	src .and. dst → dst
Description	The source operand and the destination operand are logically ANDed. The result affects only the status bits. Both operands may be located in the full address space.
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if the result is not zero, reset otherwise. C = (.not. Z) V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Test if bit 16 or 15 of R5 (20-bit data) is set. Jump to label TONI if so.

```

BITX.A    #018000h,R5      ; Test R5.16:15 bits
JNZ       TONI             ; At least 1 bit is set
...                          ; Both are reset

```

Example A table word pointed to by R5 (20-bit address) is used to test bits in R7. Jump to label TONI if at least 1 bit is set.

```

BITX.W    @R5,R7           ; Test bits in R7: C = .not.Z
JC        TONI             ; At least 1 is set
...                          ; Both are reset

```

Example A table byte pointed to by R5 (20-bit address) is used to test bits in input Port1. Jump to label TONI if no bit is set. The next table byte is addressed.

```

BITX.B    @R5+,&P1IN       ; Test input P1 bits. R5 + 1
JNC       TONI             ; No corresponding input bit is set
...                          ; At least 1 bit is set

```

4.6.3.8 CLRX

* CLRX.A	Clear destination address-word
* CLRX.[W]	Clear destination word
* CLRX.B	Clear destination byte
Syntax	CLRX.A dst CLRX dst or CLRX.W dst CLRX.B dst
Operation	$0 \rightarrow \text{dst}$
Emulation	MOVX.A #0, dst MOVX #0, dst MOVX.B #0, dst
Description	The destination operand is cleared.
Status Bits	Status bits are not affected.
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	RAM address-word TONI is cleared.

```
CLRX.A    TONI    ; 0 -> TONI
```

4.6.3.9 CMPX

CMPX.A	Compare source address-word and destination address-word
CMPX.[W]	Compare source word and destination word
CMPX.B	Compare source byte and destination byte
Syntax	CMPX.A src,dst CMPX src,dst or CMPX.W src,dst CMPX.B src,dst
Operation	$(\text{not. src}) + 1 + \text{dst}$ or $\text{dst} - \text{src}$
Description	The source operand is subtracted from the destination operand by adding the 1s complement of the source + 1 to the destination. The result affects only the status bits. Both operands may be located in the full address space.
Status Bits	N: Set if result is negative ($\text{src} > \text{dst}$), reset if positive ($\text{src} \leq \text{dst}$) Z: Set if result is zero ($\text{src} = \text{dst}$), reset otherwise ($\text{src} \neq \text{dst}$) C: Set if there is a carry from the MSB, reset otherwise V: Set if the subtraction of a negative source operand from a positive destination operand delivers a negative result, or if the subtraction of a positive source operand from a negative destination operand delivers a positive result, reset otherwise (no overflow)
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Compare EDE with a 20-bit constant 18000h. Jump to label TONI if EDE equals the constant.

```

CMPX.A    #018000h,EDE    ; Compare EDE with 18000h
JEQ       TONI            ; EDE contains 18000h
...       ; Not equal

```

Example A table word pointed to by R5 (20-bit address) is compared with R7. Jump to label TONI if R7 contains a lower, signed, 16-bit number.

```

CMPX.W    @R5,R7          ; Compare 2 signed numbers
JL        TONI            ; R7 < @R5
...       ; R7 >= @R5

```

Example A table byte pointed to by R5 (20-bit address) is compared to the input in I/O Port1. Jump to label TONI if the values are equal. The next table byte is addressed.

```

CMPX.B    @R5+,&P1IN      ; Compare P1 bits with table. R5 + 1
JEQ       TONI            ; Equal contents
...       ; Not equal

```

Note: Use CMPA for the following cases for better density and execution.

```

CMPA      Rsrc,Rdst
CMPA      #imm20,Rdst

```

4.6.3.10 DADCX

* DADCX.A	Add carry decimally to destination address-word
* DADCX.[W]	Add carry decimally to destination word
* DADCX.B	Add carry decimally to destination byte
Syntax	DADCX.A dst DADCX dst or DADCX.W dst DADCX.B dst
Operation	dst + C → dst (decimally)
Emulation	DADDX.A #0, dst DADDX #0, dst DADDX.B #0, dst
Description	The carry bit (C) is added decimally to the destination.
Status Bits	N: Set if MSB of result is 1 (address-word > 79999h, word > 7999h, byte > 79h), reset if MSB is 0 Z: Set if result is zero, reset otherwise C: Set if the BCD result is too large (address-word > 99999h, word > 9999h, byte > 99h), reset otherwise V: Undefined
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 40-bit counter, pointed to by R12 and R13, is incremented decimally. DADDX.A #1, 0(R12) ; Increment lower 20 bits DADCX.A 0(R13) ; Add carry to upper 20 bits

4.6.3.11 DADDX

DADDX.A	Add source address-word and carry decimally to destination address-word
DADDX.[W]	Add source word and carry decimally to destination word
DADDX.B	Add source byte and carry decimally to destination byte
Syntax	DADDX.A src,dst DADDX src,dst or DADDX.W src,dst DADDX.B src,dst
Operation	src + dst + C → dst (decimally)
Description	The source operand and the destination operand are treated as 2 (.B), 4 (.W), or 5 (.A) binary coded decimals (BCD) with positive signs. The source operand and the carry bit C are added decimally to the destination operand. The source operand is not affected. The previous contents of the destination are lost. The result is not defined for non-BCD numbers. Both operands may be located in the full address space.
Status Bits	N: Set if MSB of result is 1 (address-word > 79999h, word > 7999h, byte > 79h), reset if MSB is 0. Z: Set if result is zero, reset otherwise C: Set if the BCD result is too large (address-word > 99999h, word > 9999h, byte > 99h), reset otherwise V: Undefined
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Decimal 10 is added to the 20-bit BCD counter DECCNTR located in 2 words.

```
DADDX.A    #10h,&DECCNTR    ; Add 10 to 20-bit BCD counter
```

Example The 8-digit BCD number contained in 20-bit addresses BCD and BCD+2 is added decimally to an 8-digit BCD number contained in R4 and R5 (BCD+2 and R5 contain the MSDs).

```
CLRC                                ; Clear carry
DADDX.W    BCD,R4                   ; Add LSDs
DADDX.W    BCD+2,R5                 ; Add MSDs with carry
JC         OVERFLOW                 ; Result >99999999: go to error routine
...                                     ; Result ok
```

Example The 2-digit BCD number contained in 20-bit address BCD is added decimally to a 2-digit BCD number contained in R4.

```
CLRC                                ; Clear carry
DADDX.B    BCD,R4                   ; Add BCD to R4 decimally.
; R4: 000ddh
```

4.6.3.12 DECX

* DECX.A	Decrement destination address-word
* DECX.[W]	Decrement destination word
* DECX.B	Decrement destination byte
Syntax	DECX.A dst DECX dst or DECX.W dst DECX.B dst
Operation	$dst - 1 \rightarrow dst$
Emulation	SUBX.A #1, dst SUBX #1, dst SUBX.B #1, dst
Description	The destination operand is decremented by 1. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if dst contained 1, reset otherwise C: Reset if dst contained 0, set otherwise V: Set if an arithmetic overflow occurs, otherwise reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	RAM address-word TONI is decremented by 1.

```
DECX.A    TONI    ; Decrement TONI
```

4.6.3.13 DECDX

* DECDX.A	Double-decrement destination address-word
* DECDX.[W]	Double-decrement destination word
* DECDX.B	Double-decrement destination byte
Syntax	DECDX.A dst DECDX dst or DECDX.W dst DECDX.B dst
Operation	$dst - 2 \rightarrow dst$
Emulation	SUBX.A #2, dst SUBX #2, dst SUBX.B #2, dst
Description	The destination operand is decremented by 2. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if dst contained 2, reset otherwise C: Reset if dst contained 0 or 1, set otherwise V: Set if an arithmetic overflow occurs, otherwise reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	RAM address-word TONI is decremented by 2.

```
DECDX.A    TONI    ; Decrement TONI
```

4.6.3.14 INCX

* INCX.A	Increment destination address-word
* INCX.[W]	Increment destination word
* INCX.B	Increment destination byte
Syntax	INCX.A dst INCX dst or INCX.W dst INCX.B dst
Operation	$dst + 1 \rightarrow dst$
Emulation	ADDX.A #1, dst ADDX #1, dst ADDX.B #1, dst
Description	The destination operand is incremented by 1. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: .A: Set if dst contained 0FFFFFFh, reset otherwise .W: Set if dst contained 0FFFFFFh, reset otherwise .B: Set if dst contained 0FFh, reset otherwise C: .A: Set if dst contained 0FFFFFFh, reset otherwise .W: Set if dst contained 0FFFFFFh, reset otherwise .B: Set if dst contained 0FFh, reset otherwise V: .A: Set if dst contained 07FFFh, reset otherwise .W: Set if dst contained 07FFFh, reset otherwise .B: Set if dst contained 07Fh, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	RAM address-wordTONI is incremented by 1.
	INCX.A TONI ; Increment TONI (20-bits)

4.6.3.15 INCDX

* INCDX.A	Double-increment destination address-word
* INCDX.[W]	Double-increment destination word
* INCDX.B	Double-increment destination byte
Syntax	INCDX.A dst INCDX dst or INCDX.W dst INCDX.B dst
Operation	dst + 2 → dst
Emulation	ADDX.A #2, dst ADDX #2, dst ADDX.B #2, dst
Description	The destination operand is incremented by 2. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: .A: Set if dst contained 0FFFFFFEh, reset otherwise .W: Set if dst contained 0FFFEh, reset otherwise .B: Set if dst contained 0FEh, reset otherwise C: .A: Set if dst contained 0FFFFFFEh or 0FFFFFFh, reset otherwise .W: Set if dst contained 0FFFEh or 0FFFFh, reset otherwise .B: Set if dst contained 0FEh or 0FFh, reset otherwise V: .A: Set if dst contained 07FFFEh or 07FFFFh, reset otherwise .W: Set if dst contained 07FFFEh or 07FFFh, reset otherwise .B: Set if dst contained 07Eh or 07Fh, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	RAM byte LEO is incremented by 2; PC points to upper memory.
	INCDX.B LEO ; Increment LEO by 2

4.6.3.16 INVX

* INVX.A	Invert destination
* INVX.[W]	Invert destination
* INVX.B	Invert destination
Syntax	INVX.A dst INVX dst or INVX.W dst INVX.B dst
Operation	.NOT.dst → dst
Emulation	XORX.A #0FFFFFFh, dst XORX #0FFFFFFh, dst XORX.B #0FFh, dst
Description	The destination operand is inverted. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: .A: Set if dst contained 0FFFFFFh, reset otherwise .W: Set if dst contained 0FFFFh, reset otherwise .B: Set if dst contained 0FFh, reset otherwise C: Set if result is not zero, reset otherwise (= .NOT. Zero) V: Set if initial destination operand was negative, otherwise reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	20-bit content of R5 is negated (twos complement).

```

INVX.A    R5        ; Invert R5
INCX.A    R5        ; R5 is now negated

```

Example Content of memory byte LEO is negated. PC is pointing to upper memory.

```

INVX.B    LEO       ; Invert LEO
INCX.B    LEO       ; MEM(LEO) is negated

```

4.6.3.17 MOVX

MOVX.A	Move source address-word to destination address-word
MOVX.[W]	Move source word to destination word
MOVX.B	Move source byte to destination byte
Syntax	MOVX.A src,dst MOVX src,dst or MOVX.W src,dst MOVX.B src,dst
Operation	src → dst
Description	The source operand is copied to the destination. The source operand is not affected. Both operands may be located in the full address space.
Status Bits	N: Not affected Z: Not affected C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Move a 20-bit constant 18000h to absolute address-word EDE

```
MOVX.A    #018000h,&EDE          ; Move 18000h to EDE
```

Example The contents of table EDE (word data, 20-bit addresses) are copied to table TOM. The length of the table is 030h words.

```

Loop      MOVA    #EDE,R10          ; Prepare pointer (20-bit address)
          MOVX.W  @R10+,TOM-EDE-2(R10) ; R10 points to both tables.
          ; R10+2
          CMPA    #EDE+60h,R10      ; End of table reached?
          JLO     Loop              ; Not yet
          ...                      ; Copy completed
```

Example The contents of table EDE (byte data, 20-bit addresses) are copied to table TOM. The length of the table is 020h bytes.

```

Loop      MOVA    #EDE,R10          ; Prepare pointer (20-bit)
          MOV     #20h,R9           ; Prepare counter
          MOVX.W  @R10+,TOM-EDE-2(R10) ; R10 points to both tables.
          ; R10+1
          DEC     R9                ; Decrement counter
          JNZ     Loop              ; Not yet done
          ...                      ; Copy completed
```

10 of the 28 possible addressing combinations of the MOVX.A instruction can use the MOVA instruction. This saves 2 bytes and code cycles. Examples for the addressing combinations are:

MOVX.A	Rsrc,Rdst	MOVA	Rsrc,Rdst	; Reg/Reg
MOVX.A	#imm20,Rdst	MOVA	#imm20,Rdst	; Immediate/Reg
MOVX.A	&abs20,Rdst	MOVA	&abs20,Rdst	; Absolute/Reg
MOVX.A	@Rsrc,Rdst	MOVA	@Rsrc,Rdst	; Indirect/Reg
MOVX.A	@Rsrc+,Rdst	MOVA	@Rsrc+,Rdst	; Indirect,Auto/Reg
MOVX.A	Rsrc,&abs20	MOVA	Rsrc,&abs20	; Reg/Absolute

The next 4 replacements are possible only if 16-bit indexes are sufficient for the addressing:

MOVX.A	z20(Rsrc),Rdst	MOVA	z16(Rsrc),Rdst	; Indexed/Reg
MOVX.A	Rsrc,z20(Rdst)	MOVA	Rsrc,z16(Rdst)	; Reg/Indexed
MOVX.A	symb20,Rdst	MOVA	symb16,Rdst	; Symbolic/Reg
MOVX.A	Rsrc,symb20	MOVA	Rsrc,symb16	; Reg/Symbolic

4.6.3.18 POPM

POPM.A	Restore n CPU registers (20-bit data) from the stack
POPM.[W]	Restore n CPU registers (16-bit data) from the stack
Syntax	POPM.A #n,Rdst $1 \leq n \leq 16$
	POPM.W #n,Rdst or POPM #n,Rdst $1 \leq n \leq 16$
Operation	POPM.A: Restore the register values from stack to the specified CPU registers. The SP is incremented by 4 for each register restored from stack. The 20-bit values from stack (2 words per register) are restored to the registers. POPM.W: Restore the 16-bit register values from stack to the specified CPU registers. The SP is incremented by 2 for each register restored from stack. The 16-bit values from stack (1 word per register) are restored to the CPU registers. Note : This instruction does not use the extension word.
Description	POPM.A: The CPU registers pushed on the stack are moved to the extended CPU registers, starting with the CPU register (Rdst – n + 1). The SP is incremented by (n × 4) after the operation. POPM.W: The 16-bit registers pushed on the stack are moved back to the CPU registers, starting with CPU register (Rdst – n + 1). The SP is incremented by (n × 2) after the instruction. The MSBs (Rdst.19:16) of the restored CPU registers are cleared.
Status Bits	Status bits are not affected, except SR is included in the operation.
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Restore the 20-bit registers R9, R10, R11, R12, R13 from the stack
	<pre>POPM.A #5,R13 ; Restore R9, R10, R11, R12, R13</pre>
Example	Restore the 16-bit registers R9, R10, R11, R12, R13 from the stack.
	<pre>POPM.W #5,R13 ; Restore R9, R10, R11, R12, R13</pre>

4.6.3.19 PUSHM

PUSHM.A	Save n CPU registers (20-bit data) on the stack
PUSHM.[W]	Save n CPU registers (16-bit words) on the stack
Syntax	PUSHM.A #n, Rdst $1 \leq n \leq 16$
	PUSHM.W #n, Rdst or PUSHM #n, Rdst $1 \leq n \leq 16$
Operation	PUSHM.A: Save the 20-bit CPU register values on the stack. The SP is decremented by 4 for each register stored on the stack. The MSBs are stored first (higher address). PUSHM.W: Save the 16-bit CPU register values on the stack. The SP is decremented by 2 for each register stored on the stack.
Description	PUSHM.A: The n CPU registers, starting with Rdst backwards, are stored on the stack. The SP is decremented by $(n \times 4)$ after the operation. The data (Rn.19:0) of the pushed CPU registers is not affected. PUSHM.W: The n registers, starting with Rdst backwards, are stored on the stack. The SP is decremented by $(n \times 2)$ after the operation. The data (Rn.19:0) of the pushed CPU registers is not affected. Note : This instruction does not use the extension word.
Status Bits	Status bits are not affected.
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Save the 5 20-bit registers R9, R10, R11, R12, R13 on the stack
	<pre>PUSHM.A #5, R13 ; Save R13, R12, R11, R10, R9</pre>
Example	Save the 5 16-bit registers R9, R10, R11, R12, R13 on the stack
	<pre>PUSHM.W #5, R13 ; Save R13, R12, R11, R10, R9</pre>

4.6.3.20 POPX

*** POPX.A** Restore single address-word from the stack

*** POPX.[W]** Restore single word from the stack

*** POPX.B** Restore single byte from the stack

Syntax POPX.A dst

POPX dst or POPX.W dst

POPX.B dst

Operation Restore the 8-, 16-, 20-bit value from the stack to the destination. 20-bit addresses are possible. The SP is incremented by 2 (byte and word operands) and by 4 (address-word operand).

Emulation MOVX(.B, .A) @SP+, dst

Description The item on TOS is written to the destination operand. Register mode, indexed mode, symbolic mode, and absolute mode are possible. The SP is incremented by 2 or 4.
Note: the SP is incremented by 2 also for byte operations.

Status Bits Status bits are not affected.

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example Write the 16-bit value on TOS to the 20-bit address &EDE

```
POPX.W    &EDE    ; Write word to address EDE
```

Example Write the 20-bit value on TOS to R9

```
POPX.A    R9      ; Write address-word to R9
```


4.6.3.22 RLAM

RLAM.A	Rotate left arithmetically the 20-bit CPU register content
RLAM.[W]	Rotate left arithmetically the 16-bit CPU register content
Syntax	RLAM.A #n,Rdst $1 \leq n \leq 4$
	RLAM.W #n,Rdst or RLAM #n,Rdst $1 \leq n \leq 4$
Operation	$C \leftarrow MSB \leftarrow MSB-1 \dots LSB+1 \leftarrow LSB \leftarrow 0$
Description	The destination operand is shifted arithmetically left 1, 2, 3, or 4 positions as shown in Figure 4-44 . RLAM works as a multiplication (signed and unsigned) with 2, 4, 8, or 16. The word instruction RLAM.W clears the bits Rdst.19:16. Note : This instruction does not use the extension word.
Status Bits	N: Set if result is negative
	.A: Rdst.19 = 1, reset if Rdst.19 = 0
	.W: Rdst.15 = 1, reset if Rdst.15 = 0
	Z: Set if result is zero, reset otherwise
	C: Loaded from the MSB (n = 1), MSB-1 (n = 2), MSB-2 (n = 3), MSB-3 (n = 4)
	V: Undefined
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 20-bit operand in R5 is shifted left by 3 positions. It operates equal to an arithmetic multiplication by 8.

```
RLAM.A    #3,R5      ; R5 = R5 x 8
```

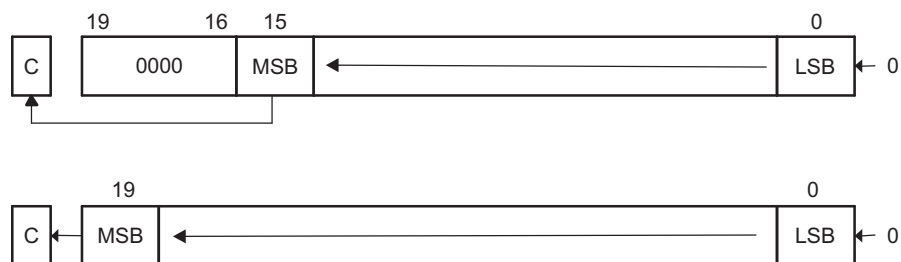


Figure 4-44. Rotate Left Arithmetically—RLAM[W] and RLAM.A

4.6.3.23 RLAX

* RLAX.A	Rotate left arithmetically address-word
* RLAX.[W]	Rotate left arithmetically word
* RLAX.B	Rotate left arithmetically byte
Syntax	RLAX.A dst RLAX dst or RLAX.W dst RLAX.B dst
Operation	$C \leftarrow \text{MSB} \leftarrow \text{MSB}-1 \dots \text{LSB}+1 \leftarrow \text{LSB} \leftarrow 0$
Emulation	ADDX.A dst, dst ADDX dst, dst ADDX.B dst, dst
Description	The destination operand is shifted left 1 position as shown in Figure 4-45 . The MSB is shifted into the carry bit (C) and the LSB is filled with 0. The RLAX instruction acts as a signed multiplication by 2.
Status Bits	N: Set if result is negative, reset if positive Z: Set if result is zero, reset otherwise C: Loaded from the MSB V: .A: Set if an arithmetic overflow occurs: the initial value is $040000\text{h} \leq \text{dst} < 0\text{C}0000\text{h}$; reset otherwise .W: Set if an arithmetic overflow occurs: the initial value is $04000\text{h} \leq \text{dst} < 0\text{C}000\text{h}$; reset otherwise .B: Set if an arithmetic overflow occurs: the initial value is $040\text{h} \leq \text{dst} < 0\text{C}0\text{h}$; reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 20-bit value in R7 is multiplied by 2

```
RLAX.A    R7    ; Shift left R7 (20-bit)
```

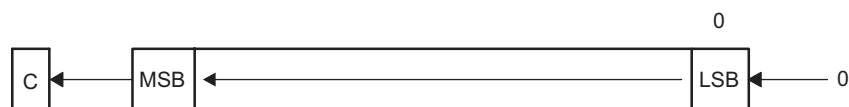


Figure 4-45. Destination Operand-Arithmetic Shift Left

4.6.3.24 RLCX

* **RLCX.A** Rotate left through carry address-word

* **RLCX.[W]** Rotate left through carry word

* **RLCX.B** Rotate left through carry byte

Syntax RLCX.A dst

RLCX dst or RLCX.W dst

RLCX.B dst

Operation $C \leftarrow \text{MSB} \leftarrow \text{MSB}-1 \dots \text{LSB}+1 \leftarrow \text{LSB} \leftarrow C$

Emulation ADDCX.A dst, dst

ADDCX dst, dst

ADDCX.B dst, dst

Description The destination operand is shifted left 1 position as shown in Figure 4-46. The carry bit (C) is shifted into the LSB and the MSB is shifted into the carry bit (C).

Status Bits N: Set if result is negative, reset if positive

Z: Set if result is zero, reset otherwise

C: Loaded from the MSB

V: .A: Set if an arithmetic overflow occurs: the initial value is $040000\text{h} \leq \text{dst} < 0\text{C}0000\text{h}$; reset otherwise

.W: Set if an arithmetic overflow occurs: the initial value is $04000\text{h} \leq \text{dst} < 0\text{C}000\text{h}$; reset otherwise

.B: Set if an arithmetic overflow occurs: the initial value is $040\text{h} \leq \text{dst} < 0\text{C}0\text{h}$; reset otherwise

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example The 20-bit value in R5 is shifted left 1 position.

```
RLCX.A    R5        ; (R5 x 2) + C -> R5
```

Example The RAM byte LEO is shifted left 1 position. PC is pointing to upper memory.

```
RLCX.B    LEO        ; RAM(LE0) x 2 + C -> RAM(LE0)
```

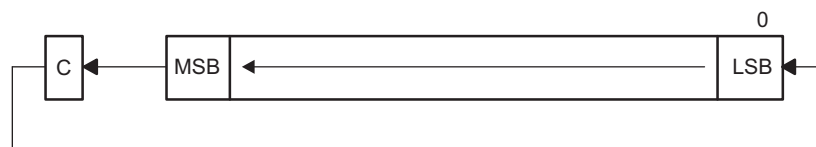


Figure 4-46. Destination Operand-Carry Left Shift

4.6.3.25 RRAM

RRAM.A	Rotate right arithmetically the 20-bit CPU register content
RRAM.[W]	Rotate right arithmetically the 16-bit CPU register content
Syntax	RRAM.A #n,Rdst $1 \leq n \leq 4$ RRAM.W #n,Rdst or RRAM #n,Rdst $1 \leq n \leq 4$
Operation	MSB $\rightarrow$ MSB $\rightarrow$ MSB-1 ... LSB+1 $\rightarrow$ LSB $\rightarrow$ C
Description	The destination operand is shifted right arithmetically by 1, 2, 3, or 4 bit positions as shown in Figure 4-47 . The MSB retains its value (sign). RRAM operates equal to a signed division by 2, 4, 8, or 16. The MSB is retained and shifted into MSB-1. The LSB+1 is shifted into the LSB, and the LSB is shifted into the carry bit C. The word instruction RRAM.W clears the bits Rdst.19:16. Note : This instruction does not use the extension word.
Status Bits	N: Set if result is negative .A: Rdst.19 = 1, reset if Rdst.19 = 0 .W: Rdst.15 = 1, reset if Rdst.15 = 0 Z: Set if result is zero, reset otherwise C: Loaded from the LSB (n = 1), LSB+1 (n = 2), LSB+2 (n = 3), or LSB+3 (n = 4) V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The signed 20-bit number in R5 is shifted arithmetically right 2 positions.

```
RRAM.A    #2,R5          ; R5/4 -> R5
```

Example The signed 20-bit value in R15 is multiplied by 0.75. $(0.5 + 0.25) \times R15$.

```
PUSHM.A   #1,R15        ; Save extended R15 on stack
RRAM.A     #1,R15        ; R15 y 0.5 -> R15
ADDX.A     @SP+,R15      ; R15 y 0.5 + R15 = 1.5 y R15 -> R15
RRAM.A     #1,R15        ; (1.5 y R15) y 0.5 = 0.75 y R15 -> R15
```

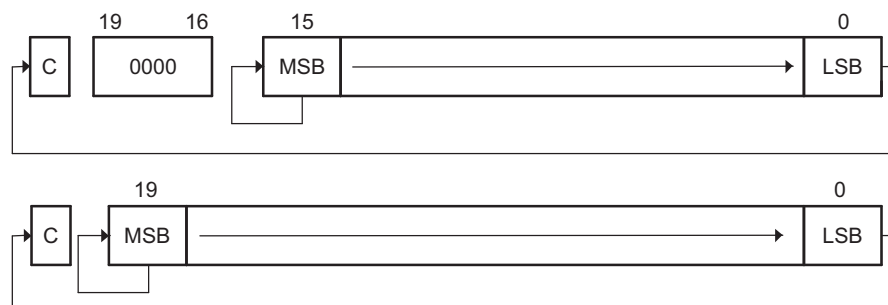


Figure 4-47. Rotate Right Arithmetically RRAM[W] and RRAM.A

4.6.3.26 RRAX

RRAX.A	Rotate right arithmetically the 20-bit operand
RRAX.[W]	Rotate right arithmetically the 16-bit operand
RRAX.B	Rotate right arithmetically the 8-bit operand
Syntax	RRAX.A Rdst RRAX.W Rdst RRAX Rdst RRAX.B Rdst RRAX.A dst RRAX dst or RRAX.W dst RRAX.B dst
Operation	MSB → MSB → MSB−1 ... LSB+1 → LSB → C
Description	Register mode for the destination: the destination operand is shifted right by 1 bit position as shown in Figure 4-48. The MSB retains its value (sign). The word instruction RRAX.W clears the bits Rdst.19:16, the byte instruction RRAX.B clears the bits Rdst.19:8. The MSB retains its value (sign), the LSB is shifted into the carry bit. RRAX here operates equal to a signed division by 2. All other modes for the destination: the destination operand is shifted right arithmetically by 1 bit position as shown in Figure 4-49. The MSB retains its value (sign), the LSB is shifted into the carry bit. RRAX here operates equal to a signed division by 2. All addressing modes, with the exception of the immediate mode, are possible in the full memory.
Status Bits	N: Set if result is negative, reset if positive .A: dst.19 = 1, reset if dst.19 = 0 .W: dst.15 = 1, reset if dst.15 = 0 .B: dst.7 = 1, reset if dst.7 = 0 Z: Set if result is zero, reset otherwise C: Loaded from the LSB V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The signed 20-bit number in R5 is shifted arithmetically right 4 positions.
	<pre>RPT #4 RRAX.A R5 ; R5/16 -> R5</pre>
Example	The signed 8-bit value in EDE is multiplied by 0.5.

RRAX.B &EDE ; EDE/2 -> EDE

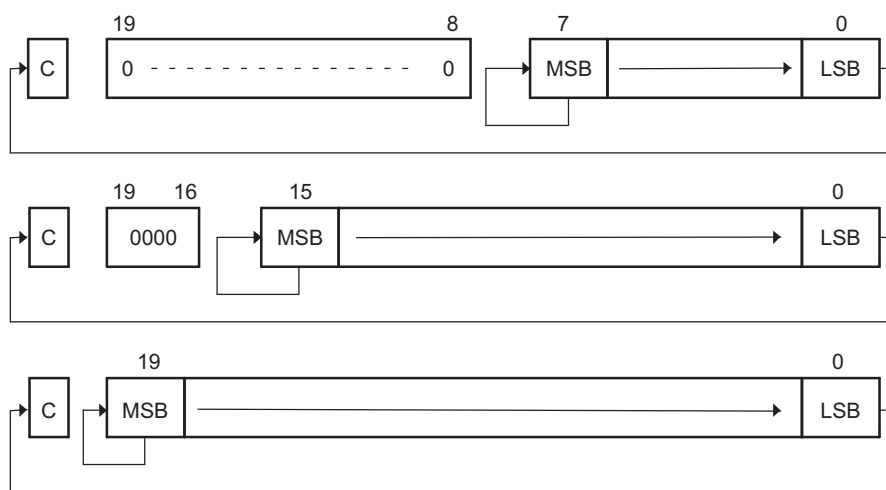


Figure 4-48. Rotate Right Arithmetically RRAX(.B,.A) – Register Mode

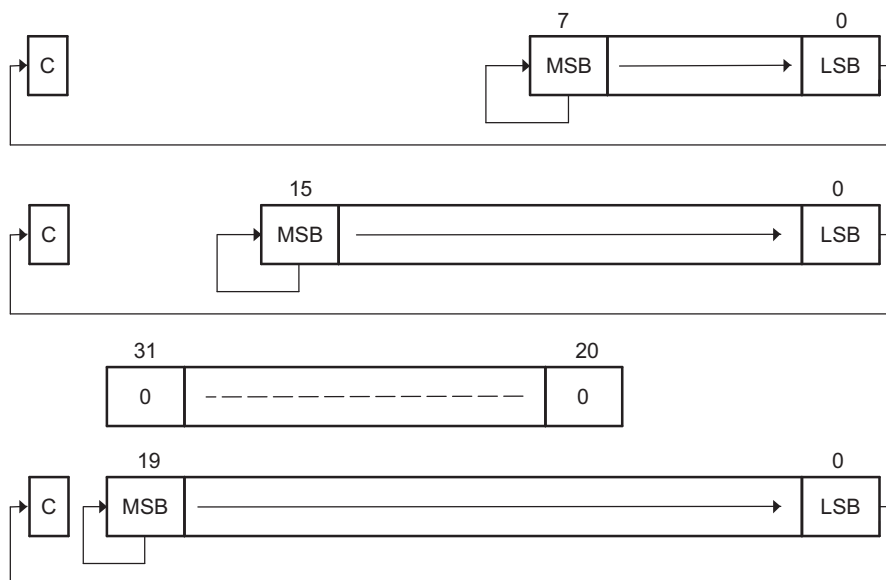


Figure 4-49. Rotate Right Arithmetically RRAX(.B,.A) – Non-Register Mode

4.6.3.27 RRCM

RRCM.A	Rotate right through carry the 20-bit CPU register content
RRCM.[W]	Rotate right through carry the 16-bit CPU register content
Syntax	RRCM.A #n,Rdst $1 \leq n \leq 4$
	RRCM.W #n,Rdst or RRCM #n,Rdst $1 \leq n \leq 4$
Operation	$C \rightarrow \text{MSB} \rightarrow \text{MSB}-1 \dots \text{LSB}+1 \rightarrow \text{LSB} \rightarrow C$
Description	The destination operand is shifted right by 1, 2, 3, or 4 bit positions as shown in Figure 4-50 . The carry bit C is shifted into the MSB, the LSB is shifted into the carry bit. The word instruction RRCM.W clears the bits Rdst.19:16. Note : This instruction does not use the extension word.
Status Bits	N: Set if result is negative
	.A: Rdst.19 = 1, reset if Rdst.19 = 0
	.W: Rdst.15 = 1, reset if Rdst.15 = 0
	Z: Set if result is zero, reset otherwise
	C: Loaded from the LSB (n = 1), LSB+1 (n = 2), LSB+2 (n = 3), or LSB+3 (n = 4)
Mode Bits	V: Reset
	OSCOFF, CPUOFF, and GIE are not affected.

Example The address-word in R5 is shifted right by 3 positions. The MSB-2 is loaded with 1.

```
SETC                ; Prepare carry for MSB-2
RRCM.A    #3,R5      ; R5 = R5 » 3 + 20000h
```

Example The word in R6 is shifted right by 2 positions. The MSB is loaded with the LSB. The MSB-1 is loaded with the contents of the carry flag.

```
RRCM.W    #2,R6      ; R6 = R6 » 2. R6.19:16 = 0
```

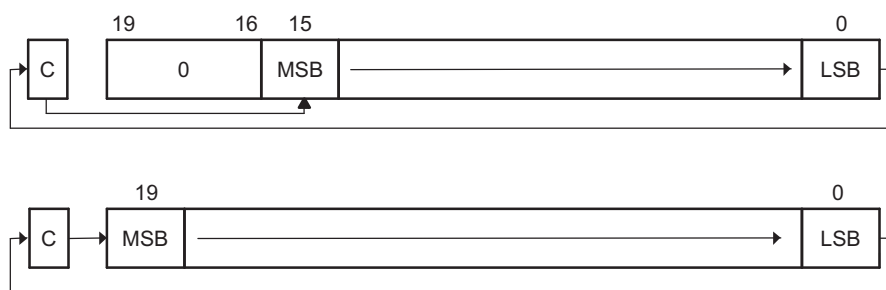


Figure 4-50. Rotate Right Through Carry RRCM[W] and RRCM.A

4.6.3.28 RRCX

RRCX.A	Rotate right through carry the 20-bit operand
RRCX.[W]	Rotate right through carry the 16-bit operand
RRCX.B	Rotate right through carry the 8-bit operand
Syntax	RRCX.A Rdst RRCX.W Rdst RRCX Rdst RRCX.B Rdst RRCX.A dst RRCX dst or RRCX.W dst RRCX.B dst
Operation	$C \rightarrow \text{MSB} \rightarrow \text{MSB}-1 \dots \text{LSB}+1 \rightarrow \text{LSB} \rightarrow C$
Description	Register mode for the destination: the destination operand is shifted right by 1 bit position as shown in Figure 4-51. The word instruction RRCX.W clears the bits Rdst.19:16, the byte instruction RRCX.B clears the bits Rdst.19:8. The carry bit C is shifted into the MSB, the LSB is shifted into the carry bit. All other modes for the destination: the destination operand is shifted right by 1 bit position as shown in Figure 4-52. The carry bit C is shifted into the MSB, the LSB is shifted into the carry bit. All addressing modes, with the exception of the immediate mode, are possible in the full memory.
Status Bits	N: Set if result is negative .A: dst.19 = 1, reset if dst.19 = 0 .W: dst.15 = 1, reset if dst.15 = 0 .B: dst.7 = 1, reset if dst.7 = 0 Z: Set if result is zero, reset otherwise C: Loaded from the LSB V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 20-bit operand at address EDE is shifted right by 1 position. The MSB is loaded with 1.
	<pre> SETC ; Prepare carry for MSB RRCX.A EDE ; EDE = EDE » 1 + 80000h </pre>
Example	The word in R6 is shifted right by 12 positions.

RPT #12
RRCX.W R6 ; R6 = R6 » 12. R6.19:16 = 0

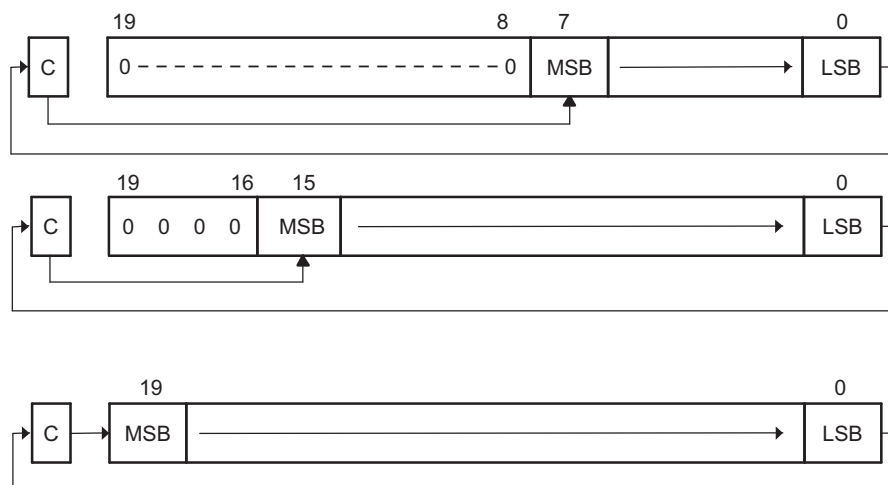


Figure 4-51. Rotate Right Through Carry RRCX(.B,.A) – Register Mode

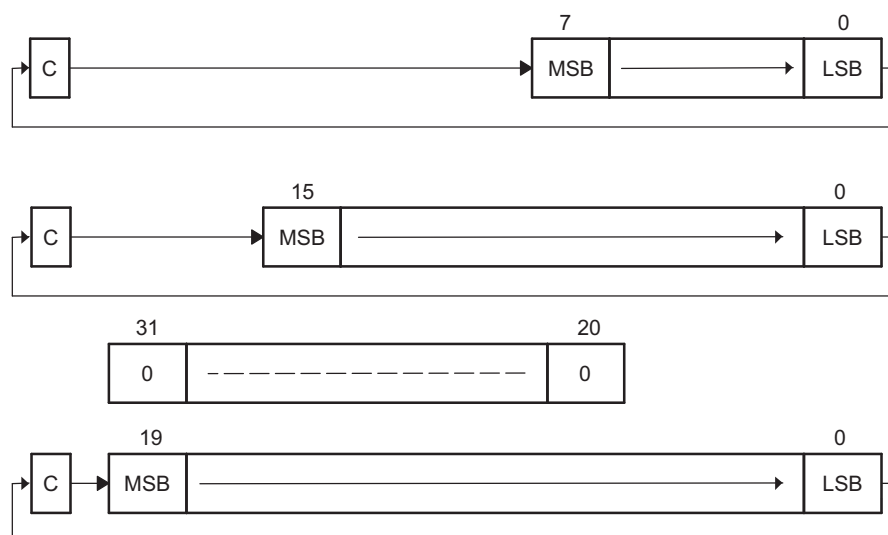


Figure 4-52. Rotate Right Through Carry RRCX(.B,.A) – Non-Register Mode

4.6.3.29 RRUM

RRUM.A	Rotate right through carry the 20-bit CPU register content
RRUM.[W]	Rotate right through carry the 16-bit CPU register content
Syntax	RRUM.A #n,Rdst $1 \leq n \leq 4$ RRUM.W #n,Rdst or RRUM #n,Rdst $1 \leq n \leq 4$
Operation	$0 \rightarrow \text{MSB} \rightarrow \text{MSB}-1 \dots \text{LSB}+1 \rightarrow \text{LSB} \rightarrow \text{C}$
Description	The destination operand is shifted right by 1, 2, 3, or 4 bit positions as shown in Figure 4-53 . Zero is shifted into the MSB, the LSB is shifted into the carry bit. RRUM works like an unsigned division by 2, 4, 8, or 16. The word instruction RRUM.W clears the bits Rdst.19:16. Note : This instruction does not use the extension word.
Status Bits	N: Set if result is negative .A: Rdst.19 = 1, reset if Rdst.19 = 0 .W: Rdst.15 = 1, reset if Rdst.15 = 0 Z: Set if result is zero, reset otherwise C: Loaded from the LSB (n = 1), LSB+1 (n = 2), LSB+2 (n = 3), or LSB+3 (n = 4) V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The unsigned address-word in R5 is divided by 16.

```
RRUM.A    #4,R5        ; R5 = R5 » 4. R5/16
```

Example The word in R6 is shifted right by 1 bit. The MSB R6.15 is loaded with 0.

```
RRUM.W    #1,R6        ; R6 = R6/2. R6.19:15 = 0
```

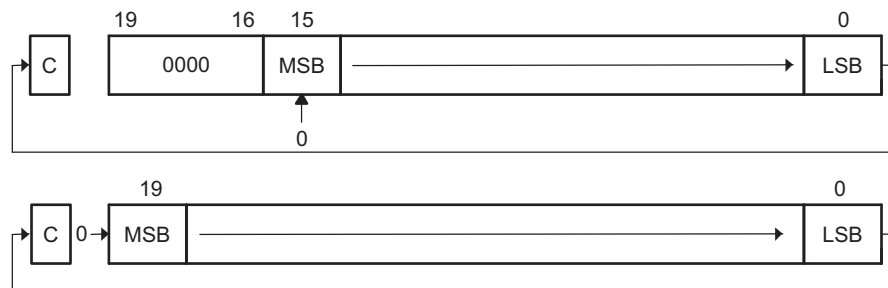


Figure 4-53. Rotate Right Unsigned RRUM[W] and RRUM.A

4.6.3.30 RRUX

RRUX.A	Shift right unsigned the 20-bit CPU register content
RRUX.[W]	Shift right unsigned the 16-bit CPU register content
RRUX.B	Shift right unsigned the 8-bit CPU register content
Syntax	RRUX.A Rdst RRUX.W Rdst RRUX Rdst RRUX.B Rdst
Operation	C=0 → MSB → MSB–1 ... LSB+1 → LSB → C
Description	RRUX is valid for register mode only: the destination operand is shifted right by 1 bit position as shown in Figure 4-54 . The word instruction RRUX.W clears the bits Rdst.19:16. The byte instruction RRUX.B clears the bits Rdst.19:8. Zero is shifted into the MSB, the LSB is shifted into the carry bit.
Status Bits	N: Set if result is negative .A: dst.19 = 1, reset if dst.19 = 0 .W: dst.15 = 1, reset if dst.15 = 0 .B: dst.7 = 1, reset if dst.7 = 0 Z: Set if result is zero, reset otherwise C: Loaded from the LSB V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The word in R6 is shifted right by 12 positions.

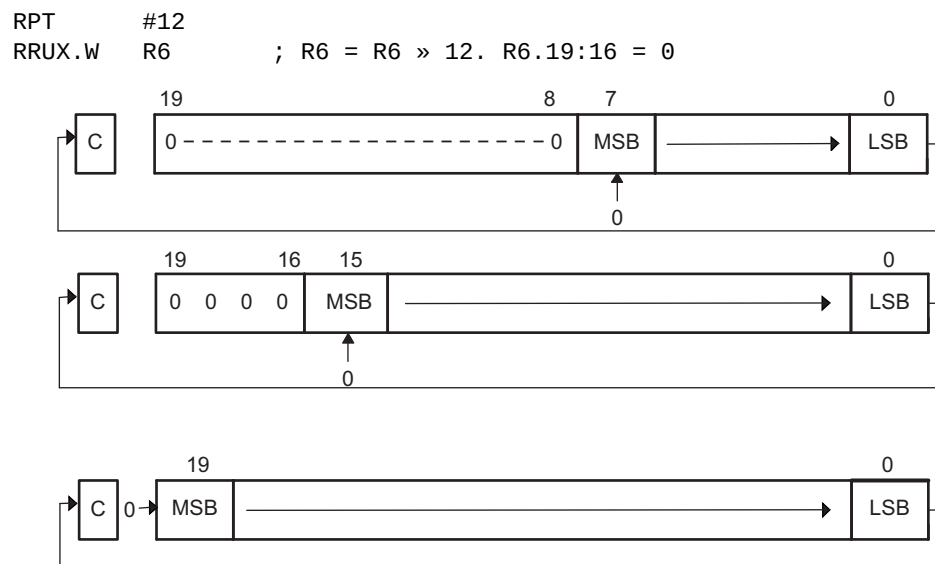


Figure 4-54. Rotate Right Unsigned RRUX(.B,.A) – Register Mode

4.6.3.31 SBCX

* SBCX.A	Subtract borrow (.NOT. carry) from destination address-word
* SBCX.[W]	Subtract borrow (.NOT. carry) from destination word
* SBCX.B	Subtract borrow (.NOT. carry) from destination byte
Syntax	SBCX.A dst SBCX dst or SBCX.W dst SBCX.B dst
Operation	$dst + 0FFFFFFh + C \rightarrow dst$ $dst + 0FFFFFFh + C \rightarrow dst$ $dst + 0FFh + C \rightarrow dst$
Emulation	SBCX.A #0, dst SBCX #0, dst SBCX.B #0, dst
Description	The carry bit (C) is added to the destination operand minus 1. The previous contents of the destination are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if result is zero, reset otherwise C: Set if there is a carry from the MSB of the result, reset otherwise Set to 1 if no borrow, reset if borrow V: Set if an arithmetic overflow occurs, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 8-bit counter pointed to by R13 is subtracted from a 16-bit counter pointed to by R12.

```

SUBX.B    @R13,0(R12)    ; Subtract LSDs
SBCX.B    1(R12)         ; Subtract carry from MSD

```

NOTE: Borrow implementation

The borrow is treated as a .NOT. carry:

Borrow	Carry Bit
Yes	0
No	1

4.6.3.32 SUBX

SUBX.A	Subtract source address-word from destination address-word
SUBX.[W]	Subtract source word from destination word
SUBX.B	Subtract source byte from destination byte
Syntax	SUBX.A src,dst SUBX src,dst or SUBX.W src,dst SUBX.B src,dst
Operation	$(\text{not. src}) + 1 + \text{dst} \rightarrow \text{dst}$ or $\text{dst} - \text{src} \rightarrow \text{dst}$
Description	The source operand is subtracted from the destination operand. This is done by adding the 1s complement of the source + 1 to the destination. The source operand is not affected. The result is written to the destination operand. Both operands may be located in the full address space.
Status Bits	N: Set if result is negative ($\text{src} > \text{dst}$), reset if positive ($\text{src} \leq \text{dst}$) Z: Set if result is zero ($\text{src} = \text{dst}$), reset otherwise ($\text{src} \neq \text{dst}$) C: Set if there is a carry from the MSB, reset otherwise V: Set if the subtraction of a negative source operand from a positive destination operand delivers a negative result, or if the subtraction of a positive source operand from a negative destination operand delivers a positive result, reset otherwise (no overflow)
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	A 20-bit constant 87654h is subtracted from EDE (LSBs) and EDE+2 (MSBs).

```
SUBX.A    #87654h,EDE        ; Subtract 87654h from EDE+2|EDE
```

Example A table word pointed to by R5 (20-bit address) is subtracted from R7. Jump to label TONI if R7 contains zero after the instruction. R5 is auto-incremented by 2. R7.19:16 = 0.

```
SUBX.W    @R5+,R7            ; Subtract table number from R7. R5 + 2
JZ        TONI                ; R7 = @R5 (before subtraction)
...       ; R7 <> @R5 (before subtraction)
```

Example Byte CNT is subtracted from the byte R12 points to in the full address space. Address of CNT is within $\text{PC} \pm 512 \text{ K}$.

```
SUBX.B    CNT,0(R12)         ; Subtract CNT from @R12
```

Note: Use SUBA for the following 2 cases for better density and execution.

```
SUBX.A    Rsrc,Rdst
SUBX.A    #imm20,Rdst
```

4.6.3.33 SUBCX

SUBCX.A	Subtract source address-word with carry from destination address-word
SUBCX.[W]	Subtract source word with carry from destination word
SUBCX.B	Subtract source byte with carry from destination byte
Syntax	SUBCX.A src,dst SUBCX src,dst or SUBCX.W src,dst SUBCX.B src,dst
Operation	$(\text{not. src}) + C + \text{dst} \rightarrow \text{dst}$ or $\text{dst} - (\text{src} - 1) + C \rightarrow \text{dst}$
Description	The source operand is subtracted from the destination operand. This is made by adding the 1s complement of the source + carry to the destination. The source operand is not affected, the result is written to the destination operand. Both operands may be located in the full address space.
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if there is a carry from the MSB, reset otherwise V: Set if the subtraction of a negative source operand from a positive destination operand delivers a negative result, or if the subtraction of a positive source operand from a negative destination operand delivers a positive result, reset otherwise (no overflow).
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	A 20-bit constant 87654h is subtracted from R5 with the carry from the previous instruction.

```
SUBCX.A    #87654h,R5        ; Subtract 87654h + C from R5
```

Example A 48-bit number (3 words) pointed to by R5 (20-bit address) is subtracted from a 48-bit counter in RAM, pointed to by R7. R5 auto-increments to point to the next 48-bit number.

```
SUBX.W     @R5+,0(R7)        ; Subtract LSBs. R5 + 2
SUBCX.W     @R5+,2(R7)        ; Subtract MIDs with C. R5 + 2
SUBCX.W     @R5+,4(R7)        ; Subtract MSBs with C. R5 + 2
```

Example Byte CNT is subtracted from the byte R12 points to. The carry of the previous instruction is used. 20-bit addresses.

```
SUBCX.B     &CNT,0(R12)      ; Subtract byte CNT from @R12
```

4.6.3.34 SWPBX

SWPBX.A	Swap bytes of lower word
SWPBX.[W]	Swap bytes of word
Syntax	SWPBX.A dst SWPBX dst or SWPBX.W dst
Operation	dst.15:8 ↔ dst.7:0
Description	Register mode: Rn.15:8 are swapped with Rn.7:0. When the .A extension is used, Rn.19:16 are unchanged. When the .W extension is used, Rn.19:16 are cleared. Other modes: When the .A extension is used, bits 31:20 of the destination address are cleared, bits 19:16 are left unchanged, and bits 15:8 are swapped with bits 7:0. When the .W extension is used, bits 15:8 are swapped with bits 7:0 of the addressed word.
Status Bits	Status bits are not affected.
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Exchange the bytes of RAM address-word EDE

```
MOVX.A    #23456h, &EDE    ; 23456h -> EDE
SWPBX.A   EDE               ; 25634h -> EDE
```

Example Exchange the bytes of R5

```
MOVA      #23456h, R5       ; 23456h -> R5
SWPBX.W   R5                ; 05634h -> R5
```

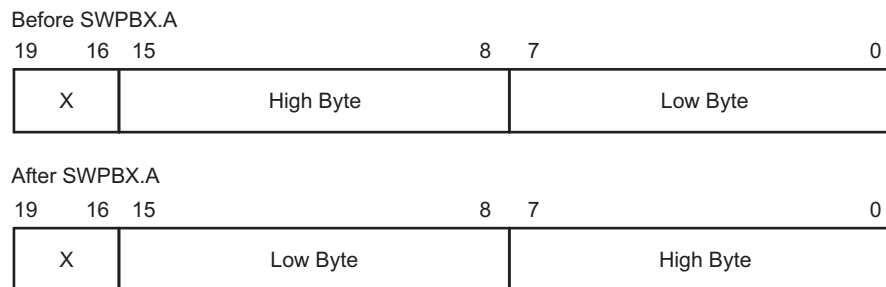


Figure 4-55. Swap Bytes SWPBX.A Register Mode

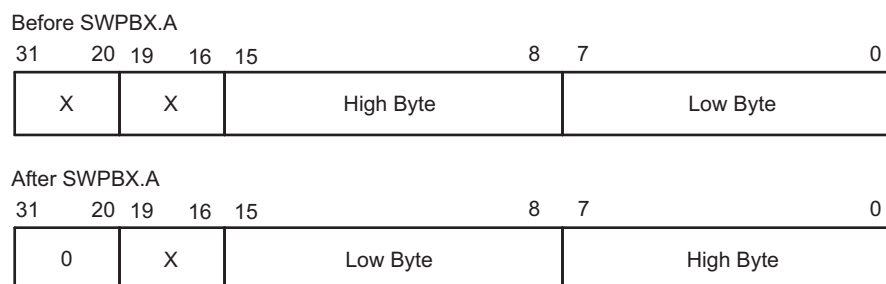


Figure 4-56. Swap Bytes SWPBX.A In Memory

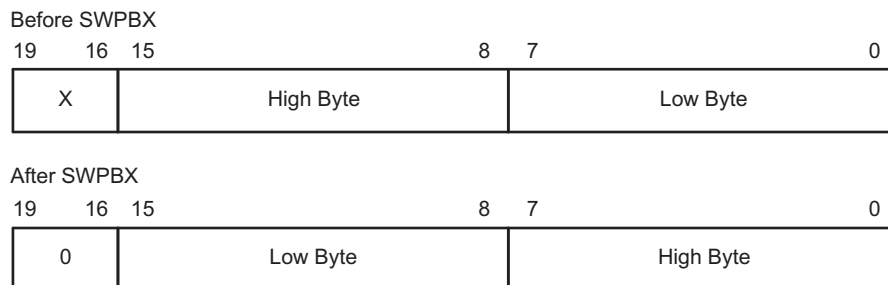


Figure 4-57. Swap Bytes SWPBX[.W] Register Mode

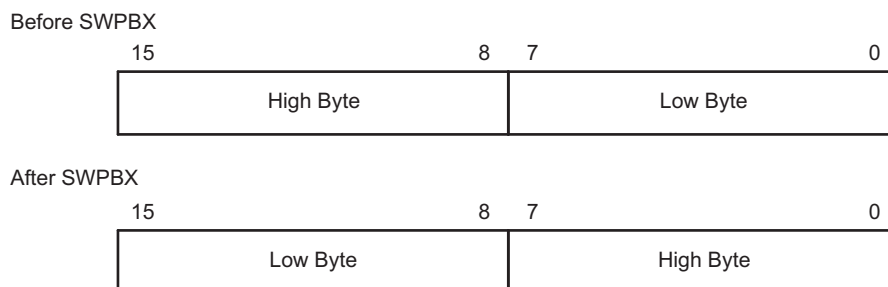


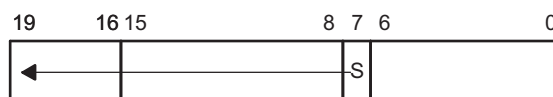
Figure 4-58. Swap Bytes SWPBX[.W] In Memory

4.6.3.35 SXTX

SXTX.A	Extend sign of lower byte to address-word
SXTX.[W]	Extend sign of lower byte to word
Syntax	SXTX.A dst SXTX dst or SXTX.W dst
Operation	dst.7 → dst.15:8, Rdst.7 → Rdst.19:8 (register mode)
Description	Register mode: The sign of the low byte of the operand (Rdst.7) is extended into the bits Rdst.19:8. Other modes: SXTX.A: the sign of the low byte of the operand (dst.7) is extended into dst.19:8. The bits dst.31:20 are cleared. SXTX.[W]: the sign of the low byte of the operand (dst.7) is extended into dst.15:8.
Status Bits	N: Set if result is negative, reset otherwise Z: Set if result is zero, reset otherwise C: Set if result is not zero, reset otherwise (C = .not.Z) V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The signed 8-bit data in EDE.7:0 is sign extended to 20 bits: EDE.19:8. Bits 31:20 located in EDE+2 are cleared.

SXTX.A &EDE ; Sign extended EDE -> EDE+2/EDE

SXTX.A Rdst



SXTX.A dst

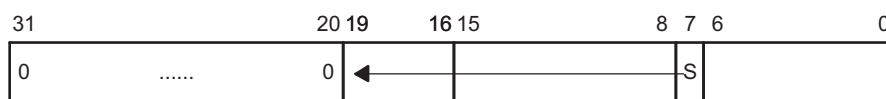
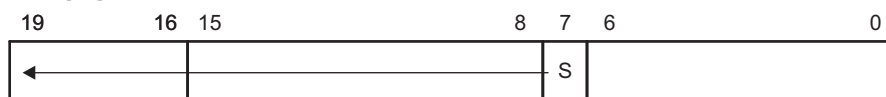


Figure 4-59. Sign Extend SXTX.A

SXTX.[W] Rdst



SXTX.[W] dst

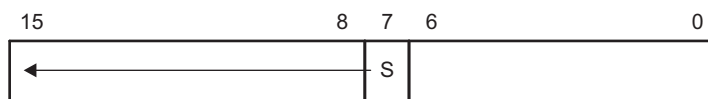


Figure 4-60. Sign Extend SXTX.[W]

4.6.3.36 TSTX

* TSTX.A	Test destination address-word
* TSTX.[W]	Test destination word
* TSTX.B	Test destination byte
Syntax	TSTX.A dst TSTX dst or TSTX.W dst TSTX.B dst
Operation	dst + 0FFFFFFh + 1 dst + 0FFFFFFh + 1 dst + 0FFh + 1
Emulation	CMPX.A #0, dst CMPX #0, dst CMPX.B #0, dst
Description	The destination operand is compared with zero. The status bits are set according to the result. The destination is not affected.
Status Bits	N: Set if destination is negative, reset if positive Z: Set if destination contains zero, reset otherwise C: Set V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	RAM byte LEO is tested; PC is pointing to upper memory. If it is negative, continue at LEONEG; if it is positive but not zero, continue at LEOPOS.

```

TSTX.B    LEO        ; Test LEO
JN        LEONEG     ; LEO is negative
JZ        LEOZERO    ; LEO is zero
LEOPOS    .....     ; LEO is positive but not zero
LEONEG    .....     ; LEO is negative
LEOZERO   .....     ; LEO is zero

```

4.6.3.37 XORX

XORX.A	Exclusive OR source address-word with destination address-word
XORX.[W]	Exclusive OR source word with destination word
XORX.B	Exclusive OR source byte with destination byte
Syntax	XORX.A src,dst XORX src,dst or XORX.W src,dst XORX.B src,dst
Operation	src .xor. dst → dst
Description	The source and destination operands are exclusively ORed. The result is placed into the destination. The source operand is not affected. The previous contents of the destination are lost. Both operands may be located in the full address space.
Status Bits	N: Set if result is negative (MSB = 1), reset if positive (MSB = 0) Z: Set if result is zero, reset otherwise C: Set if result is not zero, reset otherwise (carry = .not. Zero) V: Set if both operands are negative (before execution), reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	Toggle bits in address-word CNTR (20-bit data) with information in address-word TONI (20-bit address)

```
XORX.A  TONI,&CNTR      ; Toggle bits in CNTR
```

Example A table word pointed to by R5 (20-bit address) is used to toggle bits in R6.

```
XORX.W  @R5,R6          ; Toggle bits in R6. R6.19:16 = 0
```

Example Reset to zero those bits in the low byte of R7 that are different from the bits in byte EDE (20-bit address)

```
XORX.B  EDE,R7           ; Set different bits to 1 in R7
INV.B    R7               ; Invert low byte of R7. R7.19:8 = 0.
```


4.6.4 Address Instructions

MSP430X address instructions are instructions that support 20-bit operands but have restricted addressing modes. The addressing modes are restricted to the register mode and the immediate mode, except for the MOVA instruction. Restricting the addressing modes removes the need for the additional extension-word op-code improving code density and execution time. The MSP430X address instructions are listed and described in the following pages.

4.6.4.1 ADDA

ADDA	Add 20-bit source to a 20-bit destination register		
Syntax	ADDA Rsrc, Rdst ADDA #imm20, Rdst		
Operation	src + Rdst → Rdst		
Description	The 20-bit source operand is added to the 20-bit destination CPU register. The previous contents of the destination are lost. The source operand is not affected.		
Status Bits	N: Set if result is negative (Rdst.19 = 1), reset if positive (Rdst.19 = 0) Z: Set if result is zero, reset otherwise C: Set if there is a carry from the 20-bit result, reset otherwise V: Set if the result of 2 positive operands is negative, or if the result of 2 negative numbers is positive, reset otherwise		
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.		
Example	R5 is increased by 0A4320h. The jump to TONI is performed if a carry occurs.		
	ADDA	#0A4320h, R5	; Add A4320h to 20-bit R5
	JC	TONI	; Jump on carry
	...		; No carry occurred

4.6.4.2 BRA

* BRA	Branch to destination		
Syntax	BRA dst		
Operation	dst → PC		
Emulation	MOVA dst, PC		
Description	An unconditional branch is taken to a 20-bit address anywhere in the full address space. All 7 source addressing modes can be used. The branch instruction is an address-word instruction. If the destination address is contained in a memory location X, it is contained in 2 ascending words: X (LSBs) and (X + 2) (MSBs).		
Status Bits	N: Not affected Z: Not affected C: Not affected V: Not affected		
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.		
Examples	Examples for all addressing modes are given. Immediate mode: Branch to label EDE located anywhere in the 20-bit address space or branch directly to address.		
BRA	#EDE	; MOVA	#imm20, PC
BRA	#01AA04h		
	Symbolic mode: Branch to the 20-bit address contained in addresses EXEC (LSBs) and EXEC+2 (MSBs). EXEC is located at the address (PC + X) where X is within ±32 K. Indirect addressing.		
BRA	EXEC	; MOVA	z16(PC), PC
	Note: If the 16-bit index is not sufficient, a 20-bit index may be used with the following instruction.		
MOVX.A	EXEC, PC	; 1M byte range with 20-bit index	
	Absolute mode: Branch to the 20-bit address contained in absolute addresses EXEC (LSBs) and EXEC+2 (MSBs). Indirect addressing.		
BRA	&EXEC	; MOVA	&abs20, PC
	Register mode: Branch to the 20-bit address contained in register R5. Indirect R5.		
BRA	R5	; MOVA	R5, PC
	Indirect mode: Branch to the 20-bit address contained in the word pointed to by register R5 (LSBs). The MSBs have the address (R5 + 2). Indirect, indirect R5.		
BRA	@R5	; MOVA	@R5, PC

Indirect, Auto-Increment mode: Branch to the 20-bit address contained in the words pointed to by register R5 and increment the address in R5 afterwards by 4. The next time the software flow uses R5 as a pointer, it can alter the program execution due to access to the next address in the table pointed to by R5. Indirect, indirect R5.

```
BRA      @R5+          ; MOVA    @R5+,PC. R5 + 4
```

Indexed mode: Branch to the 20-bit address contained in the address pointed to by register (R5 + X) (for example, a table with addresses starting at X). (R5 + X) points to the LSBs, (R5 + X + 2) points to the MSBs of the address. X is within $R5 \pm 32\text{ K}$. Indirect, indirect (R5 + X).

```
BRA      X(R5)          ; MOVA    z16(R5),PC
```

Note: If the 16-bit index is not sufficient, a 20-bit index X may be used with the following instruction:

```
MOVX.A   X(R5),PC      ; 1M byte range with 20-bit index
```

4.6.4.3 CALLA

CALLA Call a subroutine

Syntax CALLA dst

Operation dst → tmp 20-bit dst is evaluated and stored
 SP – 2 → SP
 PC.19:16 → @SP updated PC with return address to TOS (MSBs)
 SP – 2 → SP
 PC.15:0 → @SP updated PC to TOS (LSBs)
 tmp → PC saved 20-bit dst to PC

Description A subroutine call is made to a 20-bit address anywhere in the full address space. All 7 source addressing modes can be used. The call instruction is an address-word instruction. If the destination address is contained in a memory location X, it is contained in 2 ascending words, X (LSBs) and (X + 2) (MSBs). 2 words on the stack are needed for the return address. The return is made with the instruction RETA.

Status Bits N: Not affected
 Z: Not affected
 C: Not affected
 V: Not affected

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Examples Examples for all addressing modes are given.
 Immediate mode: Call a subroutine at label EXEC or call directly an address.

```
CALLA #EXEC          ; Start address EXEC
CALLA #01AA04h       ; Start address 01AA04h
```

Symbolic mode: Call a subroutine at the 20-bit address contained in addresses EXEC (LSBs) and EXEC+2 (MSBs). EXEC is located at the address (PC + X) where X is within ±32 K. Indirect addressing.

```
CALLA EXEC          ; Start address at @EXEC. z16(PC)
```

Absolute mode: Call a subroutine at the 20-bit address contained in absolute addresses EXEC (LSBs) and EXEC+2 (MSBs). Indirect addressing.

```
CALLA &EXEC          ; Start address at @EXEC
```

Register mode: Call a subroutine at the 20-bit address contained in register R5. Indirect R5.

```
CALLA R5             ; Start address at @R5
```

Indirect mode: Call a subroutine at the 20-bit address contained in the word pointed to by register R5 (LSBs). The MSBs have the address (R5 + 2). Indirect, indirect R5.

```
CALLA @R5            ; Start address at @R5
```

Indirect, auto-increment mode: Call a subroutine at the 20-bit address contained in the words pointed to by register R5 and increment the 20-bit address in R5 afterwards by 4. The next time the software flow uses R5 as a pointer, it can alter the program execution due to access to the next word address in the table pointed to by R5. Indirect, indirect R5.

CALLA @R5+ ; Start address at @R5. R5 + 4

Indexed mode: Call a subroutine at the 20-bit address contained in the address pointed to by register (R5 + X); for example, a table with addresses starting at X. (R5 + X) points to the LSBs, (R5 + X + 2) points to the MSBs of the word address. X is within R5 $\pm$ 32 K. Indirect, indirect (R5 + X).

CALLA X(R5) ; Start address at @(R5+X). z16(R5)

4.6.4.4 CLRA

* CLRA	Clear 20-bit destination register
Syntax	CLRA Rdst
Operation	$0 \rightarrow \text{Rdst}$
Emulation	MOVA #0, Rdst
Description	The destination register is cleared.
Status Bits	Status bits are not affected.
Example	The 20-bit value in R10 is cleared.

```
CLRA    R10        ; 0 -> R10
```

4.6.4.5 CMPA

CMPA	Compare the 20-bit source with a 20-bit destination register
Syntax	CMPA Rsrc, Rdst CMPA #imm20, Rdst
Operation	$(\text{not. src}) + 1 + \text{Rdst}$ or $\text{Rdst} - \text{src}$
Description	The 20-bit source operand is subtracted from the 20-bit destination CPU register. This is made by adding the 1s complement of the source + 1 to the destination register. The result affects only the status bits.
Status Bits	N: Set if result is negative ($\text{src} > \text{dst}$), reset if positive ($\text{src} \leq \text{dst}$) Z: Set if result is zero ($\text{src} = \text{dst}$), reset otherwise ($\text{src} \neq \text{dst}$) C: Set if there is a carry from the MSB, reset otherwise V: Set if the subtraction of a negative source operand from a positive destination operand delivers a negative result, or if the subtraction of a positive source operand from a negative destination operand delivers a positive result, reset otherwise (no overflow)
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	A 20-bit immediate operand and R6 are compared. If they are equal, the program continues at label EQUAL.

```

CMPA    #12345h, R6        ; Compare R6 with 12345h
JEQ     EQUAL              ; R6 = 12345h
...     ; Not equal

```

Example The 20-bit values in R5 and R6 are compared. If R5 is greater than (signed) or equal to R6, the program continues at label GRE.

```

CMPA    R6, R5              ; Compare R6 with R5 (R5 - R6)
JGE     GRE                 ; R5 >= R6
...     ; R5 < R6

```


4.6.4.6 DECDA

* DECDA	Double-decrement 20-bit destination register
Syntax	DECDA Rdst
Operation	$Rdst - 2 \rightarrow Rdst$
Emulation	SUBA #2, Rdst
Description	The destination register is decremented by 2. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if Rdst contained 2, reset otherwise C: Reset if Rdst contained 0 or 1, set otherwise V: Set if an arithmetic overflow occurs, otherwise reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 20-bit value in R5 is decremented by 2.
DECDA	R5 ; Decrement R5 by 2

4.6.4.7 INCDA

* INCDA	Double-increment 20-bit destination register
Syntax	INCDA Rdst
Operation	$Rdst + 2 \rightarrow Rdst$
Emulation	ADDA #2, Rdst
Description	The destination register is incremented by 2. The original contents are lost.
Status Bits	N: Set if result is negative, reset if positive Z: Set if Rdst contained 0FFFFEh, reset otherwise Set if Rdst contained 0FFFEh, reset otherwise Set if Rdst contained 0FEh, reset otherwise C: Set if Rdst contained 0FFFFEh or 0FFFFFh, reset otherwise Set if Rdst contained 0FFFEh or 0FFFFh, reset otherwise Set if Rdst contained 0FEh or 0FFh, reset otherwise V: Set if Rdst contained 07FFFEh or 07FFFFh, reset otherwise Set if Rdst contained 07FFEh or 07FFFh, reset otherwise Set if Rdst contained 07Eh or 07Fh, reset otherwise
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 20-bit value in R5 is incremented by 2.
INCDA	R5 ; Increment R5 by 2

4.6.4.8 MOVA

MOVA	Move the 20-bit source to the 20-bit destination
Syntax	MOVA Rsrc, Rdst MOVA #imm20, Rdst MOVA z16(Rsrc), Rdst MOVA EDE, Rdst MOVA &abs20, Rdst MOVA @Rsrc, Rdst MOVA @Rsrc+, Rdst MOVA Rsrc, z16(Rdst) MOVA Rsrc, &abs20
Operation	src → Rdst Rsrc → dst
Description	The 20-bit source operand is moved to the 20-bit destination. The source operand is not affected. The previous content of the destination is lost.
Status Bits	N: Not affected Z: Not affected C: Not affected V: Not affected
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Examples	Copy 20-bit value in R9 to R8

MOVA R9, R8 ; R9 -> R8

Write 20-bit immediate value 12345h to R12

MOVA #12345h, R12 ; 12345h -> R12

Copy 20-bit value addressed by (R9 + 100h) to R8. Source operand in addresses (R9 + 100h) LSBs and (R9 + 102h) MSBs.

MOVA 100h(R9), R8 ; Index: + 32 K. 2 words transferred

Move 20-bit value in 20-bit absolute addresses EDE (LSBs) and EDE+2 (MSBs) to R12

MOVA &EDE, R12 ; &EDE -> R12. 2 words transferred

Move 20-bit value in 20-bit addresses EDE (LSBs) and EDE+2 (MSBs) to R12. PC index ± 32 K.

MOVA EDE, R12 ; EDE -> R12. 2 words transferred

Copy 20-bit value R9 points to (20 bit address) to R8. Source operand in addresses @R9 LSBs and @(R9 + 2) MSBs.

MOVA @R9, R8 ; @R9 -> R8. 2 words transferred

Copy 20-bit value R9 points to (20 bit address) to R8. R9 is incremented by 4 afterwards. Source operand in addresses @R9 LSBs and @(R9 + 2) MSBs.

MOVA @R9+, R8 ; @R9 -> R8. R9 + 4. 2 words transferred.

Copy 20-bit value in R8 to destination addressed by (R9 + 100h). Destination operand in addresses @(R9 + 100h) LSBs and @(R9 + 102h) MSBs.

MOVA R8, 100h(R9) ; Index: +- 32 K. 2 words transferred

Move 20-bit value in R13 to 20-bit absolute addresses EDE (LSBs) and EDE+2 (MSBs)

MOVA R13, &EDE ; R13 -> EDE. 2 words transferred

Move 20-bit value in R13 to 20-bit addresses EDE (LSBs) and EDE+2 (MSBs). PC index ± 32 K.

MOVA R13, EDE ; R13 -> EDE. 2 words transferred

4.6.4.9 RETA

* RETA	Return from subroutine		
Syntax	RETA		
Operation	@SP → PC.15:0 LSBs (15:0) of saved PC to PC.15:0		
	SP + 2 → SP		
	@SP → PC.19:16 MSBs (19:16) of saved PC to PC.19:16		
	SP + 2 → SP		
Emulation	MOVA @SP+, PC		
Description	The 20-bit return address information, pushed onto the stack by a CALLA instruction, is restored to the PC. The program continues at the address following the subroutine call. The SR bits SR.11:0 are not affected. This allows the transfer of information with these bits.		
Status Bits	N: Not affected		
	Z: Not affected		
	C: Not affected		
	V: Not affected		
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.		
Example	Call a subroutine SUBR from anywhere in the 20-bit address space and return to the address after the CALLA		
	CALLA	#SUBR	; Call subroutine starting at SUBR
	...		; Return by RETA to here
SUBR	PUSHM.A	#2, R14	; Save R14 and R13 (20 bit data)
	...		; Subroutine code
	POPM.A	#2, R14	; Restore R13 and R14 (20 bit data)
	RETA		; Return (to full address space)

4.6.4.10 SUBA

SUBA	Subtract 20-bit source from 20-bit destination register		
Syntax	SUBA Rsrc, Rdst SUBA #imm20, Rdst		
Operation	$(\text{not.src}) + 1 + \text{Rdst} \rightarrow \text{Rdst}$ or $\text{Rdst} - \text{src} \rightarrow \text{Rdst}$		
Description	The 20-bit source operand is subtracted from the 20-bit destination register. This is made by adding the 1s complement of the source + 1 to the destination. The result is written to the destination register, the source is not affected.		
Status Bits	N: Set if result is negative ($\text{src} > \text{dst}$), reset if positive ($\text{src} \leq \text{dst}$) Z: Set if result is zero ($\text{src} = \text{dst}$), reset otherwise ($\text{src} \neq \text{dst}$) C: Set if there is a carry from the MSB (Rdst.19), reset otherwise V: Set if the subtraction of a negative source operand from a positive destination operand delivers a negative result, or if the subtraction of a positive source operand from a negative destination operand delivers a positive result, reset otherwise (no overflow)		
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.		
Example	The 20-bit value in R5 is subtracted from R6. If a carry occurs, the program continues at label TONI.		
	SUBA	R5, R6	; R6 - R5 -> R6
	JC	TONI	; Carry occurred
	...		; No carry

4.6.4.11 TSTA

* TSTA	Test 20-bit destination register
Syntax	TSTA Rdst
Operation	dst + 0FFFFFFh + 1 dst + 0FFFFFFh + 1 dst + 0FFh + 1
Emulation	CMPA #0, Rdst
Description	The destination register is compared with zero. The status bits are set according to the result. The destination register is not affected.
Status Bits	N: Set if destination register is negative, reset if positive Z: Set if destination register contains zero, reset otherwise C: Set V: Reset
Mode Bits	OSCOFF, CPUOFF, and GIE are not affected.
Example	The 20-bit value in R7 is tested. If it is negative, continue at R7NEG; if it is positive but not zero, continue at R7POS.

```

TSTA    R7            ; Test R7
JN      R7NEG         ; R7 is negative
JZ      R7ZERO        ; R7 is zero
R7POS   .....        ; R7 is positive but not zero
R7NEG   .....        ; R7 is negative
R7ZERO  .....        ; R7 is zero

```

32-Bit Hardware Multiplier (MPY32)

This chapter describes the 32-bit hardware multiplier (MPY32). The MPY32 module is implemented in all devices.

Topic	Page
5.1 32-Bit Hardware Multiplier (MPY32) Introduction	271
5.2 MPY32 Operation	273
5.3 MPY32 Registers	285

5.1 32-Bit Hardware Multiplier (MPY32) Introduction

The MPY32 is a peripheral and is not part of the CPU. This means its activities do not interfere with the CPU activities. The multiplier registers are peripheral registers that are loaded and read with CPU instructions.

The MPY32 supports:

- Unsigned multiply
- Signed multiply
- Unsigned multiply accumulate
- Signed multiply accumulate
- 8-bit, 16-bit, 24-bit, and 32-bit operands
- Saturation
- Fractional numbers
- 8-bit and 16-bit operation compatible with 16-bit hardware multiplier
- 8-bit and 24-bit multiplications without requiring a "sign extend" instruction

The MPY32 block diagram is shown in [Figure 5-1](#).

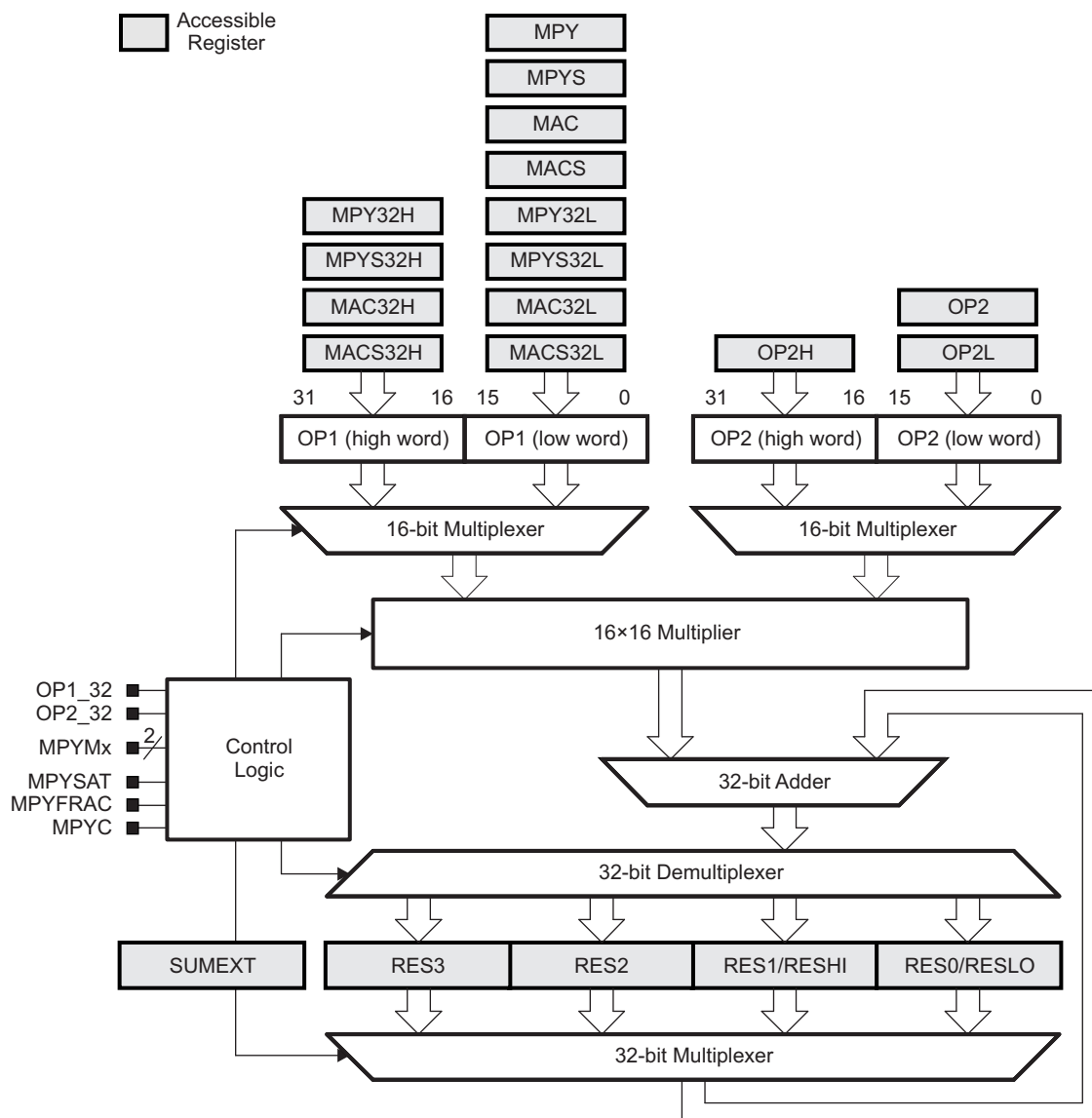


Figure 5-1. MPY32 Block Diagram

5.2 MPY32 Operation

The MPY32 supports 8-bit, 16-bit, 24-bit, and 32-bit operands with unsigned multiply, signed multiply, unsigned multiply-accumulate, and signed multiply-accumulate operations. The size of the operands are defined by the address the operand is written to and if it is written as word or byte. The type of operation is selected by the address the first operand is written to.

The hardware multiplier has two 32-bit operand registers – operand one (OP1) and operand two (OP2), and a 64-bit result register accessible through registers RES0 to RES3. For compatibility with the 16×16 hardware multiplier, the result of a 8-bit or 16-bit operation is accessible through RESLO, RESHI, and SUMEXT, as well. RESLO stores the low word of the 16×16-bit result, RESHI stores the high word of the result, and SUMEXT stores information about the result.

The result of a 8-bit or 16-bit operation is ready in three MCLK cycles and can be read with the next instruction after writing to OP2, except when using an indirect addressing mode to access the result. When using indirect addressing for the result, a NOP is required before the result is ready.

The result of a 24-bit or 32-bit operation can be read with successive instructions after writing OP2 or OP2H starting with RES0, except when using an indirect addressing mode to access the result. When using indirect addressing for the result, a NOP is required before the result is ready.

[Table 5-1](#) summarizes when each word of the 64-bit result is available for the various combinations of operand sizes. With a 32-bit-wide second operand, OP2L and OP2H must be written. Depending on when the two 16-bit parts are written, the result availability may vary; thus, the table shows two entries, one for OP2L written and one for OP2H written. The worst case defines the actual result availability.

Table 5-1. Result Availability (MPYFRAC = 0, MPYSAT = 0)

Operation (OP1 × OP2)	Result Ready in MCLK Cycles					After
	RES0	RES1	RES2	RES3	MPYC Bit	
8/16 × 8/16	3	3	4	4	3	OP2 written
24/32 × 8/16	3	5	6	7	7	OP2 written
8/16 × 24/32	3	5	6	7	7	OP2L written
	N/A	3	4	4	4	OP2H written
24/32 × 24/32	3	8	10	11	11	OP2L written
	N/A	3	5	6	6	OP2H written

5.2.1 Operand Registers

Operand one (OP1) has 12 registers (see [Table 5-2](#)) used to load data into the multiplier and also select the multiply mode. Writing the low word of the first operand to a given address selects the type of multiply operation to be performed, but does not start any operation. When writing a second word to a high-word register with suffix 32H, the multiplier assumes a 32-bit-wide OP1, otherwise, 16 bits are assumed. The last address written prior to writing OP2 defines the width of the first operand. For example, if MPY32L is written first followed by MPY32H, all 32 bits are used and the data width of OP1 is set to 32 bits. If MPY32H is written first followed by MPY32L, the multiplication ignores MPY32H and assumes a 16-bit-wide OP1 using the data written into MPY32L.

Repeated multiply operations may be performed without reloading OP1 if the OP1 value is used for successive operations. It is not necessary to rewrite the OP1 value to perform the operations.

Table 5-2. OP1 Registers

OP1 Register	Operation
MPY	Unsigned multiply – operand bits 0 up to 15
MPYS	Signed multiply – operand bits 0 up to 15
MAC	Unsigned multiply accumulate –operand bits 0 up to 15
MACS	Signed multiply accumulate – operand bits 0 up to 15
MPY32L	Unsigned multiply – operand bits 0 up to 15
MPY32H	Unsigned multiply – operand bits 16 up to 31
MPYS32L	Signed multiply – operand bits 0 up to 15
MPYS32H	Signed multiply – operand bits 16 up to 31
MAC32L	Unsigned multiply accumulate – operand bits 0 up to 15
MAC32H	Unsigned multiply accumulate – operand bits 16 up to 31
MACS32L	Signed multiply accumulate – operand bits 0 up to 15
MACS32H	Signed multiply accumulate – operand bits 16 up to 31

Writing the second operand to the OP2 initiates the multiply operation. Writing OP2 starts the selected operation with a 16-bit-wide second operand together with the values stored in OP1. Writing OP2L starts the selected operation with a 32-bit-wide second operand and the multiplier expects a the high word to be written to OP2H. Writing to OP2H without a preceding write to OP2L is ignored.

Table 5-3. OP2 Registers

OP2 Register	Operation
OP2	Start multiplication with 16-bit-wide OP2 – operand bits 0 up to 15
OP2L	Start multiplication with 32-bit-wide OP2 – operand bits 0 up to 15
OP2H	Continue multiplication with 32-bit-wide OP2 – operand bits 16 up to 31

For 8-bit or 24-bit operands, the operand registers can be accessed with byte instructions. Accessing the multiplier with a byte instruction during a signed operation automatically causes a sign extension of the byte within the multiplier module. For 24-bit operands, only the high word should be written as byte. If the 24-bit operands are sign-extended as defined by the register, that is used to write the low word to, because this register defines if the operation is unsigned or signed.

The high-word of a 32-bit operand remains unchanged when changing the size of the operand to 16 bit, either by modifying the operand size bits or by writing to the respective operand register. During the execution of the 16-bit operation, the content of the high-word is ignored.

NOTE: Changing of first or second operand during multiplication

By default, changing OP1 or OP2 while the selected multiply operation is being calculated renders any results invalid that are not ready at the time the new operands are changed. Writing OP2 or OP2L aborts any ongoing calculation and starts a new operation. Results that are not ready at that time are also invalid for following MAC or MACS operations.

To avoid this behavior, the MPYDLYWRNEN bit can be set to 1. Then, all writes to any MPY32 registers are delayed with MPYDLY32 = 0 until the 64-bit result is ready or with MPYDLY32 = 1 until the 32-bit result is ready. For MAC and MACS operations, the complete 64-bit result should always be ready.

See [Table 5-1](#) for how many CPU cycles are needed until a certain result register is ready and valid for each of the different modes.

5.2.2 Result Registers

The multiplication result is always 64 bits wide. It is accessible through registers RES0 to RES3. Used with a signed operation, MPYS or MACS, the results are appropriately sign extended. If the result registers are loaded with initial values before a MACS operation, the user software must take care that the written value is properly sign extended to 64 bits.

NOTE: Changing of result registers during multiplication

The result registers must not be modified by the user software after writing the second operand into OP2 or OP2L until the initiated operation is completed.

In addition to RES0 to RES3, for compatibility with the 16×16 hardware multiplier, the 32-bit result of a 8-bit or 16-bit operation is accessible through RESLO, RESHI, and SUMEXT. In this case, the result low register RESLO holds the lower 16 bits of the calculation result and the result high register RESHI holds the upper 16 bits. RES0 and RES1 are identical to RESLO and RESHI, respectively, in usage and access of calculated results.

The sum extension register SUMEXT contents depend on the multiply operation and are listed in [Table 5-4](#). If all operands are 16 bits wide or less, the 32-bit result is used to determine sign and carry. If one of the operands is larger than 16 bits, the 64-bit result is used.

The MPYC bit reflects the multiplier's carry as listed in [Table 5-4](#) and, thus, can be used as 33rd or 65th bit of the result, if fractional or saturation mode is not selected. With MAC or MACS operations, the MPYC bit reflects the carry of the 32-bit or 64-bit accumulation and is not taken into account for successive MAC and MACS operations as the 33rd or 65th bit.

Table 5-4. SUMEXT and MPYC Contents

Mode	SUMEXT	MPYC
MPY	SUMEXT is always 0000h.	MPYC is always 0.
MPYS	SUMEXT contains the extended sign of the result. 00000h = Result was positive or zero 0FFFFh = Result was negative	MPYC contains the sign of the result. 0 = Result was positive or zero 1 = Result was negative
MAC	SUMEXT contains the carry of the result. 0000h = No carry for result 0001h =	MPYC contains the carry of the result. 0 = No carry for result 1 = Result has a carry
MACS	SUMEXT contains the extended sign of the result. 00000h = Result was positive or zero 0FFFFh = Result was negative	MPYC contains the carry of the result. 0 = No carry for result 1 = Result has a carry

5.2.2.1 MACS Underflow and Overflow

The multiplier does not automatically detect underflow or overflow in MACS mode. For example, working with 16-bit input data and 32-bit results (that is, using only RESLO and RESHI), the available range for positive numbers is 0 to 07FFF FFFFh and for negative numbers is 0FFFF FFFFh to 08000 0000h. An underflow occurs when the sum of two negative numbers yields a result that is in the range for a positive number. An overflow occurs when the sum of two positive numbers yields a result that is in the range for a negative number.

The SUMEXT register contains the sign of the result in both cases described above, 0FFFFh for a 32-bit overflow and 0000h for a 32-bit underflow. The MPYC bit in MPY32CTL0 can be used to detect the overflow condition. If the carry is different from the sign reflected by the SUMEXT register, an overflow or underflow occurred. User software must handle these conditions appropriately.

5.2.3 Software Examples

Examples for all multiplier modes follow. All 8×8 modes use the absolute address for the registers, because the assembler does not allow .B access to word registers when using the labels from the standard definitions file.

There is no sign extension necessary in software. Accessing the multiplier with a byte instruction during a signed operation automatically causes a sign extension of the byte within the multiplier module.

```
; 32x32 Unsigned Multiply
    MOV    #01234h,&MPY32L    ; Load low word of 1st operand
    MOV    #01234h,&MPY32H    ; Load high word of 1st operand
    MOV    #05678h,&OP2L      ; Load low word of 2nd operand
    MOV    #05678h,&OP2H      ; Load high word of 2nd operand
;    ...                      ; Process results

; 16x16 Unsigned Multiply
    MOV    #01234h,&MPY        ; Load 1st operand
    MOV    #05678h,&OP2        ; Load 2nd operand
;    ...                      ; Process results

; 8x8 Unsigned Multiply. Absolute addressing.
    MOV.B  #012h,&MPY_B        ; Load 1st operand
    MOV.B  #034h,&OP2_B        ; Load 2nd operand
;    ...                      ; Process results

; 32x32 Signed Multiply
    MOV    #01234h,&MPYS32L    ; Load low word of 1st operand
    MOV    #01234h,&MPYS32H    ; Load high word of 1st operand
    MOV    #05678h,&OP2L      ; Load low word of 2nd operand
    MOV    #05678h,&OP2H      ; Load high word of 2nd operand
;    ...                      ; Process results

; 16x16 Signed Multiply
    MOV    #01234h,&MPYS        ; Load 1st operand
    MOV    #05678h,&OP2        ; Load 2nd operand
;    ...                      ; Process results

; 8x8 Signed Multiply. Absolute addressing.
    MOV.B  #012h,&MPYS_B        ; Load 1st operand
    MOV.B  #034h,&OP2_B        ; Load 2nd operand
;    ...                      ; Process results
```

5.2.4 Fractional Numbers

The MPY32 provides support for fixed-point signal processing. In fixed-point signal processing, fractional numbers are numbers that have a fixed number of digits after (and sometimes also before) the radix point. To classify different ranges of binary fixed-point numbers, a Q-format is used. Different Q-formats represent different locations of the radix point. Figure 5-2 shows the format of a signed Q15 number using 16 bits. Every bit after the radix point has a resolution of $1/2$, and the most significant bit (MSB) is used as the sign bit. The most negative number is 08000h and the maximum positive number is 07FFFh. This gives a range from -1.0 to $0.999969482 \approx 1.0$ for the signed Q15 format with 16 bits.

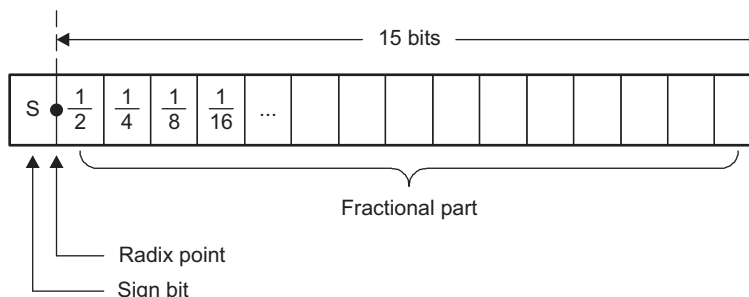


Figure 5-2. Q15 Format Representation

The range can be increased by shifting the radix point to the right as shown in Figure 5-3. The signed Q14 format with 16 bits gives a range from -2.0 to $1.999938965 \approx 2.0$.

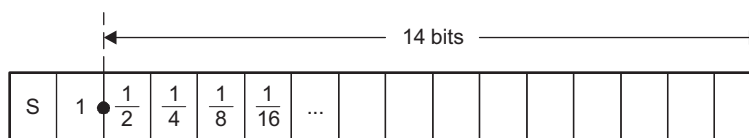


Figure 5-3. Q14 Format Representation

The benefit of using 16-bit signed Q15 or 32-bit signed Q31 numbers with multiplication is that the product of two numbers in the range from -1.0 to 1.0 is always in that same range.

5.2.4.1 Fractional Number Mode

Multiplying two fractional numbers using the default multiplication mode with $MPYFRAC = 0$ and $MPYSAT = 0$ gives a result with two sign bits. For example, if two 16-bit Q15 numbers are multiplied, a 32-bit result in Q30 format is obtained. To convert the result into Q15 format manually, the first 15 trailing bits and the extended sign bit must be removed. However, when the fractional mode of the multiplier is used, the redundant sign bit is automatically removed, yielding a result in Q31 format for the multiplication of two 16-bit Q15 numbers. Reading the result register RES1 gives the result as 16-bit Q15 number. The 32-bit Q31 result of a multiplication of two 32-bit Q31 numbers is accessed by reading registers RES2 and RES3.

The fractional mode is enabled with $MPYFRAC = 1$ in register MPY32CTL0. The actual content of the result registers is not modified when $MPYFRAC = 1$. When the result is accessed using software, the value is left shifted one bit, resulting in the final Q formatted result. This allows user software to switch between reading both the shifted (fractional) and the unshifted result. The fractional mode should only be enabled when required and disabled after use.

In fractional mode, the SUMEXT register contains the sign extended bits 32 and 33 of the shifted result for 16×16 -bit operations and bits 64 and 65 for 32×32 -bit operations – not only bits 32 or 64, respectively.

The MPYC bit is not affected by the fractional mode. It always reads the carry of the nonfractional result.

```
; Example using
; Fractional 16x16 multiplication
BIS      #MPYFRAC,&MPY32CTL0 ; Turn on fractional mode
MOV      &FRACT1,&MPYS        ; Load 1st operand as Q15
MOV      &FRACT2,&OP2          ; Load 2nd operand as Q15
MOV      &RES1,&PROD           ; Save result as Q15
BIC      #MPYFRAC,&MPY32CTL0 ; Back to normal mode
```

Table 5-5. Result Availability in Fractional Mode (MPYFRAC = 1, MPYSAT = 0)

Operation (OP1 × OP2)	Result Ready in MCLK Cycles					After
	RES0	RES1	RES2	RES3	MPYC Bit	
8/16 × 8/16	3	3	4	4	3	OP2 written
24/32 × 8/16	3	5	6	7	7	OP2 written
8/16 × 24/32	3	5	6	7	7	OP2L written
	N/A	3	4	4	4	OP2H written
24/32 × 24/32	3	8	10	11	11	OP2L written
	N/A	3	5	6	6	OP2H written

5.2.4.2 Saturation Mode

The multiplier prevents overflow and underflow of signed operations in saturation mode. The saturation mode is enabled with MPYSAT = 1 in register MPY32CTL0. If an overflow occurs, the result is set to the most-positive value available. If an underflow occurs, the result is set to the most-negative value available. This is useful to reduce mathematical artifacts in control systems on overflow and underflow conditions. The saturation mode should only be enabled when required and disabled after use.

The actual content of the result registers is not modified when MPYSAT = 1. When the result is accessed using software, the value is automatically adjusted to provide the most-positive or most-negative result when an overflow or underflow has occurred. The adjusted result is also used for successive multiply-and-accumulate operations. This allows user software to switch between reading the saturated and the nonsaturated result.

With 16×16 operations, the saturation mode only applies to the least significant 32 bits; that is, the result registers RES0 and RES1. Using the saturation mode in MAC or MACS operations that mix 16×16 operations with 32×32, 16×32, or 32×16 operations leads to unpredictable results.

With 32×32, 16×32, and 32×16 operations, the saturated result can only be calculated when RES3 is ready.

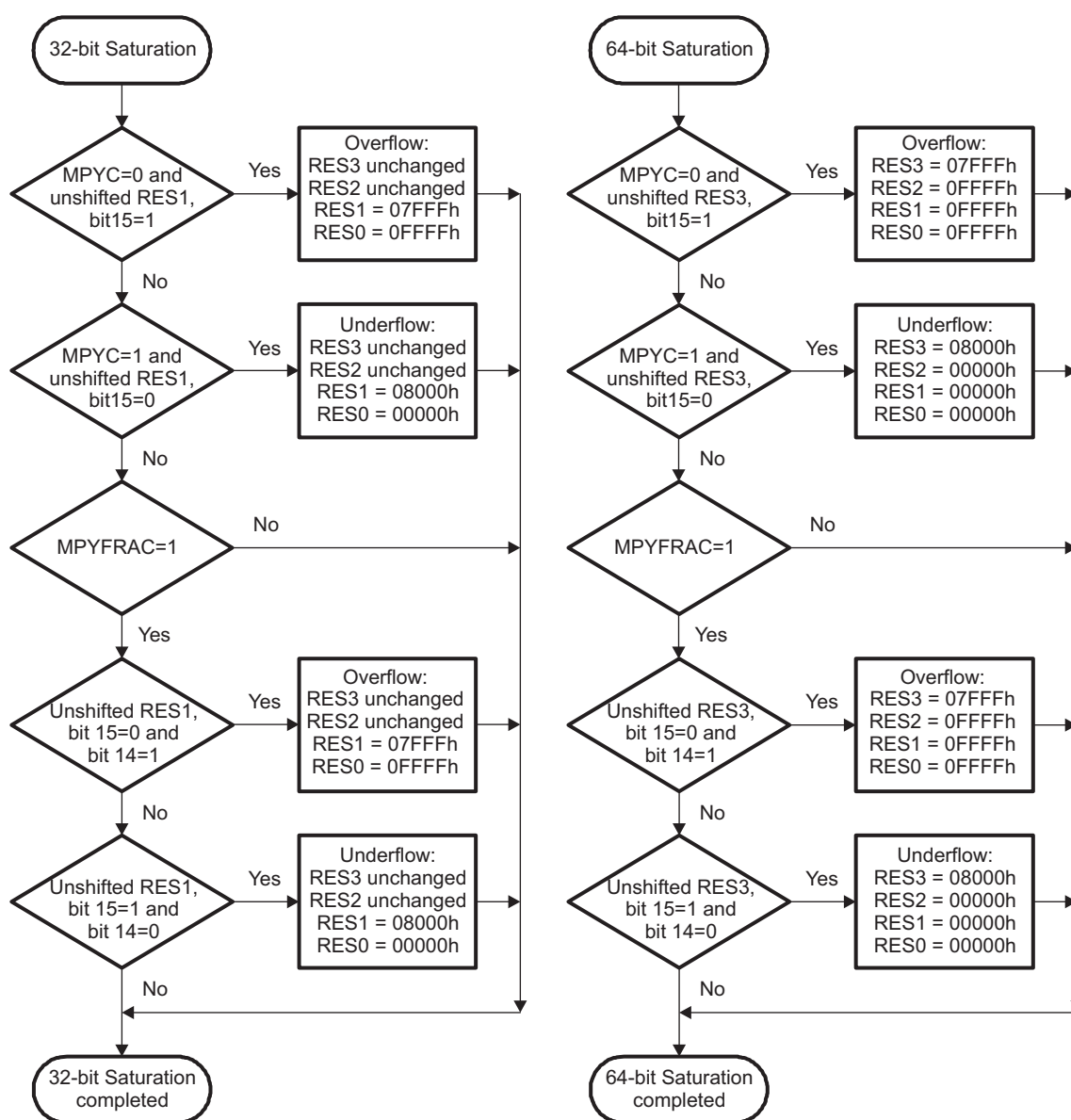
Enabling the saturation mode does not affect the content of the SUMEXT register nor the content of the MPYC bit.

```
; Example using
; Fractional 16x16 multiply accumulate with Saturation
; Turn on fractional and saturation mode:
BIS      #MPYSAT+MPYFRAC,&MPY32CTL0
MOV      &A1,&MPYS              ; Load A1 for 1st term
MOV      &K1,&OP2              ; Load K1 to get A1*K1
MOV      &A2,&MACS             ; Load A2 for 2nd term
MOV      &K2,&OP2              ; Load K2 to get A2*K2
MOV      &RES1,&PROD           ; Save A1*K1+A2*K2 as result
BIC      #MPYSAT+MPYFRAC,&MPY32CTL0 ; turn back to normal
```


Table 5-6. Result Availability in Saturation Mode (MPYSAT = 1)

Operation (OP1 × OP2)	Result Ready in MCLK Cycles					After
	RES0	RES1	RES2	RES3	MPYC Bit	
8/16 × 8/16	3	3	N/A	N/A	3	OP2 written
24/32 × 8/16	7	7	7	7	7	OP2 written
8/16 × 24/32	7	7	7	7	7	OP2L written
	4	4	4	4	4	OP2H written
24/32 × 24/32	11	11	11	11	11	OP2L written
	6	6	6	6	6	OP2H written

Figure 5-4 shows the flow for 32-bit saturation used for 16×16 bit multiplications and the flow for 64-bit saturation used in all other cases. Primarily, the saturated results depends on the carry bit MPYC and the MSB of the result. Secondly, if the fractional mode is enabled, it depends also on the two MSBs of the unshift result, that is, the result that is read with fractional mode disabled.


Figure 5-4. Saturation Flow Chart

NOTE: Saturation in fractional mode

In case of multiplying -1.0×-1.0 in fractional mode, the result of $+1.0$ is out of range, thus, the saturated result gives the most positive result.

When using multiply-and-accumulate operations, the accumulated values are saturated as if $MPYFRAC = 0$; only during read accesses to the result registers the values are saturated taking the fractional mode into account. This provides additional dynamic range during the calculation and only the end result is then saturated if needed.

The following example illustrates a special case showing the saturation function in fractional mode. It also uses the 8-bit functionality of the MPY32 module.

```
; Turn on fractional and saturation mode,
; clear all other bits in MPY32CTL0:
MOV    #MPYSAT+MPYFRAC,&MPY32CTL0
;Pre-load result registers to demonstrate overflow
MOV     #0,&RES3      ;
MOV     #0,&RES2      ;
MOV     #07FFFh,&RES1  ;
MOV     #0FA60h,&RES0  ;
MOV.B   #050h,&MACS_B   ; 8-bit signed MAC operation
MOV.B   #012h,&OP2_B    ; Start 16x16 bit operation
MOV     &RES0,R6       ; R6 = 0FFFFh
MOV     &RES1,R7       ; R7 = 07FFFh
```

The result is saturated because already the result not converted into a fractional number shows an overflow. The multiplication of the two positive numbers 00050h and 00012h gives 005A0h. 005A0h added to 07FFF FA60h results in 8000 059Fh, without MPYC being set. Because the MSB of the unmodified result RES1 is 1 and $MPYC = 0$, the result is saturated according [Figure 5-4](#).

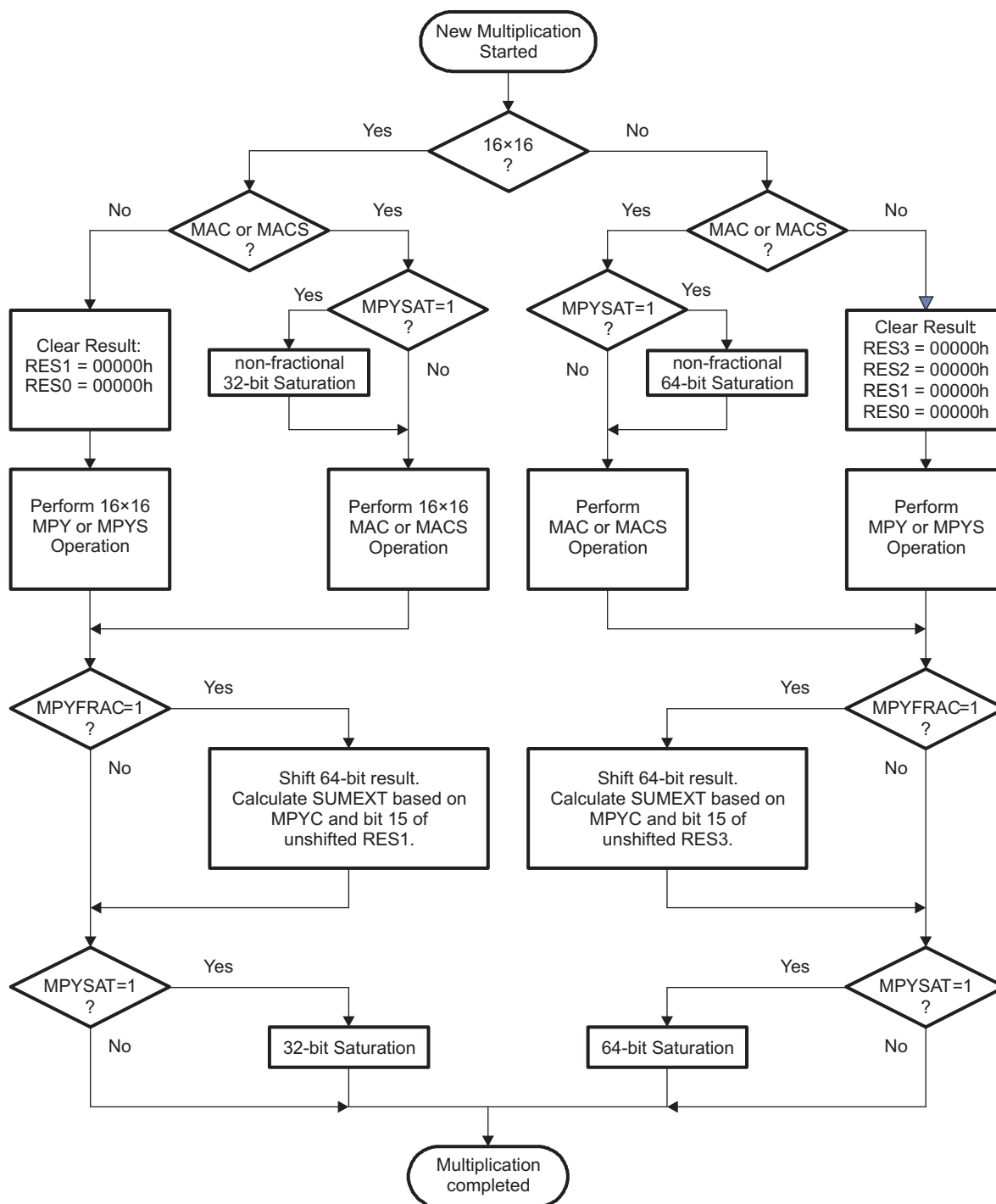
NOTE: Validity of saturated result

The saturated result is valid only if the registers RES0 to RES3, the size of OP1 and OP2, and MPYC are not modified.

If the saturation mode is used with a preloaded result, user software must ensure that MPYC in the MPY32CTL0 register is loaded with the sign bit of the written result; otherwise, the saturation mode erroneously saturates the result.

5.2.5 Putting It All Together

[Figure 5-5](#) shows the complete multiplication flow, depending on the various selectable modes for the MPY32 module.


Figure 5-5. Multiplication Flow Chart

Given the separation in processing of 16-bit operations (32-bit results) and 32-bit operations (64-bit results) by the module, it is important to understand the implications when using MAC/MACS operations and mixing 16-bit operands or results with 32-bit operands or results. User software must address these points during use when mixing these operations. The following code illustrates the issue.

```
; Mixing 32x24 multiplication with 16x16 MACS operation
MOV     #MPYSAT,&MPY32CTL0    ; Saturation mode
MOV     #052C5h,&MPY32L       ; Load low word of 1st operand
MOV     #06153h,&MPY32H       ; Load high word of 1st operand
MOV     #001ABh,&OP2L          ; Load low word of 2nd operand
MOV.B   #023h,&OP2H_B          ; Load high word of 2nd operand
                                           ; ... 5 NOPs required

MOV     &RES0,R6               ; R6 = 00E97h
MOV     &RES1,R7               ; R7 = 0A6EAh
MOV     &RES2,R8               ; R8 = 04F06h
MOV     &RES3,R9               ; R9 = 0000Dh
                                           ; Note that MPYC = 0!

MOV     #0CCC3h,&MACS           ; Signed MAC operation
MOV     #0FFB6h,&OP2           ; 16x16 bit operation
MOV     &RESL0,R6              ; R6 = 0FFFFh
MOV     &RESH1,R7              ; R7 = 07FFFh
```

The second operation gives a saturated result because the 32-bit value used for the 16×16-bit MACS operation was already saturated when the operation was started; the carry bit MPYC was 0 from the previous operation, but the MSB in result register RES1 is set. As one can see in the flow chart, the content of the result registers are saturated for multiply-and-accumulate operations after starting a new operation based on the previous results, but depending on the size of the result (32 bit or 64 bit) of the newly initiated operation.

The saturation before the multiplication can cause issues if the MPYC bit is not properly set as the following code shows.

```
;Pre-load result registers to demonstrate overflow
MOV     #0,&RES3               ;
MOV     #0,&RES2               ;
MOV     #0,&RES1               ;
MOV     #0,&RES0               ;
; Saturation mode and set MPYC:
MOV     #MPYSAT+MPYC,&MPY32CTL0
MOV.B   #082h,&MACS_B          ; 8-bit signed MAC operation
MOV.B   #04Fh,&OP2_B           ; Start 16x16 bit operation
MOV     &RES0,R6               ; R6 = 00000h
MOV     &RES1,R7               ; R7 = 08000h
```

Even though the result registers were loaded with all zeros, the final result is saturated. This is because the MPYC bit was set, causing the result used for the multiply-and-accumulate to be saturated to 08000 0000h. Adding a negative number to it would again cause an underflow, thus, the final result is also saturated to 08000 0000h.

5.2.6 Indirect Addressing of Result Registers

When using indirect or indirect autoincrement addressing mode to access the result registers and the multiplier requires three cycles until result availability according to [Table 5-1](#), at least one instruction is needed between loading the second operand and accessing the result registers:

```
; Access multiplier 16x16 results with indirect addressing
MOV    #RES0,R5          ; RES0 address in R5 for indirect
MOV    &OPER1,&MPY        ; Load 1st operand
MOV    &OPER2,&OP2        ; Load 2nd operand
NOP                      ; Need one cycle
MOV    @R5+,&xxx          ; Move RES0
MOV    @R5,&xxx           ; Move RES1
```

In case of a 32×16 multiplication, there is also one instruction required between reading the first result register RES0 and the second result register RES1:

```
; Access multiplier 32x16 results with indirect addressing
MOV    #RES0,R5          ; RES0 address in R5 for indirect
MOV    &OPER1L,&MPY32L    ; Load low word of 1st operand
MOV    &OPER1H,&MPY32H    ; Load high word of 1st operand
MOV    &OPER2,&OP2        ; Load 2nd operand (16 bits)
NOP                      ; Need one cycle
MOV    @R5+,&xxx          ; Move RES0
NOP                      ; Need one additional cycle
MOV    @R5,&xxx           ; Move RES1
                          ; No additional cycles required!
MOV    @R5,&xxx           ; Move RES2
```

5.2.7 Using Interrupts

If an interrupt occurs after writing OP, but before writing OP2, and the multiplier is used in servicing that interrupt, the original multiplier mode selection is lost and the results are unpredictable. To avoid this, disable interrupts before using the MPY32, do not use the MPY32 in interrupt service routines, or use the save and restore functionality of the MPY32.

```
; Disable interrupts before using the hardware multiplier
DINT                      ; Disable interrupts
NOP                      ; Required for DINT
MOV    #xxh,&MPY          ; Load 1st operand
MOV    #xxh,&OP2          ; Load 2nd operand
EINT                      ; Interrupts may be enabled before
                          ; processing results if result
                          ; registers are stored and restored in
                          ; interrupt service routines
```

5.2.7.1 Save and Restore

If the multiplier is used in interrupt service routines, its state can be saved and restored using the MPY32CTL0 register. The following code example shows how the complete multiplier status can be saved and restored to allow interruptible multiplications together with the usage of the multiplier in interrupt service routines. Because the state of the MPYSAT and MPYFRAC bits are unknown, they should be cleared before the registers are saved as shown in the code example.

```
; Interrupt service routine using multiplier
MPY_USING_ISR
    PUSH    &MPY32CTL0      ; Save multiplier mode, etc.
    BIC     #MPYSAT+MPYFRAC,&MPY32CTL0
                                ; Clear MPYSAT+MPYFRAC

    PUSH    &RES3            ; Save result 3
    PUSH    &RES2            ; Save result 2
    PUSH    &RES1            ; Save result 1
    PUSH    &RES0            ; Save result 0
    PUSH    &MPY32H          ; Save operand 1, high word
    PUSH    &MPY32L          ; Save operand 1, low word
    PUSH    &OP2H            ; Save operand 2, high word
    PUSH    &OP2L            ; Save operand 2, low word
                                ;
    ...                        ; Main part of ISR
                                ; Using standard MPY routines
                                ;
    POP     &OP2L            ; Restore operand 2, low word
    POP     &OP2H            ; Restore operand 2, high word
                                ; Starts dummy multiplication but
                                ; result is overwritten by
                                ; following restore operations:
    POP     &MPY32L          ; Restore operand 1, low word
    POP     &MPY32H          ; Restore operand 1, high word
    POP     &RES0            ; Restore result 0
    POP     &RES1            ; Restore result 1
    POP     &RES2            ; Restore result 2
    POP     &RES3            ; Restore result 3
    POP     &MPY32CTL0       ; Restore multiplier mode, etc.
    reti                                ; End of interrupt service routine
```

5.2.8 Using DMA

In devices with a DMA controller, the multiplier can trigger a transfer when the complete result is available. The DMA controller needs to start reading the result with MPY32RES0 successively up to MPY32RES3. Not all registers need to be read. The trigger timing is such that the DMA controller starts reading MPY32RES0 when its ready, and that the MPY32RES3 can be read exactly in the clock cycle when it is available to allow the fastest access through the DMA. The signal into the DMA controller is 'Multiplier ready' (see the *DMA Controller* chapter for details).

5.3 MPY32 Registers

MPY32 registers are listed in [Table 5-7](#). The base address can be found in the device-specific data sheet. The address offsets are listed in [Table 5-7](#).

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 5-7. MPY32 Registers

Offset	Acronym	Register Name	Type	Access	Reset
00h	MPY	16-bit operand one – multiply	Read/write	Word	Undefined
00h	MPY_L		Read/write	Byte	Undefined
01h	MPY_H		Read/write	Byte	Undefined
00h	MPY_B	8-bit operand one – multiply	Read/write	Byte	Undefined
02h	MPYS	16-bit operand one – signed multiply	Read/write	Word	Undefined
02h	MPYS_L		Read/write	Byte	Undefined
03h	MPYS_H		Read/write	Byte	Undefined
02h	MPYS_B	8-bit operand one – signed multiply	Read/write	Byte	Undefined
04h	MAC	16-bit operand one – multiply accumulate	Read/write	Word	Undefined
04h	MAC_L		Read/write	Byte	Undefined
05h	MAC_H		Read/write	Byte	Undefined
04h	MAC_B	8-bit operand one – multiply accumulate	Read/write	Byte	Undefined
06h	MACS	16-bit operand one – signed multiply accumulate	Read/write	Word	Undefined
06h	MACS_L		Read/write	Byte	Undefined
07h	MACS_H		Read/write	Byte	Undefined
06h	MACS_B	8-bit operand one – signed multiply accumulate	Read/write	Byte	Undefined
08h	OP2	16-bit operand two	Read/write	Word	Undefined
08h	OP2_L		Read/write	Byte	Undefined
09h	OP2_H		Read/write	Byte	Undefined
08h	OP2_B	8-bit operand two	Read/write	Byte	Undefined
0Ah	RESLO	16x16-bit result low word	Read/write	Word	Undefined
0Ah	RESLO_L		Read/write	Byte	Undefined
0Ch	RESHI	16x16-bit result high word	Read/write	Word	Undefined
0Eh	SUMEXT	16x16-bit sum extension register	Read	Word	Undefined
10h	MPY32L	32-bit operand 1 – multiply – low word	Read/write	Word	Undefined
10h	MPY32L_L		Read/write	Byte	Undefined
11h	MPY32L_H		Read/write	Byte	Undefined
12h	MPY32H	32-bit operand 1 – multiply – high word	Read/write	Word	Undefined
12h	MPY32H_L		Read/write	Byte	Undefined
13h	MPY32H_H		Read/write	Byte	Undefined
12h	MPY32H_B	24-bit operand 1 – multiply – high byte	Read/write	Byte	Undefined
14h	MPYS32L	32-bit operand 1 – signed multiply – low word	Read/write	Word	Undefined
14h	MPYS32L_L		Read/write	Byte	Undefined
15h	MPYS32L_H		Read/write	Byte	Undefined
16h	MPYS32H	32-bit operand 1 – signed multiply – high word	Read/write	Word	Undefined
16h	MPYS32H_L		Read/write	Byte	Undefined
17h	MPYS32H_H		Read/write	Byte	Undefined
16h	MPYS32H_B	24-bit operand 1 – signed multiply – high byte	Read/write	Byte	Undefined
18h	MAC32L	32-bit operand 1 – multiply accumulate – low word	Read/write	Word	Undefined

Table 5-7. MPY32 Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset
18h	MAC32L_L	32-bit operand 1 – multiply accumulate – high word	Read/write	Byte	Undefined
19h	MAC32L_H		Read/write	Byte	Undefined
1Ah	MAC32H		Read/write	Word	Undefined
1Ah	MAC32H_L	24-bit operand 1 – multiply accumulate – high byte	Read/write	Byte	Undefined
1Bh	MAC32H_H		Read/write	Byte	Undefined
1Ah	MAC32H_B		Read/write	Byte	Undefined
1Ch	MACS32L	32-bit operand 1 – signed multiply accumulate – low word	Read/write	Word	Undefined
1Ch	MACS32L_L		Read/write	Byte	Undefined
1Dh	MACS32L_H		Read/write	Byte	Undefined
1Eh	MACS32H	32-bit operand 1 – signed multiply accumulate – high word	Read/write	Word	Undefined
1Eh	MACS32H_L		Read/write	Byte	Undefined
1Fh	MACS32H_H		Read/write	Byte	Undefined
1Eh	MACS32H_B	24-bit operand 1 – signed multiply accumulate – high byte	Read/write	Byte	Undefined
20h	OP2L	32-bit operand 2 – low word	Read/write	Word	Undefined
20h	OP2L_L		Read/write	Byte	Undefined
21h	OP2L_H		Read/write	Byte	Undefined
22h	OP2H	32-bit operand 2 – high word	Read/write	Word	Undefined
22h	OP2H_L		Read/write	Byte	Undefined
23h	OP2H_H		Read/write	Byte	Undefined
22h	OP2H_B	24-bit operand 2 – high byte	Read/write	Byte	Undefined
24h	RES0	32x32-bit result 0 – least significant word	Read/write	Word	Undefined
24h	RES0_L		Read/write	Byte	Undefined
26h	RES1	32x32-bit result 1	Read/write	Word	Undefined
28h	RES2	32x32-bit result 2	Read/write	Word	Undefined
2Ah	RES3	32x32-bit result 3 – most significant word	Read/write	Word	Undefined
2Ch	MPY32CTL0	MPY32 control register 0	Read/write	Word	Undefined
2Ch	MPY32CTL0_L		Read/write	Byte	Undefined
2Dh	MPY32CTL0_H		Read/write	Byte	00h

The registers listed in [Table 5-8](#) are treated equally.

Table 5-8. Alternative Registers

Register	Alternative 1	Alternative 2
16-bit operand one – multiply	MPY	MPY32L
8-bit operand one – multiply	MPY_B or MPY_L	MPY32L_B or MPY32L_L
16-bit operand one – signed multiply	MPYS	MPYS32L
8-bit operand one – signed multiply	MPYS_B or MPYS_L	MPYS32L_B or MPYS32L_L
16-bit operand one – multiply accumulate	MAC	MAC32L
8-bit operand one – multiply accumulate	MAC_B or MAC_L	MAC32L_B or MAC32L_L
16-bit operand one – signed multiply accumulate	MACS	MACS32L
8-bit operand one – signed multiply accumulate	MACS_B or MACS_L	MACS32L_B or MACS32L_L
16x16-bit result low word	RESLO	RES0
16x16-bit result high word	RESHI	RES1

5.3.1 MPY32CTL0 Register

32-Bit Hardware Multiplier Control 0 Register

Figure 5-6. MPY32CTL0 Register

15	14	13	12	11	10	9	8
Reserved						MPYDLY32	MPYDLYWRN
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0
7	6	5	4	3	2	1	0
MPYOP2_32	MPYOP1_32	MPYMx		MPYSAT	MPYFRAC	Reserved	MPYC
rw	rw	rw	rw	rw-0	rw-0	rw-0	rw

Table 5-9. MPY32CTL0 Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved. Always reads as 0.
9	MPYDLY32	RW	0h	Delayed write mode 0b = Writes are delayed until 64-bit result (RES0 to RES3) is available. 1b = Writes are delayed until 32-bit result (RES0 to RES1) is available.
8	MPYDLYWRN	RW	0h	Delayed write enable All writes to any MPY32 register are delayed until the 64-bit (MPYDLY32 = 0) or 32-bit (MPYDLY32 = 1) result is ready. 0b = Writes are not delayed. 1b = Writes are delayed.
7	MPYOP2_32	RW	0h	Multiplier bit width of operand 2 0b = 16 bits 1b = 32 bits
6	MPYOP1_32	RW	0h	Multiplier bit width of operand 1 0b = 16 bits 1b = 32 bits
5-4	MPYMx	RW	0h	Multiplier mode 00b = MPY – Multiply 01b = MPYS – Signed multiply 10b = MAC – Multiply accumulate 11b = MACS – Signed multiply accumulate
3	MPYSAT	RW	0h	Saturation mode 0b = Saturation mode disabled 1b = Saturation mode enabled
2	MPYFRAC	RW	0h	Fractional mode 0b = Fractional mode disabled 1b = Fractional mode enabled
1	Reserved	RW	0h	Reserved. Always reads as 0.
0	MPYC	RW	0h	Carry of the multiplier. It can be considered as 33rd or 65th bit of the result if fractional or saturation mode is not selected, because the MPYC bit does not change when switching to saturation or fractional mode. It is used to restore the SUMEXT content in MAC mode. 0b = No carry for result 1b = Result has a carry

FRAM Controller Overview

6.1 FRAM Controller Overview

Table 6-1 summarizes the differences between the FRCTL and FRCTL_A modules. See the full feature descriptions in the [FRCTL chapter](#) and [FRCTL_A chapter](#) for details. A device can include only one FRAM controller, either FRCTL or FRCTL_A. See the functional block diagram in the device-specific data sheet to determine the supported FRAM controller (if FRCTL_A is not specified in the block diagram, the device supports FRCTL).

Table 6-1. FRAM Controller Overview

Feature		FRCTL	FRCTL_A
Wait state control	Automatic wait state mode	No	Yes
	User wait state mode	Yes	Yes
		Control Bits: NWAITS[2:0]	Control Bits: NWAITS[3:0]
		Timing violation interrupt: ACCTEIFG bit and a reset (PUC); always enabled	Timing violation interrupt: ACCTEIFG bit; enabled by the ACCTEIE bit
Write protection	MPU (see Chapter 9)	Yes	Yes
	Temporary protection - whole FRAM memory	No	Yes (WPROT bit)
Power control	FRAM on or off in AM	FRPWR bit	FRPWR bit
	FRAM power status when the device wakes up from a LPM	FRLPMPWR bit	FRPWR bit

FRAM Controller (FRCTL)

This chapter describes the operation of the FRAM controller.

Topic	Page
7.1 FRAM Introduction.....	290
7.2 FRAM Organization.....	290
7.3 FRCTL Module Operation	290
7.4 Programming FRAM Devices	291
7.5 Wait State Control	291
7.6 FRAM ECC.....	292
7.7 FRAM Write Back	292
7.8 FRAM Power Control	292
7.9 FRAM Cache	293
7.10 FRCTL Registers	294

7.1 FRAM Introduction

FRAM is a nonvolatile memory that reads and writes like standard SRAM. The MSP430 FRAM features include:

- Byte or word write access
- Automatic and programmable wait state control with independent wait state settings for access and cycle times
- Error correction code with bit error correction, extended bit error detection and flag indicators
- Cache for fast read
- Power control for disabling FRAM if it is not used

For important software design information regarding FRAM, including but not limited to partitioning the memory layout according to application-specific code, constant, and data space requirements, the use of FRAM to optimize application energy consumption, and the use of the memory protection unit (MPU) to maximize application robustness by protecting the program code against unintended write accesses, see [MSP430™ FRAM Technology – How To and Best Practices](#).

Figure 7-1 shows the block diagram of the FRAM Controller.

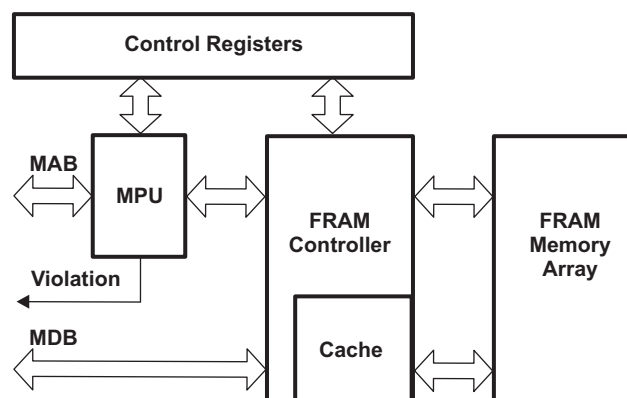


Figure 7-1. FRAM Controller Block Diagram

7.2 FRAM Organization

The FRAM can be arranged into segments by the Memory Protection Unit (MPU). See [Chapter 9, Memory Protection Unit](#), for details. The address space is linear with the exception of the User Information Memory and the Device Descriptor Information (TLV).

7.3 FRCTL Module Operation

The FRAM can be read in a similar fashion to SRAM and needs no special requirements. Similarly, any writes to unprotected segments can be written in the same fashion as SRAM. All writes to user protected segments are handled as described in [Chapter 9, Memory Protection Unit](#).

An FRAM read always requires a write back to the same memory location with the same information read. This write back is part of the FRAM module itself and requires no user interaction. These write backs are different from the normal write access from application code.

The FRAM module has built-in error correction code (ECC) logic that can correct bit errors and detect multiple bit errors. Two flags are available that indicate the presence of an error. The CBDIFG is set when a correctable bit error has been detected. If CBDIE is also set, a System NMI event (SYSNMI) occurs. The UBDIFG is set when a multiple bit error that is not correctable has been detected. If UBDIE is also set, a System NMI event (SYSNMI) occurs. Upon correctable or uncorrectable bit errors, the program vectors to the SYSSNIV if the NMI is enabled. If desired, a System Reset event (SYSRST) can be generated by setting the UBDRSTEN bit. If an uncorrectable error is detected, a PUC is initiated and the program vectors to the SYSRSTIV.

7.4 Programming FRAM Devices

There are three options for programming an MSP430 FRAM device. All options support in-system programming.

- Program with JTAG or the Spy-Bi-Wire interface
- Program with the BSL
- Program with a custom solution

7.4.1 Programming FRAM With JTAG or Spy-Bi-Wire

Devices can be programmed through the JTAG port or the Spy-Bi-Wire port. The JTAG interface requires access to TDI, TDO, TMS, TCK, TEST, ground, and optionally VCC and $\overline{\text{RST/NMI}}$. Spy-Bi-Wire interface requires access to TEST, $\overline{\text{RST/NMI}}$, ground and optionally VCC. For more details, see [MSP430 Programming With the JTAG Interface](#).

7.4.2 Programming FRAM With the Bootloader (BSL)

Every device contains a BSL stored in ROM. The BSL enables users to read or program the FRAM or RAM using a UART serial interface. Access to the FRAM through the BSL is protected by a 256-bit user-defined password. For more details, see the [MSP430FR57xx](#), [MSP430FR58xx](#), [MSP430FR59xx](#), [MSP430FR68xx](#), and [MSP430FR69xx Bootloader \(BSL\) User's Guide](#).

7.4.3 Programming FRAM With a Custom Solution

The ability of the CPU to write to its own FRAM allows for in-system and external custom programming solutions. The user can choose to provide data to the device through any means available (for example, UART or SPI). User-developed software can receive the data and program the FRAM. Because this type of solution is developed by the user, it can be completely customized to fit the application needs for programming or updating the FRAM.

7.5 Wait State Control

The system clock for the CPU or DMA can exceed the FRAM access and cycle time requirements. For these scenarios, a wait state generator mechanism is implemented. The *Recommended Operating Conditions* of the device-specific data sheet list the frequency ranges with the required wait state settings. The number of wait states is controlled by the NWAITS[2:0] bits in the FRCTL0 register.

To increase the system clock frequency beyond the maximum frequency allowed by the current wait state setting, the following steps are required:

1. Increase the number of wait states by configuring NWAITS[2:0] according to the target frequency.
2. Increase the frequency to the new target.

To decrease the system clock frequency to a range that supports fewer wait states, the following steps are required:

1. Decrease frequency to the new target.
2. Decrease number of wait states by configuring NWAITS[2:0] according to the new frequency setting.

To ensure memory integrity, a mechanism is implemented to reset the device with a PUC if the system clock frequency and the wait state settings violate the FRAM access timing.

NOTE: Wait State Settings

- The device starts with zero wait states.
 - Correct wait state settings must be ensured, otherwise a PUC might be generated to avoid erratic FRAM accesses.
-

7.5.1 Wait State and Cache Hit

The FRAM controller contains a cache with two cache sets. Each of these cache sets contains two lines that are preloaded with four words (64 bits) during one access cycle. An intelligent logic selects one of the cache lines to preload FRAM data and preserves recently accessed data in the other cache. If one of the four words stored in one of the cache lines is requested (a cache hit), no FRAM access occurs; instead, a cache request occurs. No wait state is needed for a cache request, and the data is accessed with full system speed. However, if none of the words that are available in the cache are requested (a cache miss), the wait state controls the CPU to ensure proper FRAM access.

7.6 FRAM ECC

FRAM ECC supports bit error correction and uncorrectable bit error detection.

Correctable errors are generally single-bit errors that are detected and corrected by the hardware, so they do not result in data corruption or system failure. The CBDIFG FRAM correctable bit error flag is set if a correctable bit error has been detected and corrected. CBDIE can be used to enable an NMI event.

Uncorrectable bit errors are always multiple-bit errors, and they indicate memory corruption. The UBDIFG FRAM uncorrectable bit error flag is set if an uncorrectable bit error has been detected in the FRAM error detection logic. UBDIRSTEN can be used to enable a power up clear (PUC) reset, or UBDIE can be used to enable an NMI event. UBDIRSTEN and UBDIE are mutually exclusive and are not allowed to be set simultaneously.

For more information, refer to the [MSP430 FRAM Quality and Reliability](#) application report.

7.7 FRAM Write Back

All reads from FRAM requires a write back of the previously read content. This write back is performed under all circumstances without any interaction from a user.

7.8 FRAM Power Control

The FRAM controller can disable the power supply for the FRAM array. By setting FRPWR = 0, the FRAM array supply is disabled. Register accesses in the FRAM controller are still possible. Memory accesses pointing into the FRAM address space automatically reset the FRPWR = 1 and re-enable the power supply of the FRAM. A second control bit, FRLPMPWR, delays the power-up of the FRAM after LPM exit. With FRLPMPWR = 1, the FRAM is activated directly on exit from LPM. FRLPMPWR = 0 delays the activation of the FRAM to the first access into the FRAM address space. For LPM0, the FRAM power state during LPM0 is determined from the previous state in active mode. If FRAM power is disabled, a memory access automatically inserts wait states to ensure sufficient timing for the FRAM power-up and access. Access to FRAM that can be served from cache does not change the power state of the FRAM power control.

A PUC reset forces the state machine to Active mode with FRAM enabled. The CPU must execute from RAM to clear the FRPWR bit for turning off power to FRAM.

[Figure 7-2](#) shows the activation flow of the FRAM controller.

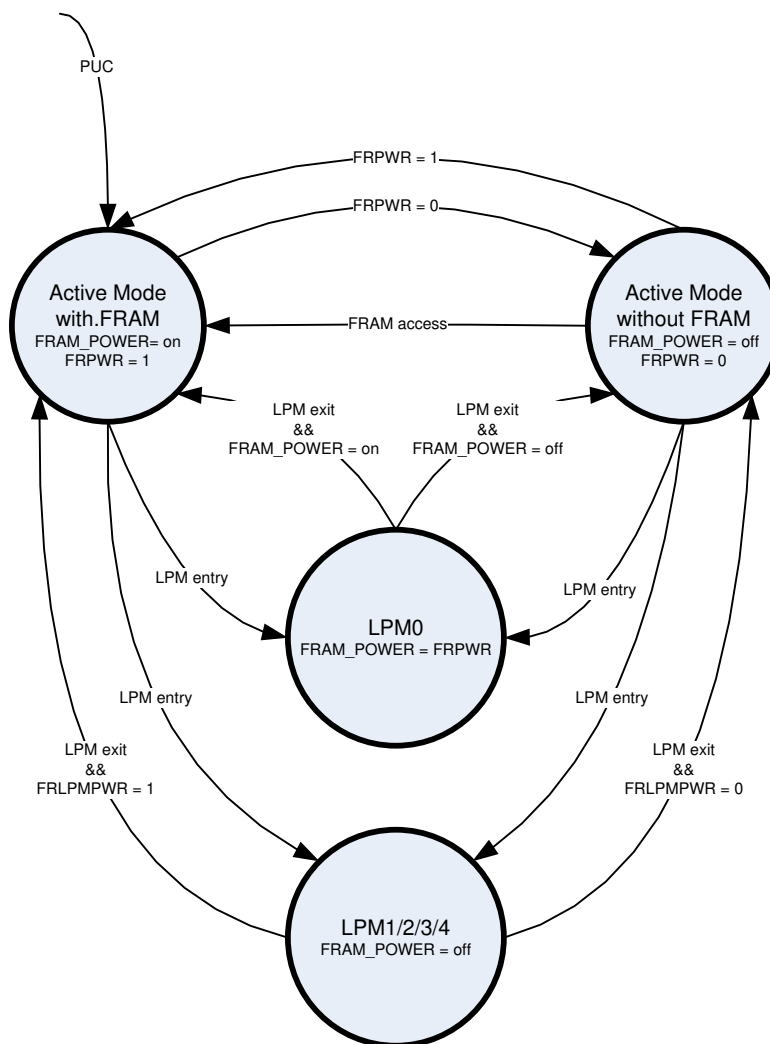


Figure 7-2. FRAM Power Control Diagram

7.9 FRAM Cache

The FRAM controller implements a read cache to provide a speed benefit when running the CPU at higher speeds than the FRAM supports without wait states. The cache implemented is a 2-way associative cache with 4 cache lines of 64 bit size. Memory read accesses on consecutive addresses can be executed without wait states when they are within the same cache line.

7.10 FRCTL Registers

The FRCTL registers and their address offsets are listed in [Table 7-1](#) . The base address of the FRCTL module can be found in the device-specific data sheet.

The password defined in the FRCTL0 register controls access to all FRCTL registers. When the correct password is written, write access to the registers is enabled. The write access is disabled by writing a wrong password in byte mode to the FRCTL upper byte. Word accesses to FRCTL with a wrong password triggers a PUC. A write access to a register other than FRCTL while write access is not enabled causes a PUC.

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 7-1. FRCTL Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	FRCTL0	FRAM Controller Control 0	Read/write	Word	9600h	Section 7.10.1
00h	FRCTL0_L		Read/Write	Byte	00h	
01h	FRCTL0_H		Read/Write	Byte	96h	
04h	GCCTL0	General Control 0	Read/write	Word	0006h	Section 7.10.2
04h	GCCTL0_L		Read/Write	Byte	06h	
05h	GCCTL0_H		Read/Write	Byte	00h	
06h	GCCTL1	General Control 1	Read/write	Word	0000h	Section 7.10.3
06h	GCCTL1_L		Read/Write	Byte	00h	
07h	GCCTL1_H		Read/Write	Byte	00h	

7.10.1 FRCTL0 Register

FRAM Controller Control Register 0

Figure 7-3. FRCTL0 Register

15	14	13	12	11	10	9	8
FRCTLPW							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
Reserved	NWAITS			Reserved			
r-0	rw-[0]	rw-[0]	rw-[0]	r-0	r-0	r-0	r-0

Table 7-2. FRCTL0 Register Description

Bit	Field	Type	Reset	Description
15-8	FRCTLPW	RW	96h	FRCTLPW password. Always reads as 96h. To enable write access to the FRCTL registers, write A5h. A word write of any other value causes a PUC. After a correct password is written and register access is enabled, write a wrong password in byte mode to disable the access. In this case, no PUC is generated.
7	Reserved	R	0h	Reserved. Always reads as 0.
6-4	NWAITS	RW	0h	Wait state control. Specifies number of wait states (0 to 7) required for an FRAM access (cache miss). 0 implies no wait states.
3	Reserved	R	0h	Reserved. Must be written as 0.
2-0	Reserved	R	0h	Reserved. Always reads as 0.

7.10.2 GCCTL0 Register

General Control Register 0

Figure 7-4. GCCTL0 Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
UBDRSTEN	UBDIE	CBDIE	Reserved	Reserved	FRPWR	FRLPMPWR	Reserved
rw-[0]	rw-[0]	rw-[0]	r-0	rw-0	rw-1	rw-1	r-0

Table 7-3. GCCTL0 Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved. Always reads as 0.
7	UBDRSTEN	RW	0h	Enable power up clear (PUC) reset if FRAM uncorrectable bit error detected. The bits UBDRSTEN and UBDIE are mutual exclusive and are not allowed to be set simultaneously. Only one error handling can be selected at one time. 0b = PUC not initiated on uncorrectable bit detection flag. 1b = PUC initiated on uncorrectable bit detection flag. Generates vector in SYSRSTIV.
6	UBDIE	RW	0h	Enable NMI event if uncorrectable bit error detected. The bits UBDRSTEN and UBDIE are mutual exclusive and are not allowed to be set simultaneously. Only one error handling can be selected at one time. 0b = Uncorrectable bit detection interrupt disabled. 1b = Uncorrectable bit detection interrupt enabled. Generates vector in SYSSNIV.
5	CBDIE	RW	0h	Enable NMI event if correctable bit error detected. 0b = Correctable bit detection interrupt disabled. 1b = Correctable bit detection interrupt enabled. Generates vector in SYSSNIV.
4	Reserved	R	0h	Reserved. Always reads as 0.
3	Reserved	RW	0h	Reserved. Must be written as 0.
2	FRPWR	RW	1h	FRAM power control. Writing to the register enables or disables the FRAM power supply. The read of the register returns the actual state of the FRAM power supply, also reflecting a possible delay after enabling the power supply. FRPWR = 1 indicates that the FRAM power is up and ready. 0b = FRAM power supply disabled 1b = FRAM power supply enabled
1	FRLPMPWR	RW	1h	Enables FRAM auto power up after LPM 0b = FRAM startup is delayed to the first FRAM access after LPM exit 1b = FRAM is powered up instantly with LPM exit.
0	Reserved	R	0h	Reserved. Always reads as 0.

7.10.3 GCCTL1 Register

General Control Register 1

Figure 7-5. GCCTL1 Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved				ACCTEIFG	UBDIFG	CBDIFG	Reserved
r-0	r-0	r-0	r-0	rw-0	rw-[0]	rw-[0]	r-0

Table 7-4. GCCTL1 Register Description

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0h	Reserved. Always reads as 0.
3	ACCTEIFG	RW	0h	Access time error flag. This flag is set and a reset PUC is generated if a wrong setting for NWAITS is set and the FRAM access time is violated. This bit is cleared by software or by reading the system reset vector word SYSRSTIV if it is the highest pending flag. This bit is write 0 only, write 1 has no effect. Note: The ACCTEIFG bit may be set in debug mode when the system frequency is configured to be greater than 8 MHz, regardless of the wait states (NWAITS). In the case, it is not an FRAM access violation. The ACCTEIFG bit does not trigger a PUC or change the SYSRSTIV register value. The ACCTEIFG bit is cleared only by writing 0. It is recommended to use SYSRESTIV register to check FRAM access violation error to avoid confusion.
2	UBDIFG	RW	0h	FRAM uncorrectable bit error flag. This interrupt flag is set if an uncorrectable bit error has been detected in the FRAM memory error detection logic. This bit is cleared by software or by reading the system NMI vector word SYSSNIV if it is the highest pending interrupt flag. This bit is write 0 only and write 1 has no effect. 0b = No interrupt pending 1b = Interrupt pending. Can be cleared by user or by reading SYSSNIV.
1	CBDIFG	RW	0h	FRAM correctable bit error flag. This interrupt flag is set if a correctable bit error has been detected and corrected in the FRAM memory error detection logic. This bit is cleared by software or by reading the system NMI vector word SYSSNIV if it is the highest pending interrupt flag. This bit is write 0 only and write 1 has no effect. 0b = No interrupt pending 1b = Interrupt pending. Can be cleared by user or by reading SYSSNIV
0	Reserved	R	0h	Reserved. Always reads as 0.

FRAM Controller A (FRCTL_A)

This chapter describes the operation of the FRAM controller A (FRCTL_A) . The FRCTL_A and FRCTL are almost identical in terms of the features that they support. For a summary of the differences between the two modules, see the *FRAM Controller Overview* chapter.

Topic	Page
8.1 FRAM Controller A (FRCTL_A) Introduction.....	299
8.2 FRAM Controller A (FRCTL_A) Operation	299
8.3 FRAM ECC.....	302
8.4 FRAM Power Control	302
8.5 FRAM Cache	303
8.6 FRCTL_A Registers	304

8.1 FRAM Controller A (FRCTL_A) Introduction

The FRAM Controller A includes the following features :

- Byte (8 bit) or word (16 bit) write access
- Automatic and programmable wait state control with independent wait state settings for access and cycle times
- Timing violation detection to ensure proper interrupt handling with incorrect wait state setting
- Error correction code with bit error correction, extended bit error detection, and flag indicators
- Cache for energy-efficient read
- Power control for disabling FRAM when it is not in use, including automatic wake up

For important software design information regarding FRAM, including but not limited to partitioning the memory layout according to application-specific code, constant, and data space requirements, the use of FRAM to optimize application energy consumption, and the use of the memory protection unit (MPU) to maximize application robustness by protecting the program code against unintended write accesses, see [MSP430™ FRAM Technology – How To and Best Practices](#).

Figure 8-1 shows the block diagram of the FRCTL_A.

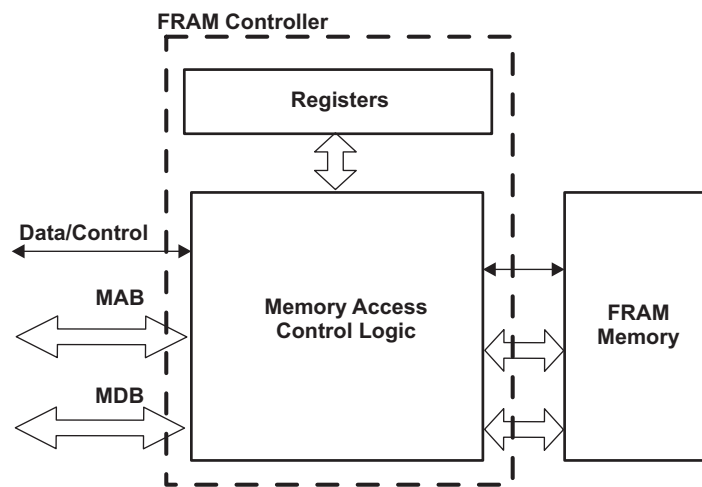


Figure 8-1. FRCTL_A Block Diagram

8.2 FRAM Controller A (FRCTL_A) Operation

FRAM is a nonvolatile memory that eliminates the slow writing barrier of flash memory. The read and write operations of FRAM is just like the way that the standard SRAM works. The FRAM features SRAM-like operation with nonvolatility.

8.2.1 FRCTL_A Error Detection

The FRAM module has a built-in error correction code (ECC) block that can correct bit errors and detect multiple bit errors. Two flags, the CBDIFG and UBDIFG bits, are used to report the status of errors.

The correctable bit detection interrupt flag (CBDIFG) is set when a correctable bit error is detected. In this case, the error generates a system NMI (SYSNMI) if the correctable bit detection interrupt enable bit (CBDIE) is set.

The uncorrectable bit detection interrupt flag (UBDIFG) is set when a multiple bit error, which is not correctable, is detected. In this case, cache is flushed and either a system NMI (SYSNMI), if the uncorrectable bit detection interrupt enable bit (UBDIE) is set, or a power-up-clear (PUC) reset, if the uncorrectable bit detection reset enable bit (UBDRSTEN) is set, can be generated.

The UBDRSTEN bit and the UBDIE bit are mutually exclusive. The UBDRSTEN bit has a higher priority—if both bits are set, the UBDIE bit is ignored and the UBDRSTEN bit remains active.

8.2.2 Programming FRAM Memory Devices

There are three options for programming an MSP430 FRAM device. All options support in-system programming.

- Program with JTAG or the Spy-Bi-Wire interface
- Program with the BSL
- Program with a custom solution

8.2.2.1 Programming FRAM Memory With JTAG or Spy-Bi-Wire

Devices can be programmed through the JTAG port or the Spy-Bi-Wire port. The JTAG interface requires access to TDI, TDO, TMS, TCK, TEST, ground, and optionally V_{CC} and $\overline{RST}/NMI$. Spy-Bi-Wire interface requires access to TEST, $\overline{RST}/NMI$, ground and optionally V_{CC} . For more details, see the [MSP430 Programming With the JTAG Interface](#).

8.2.2.2 Programming FRAM Memory With the Bootstrap Loader (BSL)

Every device contains a BSL stored in ROM. The BSL enables users to read or program the FRAM or RAM using a UART serial interface. Access to the FRAM through the BSL is protected by a 256-bit user-defined password. For more details, see the [MSP430 Programming With the Bootloader \(BSL\)](#).

8.2.2.3 Programming FRAM Memory With a Custom Solution

The ability of the CPU to write to its own FRAM allows for in-system and external custom programming solutions. The user can choose to provide data to the device through any means available (for example, UART or SPI). User-developed software can receive the data and program the FRAM. Because this type of solution is developed by the user, it can be completely customized to fit the application needs for programming or updating the FRAM.

8.2.3 Access Control

8.2.3.1 Write Protection

The WPROT bit can be used to protect the contents of FRAM from being unintentionally modified. When the WPROT is set, reading is allowed, but no writing to FRAM memory is allowed. If a write access is attempted with WPROT = 1, the WPIFG (write protection flag) bit is set. In this case, the error generates a system NMI (SYSNMI) if the WPIE (write protection interrupt enable) bit is set. Note that writing-to-FRAM is also blocked when the ACCTEIFG bit is set due to a timing violation. The WPIFG bit is set when a write access is attempted with ACCTEIFG = 1.

The WPROT bit protects the entire FRAM from unintended writes regardless of MPU configurations, so this bit should be used as temporary protection. To protect a portion of the FRAM permanently, use the MPU module (see [Chapter 9, Memory Protection Unit](#)). Write protection is disabled after BOR (WPROT = 0).

8.2.3.2 Two Wait State Modes

FRAM memory has limited access speed (see the device data sheet for details), but that does not limit the speed of CPU and DMA in the device. When the running speed of the CPU and DMA exceeds the FRAM access speed, a wait state control mechanism is implemented. The FRAM controller A (FRCTL_A) supports two wait state modes, user wait state mode and automatic wait state mode.

8.2.3.3 User Wait State Mode

User wait state mode and automatic wait state mode are mutually exclusive. User wait state mode is automatically enabled after device reset (BOR), but the wait state mode can be switched to automatic wait state mode by setting the AUTO bit. The FRAM access speed can be maximized in user wait state mode by writing an optimized wait state number to NWAITS[3:0]. However, incorrect wait state numbers may cause a timing violation error. Thus, the application must write a proper wait state to NWAITS[3:0] before accessing FRAM. See [Table 8-1](#) for optimized wait states with different system frequencies.

8.2.3.4 Timing Violation Detection

In user wait state mode (AUTO = 0), if NWAITS[3:0] bit has been configured with an incorrect wait state, a timing violation can occur when accessing FRAM. Upon detecting a timing violation, the FRAM controller responds to the timing violation event with three actions:

- Sets the access timer error flag (ACCTEIFG)
- Ignores the NWAITS[3:0] bits and internally applies the maximum wait state (15) (the NWAITS[3:0] bits are not changed)
- Flushes the cache
- Disables write access to FRAM regardless of the WPROT bit (the WPROT bit is not changed)

The FRAM controller A (FRCTL_A) keeps the maximum wait state and blocks write access if the ACCTEIFG bit is set in order to avoid further timing violations. It is recommended to configure the NWAITS[3:0] bits based on the table shown in [Table 8-1](#) and complete any necessary actions prior to clearing the ACCTEIFG bit. When the ACCTEIFG bit is cleared, the FRAM controller A (FRCTL_A) takes the value written to NWAITS[3:0] bits as wait state and enables write access to the FRAM if the WPROT bit is cleared. The timing violation (ACCTEIFG) generates a system NMI (SYSNMI) if the access time error interrupt enable (ACCTEIE) bit is set.

8.2.3.5 Automatic Wait State Mode

Automatic wait state mode is enabled when the AUTO bit is set. In this mode, the FRAM controller A (FRCTL_A) takes a control of choosing a wait state. So, it is not required for user to configure NWAITS[3:0] bits. The value written to NWAITS[3:0] has no influence in this mode. In order to determine the wait state automatically, the FRAM controller A (FRCTL_A) adds a delay so that no maximum FRAM access speed is reached and no timing violation is guaranteed. See [Table 8-1](#) for wait state numbers in automatic mode with different system frequencies.

Table 8-1. FRAM memory Access Speed

System Bus Frequency	Required Wait States		FRAM Access Speed (Without Cache Hit)	
	User Mode	Automatic Mode	User Mode	Automatic Mode
4 MHz	0	3	4 MHz	1 MHz
8 MHz	0	3	8 MHz	2 MHz
10 MHz	1	3	5 MHz	2.5 MHz
12 MHz	1	3	6 MHz	3 MHz
14 MHz	1	3	7 MHz	3.5 MHz
16 MHz	1	3	8 MHz	4 MHz
24 MHz	2	3	8 MHz	6 MHz
32 MHz	3	4	8 MHz	6.4 MHz

8.2.3.6 Wait State and Cache Hit

The FRAM controller A (FRCTL_A) has a cache that contains four 64-bit lines. The cache keeps up to 32 bytes (4 × 64 bit) from the latest accesses to FRAM. When a read is requested, the FRAM controller A (FRCTL_A) first determines if the requested data is in the cache. If a match is found (a cache hit), then the data is read from the cache and no physical FRAM memory access occurs. In this case, no wait state is required and the data is accessed at the full system bus speed. If no match is found (no cache hit), then the data is read from FRAM memory and the new data replaces one of the four 64-bit lines in the cache.

8.2.3.7 Wait State in Debug Mode

When the device is in debug mode, no wait state is applied. The NWAITS[3:0] has no influence in debug mod. In debug mode (for example, during JTAG access to FRAM), the device system clock is controlled externally and can be stopped at any time, thus FRAM access needs to be completed without wait state cycles. The running speed of the CPU and DMA never exceeds the maximum FRAM access speed limit in debug mode.

8.3 FRAM ECC

FRAM ECC supports bit error correction and uncorrectable bit error detection.

Correctable errors are generally single-bit errors that are detected and corrected by the hardware, so they do not result in data corruption or system failure. The CBDIFG FRAM correctable bit error flag is set if a correctable bit error has been detected and corrected. CBDIE can be used to enable an NMI event.

Uncorrectable bit errors are always multiple-bit errors, and they indicate memory corruption. The UBDIFG FRAM uncorrectable bit error flag is set if an uncorrectable bit error has been detected in the FRAM error detection logic. UBDRSTEN can be used to enable a power up clear (PUC) reset, or UBDIE can be used to enable an NMI event. UBDRSTEN and UBDIE are mutually exclusive and are not allowed to be set simultaneously.

For more information, refer to the [MSP430 FRAM Quality and Reliability](#) application report.

8.4 FRAM Power Control

To achieve maximum power efficiency of FRAM operations, the FRAM controller A (FRCTL_A) supports a power control mode. There are three inputs that influence the power state of FRAM: the FRPWR bit, FRAM access (read or write), and the device power mode.

[Table 8-2](#) summarizes how FRAM power modes are controlled by the source. [Figure 8-2](#) shows the flow of FRAM power mode transitions.

While the device is in active mode (AM), FRAM power is controlled by the FRPWR bit and FRAM access. When the FRPWR is set, FRAM goes to ACTIVE mode regardless of FRAM access. When the FRPWR is cleared by the CPU and there is no access to FRAM, the FRAM goes into INACTIVE mode so that the FRAM does not consume power.

INACTIVE mode can be used if FRAM access is not required for a significant amount of time. For example, short tasks can be executed from RAM, so while CPU runs from RAM, FRAM can be powered off. When the FRAM is in the INACTIVE mode, wake-up is automatic. An access to FRAM (read or write) wakes up the FRAM before performing the access. In this case, the FRPWR bit is set automatically by the FRAM controller A (FRCTL_A).

Care must be taken when using the FRPWR bit. When the FRAM is powered off, there is a wake-up time delay before the FRAM can be accessed again. The delay should be considered to avoid affecting system performance. See the device data sheet for the delay time.

When the device enters LPM0, LPM1, LPM2, LPM3, or LPM4, the FRAM also enters INACTIVE mode regardless of FRPWR bit status, however FRPWR bit determines the power status when the device wakes up from a LPM. When the device wakes up from a low-power mode to active mode (AM), FRAM Controller A (FRCTL_A) immediately wakes up FRAM memory if the FRPWR is set. If the FRPWR bit is cleared, FRAM memory remains in INACTIVE mode until an access to FRAM occurs (read or write). The latter case can be used to reduce the device power consumption if the device wakes up only for a short amount of time, and the task during device active mode can be executed from RAM with no need to access FRAM memory. See [Table 8-2](#) and [Figure 8-2](#) for details.

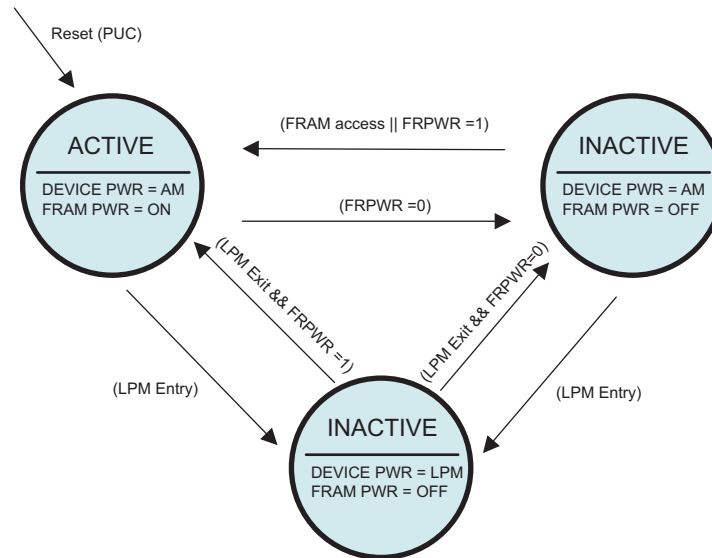
Table 8-2. FRAM Power Mode Transition

Device Power Mode	Power Control Source		FRAM Power State (Start)	FRAM Power State (End)
	FRPWR Bit	FRAM Access		
AM	1 (after PUC)	Don't care	ACTIVE	ACTIVE
AM	1 → 0	No	ACTIVE	INACTIVE
AM	0	No → Yes	INACTIVE	ACTIVE (FRPWR bit is set automatically)
AM	0 → 1	No	INACTIVE	ACTIVE
AM → LPM0, LPM1, LPM2, LPM3, or LPM4	Don't care	No	Don't care	INACTIVE
LPM0	Don't care	No → Yes	Don't care	ACTIVE (FRPWR bit is set automatically)

Table 8-2. FRAM Power Mode Transition (continued)

Device Power Mode	Power Control Source		FRAM Power State (Start)	FRAM Power State (End)
	FRPWR Bit	FRAM Access		
LPM0, LPM1, LPM2, LPM3, or LPM4 → AM	1	No	INACTIVE	ACTIVE
LPM0, LPM1, LPM2, LPM3, or LPM4 → AM	0	No	INACTIVE	INACTIVE

Figure 8-2 shows the flow of the FRAM power transitions.


Figure 8-2. FRAM Power Control Diagram

8.5 FRAM Cache

The FRAM controller A (FRCTL_A) has a cache that contains four 64-bit lines. One of the 64-bit lines is preloaded during one access cycle and the cache can keep up to 32 bytes (4 × 64 bit) from the latest accesses to FRAM memory. When an FRAM read is requested, the FRAM controller A (FRCTL_A) first checks cache. If the requested data is found in cache (a cache hit), then the data is read from the cache and no physical FRAM access occurs. In this case, no wait state is required and the data is accessed at the full system bus speed.

8.6 FRCTL_A Registers

[Table 8-3](#) lists the memory-mapped registers for the FRCTL_A. All register offset addresses not listed in [Table 8-3](#) should be considered as reserved locations and the register contents should not be modified.

The password defined in the FRCTL0 register controls access to all FRAM Controller A registers. When the correct password is written, write access to the registers is enabled. The write access is disabled by writing a wrong password in byte mode to FRCTL0 upper byte. Word accesses to FRCTL0 with a wrong password triggers a PUC. A write access to a register other than FRCTL0 while write access is not enabled causes a PUC.

Note 1: The correct password (A5h) is written to the FRCTLPW bits by the bootcode during the device boot-up process; therefore, the FRCTL0 (low byte), GCCTL0, and GCCTL1 registers are unlocked after the device is powered up or reset (BOR) or after LPMx.5 wakeup.

Note 2: All registers have word or byte register access. For a generic register ANYREG, the suffix "_L" (ANYREG_L) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (ANYREG_H) refers to the upper byte of the register (bits 8 through 15).

Table 8-3. FRCTL_A Registers

Offset	Acronym	Register Name	Type	Reset	Section
0h	FRCTL0	FRAM Controller A Control Register 0	Read-Write	9600h	Section 8.6.1
4h	GCCTL0	General Control Register 0	Read-Write	4h	Section 8.6.2
6h	GCCTL1	General Control Register 1	Read-Write	0h	Section 8.6.3

8.6.1 FRCTL0 Register (Offset = 0h) [reset = 9600h]

FRCTL0 is shown in [Figure 8-3](#) and described in [Table 8-4](#).

Return to [Summary Table](#).

FRAM Controller A Control Register 0

Figure 8-3. FRCTL0 Register

15	14	13	12	11	10	9	8
FRCTLPW							
R/W-96h							
7	6	5	4	3	2	1	0
NWAITS				AUTO	Reserved		WPROT
R/W-0h				R/W-0h	R-0h		R/W-0h

Table 8-4. FRCTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	FRCTLPW	R/W	96h	FRCTLPW password. Always read as 96h. Note: The correct password (A5h) is written to the FRCTLPW bits by the bootcode during the device boot-up process; therefore, the FRCTL0 (low byte), GCCTL0, and GCCTL1 registers are unlocked after the device is powered up or reset (BOR) or after LPMx.5 wakeup. 96h (R) = FRPW : Read value while locked A5h (W) = FWPW : Must be written as A5h or a PUC is generated on word write. After a correct password is written and register access is enabled, a wrong password write in byte mode disables the access and no PUC is generated.
7-4	NWAITS	R/W	0h	Wait state generator access time control when AUTO =0. Each wait state adds a N integer multiple increase of the IFCLK period where N = 0 through 15. N = 0 implies no wait states. When a timing violation is detected, the Access Time Error Flag (ACCTEIFG) is set and the maximum wait state, 15, is automatically applied to the NWAITS[3:0] to avoid further timing violation. While the ACCTEIFG bit is set, the NWAITS[3:0] cannot be overwritten and writing to the FRAM memory is prohibited regardless of the WPROT bit. Only reading is allowed. The ACCTEIFG bit must be cleared prior to applying a new value to NWAITS[3:0] or writing access to the FRAM memory. The timing violation (ACCTEIFG) can generate a system NMI (SYSNMI) if the Access Time Error Interrupt Enable (ACCTEIE) bit is set. When a timing violation occurs for reading, the data from FRAM memory could be incorrect, thus proper error handling is recommended before proceeding. Reset type: BOR 0h (R/W) = FRAM wait states: 0 1h (R/W) = FRAM wait states: 1 2h (R/W) = FRAM wait states: 2 3h (R/W) = FRAM wait states: 3 4h (R/W) = FRAM wait states: 4 5h (R/W) = FRAM wait states: 5 6h (R/W) = FRAM wait states: 6 7h (R/W) = FRAM wait states: 7 8h (R/W) = FRAM wait states: 8 9h (R/W) = FRAM wait states: 9 Ah (R/W) = FRAM wait states: 10 Bh (R/W) = FRAM wait states: 11 Ch (R/W) = FRAM wait states: 12 Dh (R/W) = FRAM wait states: 13 Eh (R/W) = FRAM wait states: 14 Fh (R/W) = FRAM wait states: 15

Table 8-4. FRCTL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	AUTO	R/W	0h	Enable automatic Wait State Mode. Reset type: BOR 0h (R/W) = AUTO_0 : User Wait State Mode. The NWAITS[3:0] is used for the FRAM wait state. 1h (R/W) = AUTO_1 : Auto mode. The NWAITS[3:0] is ignored. Wait states are generated automatically by the internal FRAM controller state machine.
2-1	Reserved	R	0h	Reserved. Always read 0. Reset type: PUC
0	WPROT	R/W	0h	Write Protection Enable. This bit is set after BOR. This bit must be cleared before accessing FRAM for write. This bit does not block read operation. Note that the WPROT bit protects the entire FRAM memory from unintended write, so it should be used as temporary protection. If it is desired to protect a portion of the FRAM memory permanently, it should be done via MPU segments. See the MPU module for details. Reset type: BOR 0h (R/W) = WPROT_0 : Disable Write Protection. Write to FRAM memory is allowed. 1h (R/W) = WPROT_1 : Enable Write Protection. Write to FRAM memory is not allowed. If a write access is attempted, the WPIFG (Write Protection Flag) bit will be set.

8.6.2 GCCTL0 Register (Offset = 4h) [reset = 4h]

GCCTL0 is shown in [Figure 8-4](#) and described in [Table 8-5](#).

Return to [Summary Table](#).

General Control Register 0

Figure 8-4. GCCTL0 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							
7	6	5	4	3	2	1	0
UBDRSTEN	UBDIE	CBDIE	WPIE	ACCTEIE	FRPWR	Reserved	
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-1h	R-0h	

Table 8-5. GCCTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved. Always read 0. Reset type: PUC
7	UBDRSTEN	R/W	0h	Enable Power Up Clear (PUC) reset for the uncorrectable bit error detection flag (UBDIFG). The UBDRSTEN and UBDIE must be mutual exclusive. The FRAM controller does not allow the status of both bits are being set. Writing 1 to one of the bits will automatically clear the other bit. Reset type: BOR 0h (R/W) = UBDRSTEN_0 : PUC not initiated on uncorrectable bit error detection flag. 1h (R/W) = UBDRSTEN_1 : PUC initiated on uncorrectable bit error detection flag. Generates vector in SYSRSTIV. Clear the UBDIE bit.
6	UBDIE	R/W	0h	Enable NMI event for the uncorrectable bit error detection flag (UBDIFG). The UBDRSTEN and UBDIE must be mutual exclusive. The FRAM controller does not allow the status of both bits are being set. Writing 1 to one of the bits will automatically clear the other bit. Reset type: BOR 0h (R/W) = UBDIE_0 : Disable NMI for the uncorrectable bit error detection flag (UBDIFG). 1h (R/W) = UBDIE_1 : Enable NMI for the uncorrectable bit error detection flag (UBDIFG). Generates vector in SYSSNIV. Clear the UBDRSTEN bit.
5	CBDIE	R/W	0h	Enable NMI event for the correctable bit error detection flag (CBDIFG). Reset type: BOR 0h (R/W) = CBDIE_0 : Disable NMI for the correctable bit error detection flag (CBDIFG). 1h (R/W) = CBDIE_1 : Disable NMI for the correctable bit error detection flag (CBDIFG). Generates vector in SYSSNIV.
4	WPIE	R/W	0h	Enable NMI event for the Write Protection Detection flag (WPIFG). Reset type: BOR 0h (R/W) = WPIE_0 : Disable NMI for the Write Protection Detection flag (WPIFG). 1h (R/W) = WPIE_1 : Enable NMI for the Write Protection Detection flag (WPIFG). Generates vector in SYSSNIV.

Table 8-5. GCCTL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	ACCTEIE	R/W	0h	Enable NMI event for the Access time error flag (ACCTEIFG). Reset type: BOR 0h (R/W) = ACCTEIE_0 : Disable NMI for the Access time error flag (ACCTEIFG). 1h (R/W) = ACCTEIE_1 : Enable NMI for the Access time error flag (ACCTEIFG). Generates vector in SYSSNIV.
2	FRPWR	R/W	1h	FRAM Memory Power Control Request While the device is in AM (Active mode), the FRAM memory power is controlled by the FRPWR bit and FRAM access. When the FRPWR is set, the FRAM memory is in ACTIVE mode. When the FRPWR is cleared by CPU, the FRAM memory goes into INACTIVE mode so that the FRAM memory does not consumes power. The INACTIVE mode can be used if no FRAM access is required for a significant amount of time. Once the FRAM memory is in the INACTIVE mode, wake-up is automatic. An access to FRAM (read or write) will wake up the FRAM memory before performing the access. In this case, the FRPWR bit is set automatically by the FRAM controller. When the device enters LPM0/1/2/3/4 modes, the FRAM memory also enters INACTIVE mode regardless of the FRPWR bit status. When the device wakes up from LPM0/1/2/3/4, the FRAM memory will be immediately powered up (ACTIVE mode) if the FRPWR is set, but if the FRPWR bit is cleared, the FRAM memory will remain in INACTIVE mode until the FRAM memory is actually accessed (read or write). The latter case can be used to save the device power consumption in case the device wakes up only for a short amount of time, and the task during the wake-up can be executed from RAM, so no need of FRAM access. Reset type: PUC 0h (R/W) = FRPWR_0 : Enable INACTIVE mode. 1h (R/W) = FRPWR_1 : Enable ACTIVE mode.
1-0	Reserved	R	0h	Reserved. Always read 0. Reset type: PUC

8.6.3 GCCTL1 Register (Offset = 6h) [reset = 0h]

GCCTL1 is shown in [Figure 8-5](#) and described in [Table 8-6](#).

Return to [Summary Table](#).

General Control Register 1

Figure 8-5. GCCTL1 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							
7	6	5	4	3	2	1	0
Reserved			WPIFG	ACCTEIFG	UBDIFG	CBDIFG	Reserved
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

Table 8-6. GCCTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Always read 0. Reset type: PUC
4	WPIFG	R/W	0h	Write Protection Detection flag. This flag is set when a write access is attempted while the WPROT bit is set. This bit can generate a system NMI if the WPIE bit is set (see the GCCTL0 register). This bit can be cleared by writing 0 directly or by reading the system reset vector word SYSRSTIV if it is the highest pending interrupt flag. This bit is write 0 only, write 1 has no effect. Reset type: BOR 0h (R/W) = WPIFG_0 : No interrupt pending. 1h (R/W) = Interrupt pending. Can be cleared by writing 0 or by reading SYSSNIV when it is the highest pending interrupt.
3	ACCTEIFG	R/W	0h	Access time error flag. This flag is set when a timing violation is detected, which indicates that NWAITS[3:0] is improperly configured. When a timing violation is detected, the maximum wait state will be automatically applied to the NWAITS[3:0] to avoid further timing violation. While the ACCTEIFG bit is set, the NWAITS[3:0] cannot be overwritten and the FRAM memory write access is prohibited regardless of the WPROT bit. The ACCTEIFG bit must be cleared prior to applying a new value to NWAITS[3:0] or writing access to the FRAM memory. The timing violation (ACCTEIFG) can generate a system NMI (SYSNMI) if the Access Time Error Interrupt Enable (ACCTEIE) bit is set. This bit can be cleared by writing 0 directly or by reading the system reset vector word SYSRSTIV if it is the highest pending interrupt flag. This bit is write 0 only, write 1 has no effect. Reset type: BOR 0h (R/W) = ACCTEIFG_0 : No interrupt pending. 1h (R/W) = ACCTEIFG_1 : Interrupt pending. Can be cleared by writing 0 or by reading SYSSNIV when it is the highest pending interrupt.
2	UBDIFG	R/W	0h	FRAM uncorrectable bit error detection flag. This flag is set when an uncorrectable bit error is detected in the FRAM memory error detection logic. This bit can generate either a system NMI or a system reset (PUC). If the UBDIE bit is set, then this bit generates a system NMI, if the UBDRSTEN bit is set, then this bit generates a system reset (PUC) - see the GCCTL0 register. This bit can be cleared by writing 0 directly or by reading the system NMI vector word SYSSNIV when it is the highest pending interrupt flag. This bit is write 0 only and write 1 has no effect. Reset type: BOR 0h (R/W) = UBDIFG_0 : No interrupt pending. 1h (R/W) = UBDIFG_1 : Interrupt pending. Can be cleared by writing 0 or by reading SYSSNIV when it is the highest pending interrupt.

Table 8-6. GCCTL1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	CBDIFG	R/W	0h	FRAM correctable bit error detection flag. This flag is set when a correctable bit error is detected and corrected in the FRAM memory error detection logic. This bit can generate a system NMI if the CBDIE bit is set (see the GCCTL0 register). This bit can be cleared by software or by reading the system NMI vector word SYSSNIV if it is the highest pending interrupt flag. This bit is write 0 only and write 1 has no effect. Reset type: BOR 0h (R/W) = CBDIFG_0 : No interrupt is pending 1h (R/W) = CBDIFG_1 : Interrupt pending. Can be cleared by writing 0 or by reading SYSSNIV if it is the highest pending interrupt.
0	Reserved	R	0h	Reserved. Always read 0. Reset type: PUC

Memory Protection Unit (MPU)

This chapter describes the operation of the Memory Protection Unit (MPU). The MPU is family specific.

Topic	Page
9.1 Memory Protection Unit (MPU) Introduction.....	312
9.2 MPU Segments.....	313
9.3 MPU Access Management Settings.....	317
9.4 MPU Violations.....	318
9.5 MPU Lock	318
9.6 How to Enable MPU and IPE Segments	318
9.7 MPU Registers	321

9.1 Memory Protection Unit (MPU) Introduction

The MPU protects against accidental writes to designated read-only memory segments or execution of code from a constant memory segment. Clearing the MPUENA bit disables the MPU, and the complete memory is accessible for read, write, and execute operations. After a BOR, the complete memory is accessible without restrictions to read, write, and execute operations.

The Memory Protection Unit features include:

- Configuration of main memory into three variable-sized segments
- Access rights for each segment can be set independently
- Fixed-size constant user information memory segment with selectable access rights
- Protection of MPU registers by password

NOTE: After BOR, no segmentation is initiated, and the main memory and information memory are accessible by read, write, and execute operations.

For important software design information regarding FRAM, including but not limited to partitioning the memory layout according to application-specific code, constant, and data space requirements, the use of FRAM to optimize application energy consumption, and the use of the memory protection unit (MPU) to maximize application robustness by protecting the program code against unintended write accesses, see [MSP430™ FRAM Technology – How To and Best Practices](#).

Figure 9-1 shows an overview of the Memory Protection Unit.

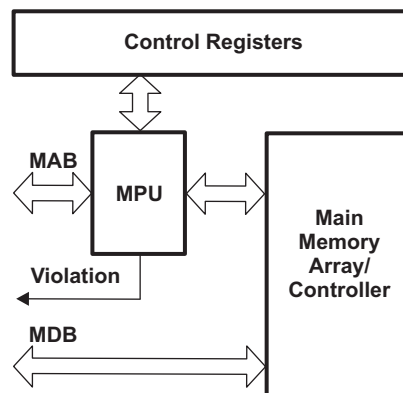


Figure 9-1. Memory Protection Unit Overview

9.2 MPU Segments

9.2.1 Main Memory Segments

The MPU can logically divide the main memory into three segments. The size of each segment is defined by setting the borders between adjacent segments. To configure three segments, a lower border (B1) and a higher border (B2) are positioned by control register bits MPUSEGB1[15:0] and MPUSEGB2[15:0], respectively, in the MPUSEGBx registers. The position of both borders is limited to the 16 most significant bits of the memory address space (20-bit). Therefore, the segment borders registers are equivalent to the memory address bus, shifted right by 4 bits (see [Figure 9-2](#)).

[Table 9-1](#) shows the minimum segment size. Depending on the total memory size, some of the border register bits must be written as zero. [Table 9-1](#) summarizes the user-selectable bits and fixed bits for different memory sizes (see the device-specific data sheet for total memory size). [Figure 9-3](#) and [Figure 9-4](#) show fixed bits of the segment register when memory size is 128KB and 256KB respectively.

The beginning of segment 1 is the lowest available address for the main memory as defined in the device-specific data sheet. The lower border (B1) defines the end of segment 1 and the beginning of segment 2. The higher border (B2) defines the end of segment 2 and beginning of segment 3. The end of segment 3 is the highest main memory address as defined in the device-specific data sheet. For example, devices with up to 64KB of FRAM, the highest memory address is 013FFFh. Segment 2 includes the address defined by the lower border (B1) but excludes the higher border (B2).

The address bus (MAB) is analyzed by the MPU using the 16 most significant bits along with the current border settings.

- If the significant address bits are lower than MPUSEGB1[15:0], segment 1 is selected.
- If the significant address bits are equal to or greater than MPUSEGB1[15:0] and less than MPUSEGB2[15:0], segment 2 is selected.
- If the significant address bits are equal to or greater than MPUSEGB2[15:0], segment 3 is selected.

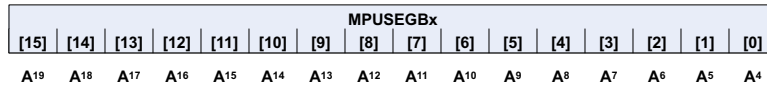


Figure 9-2. Segment Border Register

Table 9-1. Address Comparator Bit Selection

FRAM Size	Index of Used MSB Address Bus (n)	User-Selectable Border Register Bits	Fixed Border Register Bits (zero)	Segment Size (bytes)
32KB < size ≤ 128KB	17-bit	[13:6]	[15:14] and [5:0]	1k
128KB < size ≤ 256KB	18-bit	[14:6]	[15] and [5:0]	1k

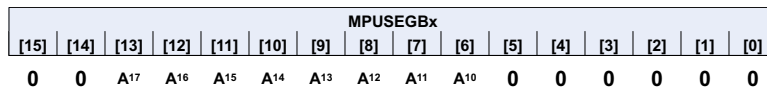


Figure 9-3. Example of Segment Border Register Fixed Bits When FRAM Size = 128KB

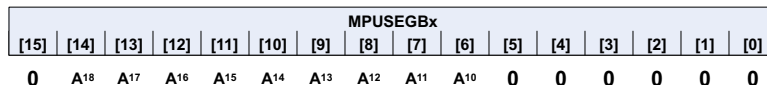


Figure 9-4. Example of Segment Border Register Fixed Bits When FRAM Size = 256KB

NOTE: The same result is calculated during MAB analysis of segment membership independent of whether the higher value is in MPUSEGB1[15:0] or MPUSEGB2[15:0]. If MPUSEGB1[15:0] = MPUSEGB2[15:0], Segment 2 is not available and the main memory only contains 2 segments.

Figure 9-5 shows an example segmentation of the main memory.

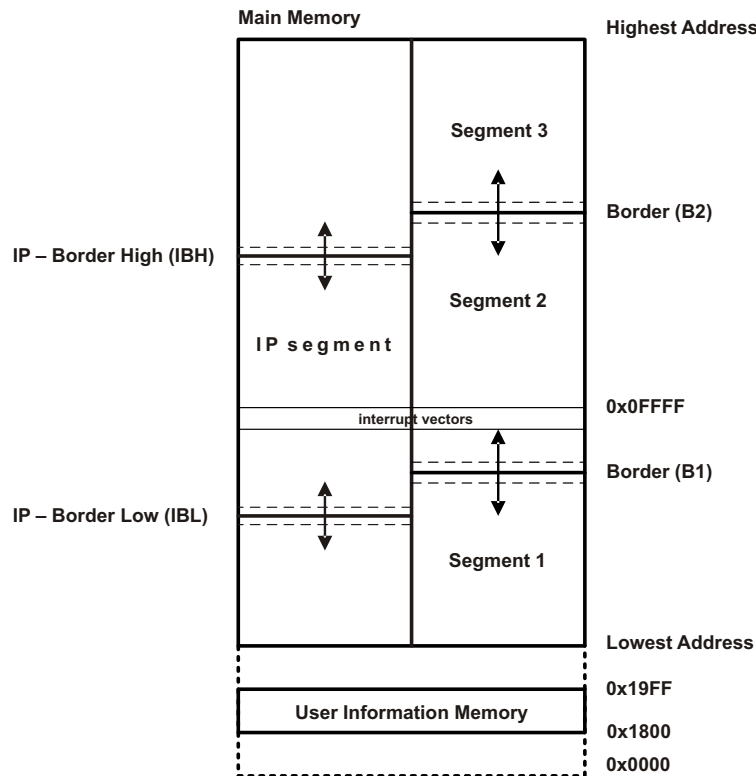


Figure 9-5. Segmentation of Main Memory

9.2.2 IP Encapsulation Segment

The MPU can protect an address range in the main memory from unconditional external access. The size of this segment is defined by setting the upper and lower borders of this segment. To configure the segments, a lower (IBL) border and a higher (IBH) border are positioned by control register bits MPUIPSEGB1[15:0] and MPUIPSEGB2[15:0], respectively, in the MPUIPSEGBx register. The position of both borders follows the same mechanism as described in for the main segments.

The beginning of the IP encapsulation segment (IPE-segment) (IBL) is defined by the lower value of either the MPUIPSEGB1[15:0] or MPUIPSEGB2[15:0] register. The end of the IPE-segment (IBH) is defined by the higher value of either the MPUIPSEGB1[15:0] or MPUIPSEGB2[15:0] register. All memory locations addressed by the 16 most significant bits of the address bus (MAB) equal to or greater than the lower border (IBL) and less than IBH are protected.

Only program code executed from the IPE-segment can access data stored in this segment. The access rights are evaluated with each code access. Each code access outside of the IP-protected area deactivates the data access into the IPE-segment. JTAG or DMA cannot access the IPE-segment. The interrupt vector table is always open for read and write accesses (for details see [Section 9.4.1](#)).

To execute code from the IPE-segment, branch into that segment or call functions stored in that segment. Interrupt service routines can be executed from the IPE-segment, too.

[Table 9-2](#) summarizes the possible combinations of code execution and memory access types and the resulting access rights.

An unauthorized access to the IPE-segment returns a value equivalent to the instruction "JMP \$" and triggers an interrupt. In addition, the generation of a PUC can be configured.

Table 9-2. IP Encapsulation Access Rights

$IBL \leq \text{Memory Address} < IBH$	$IBL \leq \text{Program Counter} < IBH$	JTAG or DMA Access	CPU Access
$0FF80h \leq \text{Memory Address} < 0FFFFh$	–	Read/Write	Read/Write
False	False	Yes	Yes
False	True	Yes	Yes
True	False	No	No
True	True	No	Yes

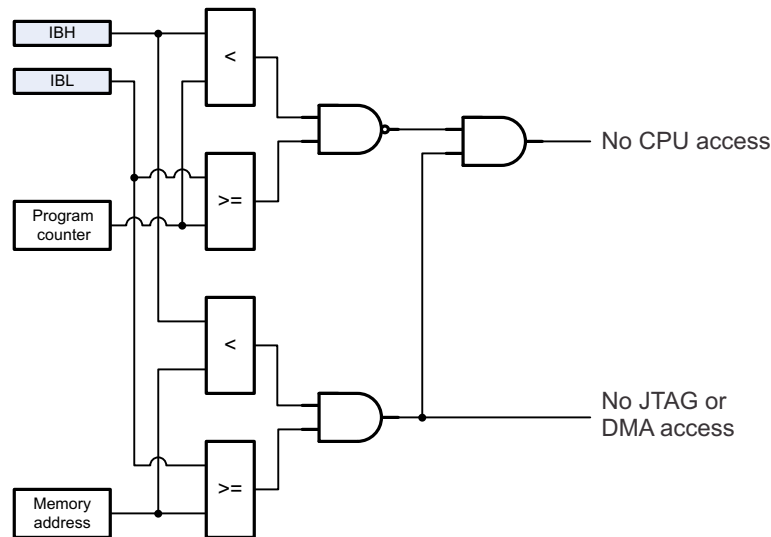


Figure 9-6. IP Encapsulation Access Rights Equivalent Schematic

NOTE: IP Encapsulation area access rights do not override MPU segment rights. The IP Encapsulation rights are evaluated and if the access is granted, access rights as describe in [Section 9.3](#) are applied.

NOTE: Code fetch from the first 8 bytes in IPE-segment does not enable data access.

The first 8 bytes within the IPE-segment do not enable data access within the IPE-segment if code is executed from that area. The start of an IPE-segment is reserved for a data structure describing the IPE-segment boundaries.

shows the segmentation of the main memory.

9.2.3 Segment Border Setting

[Section 9.2.1](#) describes the procedure of setting borders for segmentation of the main memory. This section describes how the values in MPUSEGBx[15:0] and MPUIPSEGBx[15:0] bits need to be set to achieve the desired borders for different memory sizes. The bits of the MUSBx[15:0] bits represent the 16 most significant bits of the border address that can be selected.

The setting of the MPUSEGBx[15:0] bits forms a border between two segments of main memory space. For the following examples, the segment with the higher address range formed by this border is called the higher segment. The segment with the lower address range is called the lower segment.

The lowest address in the higher segment can be calculated with the following formula:

Given:

Segment Border Address (BA) or register value MPUSEGBx

Hence follows:

$\text{MPUSEGBx} = (\text{BA}) \gg 4$

$\text{BA} = (\text{MPUSEGBx} \ll 4)$

Examples:

Segment border address = 0x0F000 → $\text{MPUSEGBx} = (0x0F000 \gg 4) = 0x0F00$

Segment border address = 0x13000 → $\text{MPUSEGBx} = (0x13000 \gg 4) = 0x1300$

$\text{MPUSEGBx} = 0x1100 \rightarrow \text{segment border address} = (0x1100 \ll 4) = 0x11000$

**Table 9-3. MPU Border Selection Example 64KB
(004000h to 013FFFh)**

Border Address	MPUSEGBx[15:0]
(outside)	0000h
⋮	⋮
(outside)	03C0h
04000h	0400h
04400h	0440h
04800h	0480h
04C00h	04C0h
05000h	0500h
⋮	⋮
0F000h	0F00h
0F400h	0F40h
0F800h	0F80h
0FC00h	0FC0h
10000h	1000h
10400h	1040h
⋮	⋮
13000h	1300h
13400h	1340h
13800h	1380h
13C00h	13C0h
14000h (top of memory)	1400h
(outside)	1440h
⋮	⋮
(outside)	3F80h
(outside)	3FC0h

NOTE: Depending on the memory size settings for MPUSEGBx[4:0], the calculation may result in a lower address space than is available. For those settings, a lower segment does not exist, and the higher segment starts with the first available memory address (see memory organization in device-specific data sheet).

9.2.4 IP Encapsulation Border Settings

The setting of the boundaries for the IP Encapsulation segment follows the same principle as the Main Segment settings.

9.2.5 Information Memory

The information memory is a partition of memory that is 512 bytes in size. The information memory can be used to store application-specific information (such as IDs or version numbers), or it can be used for executable code. It is located at address 01800h to 019FFh.

9.3 MPU Access Management Settings

Each segment described in and [Section 9.2.5](#) can have read, write, and execute access rights set independently.

The MPUSAM register allows setting the access rights for the four segments (information memory segment, three main memory segments) . MPUSEGxRE enables read access for segment x, MPUSEGxWE enables write access for segment x, and MPUSEGxXE enables code execution from segment x. JTAG and DMA accesses are treated as read or write data accesses and are evaluated according to the corresponding access bits.

[Table 9-4](#) shows the different settings of MPUSEGxXE, MPUSEGxWE, and MPUSEGxRE. Not all settings lead to a different memory protection. For example, as shown, if the execution bit MPUSEGxXE is set to 1, read access is automatically allowed independent of the setting of MPUSEGxRE. Also, setting the MPUSEGxWE bit to 1 enables the read option.

NOTE: Combinations that are not shown in [Table 9-4](#) should be avoided because they may be used in future versions of the MPU.

Table 9-4. Segment Access Rights

MPUSEGxXE	MPUSEGxWE	MPUSEGxRE	Execute Rights	Write Rights	Read Rights
0	0	0	No	No	No
0	0	1	No	No	Yes
0	1	1	No	Yes	Yes
1	0	1	Yes	No	Yes
1	1	1	Yes	Yes	Yes

NOTE: Discontinuity instructions at segment boundaries

Do not fill code segments to the last word with program code, because program discontinuity instructions like jump or branch instructions, RET, CALL, ... at a segment boundary can trigger an access right violation.

The CPU prefetches instructions beyond the one currently being executed. The MPU interprets the prefetch as read and instruction fetch accesses. For example if there is a JMP instruction (or any another discontinuity instruction) at the segment boundary, the CPU prefetches from the neighboring segment. This causes an access right violation if instruction fetches are not allowed within the neighboring segment.

9.4 MPU Violations

9.4.1 Interrupt Vector Table and Reset Vector

The interrupt vector table and the reset vector are located at addresses 0FF80h to 0FFFFh. It is possible to define a segment that includes this address space with restricted access rights. If an interrupt or a reset occurs, and this segment is read protected, the MPU automatically allows access to the Interrupt Vector memory space 0FF80h to 0FFFFh. Write rights are granted depending on MPU segment access management register MPUSAM. Only the interrupt vector table is read accessible. Access to the interrupt routine itself is not automatically enabled.

If the interrupt vector table is inside of the IP Encapsulation, the execute right is always prohibited. Code fetches at the addresses 0FF80h to 0FFFFh are always denied if IPE-segments include that memory range.

Table 9-5 shows the access right to the interrupt vector table for all possible cases.

Table 9-5. Access Rights to IVT⁽¹⁾

Condition	Read			Execute	Write		
	CPU	DMA	JTAG		CPU	DMA	JTAG
IVT belongs to ...							
an MPU segment (main memory segment)	O	O	O	O	C	C	C
the IPE-segment	O	O	O	X	O	O	O
an MPU segment and the IPE-segment	O	O	O	X	C	C	C

⁽¹⁾ O = Allowed, X = Not allowed, C = Depends on MPU segment access management register MPUSAM

NOTE: Only the interrupt table and the reset vector are opened on an interrupt or reset occurrence. If the application protects the segment that contains the interrupt routine itself from execution rights, a violation occurs.

9.4.2 Violation Handling

The handling of access rights violations can be selected for each segment with the MPUSEGxVS bit in the MPUSAM register. By default (MPUSEGxVS = 0), any access right violation causes a nonmaskable interrupt (NMI). Setting MPUSEGxVS = 1, causes a PUC to occur. In either case, the illegal instruction on a protected memory segment is not executed. Upon an access rights violation, the data bus content (MDB) is driven with 03FFFh until next valid data is available.

9.5 MPU Lock

The MPU registers can be protected from write access by setting the MPULOCK bit. Write access is not possible on all MPU registers except MPUCTL1, MPUIPC0, and MPUIPSEGBx until a BOR occurs. MPULOCK cannot be cleared manually.

MPUIPLOCK allows to separately lock the MPUIPC0 and MPUIPSEGBx registers. Write access is not possible on these registers until a BOR occurs. MPUIPLOCK cannot be cleared manually.

9.6 How to Enable MPU and IPE Segments

Both MPU and IPE-segments can be enabled at any time through the control registers. However, in typical applications, it may be required to initiate MPU segments or the IPE-segment before beginning the application program, thus the following features are offered as more secure and convenient methods.

1. Code Composer Studio for MSP430 offers an easy-to-use graphical interface to configure MPU segments and the IPE-segment. See the [Code Composer Studio for MSP430 User's Guide](#). When using the methods outlined in the document, no further user setup is required.
2. For the IPE-segment, when using the method described in the [Code Composer Studio for MSP430 User's Guide](#), the compiler and linker handle the structure generation and initialization automatically - [Section 9.6.1](#) describes what happens behind the scenes in this case, and further user setup is not required.

9.6.1 IP Encapsulation (IPE) Instantiation Using IPE Signatures

The boot code can preload user-defined setting before the start of application code. This ensures that the encapsulation is active before any user-controlled accesses to the memory can be performed.

9.6.1.1 IP Encapsulation Signatures

Two IPE signatures, IPE Signature 1 (memory location 0FF88h) and IPE Signature 2 (memory location 0FF8Ah), reside in FRAM and can be used to control the initialization of the IP Encapsulation. Write 0xAAAA to IPE Signature 1 to trigger the evaluation of the IPE Signature 2 as the IPE structure pointer. The following code for CCS is an example:

```
#define IPE_SIG_VALID    0xFF88
    // IPE signature valid flag
#define IPE_STR_PTR_SRC 0xFF8A
    // Source for pointer (nibble address) to MPU IPE structure

#pragma RETAIN(ipe_signalValid)
#pragma location=IPE_SIG_VALID
const unsigned int ipe_signalValid = 0xAAAA;

// Locate your IPE structure and it should be placed
// on the top of the IPE memory. In this example, IPE structure
// is at 0xD000
#pragma RETAIN(IPE_stringPointerSourceSource)
#pragma location=IPE_STR_PTR_SRC
const unsigned int IPE_stringPointerSourceSource = (__INSERT_IPE_STRUCT_ADDRESS__(0xD000)) >> 4;
```

Table 9-6. IPE Signatures

Signature	Address	Symbolic Name	Description
IPE Signature 1	0FF88h	IPE_SIG_VALID	IPE signature valid flag
IPE Signature 2	0FF8Ah	IPE_STR_PTR_SRC	Source for pointer (nibble address) to MPU IPE structure

9.6.1.1.1 Trapdoor Mechanism for IP Structure Pointer Transfer

The bootcode performs a sequence to ensure the integrity of the IPE structure pointer. On bootcode execution, a valid IPE Signature 1 triggers the transfer of the IPE Signature 2 (IPE structure pointer source) to a secured nonvolatile system data area (saved IPE structure pointer). This transfer only happens once if no previous secured IPE structure pointer exist. Subsequent of a successful transfer of the IPE structure pointer, the IPE Signatures can be overwritten by any value without compromising the existing IP Encapsulation.

NOTE: Memory locations for IPE Signatures are shared with the JTAG password. This gives the limitation that the first word of the JTAG password cannot be set to 0AAAAh for a nonprotected device, because this would unintentionally trigger the trapdoor mechanism.

9.6.1.2 IP Encapsulation Init Structure

By evaluating the saved IPE structure pointer, the bootcode can program the IP Encapsulation related register by transferring the values defined in the IP Encapsulation init structure to the corresponding fields in the MPU control registers. The definition of the structure can be seen in [Table 9-7](#). The check code is calculated as an odd bit interleaved parity of the previous three words. As an example, see the following code for CCS:

```
// IPE data structures definition, reusable for ALL projects
#define IPE_MPUIPLOCK    0x0080
#define IPE_MPUIPENANA    0x0040
#define IPE_MPUIPPUC    0x0020
#define IPE_SEGREG(a)    (a >> 4)
#define IPE_BIP(a,b,c)    (a ^ b ^ c ^ 0xFFFF)
#define IPE_FILLSTRUCT(a,b,c)
```

```
{a, IPE_SEGREG(b), IPE_SEGREG(c), IPE_BIP(a, IPE_SEGREG(b), IPE_SEGREG(c))}}

typedef struct IPE_Init_Structure {
    unsigned int MPUIPC0 ;
    unsigned int MPUIPB2 ;
    unsigned int MPUIPB1 ;
    unsigned int MPUCHECK ;
} IPE_Init_Structure;          // this struct should be placed inside IPB1/IPB2 boundaries

// This is the project dependant part

#define IPE_START 0x0D000      // This defines the Start of the IP protected area
#define IPE_END   0x0F000      // This defines the End of the IP protected area

// define borders of protected code
// ipestruct is defined in a adopted linker control file
// ipestruct is the section for protected data;
#pragma RETAIN(ipe_configStructure)
#pragma DATA_SECTION(ipe_configStructure, ".ipestruct");

const IPE_Init_Structure ipe_configStructure = IPE_FILLSTRUCT(IPE_MPUIPLOCK + IPE_MPUIPENA,
                                                             IPE_END, IPE_START);
```

Table 9-7. IPE_Init_Structure

Field Name	Address Offset	Length	Description
MPUIPC0	0h	word	Control setting for IP Encapsulation. Value is written to MPUIPC0
MPUIPB2	2h	word	Upper border of IP Encapsulation segment. Value is written to MPUIPSEGB2.
MPUIPB1	4h	word	Lower border of IP Encapsulation segment. Value is written to MPUIPSEGB1.
MPUCHECK	6h	word	Odd bit interleaved parity

NOTE: Although the user is free to select the location for the IPE Init Structure, protection against unwanted modification is given only if the structure is placed inside of the protected area checked by the structure itself. This allows a reconfiguration from within the protected area but prevents malicious modification from outside.

9.6.2 IP Encapsulation Removal

After successful instantiation of an IP protected memory area, a mass erase erases only the memory area outside of the IP Encapsulation. To perform an erase of all memory locations in main memory and to remove the IPE structure pointer, a special erase sequence must be performed. For more details, see the [MSP430 Programming With the JTAG Interface](#). For details on how to initiate this erasure from the IDE, see the [Code Composer Studio for MSP430 User's Guide](#).

NOTE: An invalid IP Encapsulation init structure or a saved IPE structure pointer with an invalid target (not pointing to a valid IP Encapsulation init structure) causes an erase of all nonvolatile memory segments including the IP Encapsulation segments and the init structure during bootcode execution. This setup error leads to a completely unprogrammed device after the next bootcode execution. This mechanism ensures that no exposure of IP code can happen by a misconfiguration or a memory corruption.

9.7 MPU Registers

The MPU registers are listed in [Table 9-8](#). The base address of the MPU module can be found in the device-specific data sheet. The address offset of each MPU register is given in [Table 9-8](#). The password defined in the MPUCTL0 register controls access to all MPU registers. Once the correct password is written, the write access is enabled. The write access is disabled by writing a wrong password in byte mode to the MPUCTL0 upper byte. Word accesses to MPUCTL0 with a wrong password triggers a PUC. A write access to a register other than MPUCTL0 while write access is not enabled causes a PUC. This behavior is independent from MPULOCK bit settings. Password write is always enabled to allow consecutive access to MPUCTL1 and independent configuration of MPU and IP-Encapsulation registers.

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 9-8. MPU Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	MPUCTL0	Memory Protection Unit Control 0	Read/write	Word	9600h	Section 9.7.1
00h	MPUCTL0_L		Read/Write	Byte	00h	
01h	MPUCTL0_H		Read/Write	Byte	96h	
02h	MPUCTL1	Memory Protection Unit Control 1	Read/write	Word	0000h	Section 9.7.2
02h	MPUCTL1_L		Read/Write	Byte	00h	
03h	MPUCTL1_H		Read/Write	Byte	00h	
04h	MPUSEGB2	Memory Protection Unit Segmentation Border 2 Register	Read/write	Word	0000h	
04h	MPUSEGB2_L		Read/Write	Byte	00h	
05h	MPUSEGB2_H		Read/Write	Byte	00h	
06h	MPUSEGB1	Memory Protection Unit Segmentation Border 1 Register	Read/Write	Word	0000h	
06h	MPUSEGB1_L		Read/Write	Byte	00h	
07h	MPUSEGB1_H		Read/Write	Byte	00h	
08h	MPUSAM	Memory Protection Unit Segmentation Access Management Register	Read/write	Word	7777h	Section 9.7.5
08h	MPUSAM_L		Read/Write	Byte	77h	
09h	MPUSAM_H		Read/Write	Byte	77h	
0Ah	MPUIPC0	Memory Protection Unit IP Control 0 Register	Read/Write	Word	0000h	Section 9.7.6
0Ah	MPUIPC0_L		Read/Write	Byte	00h	
0Bh	MPUIPC0_H		Read/Write	Byte	00h	
0Ch	MPUIPSEGB2	Memory Protection Unit IP Encapsulation Segment Border 2 Register	Read/Write	Word	0000h	
0Ch	MPUIPSEGB2_L		Read/Write	Byte	00h	
0Dh	MPUIPSEGB2_H		Read/Write	Byte	00h	
0Eh	MPUIPSEGB1	Memory Protection Unit IP Encapsulation Segment Border 1 Register	Read/Write	Word	0000h	
0Eh	MPUIPSEGB1_L		Read/Write	Byte	00h	
0Fh	MPUIPSEGB1_H		Read/Write	Byte	00h	

9.7.1 MPUCTL0 Register

Memory Protection Unit Control 0 Register

Figure 9-7. MPUCTL0 Register

15	14	13	12	11	10	9	8
MPUPW							
rw-1	rw-0	rw-0	rw-1	rw-0	rw-1	rw-1	rw-0
7	6	5	4	3	2	1	0
Reserved			MPUSEGIE	Reserved		MPULOCK	MPUENA
r-0	r-0	r-0	rw-[0]	r-0	r-0	rw-[0]	rw-[0]

Table 9-9. MPUCTL0 Register Description

Bit	Field	Type	Reset	Description
15-8	MPUPW	RW	96h	MPU Password. Always reads as 096h. Must be written as 0A5h; writing any other value with a word write generates a PUC. After a correct password is written and MPU register access is enabled, a wrong password write in byte mode disables the access and no PUC is generated. This behavior is independent from MPULOCK bit settings.
7-5	Reserved	R	0h	Reserved. Always read 0.
4	MPUSEGIE	RW	0h	Enable NMI Event if a Segment violation is detected in any Segment. 0b = Segment violation interrupt disabled 1b = Segment violation interrupt enabled
3-2	Reserved	R	0h	Reserved. Always read 0.
1	MPULOCK	RW	0h	MPU Lock. If this bit is set, access to all MPU Registers except MPUCTL1, MPUIPC0, and MPUIPSEGx are locked and they are read only until a BOR occurs. BOR sets MPULOCK to 0. 0b = Open 1b = Locked
0	MPUENA	RW	0h	MPU Enable. This bit enables the MPU operation. The enable bit can be set any time with word write and a correct password, if MPULOCK is not set 0b = Disabled 1b = Enabled

9.7.2 MPUCTL1 Register

Memory Protection Unit Control 1 Register

Figure 9-8. MPUCTL1 Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved			MPUSEG1IFG	MPUSEG2IFG	MPUSEG3IFG	MPUSEG1IFG	MPUSEG1IFG
r-0	r-0	r-0	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]

Table 9-10. MPUCTL1 Register Description

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Always read 0.
4	MPUSEG1IFG	RW	0h	IP Encapsulation Access Violation Interrupt Flag. This bit is set if an access violation in the IP Encapsulation memory segment is detected. This bit is cleared by software or by reading the reset vector word SYSRSTIV if it is the highest pending interrupt flag. This bit is write 0 only and write 1 has no effect. 0b = No interrupt pending 1b = Interrupt pending
3	MPUSEG2IFG	RW	0h	User Information Memory Violation Interrupt Flag. This bit is set if an access violation in User Information Memory is detected. This bit is cleared by software or by reading the reset vector word SYSRSTIV if it is the highest pending interrupt flag. This bit is write 0 only and write 1 has no effect. 0b = No interrupt pending 1b = Interrupt pending
2	MPUSEG3IFG	RW	0h	Main Memory Segment 3 Violation Interrupt Flag. This bit is set if an access violation in Main Memory Segment 3 is detected. This bit is cleared by software or by reading the reset vector word SYSRSTIV if it is the highest pending interrupt flag. This bit is write 0 only and write 1 has no effect. 0b = No interrupt pending 1b = Interrupt pending
1	MPUSEG2IFG	RW	0h	Main Memory Segment 2 Violation Interrupt Flag. This bit is set if an access violation in Main Memory Segment 2 is detected. This bit is cleared by software or by reading the reset vector word SYSRSTIV if it is the highest pending interrupt flag. This bit is write 0 only and write 1 has no effect. 0b = No interrupt pending 1b = Interrupt pending
0	MPUSEG1IFG	RW	0h	Main Memory Segment 1 Violation Interrupt Flag. This bit is set if an access violation in Main Memory Segment 1 is detected. This bit is cleared by software or by reading the reset vector word SYSRSTIV if it is the highest pending interrupt flag. This bit is write 0 only and write 1 has no effect. 0b = No interrupt pending 1b = Interrupt pending

9.7.3 MPUSEGB2 Register

Memory Protection Unit Segmentation Border 2 Register

Figure 9-9. MPUSEGB2 Register

15	14	13	12	11	10	9	8
MPUSEGB2							
r-0	rw-[0] or r-0	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]
7	6	5	4	3	2	1	0
MPUSEGB2							
rw-[0]	rw-[0]	r-0	r-0	r-0	r-0	r-0	r-0

Table 9-11. MPUSEGB2 Register Description

Bit	Field	Type	Reset	Description
15-0	MPUSEGB2	RW	0h	MPU Segment Border 2 address line equivalents. FRAM size $\leq$ 128KB: MPUSEGB2[15:14] = MPU Segment Border 2 address line 19-18 equivalents. Must be written as zero. MPUSEGB2[13:6] = MPU Segment Border 2 address lines 17-10. After BOR, the bits are set to 0 (if MPU is enabled and MPUSEGB1 is also 0, only Segment 3 is active). MPUSEGB2[5:0] = MPU Segment Border 2 address line 9-4 equivalents. Must be written as zero. 128KB < FRAM size $\leq$ 256KB: MPUSEGB2[15] = MPU Segment Border 2 address line 19 equivalents. Must be written as zero. MPUSEGB2[14:6] = MPU Segment Border 2 address lines 18-10. After BOR, the bits are set to 0 (if MPU is enabled and MPUSEGB1 is also 0, only Segment 3 is active). MPUSEGB2[5:0] = MPU Segment Border 2 address line 9-4 equivalents. Must be written as zero.

9.7.4 MPUSEGB1 Register

Memory Protection Unit Segmentation Border 1 Register

Figure 9-10. MPUSEGB1 Register

15	14	13	12	11	10	9	8
MPUSEGB1							
r-0	rw-[0] or r-0	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]
7	6	5	4	3	2	1	0
MPUSEGB1							
rw-[0]	rw-[0]	r-0	r-0	r-0	r-0	r-0	r-0

Table 9-12. MPUSEGB1 Register Description

Bit	Field	Type	Reset	Description
15-0	MPUSEGB1	RW	0h	MPU Segment Border 1 address line equivalents. FRAM size $\leq$ 128KB: MPUSEGB1[15:14] = MPU Segment Border 1 address line 19-18 equivalents. Must be written as zero. MPUSEGB1[13:6] = MPU Segment Border 1 address lines 17-10. After BOR, the bits are set to 0 (if MPU is enabled and MPUSEGB2 is also 0, only Segment 3 is active). MPUSEGB1[5:0] = MPU Segment Border 1 address line 9-4 equivalents. Must be written as zero. 128KB < FRAM size $\leq$ 256KB: MPUSEGB1[15] = MPU Segment Border 1 address line 19 equivalents. Must be written as zero. MPUSEGB1[14:6] = MPU Segment Border 1 address lines 18-10. After BOR, the bits are set to 0 (if MPU is enabled and MPUSEGB2 is also 0, only Segment 3 is active). MPUSEGB1[5:0] = MPU Segment Border 1 address line 9-4 equivalents. Must be written as zero.

9.7.5 MPUSAM Register

Memory Protection Unit Segmentation Access Management Register

Figure 9-11. MPUSAM Register

15	14	13	12	11	10	9	8
MPUSEGIVS	MPUSEGIXE	MPUSEGIWE	MPUSEGIRE	MPUSEG3VS	MPUSEG3XE	MPUSEG3WE	MPUSEG3RE
rw-[0]	rw-[1]	rw-[1]	rw-[1]	rw-[0]	rw-[1]	rw-[1]	rw-[1]
7	6	5	4	3	2	1	0
MPUSEG2VS	MPUSEG2XE	MPUSEG2WE	MPUSEG2RE	MPUSEG1VS	MPUSEG1XE	MPUSEG1WE	MPUSEG1RE
rw-[0]	rw-[1]	rw-[1]	rw-[1]	rw-[0]	rw-[1]	rw-[1]	rw-[1]

Table 9-13. MPUSAM Register Description

Bit	Field	Type	Reset	Description
15	MPUSEGIVS	RW	0h	MPU User Information Memory Segment Violation Select. This bit selects if additional to the interrupt flag a PUC must be executed on illegal access to User Information Memory 0b = Violation in User Information Memory asserts the MPUSEGIIFG bit and executes a SNMI if enabled by MPUSEGIE =1 1b = Violation in User Information Memory asserts the MPUSEGIIFG bit and executes a PUC
14	MPUSEGIXE	RW	1h	MPU User Information Memory Segment Execute Enable. If set, this bit enables execution on User Information Memory 0b = Execute code on User Information Memory causes a violation 1b = Execute code on User Information Memory is allowed
13	MPUSEGIWE	RW	1h	MPU User Information Memory Segment Write Enable. If set, this bit enables write access on User Information Memory 0b = Write on User Information Memory causes a violation 1b = Write on User Information Memory is allowed
12	MPUSEGIRE	RW	1h	MPU User Information Memory Segment Read Enable. If set, this bit enables read access on User Information Memory 0b = Read on User Information Memory causes a violation if MPUSEGIWE=MPUSEGIXE=0 1b = Read on User Information Memory is allowed
11	MPUSEG3VS	RW	0h	MPU Main Memory Segment 3 Violation Select. This bit selects if additional to the interrupt flag a PUC must be executed on illegal access to Main Memory segment 3 0b = Violation in Main Memory Segment 3 asserts the MPUSEG3IFG bit and executes a SNMI if enabled by MPUSEGIE =1 1b = Violation in Main Memory Segment 3 asserts the MPUSEG3IFG bit and executes a PUC
10	MPUSEG3XE	RW	1h	MPU Main Memory Segment 3 Execute Enable. If set this bit enables execution on Main Memory segment 3 0b = Execute code on Main Memory Segment 3 causes a violation 1b = Execute code on Main Memory Segment 3 is allowed
9	MPUSEG3WE	RW	1h	MPU Main Memory Segment 3 Write Enable. If set this bit enables write access on Main Memory segment 3 0b = Write on Main Memory Segment 3 causes a violation 1b = Write on Main Memory Segment 3 is allowed
8	MPUSEG3RE	RW	1h	MPU Main Memory Segment 3 Read Enable. If set this bit enables read access on Main Memory segment 3 0b = Read on Main Memory Segment 3 causes a violation if MPUSEG3WE = MPUSEG3XE = 0 1b = Read on Main Memory Segment 3 is allowed

Table 9-13. MPUSAM Register Description (continued)

Bit	Field	Type	Reset	Description
7	MPUSEG2VS	RW	0h	MPU Main Memory Segment 2 Violation Select. This bit selects if additional to the interrupt flag a PUC must be executed on illegal access to Main Memory segment 2 0b = Violation in Main Memory Segment 2 asserts the MPUSEG2IFG bit and executes a SNMI if enabled by MPUSEGIE = 1 1b = Violation in Main Memory Segment 2 asserts the MPUSEG2IFG bit and executes a PUC
6	MPUSEG2XE	RW	1h	MPU Main Memory Segment 2 Execute Enable. If set this bit enables execution on Main Memory segment 2 0b = Execute code on Main Memory Segment 2 causes a violation 1b = Execute code on Main Memory Segment 2 is allowed
5	MPUSEG2WE	RW	1h	MPU Main Memory Segment 2 Write Enable. If set this bit enables write access on Main Memory segment 2 0b = Write on Main Memory Segment 2 causes a violation 1b = Write on Main Memory Segment 2 is allowed
4	MPUSEG2RE	RW	1h	MPU Main Memory Segment 2 Read Enable. If set this bit enables read access on Main Memory segment 2 0b = Read on Main Memory Segment 2 causes a violation if MPUSEG2WE = MPUSEG2XE = 0 1b = Read on Main Memory Segment 2 is allowed
3	MPUSEG1VS	RW	0h	MPU Main Memory Segment 1 Violation Select. This bit selects if additional to the interrupt flag a PUC must be executed illegal access to Main Memory segment 1 0b = Violation in Main Memory Segment 1 asserts the MPUSEG1IFG bit and executes a SNMI if enabled by MPUSEGIE = 1 1b = Violation in Main Memory Segment 1 asserts the MPUSEG1IFG bit and executes a PUC
2	MPUSEG1XE	RW	1h	MPU Main Memory Segment 1 Execute Enable. If set this bit enables execution on Main Memory segment 1 0b = Execute code on Main Memory Segment 1 causes a violation 1b = Execute code on Main Memory Segment 1 is allowed
1	MPUSEG1WE	RW	1h	MPU Main Memory Segment 1 Write Enable. If set this bit enables write access on Main Memory segment 1 0b = Write on Main Memory Segment 1 causes a violation 1b = Write on Main Memory Segment 1 is allowed
0	MPUSEG1RE	RW	1h	MPU Main Memory Segment 1 Read Enable. If set this bit enables read access on Main Memory segment 1 0b = Read on Main Memory Segment 1 causes a violation if MPUSEG1WE = MPUSEG1XE = 0 1b = Read on Main Memory Segment 1 is allowed

9.7.6 MPUIPC0 Register

Memory Protection Unit IP Control 0 Register

Figure 9-12. MPUIPC0 Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
MPUIPLOCK	MPUIPENA	MPUIPVS	Reserved				
rw[0]	rw-[0]	rw-[0]	r-0	r-0	r-0	r-0	r-0

Table 9-14. MPUIPC0 Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved. Always read 0.
7	MPUIPLOCK	RW	0h	MPU IP Encapsulation Lock. If this bit is set, access to MPUIPC0 and MPUIPSEGBx registers is locked, and they are read-only until a BOR occurs. BOR sets the bit to 0. 0b = Open 1b = Locked
6	MPUIPENA	RW	0h	MPU IP Encapsulation Enable. This bit enables the MPU IP Encapsulation operation. The enable bit can be set any time with word write and a correct password, if MPUIPLOCK is not set 0b = Disabled 1b = Enabled
5	MPUIPVS	RW	0h	MPU IP Encapsulation segment Violation Select. This bit selects whether or not a PUC occurs on illegal access to the IPE-segment. 0b = Violation in Main Memory Segment 1 asserts the MPUSEGPIFG bit and executes a SNMI if enabled by MPUSEGIE = 1 1b = Violation in Main Memory Segment 1 asserts the MPUSEGPIFG bit and executes a PUC
4-0	Reserved	R	0h	Reserved. Always read 0.

9.7.7 MPUIPSEGB2 Register

Memory Protection Unit IP Encapsulation Segmentation Border 2 Register

Figure 9-13. MPUIPSEGB2 Register

15	14	13	12	11	10	9	8
MPUIPSEGB2							
r-0	rw-[0] or r-0	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]
7	6	5	4	3	2	1	0
MPUIPSEGB2							
rw-[0]	rw-[0]	r-0	r-0	r-0	r-0	r-0	r-0

Table 9-15. MPUIPSEGB2 Register Description

Bit	Field	Type	Reset	Description
15-0	MPUIPSEGB2	RW	0h	MPU IP Segment Border 2 address line equivalents. FRAM size $\leq$ 128KB: MPUIPSEGB2[15:14] = MPU IP Segment Border 2 address line 19-18 equivalents. Must be written as zero. MPUIPSEGB2[13:6] = MPU IP Segment Border 2 address lines 17-10. After BOR, the bits are set to 0 (if MPU is enabled and MPUSEGB1 is also 0, only Segment 3 is active). MPUIPSEGB2[5:0] = MPU IP Segment Border 2 address line 9-4 equivalents. Must be written as zero. 128KB < FRAM size $\leq$ 256KB: MPUIPSEGB2[15] = MPU IP Segment Border 2 address line 19 equivalents. Must be written as zero. MPUIPSEGB2[14:6] = MPU IP Segment Border 2 address lines 18-10. After BOR, the bits are set to 0 (if MPU is enabled and MPUSEGB1 is also 0, only Segment 3 is active). MPUIPSEGB2[5:0] = MPU IP Segment Border 2 address line 9-4 equivalents. Must be written as zero.

9.7.8 MPUIPSEGB1 Register

Memory Protection Unit IP Encapsulation Segmentation Border 1 Register

Figure 9-14. MPUIPSEGB1 Register

15	14	13	12	11	10	9	8
MPUIPSEGB1							
r-0	rw-[0] or r-0	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]	rw-[0]
7	6	5	4	3	2	1	0
MPUIPSEGB1							
rw-[0]	rw-[0]	r-0	r-0	r-0	r-0	r-0	r-0

Table 9-16. MPUIPSEGB1 Register Description

Bit	Field	Type	Reset	Description
15-0	MPUIPSEGB1	RW	0h	MPU Segment Border 1 address line equivalents. FRAM size $\leq$ 128KB: MPUIPSEGB1[15:14] = MPU Segment Border 1 address line 19-18 equivalents. Must be written as zero. MPUIPSEGB1[13:6] = MPU Segment Border 1 address lines 17-10. After BOR, the bits are is set to 0 (if MPU is enabled and MPUSEGB2 is also 0, only Segment 3 is active). MPUIPSEGB1[5:0] = MPU Segment Border 1 address line 9-4 equivalents. Must be written as zero. 128KB < FRAM size $\leq$ 256KB: MPUIPSEGB1[15] = MPU Segment Border 1 address line 19 equivalents. Must be written as zero. MPUIPSEGB1[14:6] = MPU Segment Border 1 address lines 18-10. After BOR, the bits are is set to 0 (if MPU is enabled and MPUSEGB2 is also 0, only Segment 3 is active). MPUIPSEGB1[5:0] = MPU Segment Border 1 address line 9-4 equivalents. Must be written as zero.

RAM Controller (RAMCTL)

The RAM controller (RAMCTL) allows control of the power-down behavior of the RAM.

Topic	Page
10.1 RAM Controller (RAMCTL) Introduction.....	332
10.2 RAMCTL Operation	332
10.3 RAMCTL Registers	334

10.1 RAM Controller (RAMCTL) Introduction

The RAM Controller allows reduction of the leakage current during LPM3 and LPM4. The RAM is partitioned in one to four sectors, depending on the device. See the device-specific data sheet for sector allocation and size.

10.2 RAMCTL Operation

Each sector y is controlled by a sector off control bit (RCRSyOFF0) in the RAM Controller Control Register 0 (RCCTL0). By default, the RAM content is retained in LPM3 and LPM4 (RCRSyOFF0 = 0).

By setting the RAM sector's control bit RCRSyOFF0 to 1, the respective RAM sector is powered down completely during LPM3 and LPM4 and all RAM content within the sector y is lost after a wake-up from LPM3 or LPM4. After wake-up the RAM can be accessed normally.

Figure 10-1 shows the possible transitions when entering LPM3 or LPM4 and when waking up from LPM3 or LPM4.

NOTE: After a wake-up from LPM3 and LPM4 with RCRSyOFF0 = 1, the content of powered down sectors is lost and completely undefined. Any potentially required re-initialization must be implemented in software.

The RCTL0 register is protected with a key. The RCTL0 register content can be modified only if the correct key is written during a word write. Byte write accesses or write accesses with a wrong key are ignored.

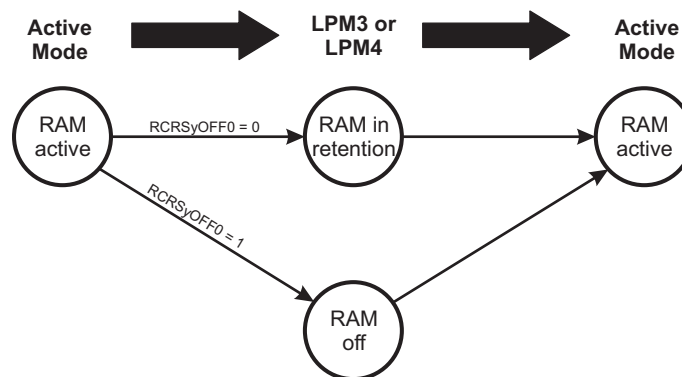


Figure 10-1. RAM Power Mode Transitions Into and Out of LPM3 or LPM4

10.2.1 Considerations for Complete Power Down

Using the power-down feature requires special care in devices with only one RAM sector or if all sectors are powered down. Usually the program stack is located in RAM; therefore, using the power-down (with RCRSyOFF0 = 1) destroys the stack content when entering LPM3 or LPM4. This is acceptable if the stack is empty when entering LPM3 or LPM4; otherwise, the stack must be located in a different memory (for example, FRAM).

10.2.2 DACCESSIE and DACCESSIFG Bits in RCTL1 Register

This section applies only to the devices that include both the USS and the LEA modules. The LEA RAM can be accessed by CPU, DMA, or DTC. Among the three bus master sources, the DTC has the highest priority. The DTC is the data transfer controller in the SDHS, which is a submodule of the USS module. The DTC transfers data from the SDHS directly to the LEA RAM. It is highly recommended not to access LEA RAM while the DTC is active. If CPU or DMA accesses the LEA RAM while the DTC is accessing the same memory:

- A write access from CPU or DMA is ignored

- A read access from CPU or DMA returns with 0x3FFF
- A instruction fetch access from CPU results in executing a jump \$ instruction (0x3FFF)
- A read or write access from CPU or DMA causes the DACCESSIFG bit to be set

This conflict does not affect the access by the DTC. The DACCESS interrupt can be enabled or disabled by the DACCESSIE bit. If DACCESSIE = 1 and DACCESSIFG = 1, then a user NMI is generated (DACCESSIFG). See the device-specific data sheet for the user NMI information.

10.3 RAMCTL Registers

[Table 10-1](#) lists the memory-mapped registers for the RAMCTL. All register offset addresses not listed in [Table 10-1](#) should be considered as reserved locations and the register contents should not be modified.

Table 10-1. RAMCTL Registers

Offset	Acronym	Register Name	Type	Reset	Section
0h	CTL0	RAM Controller Control 0	read-write	6900h	Section 10.3.1
2h	CTL1	RAM Controller Control 1	read-write	0h	Section 10.3.2

10.3.1 CTL0 Register (Offset = 0h) [reset = 6900h]

CTL0 is shown in [Figure 10-2](#) and described in [Table 10-2](#).

Return to [Summary Table](#).

RAM Controller Control 0

Figure 10-2. CTL0 Register

15	14	13	12	11	10	9	8
KEY							
R/W-69h							
7	6	5	4	3	2	1	0
RS3OFF		RS2OFF		RS1OFFx		RS0OFF	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

Table 10-2. CTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	KEY	R/W	69h	RAM controller key. Always reads as 69h. Must be written as 5Ah; any other write is ignored. 5Ah (W) = 0x5A
7-6	RS3OFF	R/W	0h	RAM controller RAM sector 3 off. 0h (R/W) = Contents of this RAM sector are retained in LPM3 and LPM4. 1h (R/W) = Turns off this RAM sector in LPM3 and LPM4, re-activates it on wake-up. All data of this RAM sector is lost after wakeup from LPM3 and LPM4. See the device-specific data sheet to find the number of available sectors, the address range, and the size of each RAM sector. 2h (R/W) = Turns off this RAM sector entering LPM3 and LPM4, the RAM sector remains off after wake-up. All data of this RAM sector is lost. See the device-specific data sheet to find the number of available sectors, the address range, and the size of each RAM sector.
5-4	RS2OFF	R/W	0h	RAM controller RAM sector 2 off. 0h (R/W) = Contents of this RAM sector are retained in LPM3 and LPM4. 1h (R/W) = Turns off this RAM sector in LPM3 and LPM4, re-activates it on wake-up. All data of this RAM sector is lost after wakeup from LPM3 and LPM4. See the device-specific data sheet to find the number of available sectors, the address range, and the size of each RAM sector. 2h (R/W) = Turns off this RAM sector entering LPM3 and LPM4, the RAM sector remains off after wake-up. All data of this RAM sector is lost. See the device-specific data sheet to find the number of available sectors, the address range, and the size of each RAM sector.

Table 10-2. CTL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-2	RS1OFFx	R/W	0h	RAM controller RAM sector 1 off. 0h (R/W) = Contents of this RAM sector are retained in LPM3 and LPM4. 1h (R/W) = Turns off this RAM sector in LPM3 and LPM4, re-activates it on wake-up. All data of this RAM sector is lost after wakeup from LPM3 and LPM4. See the device-specific data sheet to find the number of available sectors, the address range, and the size of each RAM sector. 2h (R/W) = Turns off this RAM sector entering LPM3 and LPM4, the RAM sector remains off after wake-up. All data of this RAM sector is lost. See the device-specific data sheet to find the number of available sectors, the address range, and the size of each RAM sector.
1-0	RS0OFF	R/W	0h	RAM controller RAM sector 0 off 0h (R/W) = Contents of this RAM sector are retained in LPM3 and LPM4. 1h (R/W) = Turns off this RAM sector in LPM3 and LPM4, re-activates it on wake-up. All data of this RAM sector is lost after wakeup from LPM3 and LPM4. See the device-specific data sheet to find the number of available sectors, the address range, and the size of each RAM sector. 2h (R/W) = Turns off this RAM sector entering LPM3 and LPM4, the RAM sector remains off after wake-up. All data of this RAM sector is lost. See the device-specific data sheet to find the number of available sectors, the address range, and the size of each RAM sector.

10.3.2 CTL1 Register (Offset = 2h) [reset = 0h]

CTL1 is shown in [Figure 10-3](#) and described in [Table 10-3](#).

Return to [Summary Table](#).

RAM Controller Control 1

Figure 10-3. CTL1 Register

15	14	13	12	11	10	9	8
Reserved							DACCESSIE
R-0h							R/W-0h
7	6	5	4	3	2	1	0
Reserved							DACCESSIFG
R-0h							R/W-0h

Table 10-3. CTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0h	Reserved
8	DACCESSIE	R/W	0h	DACCESS Interrupt enable bit. DACCESS Interrupt can be enabled or disabled by DACCESSIE bit. If DACCESSIE =1 and DACCESSIFG =1, then an User NMI is generated. See the device specific datasheet for details. 0h (R/W) = Disable NMI for DACCESS Interrupt 1h (R/W) = Enable NMI for DACCESS Interrupt
7-1	Reserved	R	0h	Reserved
0	DACCESSIFG	R/W	0h	DACCESS Interrupt Flag. LEA RAM can be accessed by CPU, DMA, or DTC. DTC has the highest priority. If CPU or DMA accesses LEA RAM while DTC is accessing the LEA RAM, the access by CPU or DMA is ignored and DACCESSIFG bit is set. It does not affect the access by DTC. DACCESS Interrupt can be enabled or disabled by DACCESSIE bit. If DACCESSIE =1 and DACCESSIFG =1, then an User NMI is generated (DACCESSIFG). See the device specific datasheet for details. 0h (R/W) = DACCESS Interrupt is not pending 1h (R/W) = DACCESS Interrupt is pending.

DMA Controller

The direct memory access (DMA) controller module transfers data from one address to another, without CPU intervention. This chapter describes the operation of the DMA controller.

Topic	Page
11.1 Direct Memory Access (DMA) Introduction	339
11.2 DMA Operation	341
11.3 DMA Registers	353

11.1 Direct Memory Access (DMA) Introduction

The DMA controller transfers data from one address to another, without CPU intervention, across the entire address range. For example, the DMA controller can move data from the ADC conversion memory to RAM.

Devices that contain a DMA controller can have up to eight DMA channels available. Therefore, depending on the number of DMA channels available, some features described in this chapter are not applicable to all devices. See the device-specific data sheet for the number of channels that are supported.

Using the DMA controller can increase the throughput of peripheral modules. It can also reduce system power consumption by allowing the CPU to remain in a low-power mode, without having to awaken to move data to or from a peripheral.

DMA controller features include:

- Up to eight independent transfer channels
- Configurable DMA channel priorities
- Requires only two MCLK clock cycles per transfer
- Byte, word, or mixed byte and word transfer capability
- Block sizes up to 65535 bytes or words
- Configurable transfer trigger selections
- Selectable-edge or level-triggered transfer
- Four addressing modes
- Single, block, or burst-block transfer modes

The DMA controller block diagram is shown in [Figure 11-1](#).

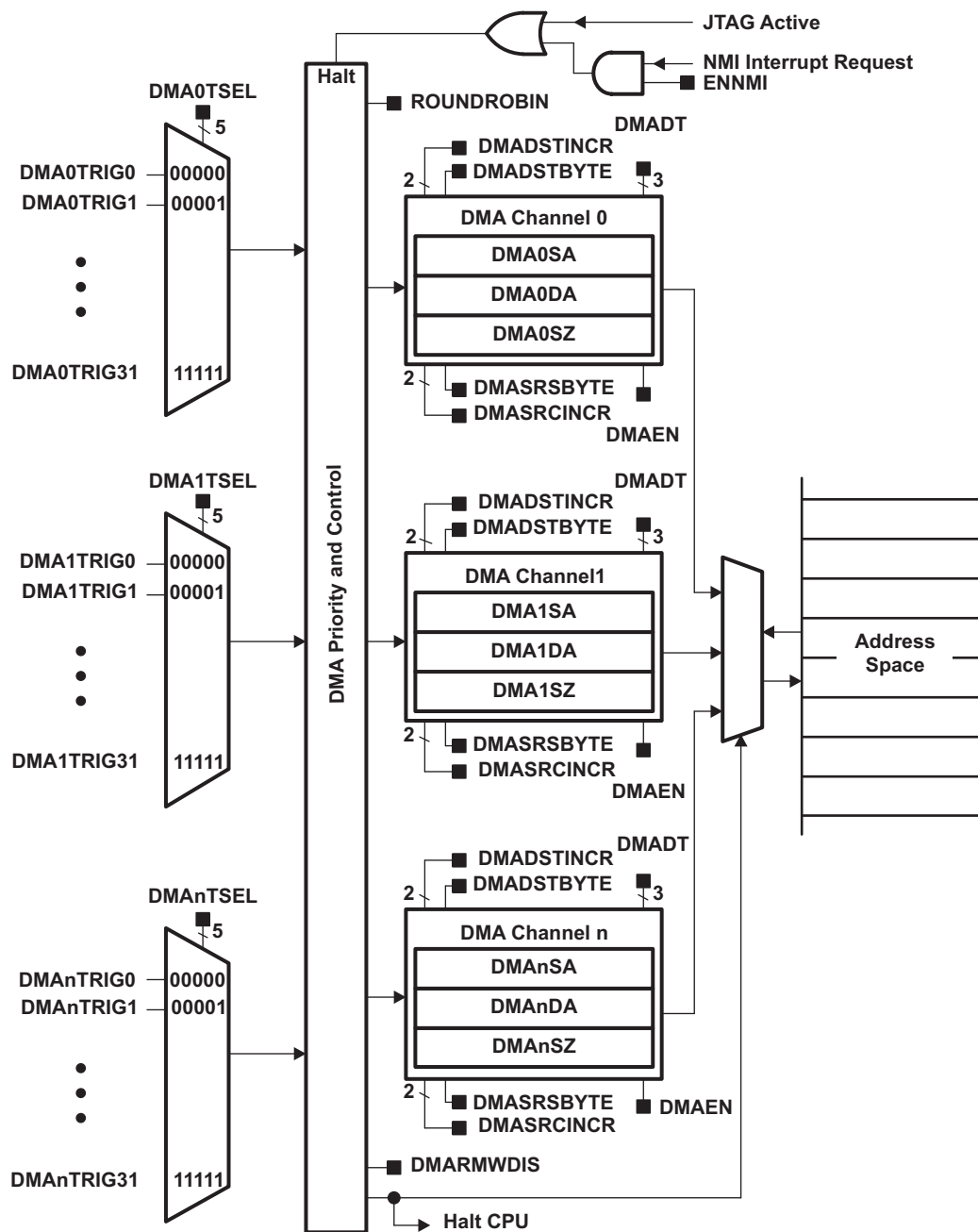


Figure 11-1. DMA Controller Block Diagram

11.2 DMA Operation

The DMA controller is configured with user software. The setup and operation of the DMA is discussed in the following sections.

11.2.1 DMA Addressing Modes

The DMA controller has four addressing modes. The addressing mode for each DMA channel is independently configurable. For example, channel 0 may transfer between two fixed addresses, while channel 1 transfers between two blocks of addresses. The addressing modes are shown in [Figure 11-2](#). The addressing modes are:

- Fixed address to fixed address
- Fixed address to block of addresses
- Block of addresses to fixed address
- Block of addresses to block of addresses

The addressing modes are configured with the DMASRCINCR and DMADSTINCR control bits. The DMASRCINCR bits select if the source address is incremented, decremented, or unchanged after each transfer. The DMADSTINCR bits select if the destination address is incremented, decremented, or unchanged after each transfer.

Transfers may be byte to byte, word to word, byte to word, or word to byte. When transferring word to byte, only the lower byte of the source word transfers. When transferring byte to word, the upper byte of the destination word is cleared when the transfer occurs.

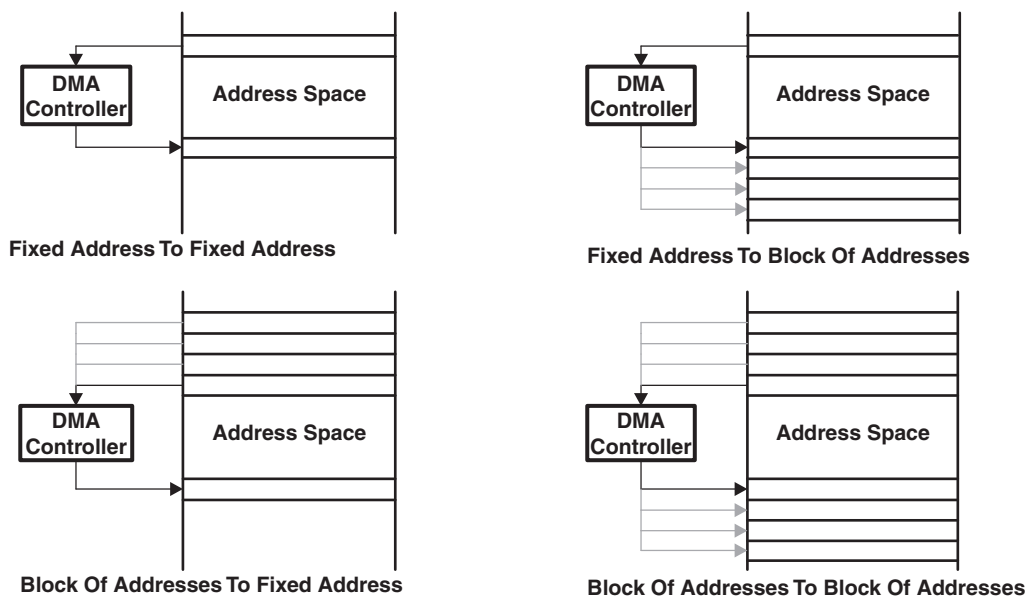


Figure 11-2. DMA Addressing Modes

11.2.2 DMA Transfer Modes

The DMA controller has six transfer modes selected by the DMADT bits as listed in [Table 11-1](#). Each channel is individually configurable for its transfer mode. For example, channel 0 may be configured in single transfer mode, while channel 1 is configured for burst-block transfer mode, and channel 2 operates in repeated block mode. The transfer mode is configured independently from the addressing mode. Any addressing mode can be used with any transfer mode.

Two types of data can be transferred selectable by the DMAxCTL DSTBYTE and SRCBYTE fields. The source and destination locations can be either byte or word data. It is also possible to transfer byte to byte, word to word, or any combination.

Table 11-1. DMA Transfer Modes

DMADT	Transfer Mode	Description
000	Single transfer	Each transfer requires a trigger. DMAEN is automatically cleared when DMAxSZ transfers have been made.
001	Block transfer	A complete block is transferred with one trigger. DMAEN is automatically cleared at the end of the block transfer.
010, 011	Burst-block transfer	CPU activity is interleaved with a block transfer. DMAEN is automatically cleared at the end of the burst-block transfer.
100	Repeated single transfer	Each transfer requires a trigger. DMAEN remains enabled.
101	Repeated block transfer	A complete block is transferred with one trigger. DMAEN remains enabled.
110, 111	Repeated burst-block transfer	CPU activity is interleaved with a block transfer. DMAEN remains enabled.

11.2.2.1 Single Transfer

In single transfer mode, each byte or word transfer requires a separate trigger. The single transfer state diagram is shown in Figure 11-3.

The DMAxSZ register defines the number of transfers to be made. The DMADSTINCR and DMASRCINCR bits select if the destination address and the source address are incremented or decremented after each transfer. If DMAxSZ = 0, no transfers occur.

The DMAxSA, DMAxDA, and DMAxSZ registers are copied into temporary registers. The temporary values of DMAxSA and DMAxDA are incremented or decremented after each transfer. The DMAxSZ register is decremented after each transfer. When the DMAxSZ register decrements to zero, it is reloaded from its temporary register and the corresponding DMAIFG flag is set. When DMA DT = 0, the DMAEN bit is cleared automatically when DMAxSZ decrements to zero and must be set again for another transfer to occur.

In repeated single transfer mode, the DMA controller remains enabled with DMAEN = 1, and a transfer occurs every time a trigger occurs.

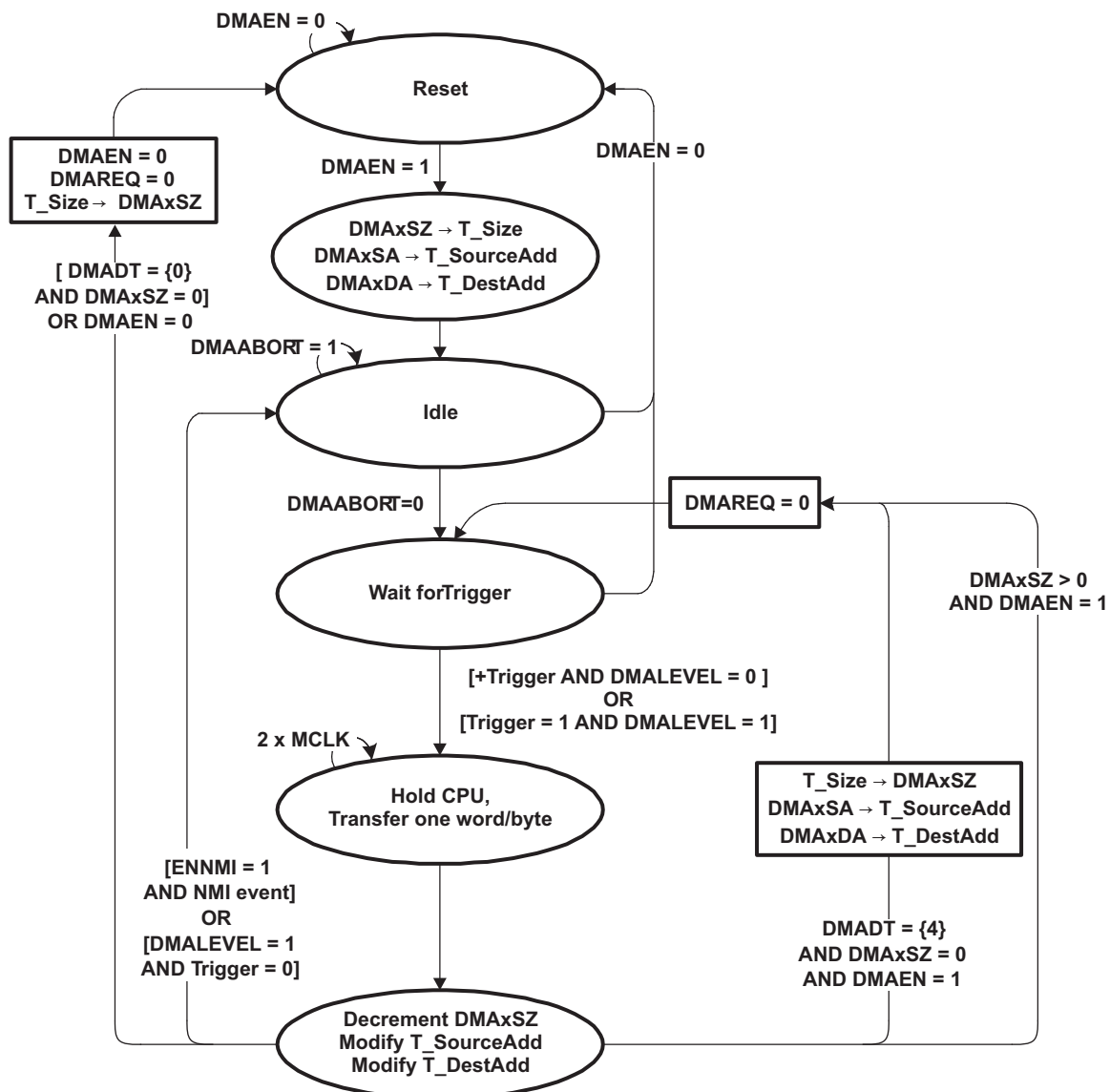


Figure 11-3. DMA Single Transfer State Diagram

11.2.2.2 Block Transfer

In block transfer mode, a transfer of a complete block of data occurs after one trigger. When $DMADT = 1$, the DMAEN bit is cleared after the completion of the block transfer and must be set again before another block transfer can be triggered. After a block transfer has started, another trigger signal that occurs during the block transfer is ignored. The block transfer state diagram is shown in [Figure 11-4](#).

The DMAxSZ register defines the size of the block, and the DMADSTINCR and DMASRCINCR bits select if the destination address and the source address are incremented or decremented after each transfer of the block. If DMAxSZ = 0, no transfers occur.

The DMAxSA, DMAxDA, and DMAxSZ registers are copied into temporary registers. The temporary values of DMAxSA and DMAxDA are incremented or decremented after each transfer in the block. The DMAxSZ register is decremented after each transfer of the block and shows the number of transfers remaining in the block. When the DMAxSZ register decrements to zero, it is reloaded from its temporary register and the corresponding DMAIFG flag is set.

During a block transfer, the CPU is halted until the complete block has been transferred. The block transfer takes $(2 \times MCLK \times DMAxSZ)$ clock cycles to complete. CPU execution resumes with its previous state after the block transfer is complete.

In repeated block transfer mode, the DMAEN bit remains set after completion of the block transfer. The next trigger after the completion of a repeated block transfer starts another block transfer.

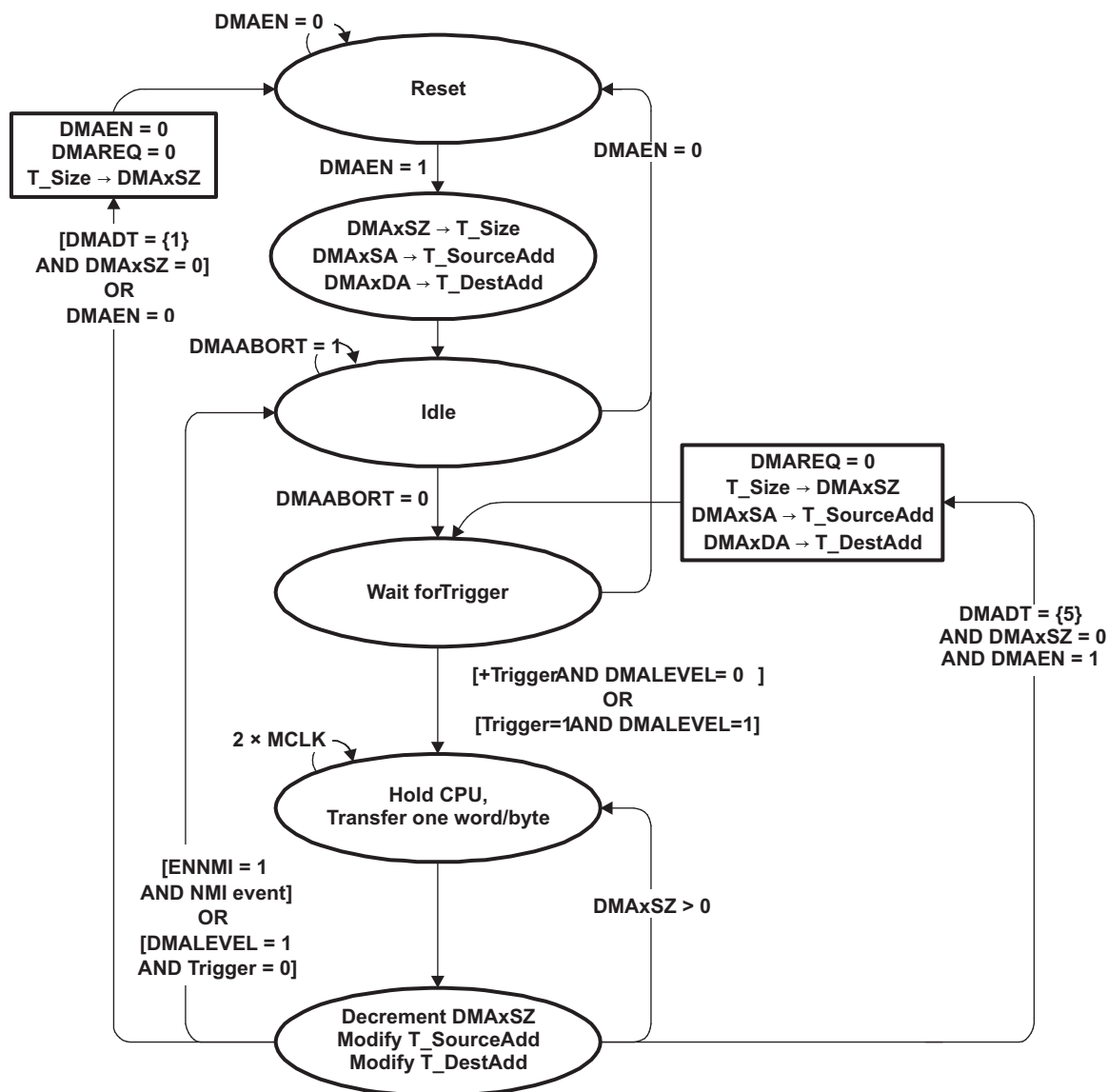


Figure 11-4. DMA Block Transfer State Diagram

11.2.2.3 Burst-Block Transfer

In burst-block mode, transfers are block transfers with CPU activity interleaved. The CPU executes two MCLK cycles after every four byte or word transfers of the block, resulting in 20% CPU execution capacity. After the burst-block, CPU execution resumes at 100% capacity and the DMAEN bit is cleared. DMAEN must be set again before another burst-block transfer can be triggered. After a burst-block transfer has been triggered, further trigger signals occurring during the burst-block transfer are ignored. The burst-block transfer state diagram is shown in [Figure 11-5](#).

The DMAxSZ register defines the size of the block, and the DMADSTINCR and DMASRCINCR bits select if the destination address and the source address are incremented or decremented after each transfer of the block. If DMAxSZ = 0, no transfers occur.

The DMAxSA, DMAxDA, and DMAxSZ registers are copied into temporary registers. The temporary values of DMAxSA and DMAxDA are incremented or decremented after each transfer in the block. The DMAxSZ register is decremented after each transfer of the block and shows the number of transfers remaining in the block. When the DMAxSZ register decrements to zero, it is reloaded from its temporary register and the corresponding DMAIFG flag is set.

In repeated burst-block mode, the DMAEN bit remains set after completion of the burst-block transfer and no further trigger signals are required to initiate another burst-block transfer. Another burst-block transfer begins immediately after completion of a burst-block transfer. In this case, the transfers must be stopped by clearing the DMAEN bit, or by an (non)maskable interrupt (NMI) when ENNMI is set. In repeated burst-block mode the CPU executes at 20% capacity continuously until the repeated burst-block transfer is stopped.

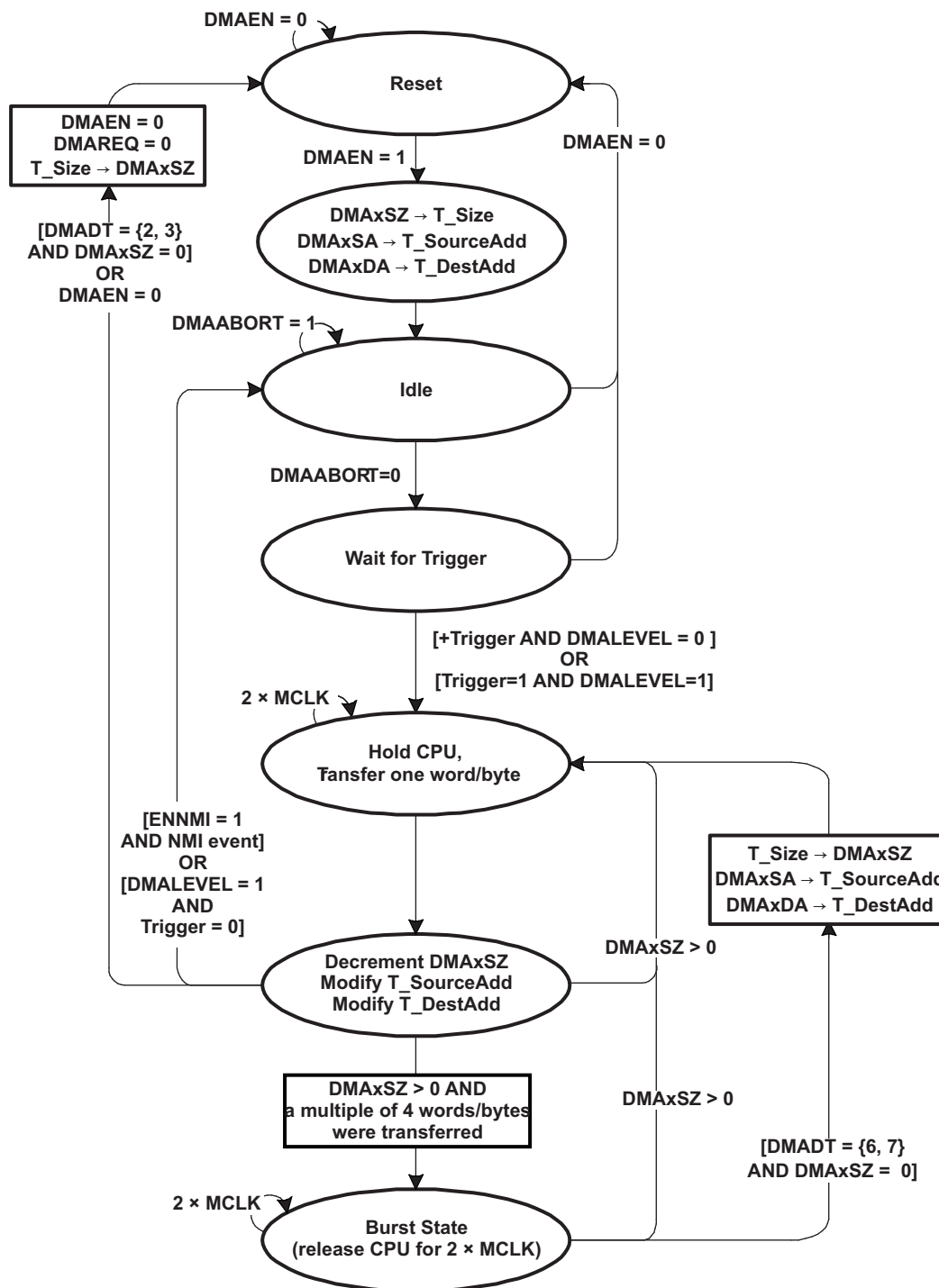


Figure 11-5. DMA Burst-Block Transfer State Diagram

11.2.3 Initiating DMA Transfers

Each DMA channel is independently configured for its trigger source with the DMAxTSEL. The DMAxTSEL bits should be modified only when the DMAEN bit in the DMAxCTL register is 0. Otherwise, unpredictable DMA triggers may occur. [Table 11-2](#) describes the trigger operation for each type of module. See the device-specific data sheet for the list of triggers available, along with their respective DMAxTSEL values.

When selecting the trigger, the trigger must not have already occurred, or the transfer does not take place.

11.2.3.1 Edge-Sensitive Triggers

When DMALEVEL = 0, edge-sensitive triggers are used, and the rising edge of the trigger signal initiates the transfer. In single-transfer mode, each transfer requires its own trigger. When using block or burst-block modes, only one trigger is required to initiate the block or burst-block transfer.

11.2.3.2 Level-Sensitive Triggers

When DMALEVEL = 1, level-sensitive triggers are used. For proper operation, level-sensitive triggers can only be used when external trigger DMAE0 is selected as the trigger. DMA transfers are triggered as long as the trigger signal is high and the DMAEN bit remains set.

The trigger signal must remain high for a block or burst-block transfer to complete. If the trigger signal goes low during a block or burst-block transfer, the DMA controller is held in its current state until the trigger goes back high or until the DMA registers are modified by software. If the DMA registers are not modified by software, when the trigger signal goes high again, the transfer resumes from where it was when the trigger signal went low.

When DMALEVEL = 1, transfer modes selected when DMADT = {0, 1, 2, 3} are recommended, because the DMAEN bit is automatically reset after the configured transfer.

11.2.4 Halting Executing Instructions for DMA Transfers

The DMARMWDIS bit controls when the CPU is halted for DMA transfers. When DMARMWDIS = 0, the CPU is halted immediately and the transfer begins when a trigger is received. In this case, it is possible that CPU read-modify-write operations can be interrupted by a DMA transfer. When DMARMWDIS = 1, the CPU finishes the currently executing read-modify-write operation before the DMA controller halts the CPU and the transfer begins (see [Table 11-2](#)).

Table 11-2. DMA Trigger Operation

Module	Operation
DMA	A transfer is triggered when the DMAREQ bit is set. The DMAREQ bit is automatically reset when the transfer starts. A transfer is triggered when the DMAxIFG flag is set. DMA0IFG triggers channel 1, DMA1IFG triggers channel 2, and DMA2IFG triggers channel 0. None of the DMAxIFG flags are automatically reset when the transfer starts. A transfer is triggered by the external trigger DMAE0.
Timer_A	A transfer is triggered when the TAxCCR0 CCIFG flag is set. The TAxCCR0 CCIFG flag is automatically reset when the transfer starts. If the TAxCCR0 CCIE bit is set, the TAxCCR0 CCIFG flag does not trigger a transfer. A transfer is triggered when the TAxCCR2 CCIFG flag is set. The TAxCCR2 CCIFG flag is automatically reset when the transfer starts. If the TAxCCR2 CCIE bit is set, the TAxCCR2 CCIFG flag does not trigger a transfer.
Timer_B	A transfer is triggered when the TBxCCR0 CCIFG flag is set. The TBxCCR0 CCIFG flag is automatically reset when the transfer starts. If the TBxCCR0 CCIE bit is set, the TBxCCR0 CCIFG flag does not trigger a transfer. A transfer is triggered when the TBxCCR2 CCIFG flag is set. The TBxCCR2 CCIFG flag is automatically reset when the transfer starts. If the TBxCCR2 CCIE bit is set, the TBxCCR2 CCIFG flag does not trigger a transfer.
eUSCI_Ax	A transfer is triggered when eUSCI_Ax receives new data. UCAXRXIFG is automatically reset when the transfer starts. If UCAXRXIE is set, the UCAXRXIFG does not trigger a transfer. A transfer is triggered when eUSCI_Ax is ready to transmit new data. UCAXTXIFG is automatically reset when the transfer starts. If UCAXTXIE is set, the UCAXTXIFG does not trigger a transfer.
eUSCI_Bx	A transfer is triggered when eUSCI_Bx receives new data. UCBxRXIFG is automatically reset when the transfer starts. If UCBxRXIE is set, the UCBxRXIFG does not trigger a transfer. A transfer is triggered when eUSCI_Bx is ready to transmit new data. UCBxTXIFG is automatically reset when the transfer starts. If UCBxTXIE is set, the UCBxTXIFG does not trigger a transfer.
ADC12_B	A transfer is triggered by an ADC12IFG flag. When single-channel conversions are performed, the corresponding ADC12IFG is the trigger. When sequences are used, the ADC12IFG for the last conversion in the sequence is the trigger. A transfer is triggered when the conversion is completed and the ADC12IFG is set. Setting the ADC12IFG with software does not trigger a transfer. All ADC12IFG flags are automatically reset when the associated ADC12MEMx register is accessed by the DMA controller.
MPY	A transfer is triggered when the hardware multiplier is ready for a new operand.
Reserved	No transfer is triggered.

11.2.5 Stopping DMA Transfers

There are two ways to stop DMA transfers in progress:

- A single, block, or burst-block transfer may be stopped with an NMI, if the ENNMI bit is set in register DMACTL1.
- A burst-block transfer may be stopped by clearing the DMAEN bit.

11.2.6 DMA Channel Priorities

The default DMA channel priorities are DMA0 through DMA7. If two or three triggers happen simultaneously or are pending, the channel with the highest priority completes its transfer (single, block, or burst-block transfer) first, then the second priority channel, then the third priority channel. Transfers in progress are not halted if a higher-priority channel is triggered. The higher-priority channel waits until the transfer in progress completes before starting.

The DMA channel priorities are configurable with the ROUNDROBIN bit. When the ROUNDROBIN bit is set, the channel that completes a transfer becomes the lowest priority. The order of the priority of the channels always stays the same, DMA0-DMA1-DMA2, for example, for three channels. When the ROUNDROBIN bit is cleared, the channel priority returns to the default priority.

DMA Priority	Transfer Occurs	New DMA Priority
DMA0→DMA1→DMA2	DMA1	DMA2→DMA0→DMA1
DMA2→DMA0→DMA1	DMA2	DMA0→DMA1→DMA2
DMA0→DMA1→DMA2	DMA0	DMA1→DMA2→DMA0

11.2.7 DMA Transfer Cycle Time

The DMA controller requires one or two MCLK clock cycles to synchronize before each single transfer or complete block or burst-block transfer. Each byte or word transfer requires two MCLK cycles after synchronization, and one cycle of wait time after the transfer. Because the DMA controller uses MCLK, the DMA cycle time is dependent on the MSP430 operating mode and clock system setup.

If the MCLK source is active but the CPU is off, the DMA controller uses the MCLK source for each transfer, without reenabling the CPU. If the MCLK source is off, the DMA controller temporarily restarts MCLK, sourced with DCOCLK, for the single transfer or complete block or burst-block transfer. The CPU remains off and, after the transfer completes, MCLK is turned off. The maximum DMA cycle time for all operating modes is shown in [Table 11-3](#).

Table 11-3. Maximum Single-Transfer DMA Cycle Time

CPU Operating Mode Clock Source	Maximum DMA Cycle Time
Active mode MCLK = DCOCLK	4 MCLK cycles
Active mode MCLK = LFXT1CLK	4 MCLK cycles
Low-power mode LPM0 or LPM1 MCLK = DCOCLK	5 MCLK cycles
Low-power mode LPM3 or LPM4 MCLK = DCOCLK	5 MCLK cycles + 5 μ s ⁽¹⁾
Low-power mode LPM0 or LPM1 MCLK = LFXT1CLK	5 MCLK cycles
Low-power mode LPM3 MCLK = LFXT1CLK	5 MCLK cycles
Low-power mode LPM4 MCLK = LFXT1CLK	5 MCLK cycles + 5 μ s ⁽¹⁾

⁽¹⁾ The additional 5 μ s are needed to start the DCOCLK. It is the $t_{(LPMx)}$ parameter in the data sheet.

11.2.8 Using DMA With System Interrupts

DMA transfers are not interruptible by system interrupts. System interrupts remain pending until the completion of the transfer. NMIs can interrupt the DMA controller if the ENNMI bit is set.

System interrupt service routines are interrupted by DMA transfers. If an interrupt service routine or other routine must execute with no interruptions, the DMA controller should be disabled before executing the routine.

11.2.9 DMA Controller Interrupts

Each DMA channel has its own DMAIFG flag. Each DMAIFG flag is set in any mode when the corresponding DMAxSZ register counts to zero. If the corresponding DMAIE and GIE bits are set, an interrupt request is generated.

All DMAIFG flags are prioritized, with DMA0IFG being the highest, and combined to source a single interrupt vector. The highest-priority enabled interrupt generates a number in the DMAIV register. This number can be evaluated or added to the program counter (PC) to automatically enter the appropriate software routine. Disabled DMA interrupts do not affect the DMAIV value.

Any access, read or write, of the DMAIV register automatically resets the highest pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. For example, assume that DMA0 has the highest priority. If the DMA0IFG and DMA2IFG flags are set when the interrupt service routine accesses the DMAIV register, DMA0IFG is reset automatically. After the RETI instruction of the interrupt service routine is executed, the DMA2IFG generates another interrupt.

11.2.9.1 DMAIV Software Example

The following software example shows the recommended use of DMAIV and the handling overhead for an eight channel DMA controller. The DMAIV value is added to the PC to automatically jump to the appropriate routine.

The numbers at the right margin show the necessary CPU cycles for each instruction. The software overhead for different interrupt sources includes interrupt latency and return-from-interrupt cycles, but not the task handling itself.

;Interrupt handler for DMAxIFG			Cycles
DMA_HND	...	; Interrupt latency	6
	ADD &DMAIV, PC	; Add offset to Jump table	3
	RETI	; Vector 0: No interrupt	5
	JMP DMA0_HND	; Vector 2: DMA channel 0	2
	JMP DMA1_HND	; Vector 4: DMA channel 1	2
	JMP DMA2_HND	; Vector 6: DMA channel 2	2
	JMP DMA3_HND	; Vector 8: DMA channel 3	2
	JMP DMA4_HND	; Vector 10: DMA channel 4	2
	JMP DMA5_HND	; Vector 12: DMA channel 5	2
	JMP DMA6_HND	; Vector 14: DMA channel 6	2
	JMP DMA7_HND	; Vector 16: DMA channel 7	2
DMA7_HND		; Vector 16: DMA channel 7	
	...	; Task starts here	
	RETI	; Back to main program	5
DMA6_HND		; Vector 14: DMA channel 6	
	...	; Task starts here	
	RETI	; Back to main program	5
DMA5_HND		; Vector 12: DMA channel 5	
	...	; Task starts here	
	RETI	; Back to main program	5
DMA4_HND		; Vector 10: DMA channel 4	
	...	; Task starts here	
	RETI	; Back to main program	5
DMA3_HND		; Vector 8: DMA channel 3	
	...	; Task starts here	
	RETI	; Back to main program	5
DMA2_HND		; Vector 6: DMA channel 2	
	...	; Task starts here	
	RETI	; Back to main program	5
DMA1_HND		; Vector 4: DMA channel 1	
	...	; Task starts here	
	RETI	; Back to main program	5
DMA0_HND		; Vector 2: DMA channel 0	
	...	; Task starts here	
	RETI	; Back to main program	5

11.2.10 Using the eUSCI_B I²C Module With the DMA Controller

The eUSCI_B I²C module provides two trigger sources for the DMA controller. The eUSCI_B I²C module can trigger a transfer when new I²C data is received and the when the transmit data is needed.

11.2.11 Using ADC12 With the DMA Controller

MSP430 devices with an integrated DMA controller can automatically move data from any ADC12MEMx register to another location. DMA transfers are done without CPU intervention and independently of any low-power modes. The DMA controller increases throughput of the ADC12 module, and enhances low-power applications allowing the CPU to remain off while data transfers occur.

DMA transfers can be triggered from any ADC12IFG flag. When CONSEQx = {0,2}, the ADC12IFG flag for the ADC12MEMx used for the conversion can trigger a DMA transfer. When CONSEQx = {1,3}, the ADC12IFG flag for the last ADC12MEMx in the sequence can trigger a DMA transfer. Any ADC12IFG flag is automatically cleared when the DMA controller accesses the corresponding ADC12MEMx.

11.3 DMA Registers

The DMA module registers are listed in [Table 11-4](#). The base addresses can be found in the device-specific data sheet. Each channel starts at its respective base address. The address offsets are listed in [Table 11-4](#).

Table 11-4. DMA Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	DMACTL0	DMA Control 0	Read/write	Word	0000h	Section 11.3.1
02h	DMACTL1	DMA Control 1	Read/write	Word	0000h	Section 11.3.2
04h	DMACTL2	DMA Control 2	Read/write	Word	0000h	Section 11.3.3
06h	DMACTL3	DMA Control 3	Read/write	Word	0000h	Section 11.3.4
08h	DMACTL4	DMA Control 4	Read/write	Word	0000h	Section 11.3.5
0Eh	DMAIV	DMA Interrupt Vector	Read only	Word	0000h	Section 11.3.10
00h	DMA0CTL	DMA Channel 0 Control	Read/write	Word	0000h	Section 11.3.6
02h	DMA0SA	DMA Channel 0 Source Address	Read/write	Word, double word	undefined	Section 11.3.7
06h	DMA0DA	DMA Channel 0 Destination Address	Read/write	Word, double word	undefined	Section 11.3.8
0Ah	DMA0SZ	DMA Channel 0 Transfer Size	Read/write	Word	undefined	Section 11.3.9
00h	DMA1CTL	DMA Channel 1 Control	Read/write	Word	0000h	Section 11.3.6
02h	DMA1SA	DMA Channel 1 Source Address	Read/write	Word, double word	undefined	Section 11.3.7
06h	DMA1DA	DMA Channel 1 Destination Address	Read/write	Word, double word	undefined	Section 11.3.8
0Ah	DMA1SZ	DMA Channel 1 Transfer Size	Read/write	Word	undefined	Section 11.3.9
00h	DMA2CTL	DMA Channel 2 Control	Read/write	Word	0000h	Section 11.3.6
02h	DMA2SA	DMA Channel 2 Source Address	Read/write	Word, double word	undefined	Section 11.3.7
06h	DMA2DA	DMA Channel 2 Destination Address	Read/write	Word, double word	undefined	Section 11.3.8
0Ah	DMA2SZ	DMA Channel 2 Transfer Size	Read/write	Word	undefined	Section 11.3.9
00h	DMA3CTL	DMA Channel 3 Control	Read/write	Word	0000h	Section 11.3.6
02h	DMA3SA	DMA Channel 3 Source Address	Read/write	Word, double word	undefined	Section 11.3.7
06h	DMA3DA	DMA Channel 3 Destination Address	Read/write	Word, double word	undefined	Section 11.3.8
0Ah	DMA3SZ	DMA Channel 3 Transfer Size	Read/write	Word	undefined	Section 11.3.9
00h	DMA4CTL	DMA Channel 4 Control	Read/write	Word	0000h	Section 11.3.6
02h	DMA4SA	DMA Channel 4 Source Address	Read/write	Word, double word	undefined	Section 11.3.7
06h	DMA4DA	DMA Channel 4 Destination Address	Read/write	Word, double word	undefined	Section 11.3.8
0Ah	DMA4SZ	DMA Channel 4 Transfer Size	Read/write	Word	undefined	Section 11.3.9
00h	DMA5CTL	DMA Channel 5 Control	Read/write	Word	0000h	Section 11.3.6
02h	DMA5SA	DMA Channel 5 Source Address	Read/write	Word, double word	undefined	Section 11.3.7
06h	DMA5DA	DMA Channel 5 Destination Address	Read/write	Word, double word	undefined	Section 11.3.8
0Ah	DMA5SZ	DMA Channel 5 Transfer Size	Read/write	Word	undefined	Section 11.3.9
00h	DMA6CTL	DMA Channel 6 Control	Read/write	Word	0000h	Section 11.3.6
02h	DMA6SA	DMA Channel 6 Source Address	Read/write	Word, double word	undefined	Section 11.3.7
06h	DMA6DA	DMA Channel 6 Destination Address	Read/write	Word, double word	undefined	Section 11.3.8

Table 11-4. DMA Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
0Ah	DMA6SZ	DMA Channel 6 Transfer Size	Read/write	Word	undefined	Section 11.3.9
00h	DMA7CTL	DMA Channel 7 Control	Read/write	Word	0000h	Section 11.3.6
02h	DMA7SA	DMA Channel 7 Source Address	Read/write	Word, double word	undefined	Section 11.3.7
06h	DMA7DA	DMA Channel 7 Destination Address	Read/write	Word, double word	undefined	Section 11.3.8
0Ah	DMA7SZ	DMA Channel 7 Transfer Size	Read/write	Word	undefined	Section 11.3.9

11.3.1 DMACTL0 Register

DMA Control 0 Register

Figure 11-6. DMACTL0 Register

15	14	13	12	11	10	9	8
Reserved			DMA1TSEL				
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
Reserved			DMA0TSEL				
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 11-5. DMACTL0 Register Description

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved. Always reads as 0.
12-8	DMA1TSEL	RW	0h	DMA trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment. 00000b = DMA1TRIG0 00001b = DMA1TRIG1 00010b = DMA1TRIG2 ⋮ 11110b = DMA1TRIG30 11111b = DMA1TRIG31
7-5	Reserved	R	0h	Reserved. Always reads as 0.
4-0	DMA0TSEL	RW	0h	DMA trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment. 00000b = DMA0TRIG0 00001b = DMA0TRIG1 00010b = DMA0TRIG2 ⋮ 11110b = DMA0TRIG30 11111b = DMA0TRIG31

11.3.2 DMACTL1 Register

DMA Control 1 Register

Figure 11-7. DMACTL1 Register

15	14	13	12	11	10	9	8
Reserved			DMA3TSEL				
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
Reserved			DMA2TSEL				
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 11-6. DMACTL1 Register Description

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved. Always reads as 0.
12-8	DMA3TSEL	RW	0h	DMA trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment. 00000b = DMA3TRIG0 00001b = DMA3TRIG1 00010b = DMA3TRIG2 ⋮ 11110b = DMA3TRIG30 11111b = DMA3TRIG31
7-5	Reserved	R	0h	Reserved. Always reads as 0.
4-0	DMA2TSEL	RW	0h	DMA trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment. 00000b = DMA2TRIG0 00001b = DMA2TRIG1 00010b = DMA2TRIG2 ⋮ 11110b = DMA2TRIG30 11111b = DMA2TRIG31

11.3.3 DMACTL2 Register

DMA Control 2 Register

Figure 11-8. DMACTL2 Register

15	14	13	12	11	10	9	8
Reserved			DMA5TSEL				
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
Reserved			DMA4TSEL				
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 11-7. DMACTL2 Register Description

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved. Always reads as 0.
12-8	DMA5TSEL	RW	0h	DMA trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment. 00000b = DMA5TRIG0 00001b = DMA5TRIG1 00010b = DMA5TRIG2 ⋮ 11110b = DMA5TRIG30 11111b = DMA5TRIG31
7-5	Reserved	R	0h	Reserved. Always reads as 0.
4-0	DMA4TSEL	RW	0h	DMA trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment. 00000b = DMA4TRIG0 00001b = DMA4TRIG1 00010b = DMA4TRIG2 ⋮ 11110b = DMA4TRIG30 11111b = DMA4TRIG31

11.3.4 DMACTL3 Register

DMA Control 3 Register

Figure 11-9. DMACTL3 Register

15	14	13	12	11	10	9	8
Reserved			DMA7TSEL				
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
Reserved			DMA6TSEL				
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 11-8. DMACTL3 Register Description

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved. Always reads as 0.
12-8	DMA7TSEL	RW	0h	DMA trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment. 00000b = DMA7TRIG0 00001b = DMA7TRIG1 00010b = DMA7TRIG2 ⋮ 11110b = DMA7TRIG30 11111b = DMA7TRIG31
7-5	Reserved	R	0h	Reserved. Always reads as 0.
4-0	DMA6TSEL	RW	0h	DMA trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment. 00000b = DMA6TRIG0 00001b = DMA6TRIG1 00010b = DMA6TRIG2 ⋮ 11110b = DMA6TRIG30 11111b = DMA6TRIG31

11.3.5 DMACTL4 Register

DMA Control 4 Register

Figure 11-10. DMACTL4 Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved					DMAARMWDIS	ROUNDROBIN	ENNMI
r0	r0	r0	r0	r0	rw-(0)	rw-(0)	rw-(0)

Table 11-9. DMACTL4 Register Description

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0h	Reserved. Always reads as 0.
2	DMAARMWDIS	RW	0h	Read-modify-write disable. When set, this bit inhibits any DMA transfers from occurring during CPU read-modify-write operations. 0b = DMA transfers can occur during read-modify-write CPU operations. 1b = DMA transfers inhibited during read-modify-write CPU operations
1	ROUNDROBIN	RW	0h	Round robin. This bit enables the round-robin DMA channel priorities. 0b = DMA channel priority is DMA0-DMA1-DMA2 - -DMA7. 1b = DMA channel priority changes with each transfer.
0	ENNMI	RW	0h	Enable NMI. This bit enables the interruption of a DMA transfer by an NMI. When an NMI interrupts a DMA transfer, the current transfer is completed normally, further transfers are stopped and DMAABORT is set. 0b = NMI does not interrupt DMA transfer 1b = NMI interrupts a DMA transfer

11.3.6 DMAxCTL Register

DMA Channel x Control Register

Figure 11-11. DMAxCTL Register

15	14	13	12	11	10	9	8
Reserved	DMADT			DMADSTINCR		DMASRCINCR	
r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
DMADSTBYTE	DMASRCBYTE	DMALEVEL	DMAEN	DMAIFG	DMAIE	DMAABORT	DMAREQ
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 11-10. DMAxCTL Register Description

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved. Always reads as 0.
14-12	DMADT	RW	0h	DMA transfer mode 000b = Single transfer 001b = Block transfer 010b = Burst-block transfer 011b = Burst-block transfer 100b = Repeated single transfer 101b = Repeated block transfer 110b = Repeated burst-block transfer 111b = Repeated burst-block transfer
11-10	DMADSTINCR	RW	0h	DMA destination increment. This bit selects automatic incrementing or decrementing of the destination address after each byte or word transfer. When DMADSTBYTE = 1, the destination address increments or decrements by one. When DMADSTBYTE = 0, the destination address increments or decrements by two. The DMAxDA is copied into a temporary register and the temporary register is incremented or decremented. DMAxDA is not incremented or decremented. 00b = Destination address is unchanged 01b = Destination address is unchanged 10b = Destination address is decremented 11b = Destination address is incremented
9-8	DMASRCINCR	RW	0h	DMA source increment. This bit selects automatic incrementing or decrementing of the source address for each byte or word transfer. When DMASRCBYTE = 1, the source address increments or decrements by one. When DMASRCBYTE = 0, the source address increments/decrements by two. The DMAxSA is copied into a temporary register and the temporary register is incremented or decremented. DMAxSA is not incremented or decremented. 00b = Source address is unchanged 01b = Source address is unchanged 10b = Source address is decremented 11b = Source address is incremented
7	DMADSTBYTE	RW	0h	DMA destination byte. This bit selects the destination as a byte or word. 0b = Word 1b = Byte
6	DMASRCBYTE	RW	0h	DMA source byte. This bit selects the source as a byte or word. 0b = Word 1b = Byte
5	DMALEVEL	RW	0h	DMA level. This bit selects between edge-sensitive and level-sensitive triggers. 0b = Edge sensitive (rising edge) 1b = Level sensitive (high level)
4	DMAEN	RW	0h	DMA enable 0b = Disabled 1b = Enabled

Table 11-10. DMAxCTL Register Description (continued)

Bit	Field	Type	Reset	Description
3	DMAIFG	RW	0h	DMA interrupt flag 0b = No interrupt pending 1b = Interrupt pending
2	DMAIE	RW	0h	DMA interrupt enable 0b = Disabled 1b = Enabled
1	DMAABORT	RW	0h	DMA abort. This bit indicates if a DMA transfer was interrupt by an NMI. 0b = DMA transfer not interrupted 1b = DMA transfer interrupted by NMI
0	DMAREQ	RW	0h	DMA request. Software-controlled DMA start. DMAREQ is reset automatically. 0b = No DMA start 1b = Start DMA

11.3.7 DMAxSA Register

DMA Source Address Register

Figure 11-12. DMAxSA Register

31	30	29	28	27	26	25	24
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
23	22	21	20	19	18	17	16
Reserved				DMAxSA			
r0	r0	r0	r0	rw	rw	rw	rw
15	14	13	12	11	10	9	8
DMAxSA							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
DMAxSA							
rw	rw	rw	rw	rw	rw	rw	rw

Table 11-11. DMAxSA Register Description

Bit	Field	Type	Reset	Description
31-20	Reserved	R	0h	Reserved. Always reads as 0.
19-0	DMAxSA	RW	undefined	DMA source address. The source address register points to the DMA source address for single transfers or the first source address for block transfers. The source address register remains unchanged during block and burst-block transfers. There are two words for the DMAxSA register. Bits 31-20 are reserved and always read as zero. Reading or writing bits 19-16 requires the use of extended instructions. When writing to DMAxSA with word instructions, bits 19-16 are cleared.

11.3.8 DMAxDA Register

DMA Destination Address Register

Figure 11-13. DMAxDA Register

31	30	29	28	27	26	25	24
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
23	22	21	20	19	18	17	16
Reserved				DMAxDA			
r0	r0	r0	r0	rw	rw	rw	rw
15	14	13	12	11	10	9	8
DMAxDA							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
DMAxDA							
rw	rw	rw	rw	rw	rw	rw	rw

Table 11-12. DMAxDA Register Description

Bit	Field	Type	Reset	Description
31-20	Reserved	R	0h	Reserved. Always reads as 0.
19-0	DMAxDA	RW	undefined	DMA destination address. The destination address register points to the DMA destination address for single transfers or the first destination address for block transfers. The destination address register remains unchanged during block and burst-block transfers. There are two words for the DMAxDA register. Bits 31–20 are reserved and always read as zero. Reading or writing bits 19–16 requires the use of extended instructions. When writing to DMAxDA with word instructions, bits 19–16 are cleared.

11.3.9 DMAxSZ Register

DMA Size Address Register

Figure 11-14. DMAxSZ Register

15	14	13	12	11	10	9	8
DMAxSZ							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
DMAxSZ							
rw	rw	rw	rw	rw	rw	rw	rw

Table 11-13. DMAxSZ Register Description

Bit	Field	Type	Reset	Description
15-0	DMAxSZ	RW	undefined	DMA size. The DMA size register defines the number of byte or word data per block transfer. DMAxSZ register decrements with each word or byte transfer. When DMAxSZ decrements to 0, it is immediately and automatically reloaded with its previously initialized value. 0000h = Transfer is disabled. 0001h = One byte or word is transferred. 0002h = Two bytes or words are transferred. ⋮ FFFFh = 65535 bytes or words are transferred.

11.3.10 DMAIV Register

DMA Interrupt Vector Register

Figure 11-15. DMAIV Register

15	14	13	12	11	10	9	8
DMAIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
DMAIV							
r0	r0	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r0

Table 11-14. DMAIV Register Description

Bit	Field	Type	Reset	Description
15-0	DMAIV	R	0h	DMA interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: DMA channel 0; Interrupt Flag: DMA0IFG; Interrupt Priority: Highest 04h = Interrupt Source: DMA channel 1; Interrupt Flag: DMA1IFG 06h = Interrupt Source: DMA channel 2; Interrupt Flag: DMA2IFG 08h = Interrupt Source: DMA channel 3; Interrupt Flag: DMA3IFG 0Ah = Interrupt Source: DMA channel 4; Interrupt Flag: DMA4IFG 0Ch = Interrupt Source: DMA channel 5; Interrupt Flag: DMA5IFG 0Eh = Interrupt Source: DMA channel 6; Interrupt Flag: DMA6IFG 10h = Interrupt Source: DMA channel 7; Interrupt Flag: DMA7IFG; Interrupt Priority: Lowest

Digital I/O

This chapter describes the operation of the digital I/O ports in all devices.

Topic	Page
12.1 Digital I/O Introduction	367
12.2 Digital I/O Operation	368
12.3 I/O Configuration	371
12.4 Digital I/O Registers	374

12.1 Digital I/O Introduction

The digital I/O features include:

- Independently programmable individual I/Os
- Any combination of input or output
- Individually configurable P1 and P2 interrupts. Some devices may include additional port interrupts.
- Independent input and output data registers
- Individually configurable pullup or pulldown resistors

Devices within the family may have up to twelve digital I/O ports implemented (P1 to P11 and PJ). Most ports contain eight I/O lines; however, some ports may contain less (see the device-specific data sheet for ports available). Each I/O line is individually configurable for input or output direction, and each can be individually read or written. Each I/O line is individually configurable for pullup or pulldown resistors.

Ports P1 and P2 always have interrupt capability. Each interrupt for the P1 and P2 I/O lines can be individually enabled and configured to provide an interrupt on a rising or falling edge of an input signal. All P1 I/O lines source a single interrupt vector ([P1IV](#)), and all P2 I/O lines source a different single interrupt vector ([P2IV](#)). Additional ports with interrupt capability may be available (see the device-specific data sheet for details) and contain their own respective interrupt vectors.

Individual ports can be accessed as byte-wide ports or can be combined into word-wide ports and accessed by word formats. Port pairs P1 and P2, P3 and P4, P5 and P6, P7 and P8, and so on, are associated with the names PA, PB, PC, PD, and so on, respectively. All port registers are handled in this manner with this naming convention except for the interrupt vector registers, P1IV and P2IV; that is, PAIV does not exist.

When writing to port PA with word operations, all 16 bits are written to the port. When writing to the lower byte of port PA using byte operations, the upper byte remains unchanged. Similarly, writing to the upper byte of port PA using byte instructions leaves the lower byte unchanged. When writing to a port that contains less than the maximum number of bits possible, the unused bits are don't care. Ports PB, PC, PD, PE, and PF behave similarly.

Reading port PA using word operations causes all 16 bits to be transferred to the destination. Reading the lower or upper byte of port PA (P1 or P2) and storing to memory using byte operations causes only the lower or upper byte to be transferred to the destination, respectively. Reading of port PA and storing to a general-purpose register using byte operations writes the byte that is transferred to the least significant byte of the register. The upper significant byte of the destination register is cleared automatically. Ports PB, PC, PD, PE, and PF behave similarly. When reading from ports that contain fewer than the maximum bits possible, unused bits are read as zeros (similarly for port PJ).

12.2 Digital I/O Operation

The digital I/O are configured with user software. The setup and operation of the digital I/O are discussed in the following sections.

12.2.1 Input Registers (PxIN)

Each bit in each PxIN register reflects the value of the input signal at the corresponding I/O pin when the pin is configured as I/O function. These registers are read only.

- Bit = 0: Input is low
- Bit = 1: Input is high

NOTE: Writing to read-only registers PxIN

Writing to these read-only registers results in increased current consumption while the write attempt is active.

12.2.2 Output Registers (PxOUT)

Each bit in each PxOUT register is the value to be output on the corresponding I/O pin when the pin is configured as I/O function, output direction.

- Bit = 0: Output is low
- Bit = 1: Output is high

If the pin is configured as I/O function, input direction and the pullup or pulldown resistor are enabled; the corresponding bit in the PxOUT register selects pullup or pulldown.

- Bit = 0: Pin is pulled down
- Bit = 1: Pin is pulled up

12.2.3 Direction Registers (PxDIR)

Each bit in each PxDIR register selects the direction of the corresponding I/O pin, regardless of the selected function for the pin. PxDIR bits for I/O pins that are selected for other functions must be set as required by the other function.

- Bit = 0: Port pin is switched to input direction
- Bit = 1: Port pin is switched to output direction

12.2.4 Pullup or Pulldown Resistor Enable Registers (PxREN)

Each bit in each PxREN register enables or disables the pullup or pulldown resistor of the corresponding I/O pin. The corresponding bit in the PxOUT register selects if the pin contains a pullup or pulldown.

- Bit = 0: Pullup or pulldown resistor disabled
- Bit = 1: Pullup or pulldown resistor enabled

[Table 12-1](#) summarizes the use of PxDIR, PxREN, and PxOUT for proper I/O configuration.

Table 12-1. I/O Configuration

PxDIR	PxREN	PxOUT	I/O Configuration
0	0	x	Input
0	1	0	Input with pulldown resistor
0	1	1	Input with pullup resistor
1	x	x	Output

12.2.5 Function Select Registers (PxSEL0, PxSEL1)

Port pins are often multiplexed with other peripheral module functions. See the device-specific data sheet to determine pin functions. Each port pin uses two bits to select the pin function – I/O port or one of the three possible peripheral module function. Table 12-2 shows how to select the various module functions. See the device-specific data sheet to determine pin functions. Each PxSEL bit is used to select the pin function – I/O port or peripheral module function.

Table 12-2. I/O Function Selection

PxSEL1	PxSEL0	I/O Function
0	0	General purpose I/O is selected
0	1	Primary module function is selected
1	0	Secondary module function is selected
1	1	Tertiary module function is selected

Setting the PxSEL1 or PxSEL0 bits to a module function does not automatically set the pin direction. Other peripheral module functions may require the PxDIR bits to be configured according to the direction needed for the module function. See the pin schematics in the device-specific data sheet.

When a port pin is selected as an input to peripheral modules, the input signal to those peripheral modules is a latched representation of the signal at the device pin. While PxSEL1 and PxSEL0 is other than 00, the internal input signal follows the signal at the pin for all connected modules. However, if PxSEL1 and PxSEL0 = 00, the input to the peripherals maintain the value of the input signal at the device pin before the PxSEL1 and PxSEL0 bits were reset.

Because the PxSEL1 and PxSEL0 bits do not reside in contiguous addresses, changing both bits at the same time is not possible. For example, an application might need to change P1.0 from general purpose I/O to the tertiary module function residing on P1.0. Initially, PxSEL1 = 00h and PxSEL0 = 00h. To change the function, it would be necessary to write both PxSEL1 = 01h and PxSEL0 = 01h. This is not possible without first passing through an intermediate configuration, and this configuration may not be desirable from an application standpoint. The PxSELC complement register can be used to handle such situations. The PxSELC register always reads 0. Each set bit of the PxSELC register complements the corresponding respective bit of the PxSEL1 and PxSEL0 registers. In the example, with PxSEL1 = 00h and PxSEL0 = 00h initially, writing PxSELC = 01h causes PxSEL1 = 01h and PxSEL0 = 01h to be written simultaneously.

NOTE: Interrupts are disabled when PxSEL1 = 1 or PxSEL0 = 1

When any PxSEL bit is set, the corresponding pin interrupt function is disabled. Therefore, signals on these pins do not generate interrupts, regardless of the state of the corresponding PxIE bit.

12.2.6 Port Interrupts

At least each pin in ports P1 and P2 have interrupt capability, configured with the PxIFG, PxIE, and PxIES registers. Some devices may contain additional port interrupts besides P1 and P2. See the device-specific data sheet to determine which port interrupts are available.

All Px interrupt flags are prioritized, with PxIFG.0 being the highest, and combined to source a single interrupt vector. The highest priority enabled interrupt generates a number in the PxIV register. This number can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled Px interrupts do not affect the PxIV value. The PxIV registers are word or byte access.

Each PxIFG bit is the interrupt flag for its corresponding I/O pin, and the flag is set when the selected input signal edge occurs at the pin. All PxIFG interrupt flags request an interrupt when their corresponding PxIE bit and the GIE bit are set. Software can also set each PxIFG flag, providing a way to generate a software-initiated interrupt.

- Bit = 0: No interrupt is pending
- Bit = 1: An interrupt is pending

Only transitions, not static levels, cause interrupts. If any PxIFG flag becomes set during a Px interrupt service routine or is set after the RETI instruction of a Px interrupt service routine is executed, the set PxIFG flag generates another interrupt. This ensures that each transition is acknowledged.

NOTE: PxIFG flags when changing PxOUT, PxDIR, or PxREN

Writing to PxOUT, PxDIR, or PxREN can result in setting the corresponding PxIFG flags.

Any access (read or write) of the lower byte of the PxIV register, either word or byte access, automatically resets the highest pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt.

For example, assume that P1IFG.0 has the highest priority. If the P1IFG.0 and P1IFG.2 flags are set when the interrupt service routine accesses the P1IV register, P1IFG.0 is reset automatically. After the RETI instruction of the interrupt service routine is executed, the P1IFG.2 generates another interrupt.

12.2.6.1 P1IV Software Example

The following software example shows the recommended use of P1IV and the handling overhead. The P1IV value is added to the PC to automatically jump to the appropriate routine. The code to handle any other PxIV register is similar.

The numbers at the right margin show the number of CPU cycles that are required for each instruction. The software overhead for different interrupt sources includes interrupt latency and return-from-interrupt cycles but not the task handling itself.

			Cycles
;Interrupt handler for P1			
P1_HND	...	; Interrupt latency	6
	ADD &P1IV,PC	; Add offset to Jump table	3
	RETI	; Vector 0: No interrupt	5
	JMP P1_0_HND	; Vector 2: Port 1 bit 0	2
	JMP P1_1_HND	; Vector 4: Port 1 bit 1	2
	JMP P1_2_HND	; Vector 6: Port 1 bit 2	2
	JMP P1_3_HND	; Vector 8: Port 1 bit 3	2
	JMP P1_4_HND	; Vector 10: Port 1 bit 4	2
	JMP P1_5_HND	; Vector 12: Port 1 bit 5	2
	JMP P1_6_HND	; Vector 14: Port 1 bit 6	2
	JMP P1_7_HND	; Vector 16: Port 1 bit 7	2
P1_7_HND		; Vector 16: Port 1 bit 7	
	...	; Task starts here	
	RETI	; Back to main program	5
P1_6_HND		; Vector 14: Port 1 bit 6	
	...	; Task starts here	
	RETI	; Back to main program	5
P1_5_HND		; Vector 12: Port 1 bit 5	
	...	; Task starts here	
	RETI	; Back to main program	5
P1_4_HND		; Vector 10: Port 1 bit 4	
	...	; Task starts here	
	RETI	; Back to main program	5
P1_3_HND		; Vector 8: Port 1 bit 3	
	...	; Task starts here	
	RETI	; Back to main program	5
P1_2_HND		; Vector 6: Port 1 bit 2	
	...	; Task starts here	
	RETI	; Back to main program	5
P1_1_HND		; Vector 4: Port 1 bit 1	
	...	; Task starts here	

```

    RETI                ; Back to main program          5
P1_0_HND                ; Vector 2: Port 1 bit 0
    ...                ; Task starts here
    RETI                ; Back to main program          5

```

12.2.6.2 Interrupt Edge Select Registers (PxIES)

Each PxIES bit selects the interrupt edge for the corresponding I/O pin.

- Bit = 0: Respective PxIFG flag is set on a low-to-high transition
- Bit = 1: Respective PxIFG flag is set on a high-to-low transition

NOTE: Writing to PxIES

Writing to [P1IES](#) or [P2IES](#) for each corresponding I/O can result in setting the corresponding interrupt flags.

PxIES	PxIN	PxIFG
0 → 1	0	Will be set
0 → 1	1	Unchanged
1 → 0	0	Unchanged
1 → 0	1	Will be set

12.2.6.3 Interrupt Enable Registers (PxIE)

Each PxIE bit enables the associated PxIFG interrupt flag.

- Bit = 0: The interrupt is disabled
- Bit = 1: The interrupt is enabled

12.3 I/O Configuration

12.3.1 Configuration After Reset

After a BOR reset, all port pins are high-impedance with Schmitt triggers and their module functions disabled to prevent any cross currents. The application must initialize all port pins including unused ones ([Section 12.3.2](#)) as input high impedance, input with pulldown, input with pullup, output high, or output low according to the application needs by configuring PxDIR, PxREN, PxOUT, and PxIES accordingly. This initialization takes effect as soon as the LOCKLPM5 bit in the PM5CTL register (described in the PMM chapter) is cleared; until then, the I/Os remain in their high-impedance state with Schmitt trigger inputs disabled. Note that this is usually the same I/O initialization that is required after a wake-up from LPMx.5. After clearing LOCKLPM5 all interrupt flags should be cleared (note, this is different to the wake-up from LPMx.5 flow). Then port interrupts can be enabled by setting the corresponding PxIE bits.

After a POR or PUC reset all port pins are configured as inputs with their module function being disabled. Also here to prevent floating inputs all port pins including unused ones ([Section 12.3.2](#)) should be configured according to the application needs as early as possible during the initialization procedure.

Note, the same I/O initialization procedure can be used for all reset cases and wake-up from LPMx.5 - except for PxIFG:

1. Initialize Ports: PxDIR, PxREN, PxOUT, and PxIES
2. Clear LOCKLPM5
3. If not wake-up from LPMx.5: clear all PxIFGs to avoid erroneous port interrupts
4. Enable port interrupts in PxIE

12.3.2 Configuration of Unused Port Pins

To prevent a floating input and to reduce power consumption, unused I/O pins should be configured as I/O function, output direction, and left unconnected on the PC board. The value of the PxOUT bit is don't care, because the pin is unconnected. Alternatively, the integrated pullup or pulldown resistor can be enabled by setting the PxREN bit of the unused pin to prevent a floating input. See the *System Resets, Interrupts, and Operating Modes, System Control Module (SYS)* chapter for termination of unused pins.

NOTE: Configuring port PJ and shared JTAG pins:

The application should make sure that port PJ is configured properly to prevent a floating input. Because port PJ is shared with the JTAG function, floating inputs may not be noticed when in an emulation environment. Port J is initialized to high-impedance inputs by default.

12.3.3 Configuration for LPMx.5 Low-Power Modes

NOTE: See , *Entering and Exiting Low-Power Modes LPMx.5*, in the *System Resets, Interrupts, and Operating Modes, System Control Module (SYS)* chapter for details about LPMx.5 low-power modes.

See the device-specific data sheet to determine which LPMx.5 low-power modes are available and which modules can operate in LPM3.5, if any.

With regard to the digital I/O, the following description is applicable to both LPM3.5 and LPM4.5.

Upon entering LPMx.5 (LPM3.5 or LPM4.5) the LDO of the PMM module is disabled, which removes the supply voltage from the core of the device. This causes all I/O register configurations to be lost, thus the configuration of I/O pins must be handled differently to ensure that all pins in the application behave in a controlled manner upon entering and exiting LPMx.5. Properly setting the I/O pins is critical to achieve the lowest possible power consumption in LPMx.5, and to prevent an uncontrolled input or output I/O state in the application. The application has complete control of the I/O pin conditions that are necessary to prevent unwanted spurious activity upon entry and exit from LPMx.5.

Before entering LPMx.5 the following operations are required for the I/Os:

- a. Set all I/Os to general-purpose I/Os (PxSEL0 = 000h and PxSEL1 = 000h) and configure as needed. Each I/O can be set to input high impedance, input with pulldown, input with pullup, output high, or output low. It is critical that no inputs are left floating in the application; otherwise, excess current may be drawn in LPMx.5.
Configuring the I/O in this manner ensures that each pin is in a safe condition before entering LPMx.5.
- b. Optionally, configure input interrupt pins for wake-up from LPMx.5. To wake the device from LPMx.5, a general-purpose I/O port must contain an input port with interrupt and wakeup capability. Not all inputs with interrupt capability offer wakeup from LPMx.5. See the device-specific data sheet for availability. To wake up the device, a port pin must be configured properly before entering LPMx.5. Each port should be configured as general-purpose input. Pulldowns or pullups can be applied if required. Setting the PxIES bit of the corresponding register determines the edge transition that wakes the device. Last, the PxIE for the port must be enabled, as well as the general interrupt enable.

NOTE: It is not possible to wake up from a port interrupt if its respective port interrupt flag is already asserted. It is recommended that the flag be cleared before entering LPMx.5. It is also recommended that GIE = 1 be set before entry into LPMx.5. Any pending flags in this case could then be serviced before LPMx.5 entry.

This completes the operations required for the I/Os before entering LPMx.5.

During LPMx.5 the I/O pin states are held and locked based on the settings before LPMx.5 entry. Note that only the pin conditions are retained. All other port configuration register settings such as PxDIR, PxREN, PxOUT, PxIES, and PxIE contents are lost.

Upon exit from LPMx.5, all peripheral registers are set to their default conditions but the I/O pins remain locked while LOCKLPM5 remains set. Keeping the I/O pins locked ensures that all pin conditions remain stable when entering the active mode, regardless of the default I/O register settings.

When back in active mode, the I/O configuration and I/O interrupt configuration such as PxDIR, PxREN, PxOUT, and PxIES should be restored to the values before entering LPMx.5. The LOCKLPM5 bit can then be cleared, which releases the I/O pin conditions and I/O interrupt configuration. Any changes to the port configuration registers while LOCKLPM5 is set have no effect on the I/O pins.

After enabling the I/O interrupts by configuring PxIE, the I/O interrupt that caused the wakeup can be serviced as indicated by the PxIFG flags. These flags can be used directly, or the corresponding PxIV register may be used. Note that the PxIFG flag cannot be cleared until the LOCKLPM5 bit has been cleared.

NOTE: It is possible that multiple events occurred on various ports. In these cases, multiple PxIFG flags are set, and it cannot be determined which port caused the I/O wakeup.

12.4 Digital I/O Registers

The digital I/O registers are listed in [Table 12-3](#). The base addresses can be found in the device-specific data sheet. Each port grouping begins at its base address. The address offsets are given in [Table 12-3](#).

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 12-3. Digital I/O Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
0Eh	P1IV	Port 1 Interrupt Vector	Read only	Word	0000h	Section 12.4.1
0Eh	P1IV_L		Read only	Byte	00h	
0Fh	P1IV_H		Read only	Byte	00h	
1Eh	P2IV	Port 2 Interrupt Vector	Read only	Word	0000h	Section 12.4.1
1Eh	P2IV_L		Read only	Byte	00h	
1Fh	P2IV_H		Read only	Byte	00h	
2Eh	P3IV	Port 3 Interrupt Vector	Read only	Word	0000h	Section 12.4.1
2Eh	P3IV_L		Read only	Byte	00h	
2Fh	P3IV_H		Read only	Byte	00h	
3Eh	P4IV	Port 4 Interrupt Vector	Read only	Word	0000h	Section 12.4.1
3Eh	P4IV_L		Read only	Byte	00h	
3Fh	P4IV_H		Read only	Byte	00h	
4Eh	P5IV	Port 5 Interrupt Vector	Read only	Word	0000h	Section 12.4.1
4Eh	P5IV_L		Read only	Byte	00h	
4Fh	P5IV_H		Read only	Byte	00h	
5Eh	P6IV	Port 6 Interrupt Vector	Read only	Word	0000h	Section 12.4.1
5Eh	P6IV_L		Read only	Byte	00h	
5Fh	P6IV_H		Read only	Byte	00h	
6Eh	P7IV	Port 7 Interrupt Vector	Read only	Word	0000h	Section 12.4.1
6Eh	P7IV_L		Read only	Byte	00h	
6Fh	P7IV_H		Read only	Byte	00h	
7Eh	P8IV	Port 8 Interrupt Vector	Read only	Word	0000h	Section 12.4.1
7Eh	P8IV_L		Read only	Byte	00h	
7Fh	P8IV_H		Read only	Byte	00h	
8Eh	P9IV	Port 9 Interrupt Vector	Read only	Word	0000h	Section 12.4.1
8Eh	P9IV_L		Read only	Byte	00h	
8Fh	P9IV_H		Read only	Byte	00h	
00h	P1IN or PAIN_L	Port 1 Input	Read only	Byte	undefined	Section 12.4.2
02h	P1OUT or PAOUT_L	Port 1 Output	Read/write	Byte	undefined	Section 12.4.3
04h	P1DIR or PADIR_L	Port 1 Direction	Read/write	Byte	00h	Section 12.4.4
06h	P1REN or PAREN_L	Port 1 Resistor Enable	Read/write	Byte	00h	Section 12.4.5
0Ah	P1SEL0 or PASEL0_L	Port 1 Select 0	Read/write	Byte	00h	Section 12.4.6
0Ch	P1SEL1 or PASEL1_L	Port 1 Select 1	Read/write	Byte	00h	Section 12.4.7
16h	P1SELC or PASELC_L	Port 1 Complement Selection	Read/write	Byte	00h	Section 12.4.8

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
18h	P1IES or PAIES_L	Port 1 Interrupt Edge Select	Read/write	Byte	undefined	Section 12.4.9
1Ah	P1IE or PAIE_L	Port 1 Interrupt Enable	Read/write	Byte	00h	Section 12.4.10
1Ch	P1IFG or PAIFG_L	Port 1 Interrupt Flag	Read/write	Byte	00h	Section 12.4.11

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
01h	P2IN or PAIN_H	Port 2 Input	Read only	Byte	undefined	Section 12.4.2
03h	P2OUT or PAOUT_H	Port 2 Output	Read/write	Byte	undefined	Section 12.4.3
05h	P2DIR or PADIR_H	Port 2 Direction	Read/write	Byte	00h	Section 12.4.4
07h	P2REN or PAREN_H	Port 2 Resistor Enable	Read/write	Byte	00h	Section 12.4.5
0Bh	P2SEL0 or PASEL0_H	Port 2 Select 0	Read/write	Byte	00h	Section 12.4.6
0Dh	P2SEL1 or PASEL1_H	Port 2 Select 1	Read/write	Byte	00h	Section 12.4.7
17h	P2SELC or PASELC_L	Port 2 Complement Selection	Read/write	Byte	00h	Section 12.4.8
19h	P2IES or PAIES_H	Port 2 Interrupt Edge Select	Read/write	Byte	undefined	Section 12.4.9
1Bh	P2IE or PAIE_H	Port 2 Interrupt Enable	Read/write	Byte	00h	Section 12.4.10
1Dh	P2IFG or PAIFG_H	Port 2 Interrupt Flag	Read/write	Byte	00h	Section 12.4.11
00h	P3IN or PBIN_L	Port 3 Input	Read only	Byte	undefined	Section 12.4.2
02h	P3OUT or PBOUT_L	Port 3 Output	Read/write	Byte	undefined	Section 12.4.3
04h	P3DIR or PBDIR_L	Port 3 Direction	Read/write	Byte	00h	Section 12.4.4
06h	P3REN or PBREN_L	Port 3 Resistor Enable	Read/write	Byte	00h	Section 12.4.5
0Ah	P3SEL0 or PBSEL0_L	Port 3 Select 0	Read/write	Byte	00h	Section 12.4.6
0Ch	P3SEL1 or PBSEL1_L	Port 3 Select 1	Read/write	Byte	00h	Section 12.4.7
16h	P3SELC or PBSELC_L	Port 3 Complement Selection	Read/write	Byte	00h	Section 12.4.8
18h	P3IES or PBIES_L	Port 3 Interrupt Edge Select	Read/write	Byte	undefined	Section 12.4.9
1Ah	P3IE or PBIE_L	Port 3 Interrupt Enable	Read/write	Byte	00h	Section 12.4.10
1Ch	P3IFG or PBIFG_L	Port 3 Interrupt Flag	Read/write	Byte	00h	Section 12.4.11

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
01h	P4IN or PBIN_H	Port 4 Input	Read only	Byte	undefined	Section 12.4.2
03h	P4OUT or PBOUT_H	Port 4 Output	Read/write	Byte	undefined	Section 12.4.3
05h	P4DIR or PBDIR_H	Port 4 Direction	Read/write	Byte	00h	Section 12.4.4
07h	P4REN or PBREN_H	Port 4 Resistor Enable	Read/write	Byte	00h	Section 12.4.5
0Bh	P4SEL0 or PBSEL0_H	Port 4 Select 0	Read/write	Byte	00h	Section 12.4.6
0Dh	P4SEL1 or PBSEL1_H	Port 4 Select 1	Read/write	Byte	00h	Section 12.4.7
17h	P4SELC or PBSELC_L	Port 4 Complement Selection	Read/write	Byte	00h	Section 12.4.8
19h	P4IES or PBIES_H	Port 4 Interrupt Edge Select	Read/write	Byte	undefined	Section 12.4.9
1Bh	P4IE or PBIE_H	Port 4 Interrupt Enable	Read/write	Byte	00h	Section 12.4.10
1Dh	P4IFG or PBIFG_H	Port 4 Interrupt Flag	Read/write	Byte	00h	Section 12.4.11
00h	P5IN or PCIN_L	Port 5 Input	Read only	Byte	undefined	Section 12.4.2
02h	P5OUT or PCOUT_L	Port 5 Output	Read/write	Byte	undefined	Section 12.4.3
04h	P5DIR or PCDIR_L	Port 5 Direction	Read/write	Byte	00h	Section 12.4.4
06h	P5REN or PCREN_L	Port 5 Resistor Enable	Read/write	Byte	00h	Section 12.4.5
0Ah	P5SEL0 or PCSEL0_L	Port 5 Select 0	Read/write	Byte	00h	Section 12.4.6
0Ch	P5SEL1 or PCSEL1_L	Port 5 Select 1	Read/write	Byte	00h	Section 12.4.7
16h	P5SELC or PCSELC_L	Port 5 Complement Selection	Read/write	Byte	00h	Section 12.4.8
18h	P5IES or PCIES_L	Port 5 Interrupt Edge Select	Read/write	Byte	undefined	Section 12.4.9
1Ah	P5IE or PCIE_L	Port 5 Interrupt Enable	Read/write	Byte	00h	Section 12.4.10
1Ch	P5IFG or PCIFG_L	Port 5 Interrupt Flag	Read/write	Byte	00h	Section 12.4.11

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
01h	P6IN or PCIN_H	Port 6 Input	Read only	Byte	undefined	Section 12.4.2
03h	P6OUT or PCOUT_H	Port 6 Output	Read/write	Byte	undefined	Section 12.4.3
05h	P6DIR or PCDIR_H	Port 6 Direction	Read/write	Byte	00h	Section 12.4.4
07h	P6REN or PCREN_H	Port 6 Resistor Enable	Read/write	Byte	00h	Section 12.4.5
0Bh	P6SEL0 or PCSEL0_H	Port 6 Select 0	Read/write	Byte	00h	Section 12.4.6
0Dh	P6SEL1 or PCSEL1_H	Port 6 Select 1	Read/write	Byte	00h	Section 12.4.7
17h	P6SELC or PCSELC_L	Port 6 Complement Selection	Read/write	Byte	00h	Section 12.4.8
19h	P6IES or PCIES_H	Port 6 Interrupt Edge Select	Read/write	Byte	undefined	Section 12.4.9
1Bh	P6IE or PCIE_H	Port 6 Interrupt Enable	Read/write	Byte	00h	Section 12.4.10
1Dh	P6IFG or PCIFG_H	Port 6 Interrupt Flag	Read/write	Byte	00h	Section 12.4.11
00h	P7IN or PDIN_L	Port 7 Input	Read only	Byte	undefined	Section 12.4.2
02h	P7OUT or PDOUT_L	Port 7 Output	Read/write	Byte	undefined	Section 12.4.3
04h	P7DIR or PDDIR_L	Port 7 Direction	Read/write	Byte	00h	Section 12.4.4
06h	P7REN or PDREN_L	Port 7 Resistor Enable	Read/write	Byte	00h	Section 12.4.5
0Ah	P7SEL0 or PDSEL0_L	Port 7 Select 0	Read/write	Byte	00h	Section 12.4.6
0Ch	P7SEL1 or PDSEL1_L	Port 7 Select 1	Read/write	Byte	00h	Section 12.4.7
16h	P7SELC or PDSELC_L	Port 7 Complement Selection	Read/write	Byte	00h	Section 12.4.8
18h	P7IES or PDIES_L	Port 7 Interrupt Edge Select	Read/write	Byte	undefined	Section 12.4.9
1Ah	P7IE or PDIE_L	Port 7 Interrupt Enable	Read/write	Byte	00h	Section 12.4.10
1Ch	P7IFG or PDIFG_L	Port 7 Interrupt Flag	Read/write	Byte	00h	Section 12.4.11

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
01h	P8IN or PDIN_H	Port 8 Input	Read only	Byte	undefined	Section 12.4.2
03h	P8OUT or PDOUT_H	Port 8 Output	Read/write	Byte	undefined	Section 12.4.3
05h	P8DIR or PDDIR_H	Port 8 Direction	Read/write	Byte	00h	Section 12.4.4
07h	P8REN or PDREN_H	Port 8 Resistor Enable	Read/write	Byte	00h	Section 12.4.5
0Bh	P8SEL0 or PDSEL0_H	Port 8 Select 0	Read/write	Byte	00h	Section 12.4.6
0Dh	P8SEL1 or PDSEL1_H	Port 8 Select 1	Read/write	Byte	00h	Section 12.4.7
17h	P8SELC or PDSELC_L	Port 8 Complement Selection	Read/write	Byte	00h	Section 12.4.8
19h	P8IES or PDIES_H	Port 8 Interrupt Edge Select	Read/write	Byte	undefined	Section 12.4.9
1Bh	P8IE or PDIE_H	Port 8 Interrupt Enable	Read/write	Byte	00h	Section 12.4.10
1Dh	P8IFG or PDIFG_H	Port 8 Interrupt Flag	Read/write	Byte	00h	Section 12.4.11
00h	P9IN or PEIN_L	Port 9 Input	Read only	Byte	undefined	Section 12.4.2
02h	P9OUT or PEOUT_L	Port 9 Output	Read/write	Byte	undefined	Section 12.4.3
04h	P9DIR or PEDIR_L	Port 9 Direction	Read/write	Byte	00h	Section 12.4.4
06h	P9REN or PEREN_L	Port 9 Resistor Enable	Read/write	Byte	00h	Section 12.4.5
0Ah	P9SEL0 or PESEL0_L	Port 9 Select 0	Read/write	Byte	00h	Section 12.4.6
0Ch	P9SEL1 or PESEL1_L	Port 9 Select 1	Read/write	Byte	00h	Section 12.4.7
16h	P9SELC or PESELC_L	Port 9 Complement Selection	Read/write	Byte	00h	Section 12.4.8
18h	P9IES or PEIES_L	Port 9 Interrupt Edge Select	Read/write	Byte	undefined	Section 12.4.9
1Ah	P9IE or PEIE_L	Port 9 Interrupt Enable	Read/write	Byte	00h	Section 12.4.10
1Ch	P9IFG or PEIFG_L	Port 9 Interrupt Flag	Read/write	Byte	00h	Section 12.4.11

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
01h	P10IN or PEIN_H	Port 10 Input	Read only	Byte	undefined	Section 12.4.2
03h	P10OUT or PEOUT_H	Port 10 Output	Read/write	Byte	undefined	Section 12.4.3
05h	P10DIR or PEDIR_H	Port 10 Direction	Read/write	Byte	00h	Section 12.4.4
07h	P10REN or PEREN_H	Port 10 Resistor Enable	Read/write	Byte	00h	Section 12.4.5
0Bh	P10SEL0 or PESEL0_H	Port 10 Select 0	Read/write	Byte	00h	Section 12.4.6
0Dh	P10SEL1 or PESEL1_H	Port 10 Select 1	Read/write	Byte	00h	Section 12.4.7
17h	P10SELC or PESELC_L	Port 10 Complement Selection	Read/write	Byte	00h	Section 12.4.8
19h	P10IES or PEIES_H	Port 10 Interrupt Edge Select	Read/write	Byte	undefined	Section 12.4.9
1Bh	P10IE or PEIE_H	Port 10 Interrupt Enable	Read/write	Byte	00h	Section 12.4.10
1Dh	P10IFG or PEIFG_H	Port 10 Interrupt Flag	Read/write	Byte	00h	Section 12.4.11
00h	P11IN or PFIN_L	Port 11 Input	Read only	Byte	undefined	Section 12.4.2
02h	P11OUT or PFOUT_L	Port 11 Output	Read/write	Byte	undefined	Section 12.4.3
04h	P11DIR or PFDIR_L	Port 11 Direction	Read/write	Byte	00h	Section 12.4.4
06h	P11REN or PFREN_L	Port 11 Resistor Enable	Read/write	Byte	00h	Section 12.4.5
0Ah	P11SEL0 or PFSEL0_L	Port 11 Select 0	Read/write	Byte	00h	Section 12.4.6
0Ch	P11SEL1 or PFSEL1_L	Port 11 Select 1	Read/write	Byte	00h	Section 12.4.7
16h	P11SELC or PFSELC_L	Port 11 Complement Selection	Read/write	Byte	00h	Section 12.4.8
18h	P11IES or PFIES_L	Port 11 Interrupt Edge Select	Read/write	Byte	undefined	Section 12.4.9
1Ah	P11IE or PFIE_L	Port 11 Interrupt Enable	Read/write	Byte	00h	Section 12.4.10
1Ch	P11IFG or PFIFG_L	Port 11 Interrupt Flag	Read/write	Byte	00h	Section 12.4.11

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	PAIN	Port A Input	Read only	Word	undefined	
00h	PAIN_L		Read only	Byte	undefined	
01h	PAIN_H		Read only	Byte	undefined	
02h	PAOUT	Port A Output	Read/write	Word	undefined	
02h	PAOUT_L		Read/write	Byte	undefined	
03h	PAOUT_H		Read/write	Byte	undefined	
04h	PADIR	Port A Direction	Read/write	Word	0000h	
04h	PADIR_L		Read/write	Byte	00h	
05h	PADIR_H		Read/write	Byte	00h	
06h	PAREN	Port A Resistor Enable	Read/write	Word	0000h	
06h	PAREN_L		Read/write	Byte	00h	
07h	PAREN_H		Read/write	Byte	00h	
0Ah	PASEL0	Port A Select 0	Read/write	Word	0000h	
0Ah	PASEL0_L		Read/write	Byte	00h	
0Bh	PASEL0_H		Read/write	Byte	00h	
0Ch	PASEL1	Port A Select 1	Read/write	Word	0000h	
0Ch	PASEL1_L		Read/write	Byte	00h	
0Dh	PASEL1_H		Read/write	Byte	00h	
16h	PASELC	Port A Complement Select	Read/write	Word	0000h	
16h	PASELC_L		Read/write	Byte	00h	
17h	PASELC_H		Read/write	Byte	00h	
18h	PAIES	Port A Interrupt Edge Select	Read/write	Word	undefined	
18h	PAIES_L		Read/write	Byte	undefined	
19h	PAIES_H		Read/write	Byte	undefined	
1Ah	PAIE	Port A Interrupt Enable	Read/write	Word	0000h	
1Ah	PAIE_L		Read/write	Byte	00h	
1Bh	PAIE_H		Read/write	Byte	00h	
1Ch	PAIFG	Port A Interrupt Flag	Read/write	Word	0000h	
1Ch	PAIFG_L		Read/write	Byte	00h	
1Dh	PAIFG_H		Read/write	Byte	00h	

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	PBIN	Port B Input	Read only	Word	undefined	
00h	PBIN_L		Read only	Byte	undefined	
01h	PBIN_H		Read only	Byte	undefined	
02h	PBOUT	Port B Output	Read/write	Word	undefined	
02h	PBOUT_L		Read/write	Byte	undefined	
03h	PBOUT_H		Read/write	Byte	undefined	
04h	PBDIR	Port B Direction	Read/write	Word	0000h	
04h	PBDIR_L		Read/write	Byte	00h	
05h	PBDIR_H		Read/write	Byte	00h	
06h	PBREN	Port B Resistor Enable	Read/write	Word	0000h	
06h	PBREN_L		Read/write	Byte	00h	
07h	PBREN_H		Read/write	Byte	00h	
0Ah	PBSEL0	Port B Select 0	Read/write	Word	0000h	
0Ah	PBSEL0_L		Read/write	Byte	00h	
0Bh	PBSEL0_H		Read/write	Byte	00h	
0Ch	PBSEL1	Port B Select 1	Read/write	Word	0000h	
0Ch	PBSEL1_L		Read/write	Byte	00h	
0Dh	PBSEL1_H		Read/write	Byte	00h	
16h	PBSELC	Port B Complement Select	Read/write	Word	0000h	
16h	PBSELC_L		Read/write	Byte	00h	
17h	PBSELC_H		Read/write	Byte	00h	
18h	PBIES	Port B Interrupt Edge Select	Read/write	Word	undefined	
18h	PBIES_L		Read/write	Byte	undefined	
19h	PBIES_H		Read/write	Byte	undefined	
1Ah	PBIE	Port B Interrupt Enable	Read/write	Word	0000h	
1Ah	PBIE_L		Read/write	Byte	00h	
1Bh	PBIE_H		Read/write	Byte	00h	
1Ch	PBIFG	Port B Interrupt Flag	Read/write	Word	0000h	
1Ch	PBIFG_L		Read/write	Byte	00h	
1Dh	PBIFG_H		Read/write	Byte	00h	

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	PCIN	Port C Input	Read only	Word	undefined	
00h	PCIN_L		Read only	Byte	undefined	
01h	PCIN_H		Read only	Byte	undefined	
02h	PCOUT	Port C Output	Read/write	Word	undefined	
02h	PCOUT_L		Read/write	Byte	undefined	
03h	PCOUT_H		Read/write	Byte	undefined	
04h	PCDIR	Port C Direction	Read/write	Word	0000h	
04h	PCDIR_L		Read/write	Byte	00h	
05h	PCDIR_H		Read/write	Byte	00h	
06h	PCREN	Port C Resistor Enable	Read/write	Word	0000h	
06h	PCREN_L		Read/write	Byte	00h	
07h	PCREN_H		Read/write	Byte	00h	
0Ah	PCSEL0	Port C Select 0	Read/write	Word	0000h	
0Ah	PCSEL0_L		Read/write	Byte	00h	
0Bh	PCSEL0_H		Read/write	Byte	00h	
0Ch	PCSEL1	Port C Select 1	Read/write	Word	0000h	
0Ch	PCSEL1_L		Read/write	Byte	00h	
0Dh	PCSEL1_H		Read/write	Byte	00h	
16h	PCSELC	Port C Complement Select	Read/write	Word	0000h	
16h	PCSELC_L		Read/write	Byte	00h	
17h	PCSELC_H		Read/write	Byte	00h	
18h	PCIES	Port C Interrupt Edge Select	Read/write	Word	undefined	
18h	PCIES_L		Read/write	Byte	undefined	
19h	PCIES_H		Read/write	Byte	undefined	
1Ah	PCIE	Port C Interrupt Enable	Read/write	Word	0000h	
1Ah	PCIE_L		Read/write	Byte	00h	
1Bh	PCIE_H		Read/write	Byte	00h	
1Ch	PCIFG	Port C Interrupt Flag	Read/write	Word	0000h	
1Ch	PCIFG_L		Read/write	Byte	00h	
1Dh	PCIFG_H		Read/write	Byte	00h	

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	PDIN	Port D Input	Read only	Word	undefined	
00h	PDIN_L		Read only	Byte	undefined	
01h	PDIN_H		Read only	Byte	undefined	
02h	PDOUT	Port D Output	Read/write	Word	undefined	
02h	PDOUT_L		Read/write	Byte	undefined	
03h	PDOUT_H		Read/write	Byte	undefined	
04h	PDDIR	Port D Direction	Read/write	Word	0000h	
04h	PDDIR_L		Read/write	Byte	00h	
05h	PDDIR_H		Read/write	Byte	00h	
06h	PDREN	Port D Resistor Enable	Read/write	Word	0000h	
06h	PDREN_L		Read/write	Byte	00h	
07h	PDREN_H		Read/write	Byte	00h	
0Ah	PDSEL0	Port D Select 0	Read/write	Word	0000h	
0Ah	PDSEL0_L		Read/write	Byte	00h	
0Bh	PDSEL0_H		Read/write	Byte	00h	
0Ch	PDSEL1	Port D Select 1	Read/write	Word	0000h	
0Ch	PDSEL1_L		Read/write	Byte	00h	
0Dh	PDSEL1_H		Read/write	Byte	00h	
16h	PDSELC	Port D Complement Select	Read/write	Word	0000h	
16h	PDSELC_L		Read/write	Byte	00h	
17h	PDSELC_H		Read/write	Byte	00h	
18h	PDIES	Port D Interrupt Edge Select	Read/write	Word	undefined	
18h	PDIES_L		Read/write	Byte	undefined	
19h	PDIES_H		Read/write	Byte	undefined	
1Ah	PDIE	Port D Interrupt Enable	Read/write	Word	0000h	
1Ah	PDIE_L		Read/write	Byte	00h	
1Bh	PDIE_H		Read/write	Byte	00h	
1Ch	PDIFG	Port D Interrupt Flag	Read/write	Word	0000h	
1Ch	PDIFG_L		Read/write	Byte	00h	
1Dh	PDIFG_H		Read/write	Byte	00h	

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	PEIN	Port E Input	Read only	Word	undefined	
00h	PEIN_L		Read only	Byte	undefined	
01h	PEIN_H		Read only	Byte	undefined	
02h	PEOUT	Port E Output	Read/write	Word	undefined	
02h	PEOUT_L		Read/write	Byte	undefined	
03h	PEOUT_H		Read/write	Byte	undefined	
04h	PEDIR	Port E Direction	Read/write	Word	0000h	
04h	PEDIR_L		Read/write	Byte	00h	
05h	PEDIR_H		Read/write	Byte	00h	
06h	PEREN	Port E Resistor Enable	Read/write	Word	0000h	
06h	PEREN_L		Read/write	Byte	00h	
07h	PEREN_H		Read/write	Byte	00h	
0Ah	PESEL0	Port E Select 0	Read/write	Word	0000h	
0Ah	PESEL0_L		Read/write	Byte	00h	
0Bh	PESEL0_H		Read/write	Byte	00h	
0Ch	PESEL1	Port E Select 1	Read/write	Word	0000h	
0Ch	PESEL1_L		Read/write	Byte	00h	
0Dh	PESEL1_H		Read/write	Byte	00h	
16h	PESELC	Port E Complement Select	Read/write	Word	0000h	
16h	PESELC_L		Read/write	Byte	00h	
17h	PESELC_H		Read/write	Byte	00h	
18h	PEIES	Port E Interrupt Edge Select	Read/write	Word	undefined	
18h	PEIES_L		Read/write	Byte	undefined	
19h	PEIES_H		Read/write	Byte	undefined	
1Ah	PEIE	Port E Interrupt Enable	Read/write	Word	0000h	
1Ah	PEIE_L		Read/write	Byte	00h	
1Bh	PEIE_H		Read/write	Byte	00h	
1Ch	PEIFG	Port E Interrupt Flag	Read/write	Word	0000h	
1Ch	PEIFG_L		Read/write	Byte	00h	
1Dh	PEIFG_H		Read/write	Byte	00h	

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	PFIN	Port F Input	Read only	Word	undefined	
00h	PFIN_L		Read only	Byte	undefined	
01h	PFIN_H		Read only	Byte	undefined	
02h	PFOUT	Port F Output	Read/write	Word	undefined	
02h	PFOUT_L		Read/write	Byte	undefined	
03h	PFOUT_H		Read/write	Byte	undefined	
04h	PFDIR	Port F Direction	Read/write	Word	0000h	
04h	PFDIR_L		Read/write	Byte	00h	
05h	PFDIR_H		Read/write	Byte	00h	
06h	PFREN	Port F Resistor Enable	Read/write	Word	0000h	
06h	PFREN_L		Read/write	Byte	00h	
07h	PFREN_H		Read/write	Byte	00h	
0Ah	PFSEL0	Port F Select 0	Read/write	Word	0000h	
0Ah	PFSEL0_L		Read/write	Byte	00h	
0Bh	PFSEL0_H		Read/write	Byte	00h	
0Ch	PFSEL1	Port F Select 1	Read/write	Word	0000h	
0Ch	PFSEL1_L		Read/write	Byte	00h	
0Dh	PFSEL1_H		Read/write	Byte	00h	
16h	PFSELC	Port F Complement Select	Read/write	Word	0000h	
16h	PFSELC_L		Read/write	Byte	00h	
17h	PFSELC_H		Read/write	Byte	00h	
18h	PFIES	Port F Interrupt Edge Select	Read/write	Word	undefined	
18h	PFIES_L		Read/write	Byte	undefined	
19h	PFIES_H		Read/write	Byte	undefined	
1Ah	PFIE	Port F Interrupt Enable	Read/write	Word	0000h	
1Ah	PFIE_L		Read/write	Byte	00h	
1Bh	PFIE_H		Read/write	Byte	00h	
1Ch	PFIFG	Port F Interrupt Flag	Read/write	Word	0000h	
1Ch	PFIFG_L		Read/write	Byte	00h	
1Dh	PFIFG_H		Read/write	Byte	00h	

Table 12-3. Digital I/O Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	PJIN	Port J Input	Read only	Word	undefined	
00h	PJIN_L		Read only	Byte	undefined	
01h	PJIN_H		Read only	Byte	undefined	
02h	PJOUT	Port J Output	Read/write	Word	undefined	
02h	PJOUT_L		Read/write	Byte	undefined	
03h	PJOUT_H		Read/write	Byte	undefined	
04h	PJDIR	Port J Direction	Read/write	Word	0000h	
04h	PJDIR_L		Read/write	Byte	00h	
05h	PJDIR_H		Read/write	Byte	00h	
06h	PJREN	Port J Resistor Enable	Read/write	Word	0000h	
06h	PJREN_L		Read/write	Byte	00h	
07h	PJREN_H		Read/write	Byte	00h	
0Ah	PJSEL0	Port J Select 0	Read/write	Word	0000h	
0Ah	PJSEL0_L		Read/write	Byte	00h	
0Bh	PJSEL0_H		Read/write	Byte	00h	
0Ch	PJSEL1	Port J Select 1	Read/write	Word	0000h	
0Ch	PJSEL1_L		Read/write	Byte	00h	
0Dh	PJSEL1_H		Read/write	Byte	00h	
16h	PJSELC	Port J Complement Select	Read/write	Word	0000h	
16h	PJSELC_L		Read/write	Byte	00h	
17h	PJSELC_H		Read/write	Byte	00h	

12.4.1 PxIV Register

Port x Interrupt Vector Register, x = 1 to 9 (see the device-specific data sheet to determine which ports support interrupts)

Figure 12-1. PxIV Register

15	14	13	12	11	10	9	8
PxIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
PxIV							
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Table 12-4. PxIV Register Description

Bit	Field	Type	Reset	Description
15-0	PxIV	R	0h	Port x interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Port x.0 interrupt; Interrupt Flag: PxIFG.0; Interrupt Priority: Highest 04h = Interrupt Source: Port x.1 interrupt; Interrupt Flag: PxIFG.1 06h = Interrupt Source: Port x.2 interrupt; Interrupt Flag: PxIFG.2 08h = Interrupt Source: Port x.3 interrupt; Interrupt Flag: PxIFG.3 0Ah = Interrupt Source: Port x.4 interrupt; Interrupt Flag: PxIFG.4 0Ch = Interrupt Source: Port x.5 interrupt; Interrupt Flag: PxIFG.5 0Eh = Interrupt Source: Port x.6 interrupt; Interrupt Flag: PxIFG.6 10h = Interrupt Source: Port x.7 interrupt; Interrupt Flag: PxIFG.7; Interrupt Priority: Lowest

12.4.2 PxIN Register

Port x Input Register

Figure 12-2. PxIN Register

7	6	5	4	3	2	1	0
PxIN							
r	r	r	r	r	r	r	r

Table 12-5. PxIN Register Description

Bit	Field	Type	Reset	Description
7-0	PxIN	R	Undefined	Port x input 0b = Input is low 1b = Input is high

12.4.3 PxOUT Register

Port x Output Register

Figure 12-3. PxOUT Register

7	6	5	4	3	2	1	0
PxOUT							
rw	rw	rw	rw	rw	rw	rw	rw

Table 12-6. PxOUT Register Description

Bit	Field	Type	Reset	Description
7-0	PxOUT	RW	Undefined	Port x output When I/O configured to output mode: 0b = Output is low. 1b = Output is high. When I/O configured to input mode and pullups/pulldowns enabled: 0b = Pulldown selected 1b = Pullup selected

12.4.4 PxDIR Register

Port x Direction Register

Figure 12-4. PxDIR Register

7	6	5	4	3	2	1	0
PxDIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-7. P1DIR Register Description

Bit	Field	Type	Reset	Description
7-0	PxDIR	RW	0h	Port x direction 0b = Port configured as input 1b = Port configured as output

12.4.5 PxREN Register

Port x Pullup or Pulldown Resistor Enable Register

Figure 12-5. PxREN Register

7	6	5	4	3	2	1	0
PxREN							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-8. PxREN Register Description

Bit	Field	Type	Reset	Description
7-0	PxREN	RW	0h	Port x pullup or pulldown resistor enable. When the port is configured as an input, setting this bit enables or disables the pullup or pulldown. 0b = Pullup or pulldown disabled 1b = Pullup or pulldown enabled

12.4.6 PxSEL0 Register

Port x Function Selection Register 0

Figure 12-6. PxSEL0 Register

7	6	5	4	3	2	1	0
PxSEL0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-9. PxSEL0 Register Description

Bit	Field	Type	Reset	Description
7-0	PxSEL0	RW	0h	Port function selection. Each bit corresponds to one channel on Port x. The values of each bit position in PxSEL1 and PxSEL0 are combined to specify the function. For example, if P1SEL1.5 = 1 and P1SEL0.5 = 0, then the secondary module function is selected for P1.5. See PxSEL1 for the definition of each value.

12.4.7 PxSEL1 Register

Port x Function Selection Register 1

Figure 12-7. PxSEL1 Register

7	6	5	4	3	2	1	0
PxSEL1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-10. PxSEL1 Register Description

Bit	Field	Type	Reset	Description
7-0	PxSEL1	RW	0h	Port function selection. Each bit corresponds to one channel on Port x. The values of each bit position in PxSEL1 and PxSEL0 are combined to specify the function. For example, if P1SEL1.5 = 1 and P1SEL0.5 = 0, then the secondary module function is selected for P1.5. 00b = General-purpose I/O is selected 01b = Primary module function is selected 10b = Secondary module function is selected 11b = Tertiary module function is selected

12.4.8 PxSELC Register

Port x Complement Selection

Figure 12-8. PxSELC Register

7	6	5	4	3	2	1	0
PxSELC							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-11. PxSELC Register Description

Bit	Field	Type	Reset	Description
7-0	PxSELC	RW	0h	Port selection complement. Each bit that is set in PxSELC complements the corresponding respective bit of both the PxSEL1 and PxSEL0 registers; that is, for each bit set in PxSELC, the corresponding bits in both PxSEL1 and PxSEL0 are both changed at the same time. Always reads as 0.

12.4.9 PxIES Register

Port x Interrupt Edge Select Register

Figure 12-9. PxIES Register

7	6	5	4	3	2	1	0
PxIES							
rw	rw	rw	rw	rw	rw	rw	rw

Table 12-12. PxIES Register Description

Bit	Field	Type	Reset	Description
7-0	PxIES	RW	Undefined	Port x interrupt edge select 0b = PxIFG flag is set with a low-to-high transition 1b = PxIFG flag is set with a high-to-low transition

12.4.10 PxIE Register

Port x Interrupt Enable Register

Figure 12-10. PxIE Register

7	6	5	4	3	2	1	0
PxIE							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-13. PxIE Register Description

Bit	Field	Type	Reset	Description
7-0	PxIE	RW	0h	Port x interrupt enable 0b = Corresponding port interrupt disabled 1b = Corresponding port interrupt enabled

12.4.11 PxIFG Register

Port x Interrupt Flag Register

Figure 12-11. PxIFG Register

7	6	5	4	3	2	1	0
PxIFG							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-14. PxIFG Register Description

Bit	Field	Type	Reset	Description
7-0	PxIFG	RW	Undefined	Port x interrupt flag 0b = No interrupt is pending. 1b = Interrupt is pending.

Capacitive Touch I/O

This chapter describes the functionality of the Capacitive Touch I/Os and related control.

Topic	Page
13.1 Capacitive Touch I/O Introduction	394
13.2 Capacitive Touch I/O Operation.....	395
13.3 CapTouch Registers	396

13.1 Capacitive Touch I/O Introduction

The Capacitive Touch I/O module allows implementation of a simple capacitive touch sense application. The module uses the integrated pullup and pulldown resistors and an external capacitor to form an oscillator by feeding back the inverted input voltage sensed by the input Schmitt triggers to the pullup and pulldown control. [Figure 13-1](#) shows the capacitive touch I/O principle

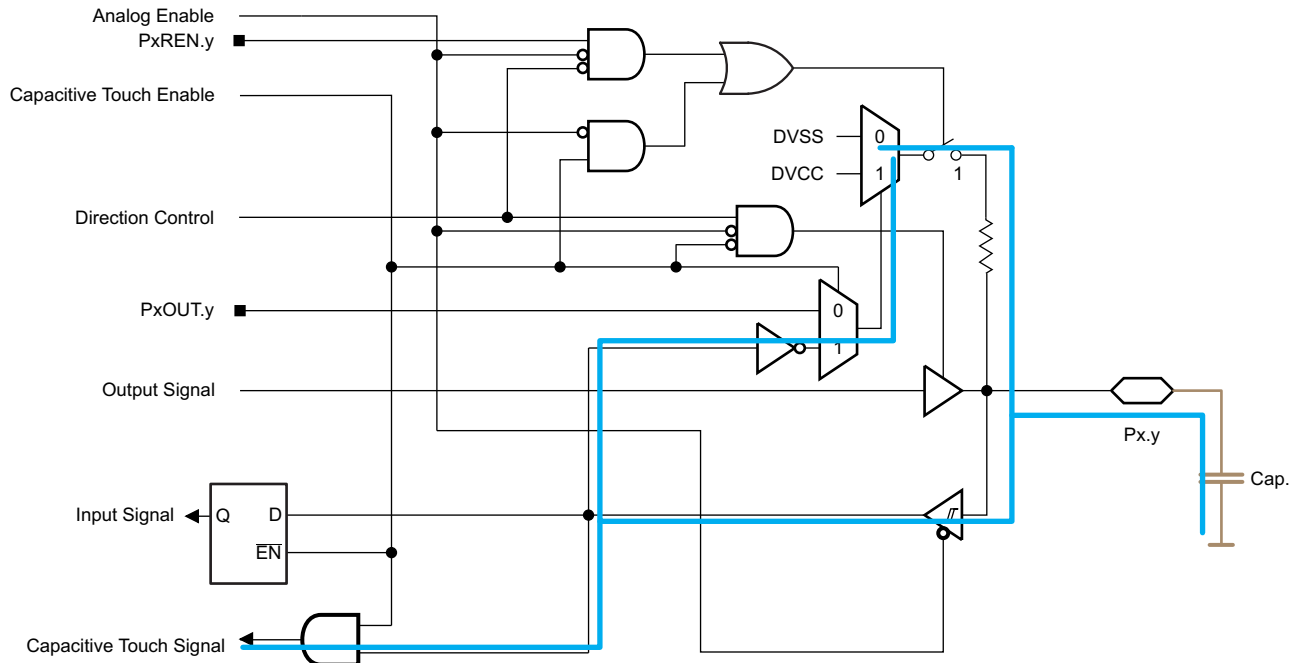


Figure 13-1. Capacitive Touch I/O Principle

Figure 13-2 shows the block diagram of the Capacitive Touch I/O module.

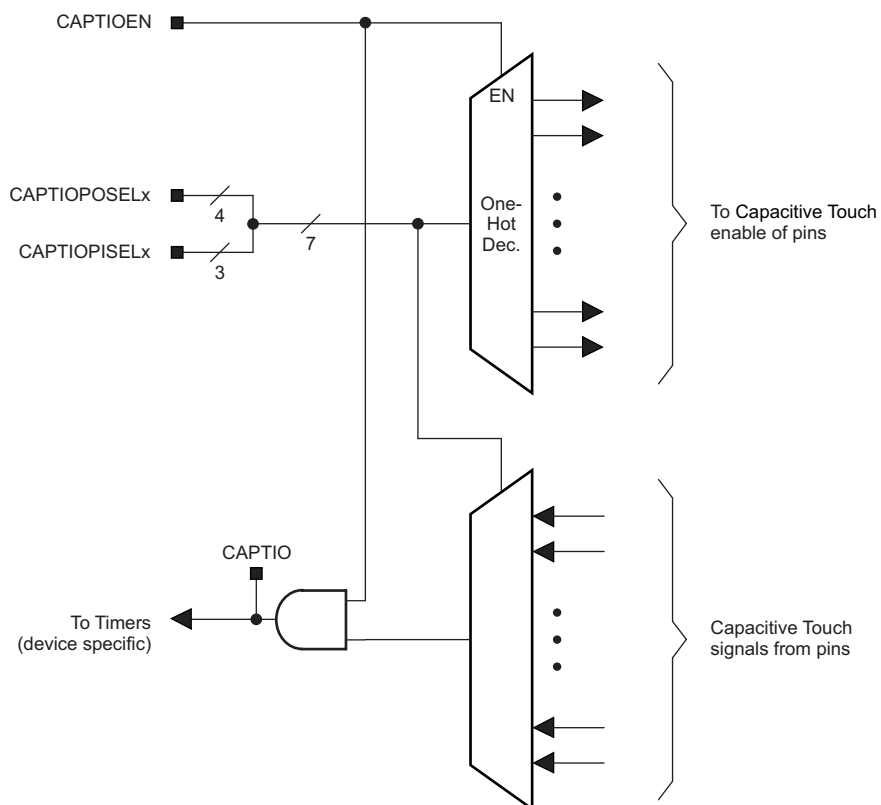


Figure 13-2. Capacitive Touch I/O Block Diagram

13.2 Capacitive Touch I/O Operation

Enable the Capacitive Touch I/O functionality with CAPTIOEN = 1 and select a port pin using CAPTIOPOSELx and CAPTIOISELx. The selected port pin is switched into the Capacitive Touch state, and the resulting oscillating signal is provided to be measured by a timer. The connected timers are device-specific (see the device-specific data sheet).

It is possible to scan to successive port pins by incrementing the low byte of the Capacitive Touch I/O control register CAPTIOCTL_L by 2.

13.3 CapTouch Registers

The Capacitive Touch I/O registers and their address offsets are listed in [Table 13-1](#). In a given device, multiple Capacitive Touch I/O registers might be available. The base address of each Capacitive Touch I/O module can be found in the device-specific data sheet.

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 13-1. CapTouch Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
0Eh	CAPTIOxCTL	Capacitive Touch I/O x control register	Read/write	Word	0000h	Section 13.3.1
0Eh	CAPTIOxCTL_L		Read/write	Byte	00h	
0Fh	CAPTIOxCTL_H		Read/write	Byte	00h	

13.3.1 CAPTIOxCTL Register (offset = 0Eh) [reset = 0000h]

Capacitive Touch I/O x Control Register

Figure 13-3. CAPTIOxCTL Register

15	14	13	12	11	10	9	8
Reserved						CAPTIO	CAPTIOEN
r0	r0	r0	r0	r0	r0	r-0	rw-0
7	6	5	4	3	2	1	0
CAPTIOPOSELx				CAPTIOISELx			Reserved
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r0

Table 13-2. CAPTIOxCTL Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved. Always reads 0.
9	CAPTIO	R	0h	Capacitive Touch I/O state. Reports the current state of the selected Capacitive Touch I/O. Reads 0, if Capacitive Touch I/O disabled. 0b = Current state 0 or Capacitive Touch I/O is disabled 1b = Current state 1
8	CAPTIOEN	RW	0h	Capacitive Touch I/O enable 0b = All Capacitive Touch I/Os are disabled. Signal toward timers is 0. 1b = Selected Capacitive Touch I/O is enabled
7-4	CAPTIOPOSELx	RW	0h	Capacitive Touch I/O port select. Selects port Px. Selecting a port pin that is not available on the device in use gives unpredictable results. 0000b = Px = PJ 0001b = Px = P1 0010b = Px = P2 0011b = Px = P3 0100b = Px = P4 0101b = Px = P5 0110b = Px = P6 0111b = Px = P7 1000b = Px = P8 1001b = Px = P9 1010b = Px = P10 1011b = Px = P11 1100b = Px = P12 1101b = Px = P13 1110b = Px = P14 1111b = Px = P15
3-1	CAPTIOISELx	RW	0h	Capacitive Touch I/O pin select. Selects the pin within selected port Px (see CAPTIOPOSELx). Selecting a port pin that is not available on the device in use gives unpredictable results. 000b = Px.0 001b = Px.1 010b = Px.2 011b = Px.3 100b = Px.4 101b = Px.5 110b = Px.6 111b = Px.7
0	Reserved	R	0h	Reserved. Always reads 0.

AES256 Accelerator

The AES256 accelerator module performs Advanced Encryption Standard (AES) encryption or decryption in hardware. It supports key lengths of 128 bits, 192 bits, and 256 bits. This chapter describes the AES256 accelerator.

Topic	Page
14.1 AES Accelerator Introduction.....	399
14.2 AES Accelerator Operation	400
14.3 AES Accelerator Registers	417

14.1 AES Accelerator Introduction

The AES accelerator module performs encryption and decryption of 128-bit data with 128-, 192-, or 256-bit keys according to the advanced encryption standard (AES) (FIPS PUB 197) in hardware.

The AES accelerator features are:

- AES encryption
 - 128 bit in 168 cycles
 - 192 bit in 204 cycles
 - 256 bit in 234 cycles
- AES decryption
 - 128 bit in 168 cycles
 - 192 bit in 206 cycles
 - 256 bit in 234 cycles
- On-the-fly key expansion for encryption and decryption
- Offline key generation for decryption
- Shadow register storing the initial key for all key lengths
- DMA support for ECB, CBC, OFB, and CFB cipher modes
- Byte and word access to key, input data, and output data
- AES ready interrupt flag

The AES accelerator block diagram is shown in [Figure 14-1](#).

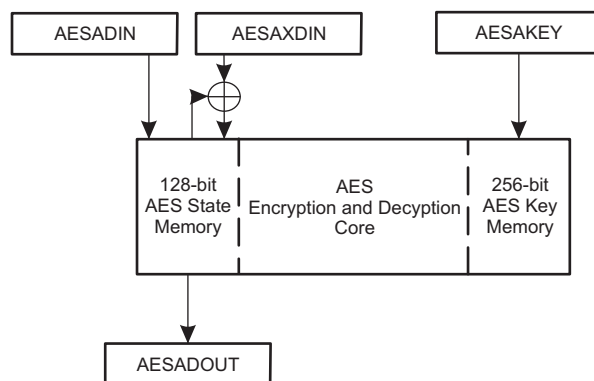


Figure 14-1. AES Accelerator Block Diagram

14.2 AES Accelerator Operation

The AES accelerator is configured with user software. The bit AESKLx determines if AES128, AES192, or AES256 is going to be performed. There are four different operation modes available, selectable by the AESOPx bits (see [Table 14-1](#)).

Table 14-1. AES Operation Modes Overview

AESOPx	AESKLx	Operation	Clock Cycles
00	00	AES128 encryption	168
	01	AES192 encryption	204
	10	AES256 encryption	234
01	00	AES128 decryption (with initial roundkey) is performed	215
	01	AES192 decryption (with initial roundkey) is performed	255
	10	AES256 decryption (with initial roundkey) is performed	292
10	00	AES128 encryption key schedule is performed	53
	01	AES192 encryption key schedule is performed	57
	10	AES256 encryption key schedule is performed	68
11	00	AES128 (with last roundkey) decryption is performed	168
	01	AES192 (with last roundkey) decryption is performed	206
	10	AES256 (with last roundkey) decryption is performed	234

The execution time of the different modes of operation is shown in [Table 14-1](#). While the AES module is operating, the AESBUSY bit is 1. As soon as the operation has finished, the AESRDYIFG bit is set.

Internally, the AES algorithm's operations are performed on a two-dimensional array of bytes called the State. The State consists of four rows of bytes, each containing four bytes, independently if AES128, AES192, or AES256 is performed. The input is assigned to the State array as shown in [Figure 14-2](#), with in[0] being the first data byte written into one of the AES accelerators input registers (AESADIN, AESAXDIN, and AESXIN). The encrypt or decrypt operations are then conducted on the State array, after which its final values can be read from the output with out[0] being the first data byte read from the AES accelerator data output register (AESADOUT).

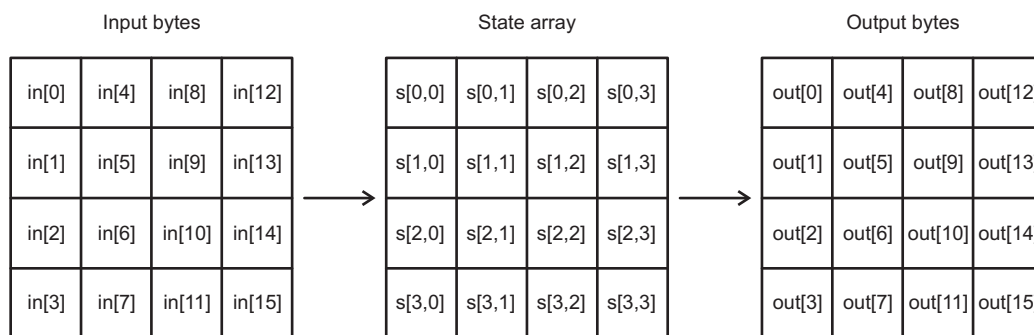


Figure 14-2. AES State Array Input and Output

If an encryption is to be performed, the initial state is called plaintext. If a decryption is to be performed, the initial state is called ciphertext.

The module allows word and byte access to all data registers—AESAKEy, AESADIN, AESAXDIN, AESAXIN, and AESADOUT. Word and byte access cannot be mixed while reading from or writing into one of the registers. However, it is possible to write one of the registers using byte access and another using word access.

NOTE: General Access Restrictions

While the AES accelerator is busy (AESBUSY = 1):

- AESADOUT always reads as zero.
- The AESDOUTCNTx counter, the AESDOUTRD flag, and the AESDINWR flag are reset.
- Any attempt to change AESOPx, AESKLx, AESDINWR, or AESKEYWR is ignored.
- Writing to AESAKEY, AESADIN, AESAXDIN, or AESAXIN aborts the current operation, the complete module is reset (except for the AESRDYIE, AESOPx, and AESKLx), and the AES error flag AESERRFG is set.

AESADIN, AESAXDIN, AESAXIN, and AESAKEY are write-only registers and always read as zero.

Writing data into AESADIN, AESAXDIN, or AESAXIN influences the content of the corresponding output data; for example, writing in[0] alters out[0], writing in[1] alters out[1], and so on. However, interleaved operation is possible; for example, first reading out[0] and then writing in[0], and continuing with reading out[1] and then writing in[1], and so on. This interleaved operation must be either byte or word access on in[x] and out[x].

Access Restriction With Cipher Modes Enabled (AESCMEN = 1)

When cipher modes are enabled (AESCMEN = 1) and a cipher block operation is being processed (AESBLKCNTx > 0), writes to the following bits are ignored, independent of AESBUSY: AESCMEN, AESCMx, AESKLx, AESOPx, and AESBLKCNTx. Writing to AESAKEY aborts the cipher block mode operation if AESBUSY = 1, and the complete module is reset (except for AESRDYIE, AESOPx, and AESKLx).

14.2.1 Load the Key (128-Bit, 192-Bit, or 256-Bit Key Length)

The key can be loaded by writing to the AESAKEY register or by setting AESKEYWR. Depending on the selected key length (AESKLx), a different number of bits must be loaded:

- If AESKLx = 00, the 128-bit key must be loaded using either 16 byte-writes or 8 word-writes to AESAKEY.
- If AESKLx = 01, the 192-bit key must be loaded using either 24 byte-writes or 12 word-writes to AESAKEY.
- If AESKLx = 10, the 256-bit key must be loaded using either 32 byte-writes or 16 word-writes to AESAKEY.

The key memory is reset after changing the AESKLx.

If a key was loaded previously without changing AESOPx, the AESKEYWR flag is cleared with the first write access to AESAKEY.

If the conversion is triggered without writing a new key, the last key is used. The key must always be written before writing the data.

14.2.2 Load the Data (128-Bit State)

The state can be loaded by writing to AESADIN, AESAXDIN, or AESAXIN with 16 byte writes or 8 word writes. Do not mix byte and word mode when writing the state. Writing to a mixture of AESADIN, AESAXDIN, and AESAXIN using the same byte or word data format is allowed. When the 16th byte or 8th word of the state is written, AESDINWR is set.

When writing to AESADIN, the corresponding byte or word of the state is overwritten. If AESADIN is used to write the last byte or word of the state, encryption or decryption starts automatically.

When writing to AESAXDIN, the corresponding byte or word is XORed with the current byte or word of the state. If AESAXDIN is used to write the last byte or word of the state, encryption or decryption starts automatically.

Writing to AESAXIN has the same behavior as writing to AESAXDIN: the corresponding byte or word is XORed with the current byte or word of the state; however, writing the last byte or word of the state using AESAXIN does not start encryption or decryption.

14.2.3 Read the Data (128-Bit State)

The state can be read if AESBUSY = 0 using 16 byte reads or 8 word reads from AESADOUT. When all 16 bytes are read, the AESDOUTRD flag indicates completion.

14.2.4 Trigger an Encryption or Decryption

The AES module's encrypt or decrypt operations are triggered if the state was completely written in the AESADIN or AESAXDIN registers. Alternatively, the bit AESDINWR can be set to trigger an operation if AESCMEN = 0.

14.2.5 Encryption

Figure 14-3 shows the encryption process with the cipher being a series of transformations that convert the plaintext written into the AESADIN register to a ciphertext that can be read from the AESADOUT register using the cipher key provided in the AESAKEY register.

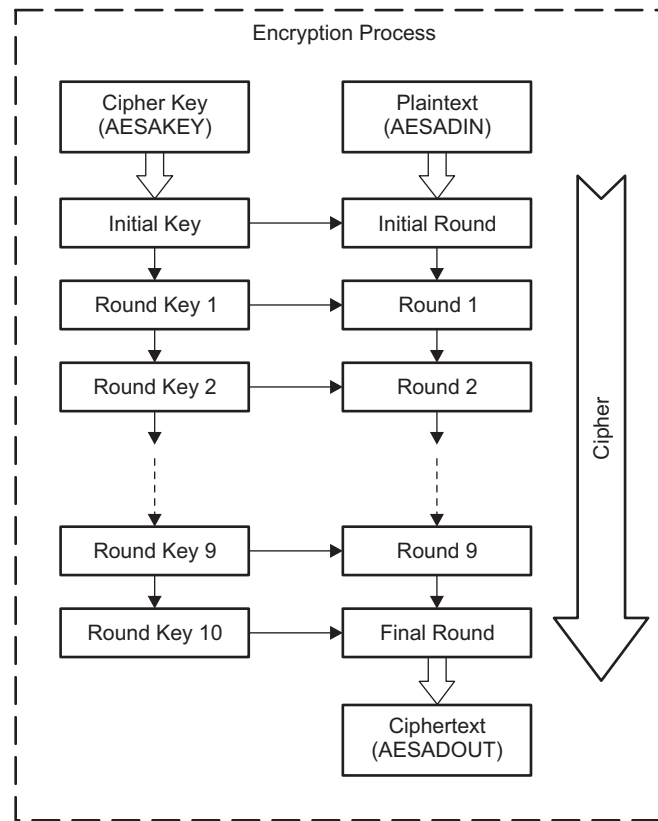


Figure 14-3. AES Encryption Process for 128-Bit Key

To perform encryption:

1. Set AESOPx = 00 to select encryption. Changing the AESOPx bits clears the AESKEYWR flag, and a new key must be loaded in the next step.
2. Load the key as described in [Section 14.2.1](#).
3. Load the state (data) as described in [Section 14.2.2](#). After the data is loaded, the AES module starts the encryption.
4. After the encryption is ready, the result can be read from AESADOUT as described in [Section 14.2.3](#).
5. To encrypt additional data with the same key loaded in step 2, write the new data into AESADIN after the results of the operation on the previous data were read from AESADOUT. When an additional 16 data bytes are written, the module automatically starts the encryption using the key loaded in step 2.

When implementing, for example, the output feedback (OFB) cipher block chaining mode, setting the AESDINWR flag triggers the next encryption, and the module starts the encryption using the output data from the previous encryption as the input data.

14.2.6 Decryption

Figure 14-4 shows the decryption process with the inverse cipher being a series of transformations that convert the ciphertext written into the AESADIN register to a plaintext that can be read from the AESADOUT register using the cipher key provided in the AESAKEY register.

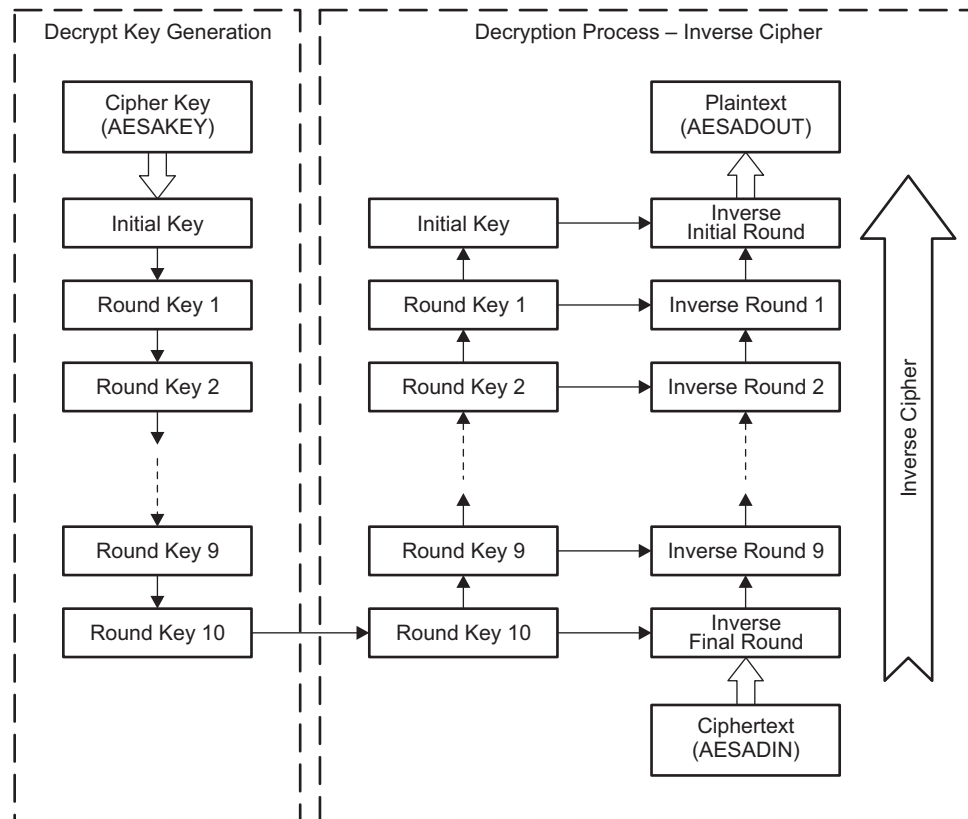


Figure 14-4. AES Decryption Process Using AESOPx = 01 for 128-Bit Key

The steps to perform decryption are:

1. Set AESOPx = 01 to select decryption using the same key used for encryption. Set AESOPx = 11 if the first-round key required for decryption (the last roundkey) is already generated and will be loaded in step 2. Changing the AESOPx bits clears the AESKEYWR flag, and a new key must be loaded in step 2.
2. Load the key according to [Section 14.2.1](#).
3. Load the state (data) according to [Section 14.2.2](#). After the data is loaded, the AES module starts the decryption.
4. After the decryption is ready, the result can be read from AESADOUT according to [Section 14.2.3](#).
5. If additional data should be decrypted with the same key loaded in step 2, new data can be written into AESADIN after the results of the operation on the previous data were read from AESADOUT. When additional 16 data bytes are written, the module automatically starts the decryption using the key loaded in step 2.

14.2.7 Decryption Key Generation

Figure 14-5 shows the decryption process with a pregenerated decryption key. In this case, the decryption key is calculated first with AESOPx = 10, then the precalculated key can be used together with the decryption operation AESOPx = 11.

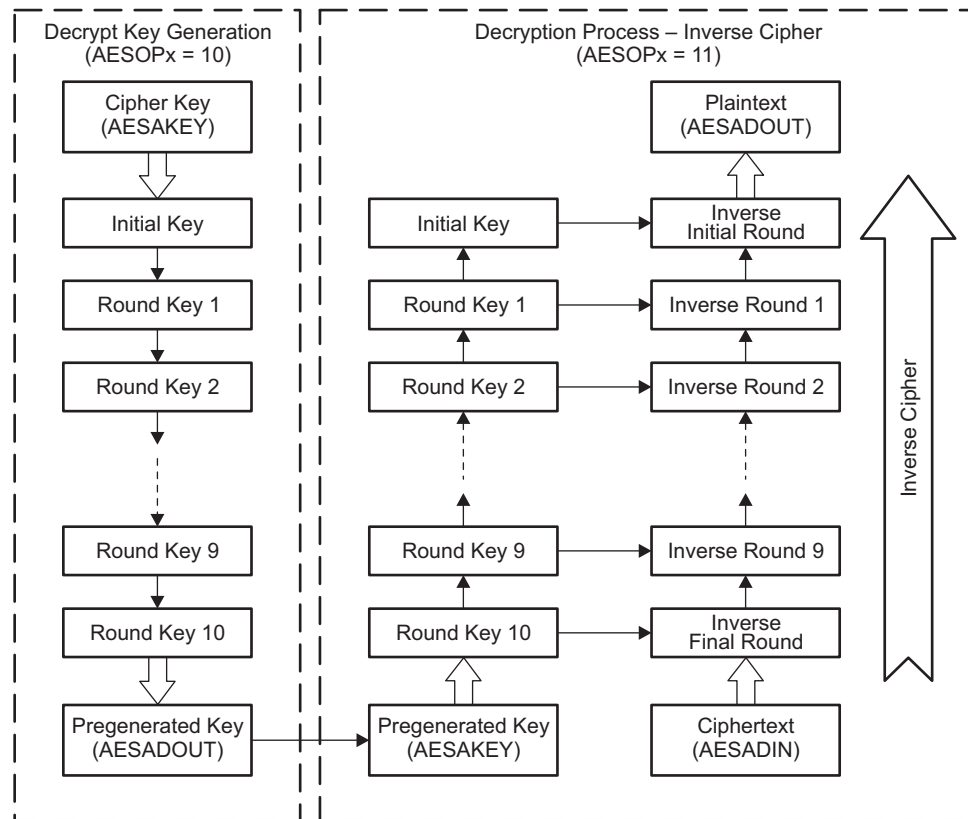


Figure 14-5. AES Decryption Process Using AESOPx = 10 and 11 for 128-bit Key

To generate the decryption key independent from the actual decryption:

1. Set AESOPx = 10 to select decryption key generation. Changing the AESOPx bits clears the AESKEYWR flag, and a new key must be loaded in step 2.
2. Load the key as described in [Section 14.2.1](#). The generation of the first round key required for decryption is started immediately.
3. While the AES module is performing the key generation, the AESBUSY bit is 1. 53 CPU clock cycles are required to complete the key generation for a 128-bit key (for other key lengths, see [Table 14-1](#)). After its completion, the AESRDYIFG is set, and the result can be read from AESADOUT. When all 16 bytes are read, the AESDOUTRD flag indicates completion.
The AESRDYIFG flag is cleared when reading AESADOUT or writing to AESKEY or AESADIN.
4. If data should be decrypted with the generated key, AESOPx must be set to 11. Then the generated key must be loaded or, if it was just generated with AESOPx = 10, set the AESKEYWR flag by software to indicate that the key is already valid.
5. See [Section 14.2.6](#) for instructions on the decryption steps, starting from step 3 (load data).

14.2.8 AES Key Buffer

The AES128, AES192, or AES256 algorithm operates not only on the state but also on the key. To avoid the need of reloading the key for each encryption or decryption, a key buffer is included in the AES accelerator.

14.2.9 Using the AES Accelerator With Low-Power Modes

The AES accelerator module provides automatic clock activation for MCLK for use with low-power modes. When the AES accelerator is busy, it automatically activates MCLK, regardless of the control-bit settings for the clock source. The clock remains active until the AES accelerator completes its operation.

The interrupt flag AESRDYIFG is reset after a PUC or with AESSWRST = 1. AESRDYIE is reset after a PUC but is not reset by AESSWRST = 1.

14.2.10 AES Accelerator Interrupts

The AESRDYIFG interrupt flag is set when the AES module completes the selected operation on the provided data. An interrupt request is generated if AESRDYIE and GIE are also set. AESRDYIFG is automatically reset if the AES interrupt is serviced, if AESADOUT is read, or if AESADIN or AESAKEY are written. AESRDYIFG is reset after a PUC or with AESSWRST = 1. AESRDYIE is reset after a PUC but is not reset by AESSWRST = 1.

14.2.11 DMA Operation and Implementing Block Cipher Modes

DMA operation, meaning the implementation of the cipher modes Electronic code book (ECB), Cipher block chaining (CBC), Output feedback (OFB), and Cipher feedback (CFB) using the DMA, supports easy and fast encryption and decryption of more than 128 bits.

When DMA cipher mode support is enabled by setting the AESCMEN bit, the AES256 module triggers 'AES trigger 0', 'AES trigger 1', and 'AES trigger 2' (also called 'AES trigger 0-2') in a certain order to execute different block cipher modes together with the DMA module.

For example, when using ECB encryption with AESCMEN = 1, 'AES trigger 0' is triggered eight times for DMA word access to read out AESADOUT, and then 'AES trigger 1' is triggered eight times to fill the next data into AESADIN. Because the AES modules generates a trigger for each word or byte the single transfer mode of the DMA must be used.

Table 14-2 shows the behavior of the 'AES trigger 0-2' for the different cipher modes selected by AESCMx.

Table 14-2. 'AES trigger 0-2' Operation When AESCMEN = 1

AESCMx	AESOPx	'AES trigger 0'	'AES trigger 1'	'AES trigger 2'
00 ECB	00 encryption	Set after encryption ready, set again until 128 bit are read from AESADOUT	Set to load the first block and set after 'AES trigger 0' was served the last time, set again until 128 bit are written to AESADIN	not set
	01 or 11 decryption	Set after decryption ready, set again until 128 bit are read from AESADOUT	Set to load the first block and set after 'AES trigger 0' was served the last time, set again until 128 bit are written to AESADIN	not set
01 CBC	00 encryption	Set after encryption ready, set again until 128 bit are read from AESADOUT	Set after 'AES trigger 0' was served the last time, set again until 128 bit are written to AESAXDIN	not set
	01 or 11 decryption	Set after decryption ready, set again until 128 bit are written to from AESAXIN	Set after 'AES trigger 0' was served the last time, set again until 128 bit are read from AESADOUT	Set after 'AES trigger 1' was served the last time, set again until 128 bit are written to AESADIN
10 OFB	00 encryption	Set after encryption ready, set again until 128 bit are written to AESAXIN	Set after 'AES trigger 0' was served the last time, set again until 128 bit are read from AESADOUT	Set after 'AES trigger 1' was served the last time, set again until 128 bit are written to AESAXDIN
	01 or 11 decryption	Set after decryption ready, set again until 128 bit are written to AESAXIN	Set after 'AES trigger 0' was served the last time, set again until 128 bit are read from AESADOUT	Set after 'AES trigger 1' was served the last time, set again until 128 bit are written to AESAXDIN
11 CFB	00 encryption	Set after encryption ready, set again until 128 bit are written to AESAXIN	Set after 'AES trigger 0' was served the last time, set again until 128 bit are read from AESADOUT	not set
	01 or 11 decryption	Set after decryption ready, set again until 128 bit are written to AESAXIN	Set after 'AES trigger 0' was served the last time, set again until 128 bit are read from AESADOUT	Set after 'AES trigger 1' was served the last time, set again until 128 bit are written to AESADIN

The retriggering of the 'AES trigger 0-2' until 128-bit of data are written or read from the corresponding register supports both byte and word access for writing and reading the state through the DMA.

For AESCMEN = 0, no DMA triggers are generated.

The following sections explain the configuration of the AES module for automatic cipher mode execution using DMA.

It is assumed that the key is written by software (or by a separate DMA transfer) before writing the first block to the AES state. The key shadow register always restores the original key, so that there is no need to reload it. The AESAKEY register should not be written after AESBLKCNTx is written to a non-zero value.

The number of blocks to be encrypted or decrypted must be programmed into the AESBLKCNTx bits before writing the first data. Writing a non-zero value into AESBLKCNTx starts the cipher mode sequence and, thus, AESBLKCNTx must be written after the DMA channels are configured.

Throughout these sections, the different DMA channels are called DMA_A, DMA_B, and so on. In the figures, these letters appear in dotted circles showing which operation is going to be executed by which DMA channel. The DMA counter must be loaded with a multiple of 8 for word mode or a multiple of 16 for byte mode and the single transfer mode of the DMA must be selected. The DMA priorities of DMA_A, DMA_B, and DMA_C do not play any role but static DMA priorities must be enabled. The DMA triggers must be configured as level-sensitive triggers.

14.2.11.1 Electronic Codebook (ECB) Mode

The electronic codebook block cipher mode is the most simple cipher mode. The data is divided into 128-bit blocks and encrypted and decrypted separately.

The disadvantage of the ECB is that the same 128-bit plaintext is always encrypted to the same ciphertext, whereas the other modes encrypt each block differently, partly dependent on the already executed encryptions.

14.2.11.1.1 ECB Encryption

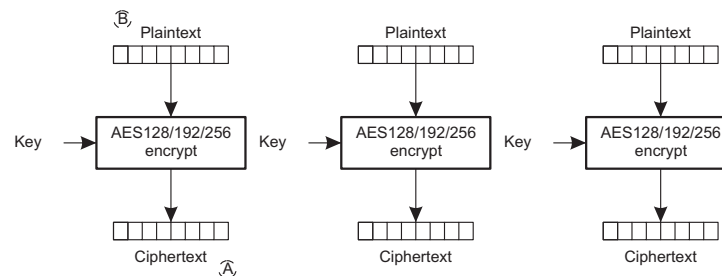


Figure 14-6. ECB Encryption

To implement the ECB encryption without CPU interaction, two DMA channels are needed. Static DMA priorities must be enabled. The DMA triggers must be configured as level-sensitive triggers.

Table 14-3. AES and DMA Configuration for ECB Encryption

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'
1	00	00	Read ciphertext from AESADOUT	Write plaintext to AESADIN, which also triggers the next encryption

The following pseudo code snippet shows the implementation of the ECB encryption in software:

```
ECB_Encryption(key, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Configure AES for block cipher:
        AESCMEN= 1; AESCMx= ECB; AESOPx= 00;

    Write key into AESAKEY;
    Setup DMA:
        DMA0: Triggered by AES trigger 0,
            Source: AESADOUT, Destination: ciphertext,
            Size: num_blocks*8 words, Single Transfer mode
        DMA1: Triggered by AES trigger 1,
            Source: plaintext, Destination: AESADIN,
            Size: num_blocks*8 words, Single Transfer mode

    Start encryption:
        AESBLKCNT= num_blocks;

    End of encryption: DMA0IFG=1
}
```

14.2.11.1.2 ECB Decryption

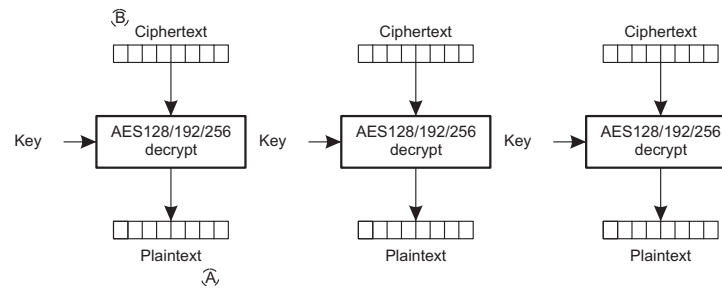


Figure 14-7. ECB Decryption

To implement the ECB decryption without CPU interaction, two DMA channels are needed. Static DMA priorities must be enabled. The DMA triggers must be configured as level-sensitive triggers.

Table 14-4. AES DMA Configuration for ECB Decryption

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'
1	00	01 or 11	Read plaintext from AESADOUT	Write ciphertext to AESADIN, which also triggers the next decryption

The following pseudo code snippet shows the implementation of the ECB decryption in software:

```
ECB_Decryption(key, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Generate Decrypt Key
    Configure AES:
        AESCMEN= 0; AESOPx= 10;
        Write key into AESAKEY;
        Wait until key generation completed.

    Configure AES for block cipher:
        AESCMEN= 1; AESCMx= ECB; AESOPx= 11;
        AESKEYWR= 1; // Use previously generated key

    Setup DMA:
        DMA0: Triggered by AES trigger 0,
              Source: AESADOUT, Destination: plaintext,
              Size: num_blocks*8 words, Single Transfer mode
        DMA1: Triggered by AES trigger 1,
              Source: ciphertext, Destination: AESADIN,
              Size: num_blocks*8 words, Single Transfer mode

    Start decryption:
        AESBLKCNT= num_blocks;

    End of decryption: DMA0IFG=1
}
```

14.2.11.2 Cipher Block Chaining (CBC) Mode

The cipher block chaining cipher mode always performs an XOR on the ciphertext of the previous block with the current block. Therefore, the encryption of each block depends not only on the key but also on the previous encryption.

14.2.11.2.1 CBC Encryption

For encryption, the initialization vector must be loaded by software (or by a separate DMA transfer) into AESXIN before the DMA can be enabled to write the first 16 bytes of the plaintext into AESAXDIN

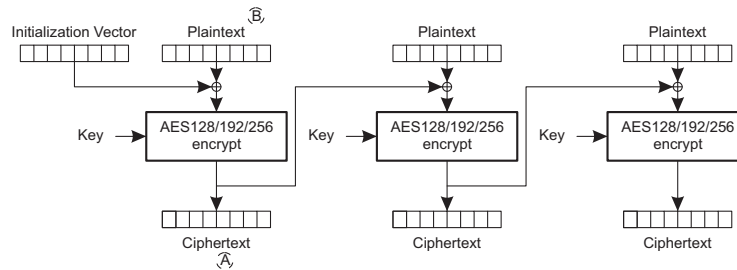


Figure 14-8. CBC Encryption

To implement the CBC encryption without CPU interaction, two DMA channels are needed. Static DMA priorities must be enabled. The DMA triggers must be configured as level-sensitive triggers.

Table 14-5. AES and DMA Configuration for CBC Encryption

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'
1	01	00	Read ciphertext from AESADOUT	Write plaintext to AESAXDIN, which also triggers the next encryption

The following pseudo code snippet shows the implementation of the CBC encryption in software:

```

CBC_Encryption(key, IV, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Reset AES Module (clears internal state memory):
        AESSWRST= 1;
    Configure AES for block cipher:
        AESCMEN= 1; AESCMx= CBC; AESOPx= 00;

    Write key into AESAKEY;
    Write IV into AESAXIN; // Does not trigger encryption.
                          // Assumes that state is reset (=> XORing with Zeros).

    Setup DMA:
        DMA0: Triggered by AES trigger 0,
              Source: AESADOUT, Destination: ciphertext,
              Size: num_blocks*8 words, Single Transfer mode
        DMA1: Triggered by AES trigger 1,
              Source: plaintext, Destination: AESAXDIN,
              Size: num_blocks*8 words, Single Transfer mode

    Start encryption:
        AESBLKCNT= num_blocks;

    End of encryption: DMA0IFG=1
}

```

14.2.11.2.2 CBC Decryption

For CBC decryption, the first block of data needs some special handling because the output must be XORed with the Initialization Vector. For that purpose, the DMA triggered by 'AES trigger 0' must be configured to read the data from the Initialization Vector first and then must be reconfigured to read from the ciphertext.

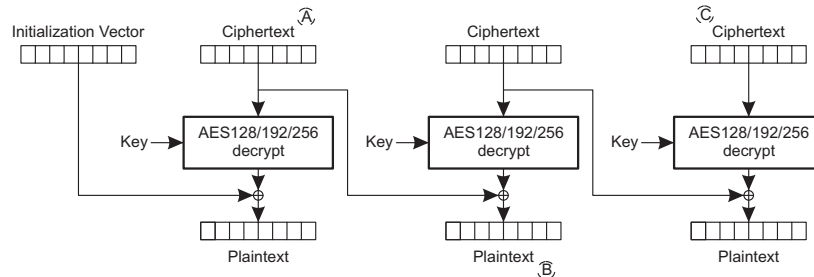


Figure 14-9. CBC Decryption

To implement the CBC decryption without CPU interaction, three DMA channels are needed. Static DMA priorities must be enabled. The DMA triggers must be configured as level-sensitive triggers.

Table 14-6. AES and DMA Configuration for CBC Decryption

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'	DMA_C Triggered by 'AES trigger 2'
1	01	01 or 11	Write the previous ciphertext block to AESAXIN	Read plaintext from AESADOUT	Write next plaintext to AESADIN, which also triggers the next decryption

The following pseudo code snippet shows the implementation of the CBC decryption in software:

```

CBC_Decryption(key, IV, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Generate Decrypt Key:
        Configure AES:
            AESCMEN= 0; AESOPx= 10;
            Write key into AESAKEY;
            Wait until key generation completed;

    Configure AES for block cipher:
        AESCMEN= 1; AESCMx= CBC; AESOPx= 11;
        AESKEYWR= 1; // Use previously generated key

    Setup DMA:
        DMA0: Triggered by AES trigger 0,
              Source: IV, Destination: AESAXIN,
              Size: 8 words, Single Transfer mode
        DMA1: Triggered by AES trigger 1,
              Source: AESADOUT, Destination: plaintext,
              Size: num_blocks*8 words, Single Transfer mode
        DMA2: Triggered by AES trigger 2,
              Source: ciphertext, Destination: AESADIN,
              Size: num_blocks*8 words, Single Transfer mode

    Start decryption:
        AESBLKCNT= num_blocks;

    Wait until first block is decrypted: DMA0IFG=1;

    Setup DMA0 for further blocks:
        DMA0: // Write previous cipher text into AES module
              Triggered by AES trigger 0,

```

```
        Source: ciphertext, Destination: AESAXIN,  
        Size: (num_blocks-1)*8 words, Single Transfer mode  
  
    End of decryption: DMA1IFG=1  
}
```

14.2.11.3 Output Feedback (OFB) Mode

The output feedback cipher mode continuously encrypts the initialization vector. The ciphertext is generated by XORing the corresponding plaintext with the encrypted initialization vector.

The initialization vector must be loaded by software (or by a separate DMA transfer).

14.2.11.3.1 OFB Encryption

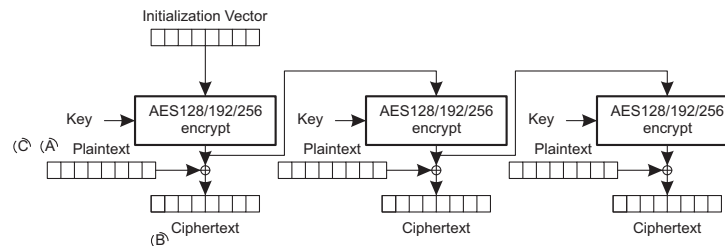


Figure 14-10. OFB Encryption

To implement the OFB encryption without CPU interaction, three DMA channels are needed. Static DMA priorities must be enabled. The DMA triggers must be configured as level-sensitive triggers.

Table 14-7. AES and DMA Configuration for OFB Encryption

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'	DMA_C Triggered by 'AES trigger 2'
1	10	00	Write the plaintext of the current block to AESAXIN	Read ciphertext from AESADOUT	Write the plaintext of the current block to AESAXDIN, which also triggers the next encryption

The following pseudo code snippet shows the implementation of the OFB encryption in software:

```

OFB_Encryption(Key, IV, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Reset AES Module (clears internal state memory):
        AESSWRST= 1;
    Configure AES:
        AESCMEN= 1; AESCMx= OFB; AESOPx= 00;

    Write Key into AESAKEY;
    Write IV into AESAXIN; // Does not trigger encryption.
                          // Assumes that state is reset (=> XORing with Zeros).

    Setup DMA:
        DMA0: Triggered by AES trigger 0,
              Source: plaintext, Destination: AESAXIN,
              Size: num_blocks*8 words, Single Transfer mode
        DMA1: Triggered by AES trigger 1,
              Source: AESADOUT, Destination: ciphertext,
              Size: num_blocks*8 words, Single Transfer mode
        DMA2: Triggered by AES trigger 2,
              Source: plaintext, Destination: AESAXDIN,
              Size: num_blocks*8 words, Single Transfer mode

    Start encryption:
        AESBLKCNT= num_blocks;
        Trigger encryption by setting AESDINWR= 1;

    End of encryption: DMA1IFG=1
}

```

14.2.11.3.2 OFB Decryption

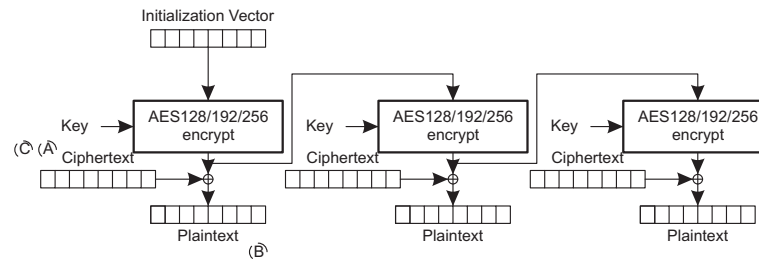


Figure 14-11. OFB Decryption

To implement the OFB decryption without CPU interaction, three DMA channels are needed. Static DMA priorities must be enabled. The DMA triggers must be configured as level-sensitive triggers.

Table 14-8. AES and DMA Configuration for OFB Decryption

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'	DMA_C Triggered by 'AES trigger 2'
1	10	01 or 11 ⁽¹⁾	Write the ciphertext of the current block to AESAXIN	Read plaintext from AESADOUT	Write the ciphertext of the current block to AESAXDIN, which also triggers the next encryption

⁽¹⁾ Note, in this cipher mode, the decryption also uses AES encryption on block level, thus the key used for decryption is identical with the key used for encryption; therefore, no decryption key generation is required.

The following pseudo code snippet shows the implementation of the OFB decryption in software:

```

OFB_Decryption(Key, IV, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Reset AES Module (clears internal state memory):
        AESSWRST= 1;
    Configure AES:
        AESCMEN= 1; AESCMx= OFB; AESOPx= 01;

    Write Key into AESAKEY;
    Write IV into AESAXIN; // Does not trigger encryption.
                          // Assumes that state is reset (=> XORing with Zeros).

    Setup DMA:
        DMA0: Triggered by AES trigger 0,
              Source: ciphertext, Destination: AESAXIN,
              Size: num_blocks*8 words, Single Transfer mode
        DMA1: Triggered by AES trigger 1,
              Source: AESADOUT, Destination: plaintext,
              Size: num_blocks*8 words, Single Transfer mode
        DMA2: Triggered by AES trigger 2,
              Source: ciphertext, Destination: AESAXDIN,
              Size: num_blocks*8 words, Single Transfer mode

    Start decryption:
        AESBLKCNT= num_blocks;
        Trigger decryption by setting AESDINWR= 1;

    End of decryption: DMA1IFG=1
}

```


14.2.11.4 Cipher Feedback (CFB) Mode

In the cipher feedback cipher mode, the plaintext of the new block is XORed to the last encryption result. The result of the encryption is the input for the new encryption.

The initialization vector must be loaded by software (or by a separate DMA transfer).

14.2.11.4.1 CFB Encryption

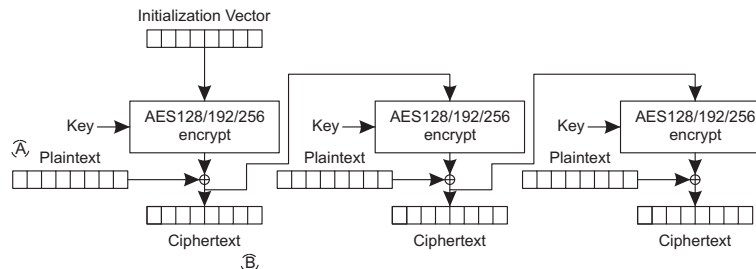


Figure 14-12. CFB Encryption

To implement the CFB encryption without CPU interaction, two DMA channels are needed. Static DMA priorities must be enabled. The DMA triggers must be configured as level-sensitive triggers.

Table 14-9. AES and DMA Configuration for CFB Encryption

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'
1	11	00	Write the plaintext of the current block to AESAXIN	Read the ciphertext from AESADOUT, which also triggers the next encryption

The following pseudo code snippet shows the implementation of the CFB encryption in software:

```
CFB_Encryption(Key, IV, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Reset AES Module (clears internal state memory):
        AESSWRST= 1;
    Configure AES:
        AESCMEN= 1; AESCMx= CFB; AESOPx= 00;

    Write Key into AESAKEY;
    Write IV into AESAXIN; // Does not trigger encryption.
                          // Assumes that state is reset (=> XORing with Zeros).

    Setup DMA:
        DMA0: Triggered by AES trigger 0,
              Source: plaintext, Destination: AESAXIN,
              Size: num_blocks*8 words, Single Transfer mode
        DMA1: Triggered by AES trigger 1,
              Source: AESADOUT, Destination: ciphertext,
              Size: num_blocks*8 words, Single Transfer mode

    Start encryption:
        AESBLKCNT= num_blocks;
        Trigger encryption by setting AESDINWR= 1;

    End of encryption: DMA1IFG=1
}
```

14.2.11.4.2 CFB Decryption

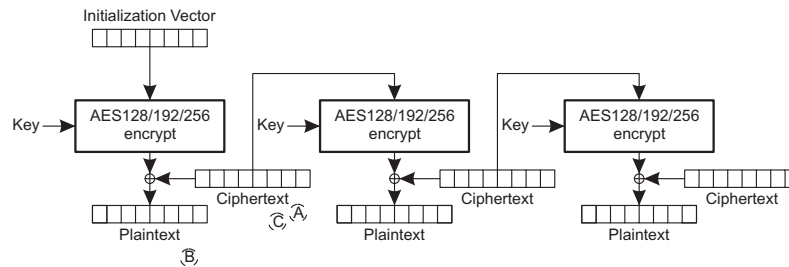


Figure 14-13. CFB Decryption

To implement the CFB decryption without CPU interaction, three DMA channels are needed. Static DMA priorities must be enabled. The DMA triggers must be configured as level-sensitive triggers.

Table 14-10. AES and DMA Configuration for CFB Decryption

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'	DMA_C Triggered by 'AES trigger 2'
1	11	01 or 11 ⁽¹⁾	Write the ciphertext of the current block to AESAXIN	Read the plaintext from AESADOUT	Write the ciphertext of the current block to AESADIN, which also triggers the next encryption

⁽¹⁾ Note, in this cipher mode, the decryption also uses AES encryption on block level thus the key used for decryption is identical with the key used for encryption; therefore, no decryption key generation is required.

The following pseudo code snippets shows the implementation of the CFB encryption and decryption in software:

```
CFB_Decryption(Key, IV, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Reset AES Module (clears internal state memory):
        AESSWRST= 1;
    Configure AES:
        AESCMEN= 1; AESCMx= CFB; AESOPx= 01;

    Write Key into AESAKEY;
    Write IV into AESAXIN; // Does not trigger encryption.
                          // Assumes that state is reset (=> XORing with Zeros).

    Setup DMA:
        DMA0: Triggered by AES trigger 0,
              Source: ciphertext, Destination: AESAXIN,
              Size: num_blocks*8 words, Single Transfer mode
        DMA1: Triggered by AES trigger 1,
              Source: AESADOUT, Destination: plaintext,
              Size: num_blocks*8 words, Single Transfer mode
        DMA2: Triggered by AES trigger 2,
              Source: ciphertext, Destination: AESADIN,
              Size: num_blocks*8 words, Single Transfer mode

    Start decryption:
        AESBLKCNT= num_blocks;
        Trigger decryption by setting AESDINWR= 1;

    End of decryption: DMA1IFG=1
}
```

14.3 AES Accelerator Registers

[Table 14-11](#) shows the memory-mapped registers for the AES256 module with their address offsets. See the device-specific data sheet for the base memory address of these registers. All other register offset addresses not listed in [Table 14-11](#) should be considered as reserved locations, and the register contents should not be modified.

Table 14-11. AES256 Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	AESACTL0	AES accelerator control register 0	Read/write	Word	00h	Section 14.3.1
02h	AESACTL1	AES accelerator control register 1	Read/write	Word	00h	Section 14.3.2
04h	AESASTAT	AES accelerator status register	Read only	Word	00h	Section 14.3.3
06h	AESAKEY	AES accelerator key register	Read/write	Word	00h	Section 14.3.4
08h	AESADIN	AES accelerator data in register	Write only	Word	00h	Section 14.3.5
0Ah	AESADOUT	AES accelerator data out register	Read/write	Word	00h	Section 14.3.6
0Ch	AESAXDIN	AES accelerator XORed data in register	Write only	Word	00h	Section 14.3.7
0Eh	AESAXIN	AES accelerator XORed data in register (no trigger)	Write only	Word	00h	Section 14.3.8

14.3.1 AESACTL0 Register

AES Accelerator Control Register 0

Figure 14-14. AESACTL0 Register

15	14	13	12	11	10	9	8
AESCMEN	Reserved		AESRDYIE	AESERRFG	Reserved		AESRDYIFG
rw-0	r0	r0	rw-0	rw-0	r0	r0	rw-0
7	6	5	4	3	2	1	0
AESSWRST	AESCMx		Reserved	AESKLx		AESOPx	
rw-0	r0	r0	r0	rw-0	rw-0	rw-0	rw-0

Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0.

Table 14-12. AESACTL0 Register Description

Bit	Field	Type	Reset	Description
15	AESCMEN	RW	0h	AESCMEN enables the support of the cipher modes ECB, CBC, OFB and CFB together with the DMA. Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0. 0 = No DMA triggers are generated 1 = DMA cipher mode support operation is enabled and the corresponding DMA triggers are generated.
14-13	Reserved	R	0h	Reserved
12	AESRDYIE	RW	0h	AES ready interrupt enable. AESRDYIE is not reset by AESSWRST = 1. 0b = Interrupt disabled 1b = Interrupt enabled
11	AESERRFG	RW	0h	AES error flag. AESAKEY or AESADIN were written while an AES operation was in progress. The bit must be cleared by software. 0b = No error 1b = Error occurred
10-9	Reserved	R	0h	Reserved
8	AESRDYIFG	RW	0h	AES ready interrupt flag. Set when the selected AES operation was completed and the result can be read from AESADOUT. Automatically cleared when AESADOUT is read or AESAKEY or AESADIN is written. 0b = No interrupt pending 1b = Interrupt pending
7	AESSWRST	RW	0h	AES software reset. Immediately resets the complete AES accelerator module even when busy except for the AESRDYIE, the AESKLx and the AESOPx bits. It also clears the (internal) state memory. The AESSWRST bit is automatically reset and is always read as zero. 0b = No reset 1b = Reset AES accelerator module
6-5	AESCMx	RW	0h	AES cipher mode select. These bits are ignored for AESCMEN = 0. Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0. 00b = ECB 01b = CBC 10b = OFB 11b = CFB
4	Reserved	R	0h	Reserved
3-2	AESKLx	RW	0h	AES key length. These bits define which of the 3 AES standards is performed. The AESKLx bits are not reset by AESSWRST = 1. Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0. 00b = AES128. The key size is 128 bit. 01b = AES192. The key size is 192 bit. 10b = AES256. The key size is 256 bit. 11b = Reserved

Table 14-12. AESACTL0 Register Description (continued)

Bit	Field	Type	Reset	Description
1-0	AESOPx	RW	0h	AES operation. The AESOPx bits are not reset by AESSWRST = 1. Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0. 00b = Encryption 01b = Decryption. The provided key is the same key used for encryption. 10b = Generate first round key required for decryption. 11b = Decryption. The provided key is the first round key required for decryption.

14.3.2 AESACTL1 Register

AES Accelerator Control Register 1

Figure 14-15. AESACTL1 Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
AESBLKCNTx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0.

Table 14-13. AESACTL1 Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved. Always reads 0.
7-0	AESBLKCNTx	RW	0h	Cipher Block Counter. Number of blocks to be encrypted or decrypted with block cipher modes enabled (AESCMEN = 1). Ignored if AESCMEN = 0. The block counter decrements with each performed encryption or decryption. Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0.

14.3.3 AESASTAT Register

AES Accelerator Status Register

Figure 14-16. AESASTAT Register

15	14	13	12	11	10	9	8
AESDOUTCNTx				AESDINCNTx			
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
AESKEYCNTx				AESDOUTRD	AESDINWR	AEKEYWR	AESBUSY
r-0	r-0	r-0	r-0	r-0	rw-0	rw-0	r-0

Table 14-14. AESASTAT Register Description

Bit	Field	Type	Reset	Description
15-12	AESDOUTCNTx	R	0h	Bytes read from AESADOUT. Reset when AESDOUTRD is reset. If AESDOUTCNTx = 0 and AESDOUTRD = 0, no bytes were read. If AESDOUTCNTx = 0 and AESDOUTRD = 1, all bytes were read.
11-8	AESDINCNTx	R	0h	Bytes written to AESADIN, AESAXDIN or AESAXIN. Reset when AESDINWR is reset. If AESDINCNTx = 0 and AESDINWR = 0, no bytes were written. If AESDINCNTx = 0 and AESDINWR = 1, all bytes were written.
7-4	AESKEYCNTx	R	0h	Bytes written to AESAKEY for AESKLx = 00, words written to AESAKEY if AESKLx = 01, 10, 11. Reset when AESKEYWR is reset. If AESKEYCNTx = 0 and AESKEYWR = 0, no bytes were written. If AESKEYCNTx = 0 and AESKEYWR = 1, all bytes were written.
3	AESDOUTRD	R	0h	All 16 bytes read from AESADOUT. AESDOUTRD is reset by PUC, AESSWRST, an error condition, changing AESOPx, changing AESKLx, when the AES accelerator is busy, and when the output data is read again. 0 = Not all bytes read 1 = All bytes read
2	AESDINWR	RW	0h	All 16 bytes written to AESADIN, AESAXDIN or AESAXIN. Changing its state by software also resets the AESDINCNTx bits. AESDINWR is reset by PUC, AESSWRST, an error condition, changing AESOPx, changing AESKLx, the start to (over)write the data, and when the AES accelerator is busy. Because it is reset when AESOPx or AESKLx is changed it can be set by software again to indicate that the current data is still valid. 0 = Not all bytes written 1 = All bytes written
1	AESKEYWR	RW	0h	All 16 bytes written to AESAKEY. This bit can be modified by software but it must not be reset by software (1→0) if AESCMEN=1. Changing its state by software also resets the AESKEYCNTx bits. AESKEYWR is reset by PUC, AESSWRST, an error condition, changing AESOPx, changing AESKLx, and the start to (over)write a new key. Because it is reset when AESOPx is changed it can be set by software again to indicate that the loaded key is still valid 0 = Not all bytes written 1 = All bytes written
0	AESBUSY	R	0h	AES accelerator module busy; encryption, decryption, or key generation in progress. 0 = Not busy 1 = Busy

14.3.4 AESAKEY Register

AES Accelerator Key Register

Figure 14-17. AESAKEY Register

15	14	13	12	11	10	9	8
AESKEY1x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0
7	6	5	4	3	2	1	0
AESKEY0x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0

Table 14-15. AESAKEY Register Description

Bit	Field	Type	Reset	Description
15-8	AESKEY1x	W	0h	AES key byte n+1 when AESAKEY is written as word. Do not use these bits for byte access. Do not mix word and byte access. Always reads as zero. The key is reset by PUC or by AESSWRST = 1.
7-0	AESKEY0x	W	0h	AES key byte n when AESAKEY is written as word. AES next key byte when AESAKEY_L is written as byte. Do not mix word and byte access. Always reads as zero. The key is reset by PUC or by AESSWRST = 1.

14.3.5 AESADIN Register

AES Accelerator Data In Register

Figure 14-18. AESADIN Register

15	14	13	12	11	10	9	8
AESDIN1x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0
7	6	5	4	3	2	1	0
AESDIN0x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0

Table 14-16. AESADIN Register Description

Bit	Field	Type	Reset	Description
15-8	AESDIN1x	W	0h	AES data in byte n+1 when AESADIN is written as word. Do not use these bits for byte access. Do not mix word and byte access. Always reads as zero.
7-0	AESDIN0x	W	0h	AES data in byte n when AESADIN is written as word. AES next data in byte when AESADIN_L is written as byte. Do not mix word and byte access. Always reads as zero.

14.3.6 AESADOUT Register

AES Accelerator Data Out Register

Figure 14-19. AESADOUT Register

15	14	13	12	11	10	9	8
AESDOUT1x							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
AESDOUT0x							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 14-17. AESADOUT Register Description

Bit	Field	Type	Reset	Description
15-8	AESDOUT1x	R	0h	AES data out byte n+1 when AESADOUT is read as word. Do not use these bits for byte access. Do not mix word and byte access.
7-0	AESDOUT0x	R	0h	AES data out byte n when AESADOUT is read as word. AES next data out byte when AESADOUT_L is read as byte. Do not mix word and byte access.

14.3.7 AESAXDIN Register

AES Accelerator XORed Data In Register

Figure 14-20. AESAXDIN Register

15	14	13	12	11	10	9	8
AESXDIN1x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0
7	6	5	4	3	2	1	0
AESXDIN0x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0

Table 14-18. AESAXDIN Register Description

Bit	Field	Type	Reset	Description
15-8	AESXDIN1x	W	0h	AES data in byte n+1 when AESAXDIN is written as word. Do not use these bits for byte access. Do not mix word and byte access. Always reads as zero.
7-0	AESXDIN0x	W	0h	AES data in byte n when AESAXDIN is written as word. AES next data in byte when AESAXDIN_L is written as byte. Do not mix word and byte access. Always reads as zero.

14.3.8 AESAXIN Register

AES Accelerator XORed Data In Register (No Trigger)

Figure 14-21. AESAXIN Register

15	14	13	12	11	10	9	8
AESXIN1x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0
7	6	5	4	3	2	1	0
AESXIN0x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0

Table 14-19. AESAXIN Register Description

Bit	Field	Type	Reset	Description
15-8	AESXIN1x	W	0h	AES data in byte n+1 when AESAXIN is written as word. Do not use these bits for byte access. Do not mix word and byte access. Always reads as zero.
7-0	AESXIN0x	W	0h	AES data in byte n when AESAXIN is written as word. AES next data in byte when AESAXIN_L is written as byte. Do not mix word and byte access. Always reads as zero.

CRC Module

The cyclic redundancy check (CRC) module provides a signature for a given data sequence. This chapter describes the operation and use of the CRC module.

Topic	Page
15.1 Cyclic Redundancy Check (CRC) Module Introduction	428
15.2 CRC Standard and Bit Order	428
15.3 CRC Checksum Generation	429
15.4 CRC Registers.....	432

15.1 Cyclic Redundancy Check (CRC) Module Introduction

The CRC module produces a signature for a given sequence of data values. The signature is generated through a feedback path from data bits 0, 4, 11, and 15 (see [Figure 15-1](#)). The CRC signature is based on the polynomial given in the CRC-CCITT-BR polynomial (see [Equation 10](#)).

$$f(x) = x^{16} + x^{12} + x^5 + 1 \quad (10)$$

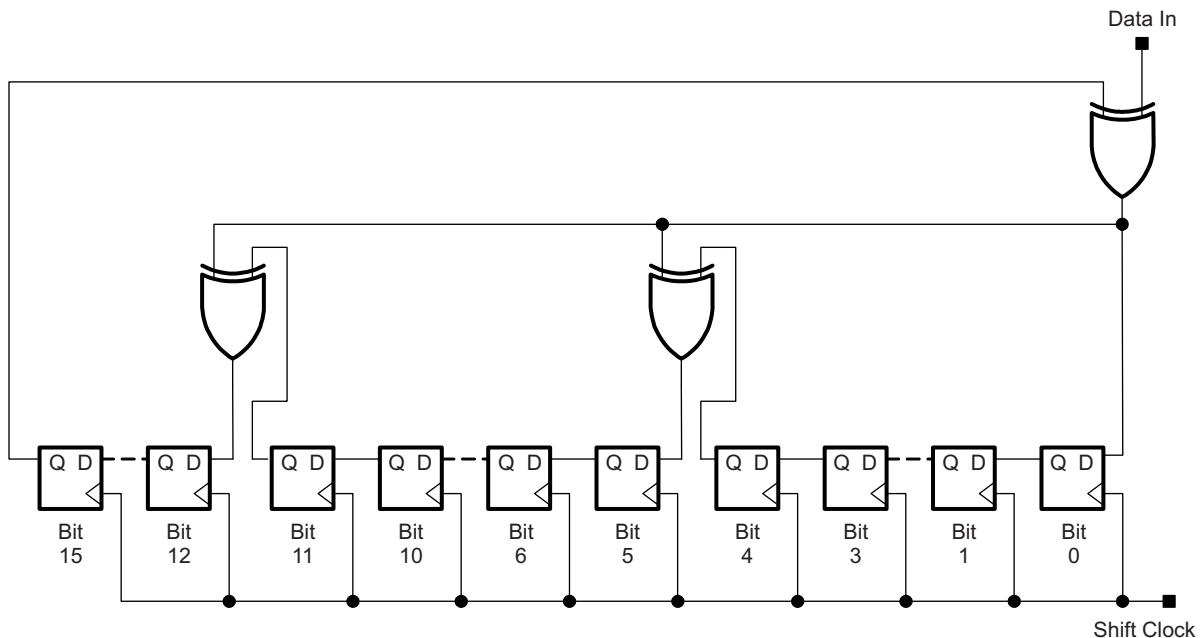


Figure 15-1. LFSR Implementation of CRC-CCITT Standard, Bit 0 is the MSB of the Result

Identical input data sequences result in identical signatures when the CRC is initialized with a fixed seed value, whereas different sequences of input data, in general, result in different signatures.

15.2 CRC Standard and Bit Order

The definitions of the various CRC standards were done in the era of main frame computers, and by convention bit 0 was treated as the MSB. Today, as in most microcontrollers such as the MSP430, bit 0 normally denotes the LSB. In [Figure 15-1](#), the bit convention shown is as given in the original standards (bit 0 is the MSB). The fact that bit 0 is treated for some as LSB, and for others as MSB, continues to cause confusion. The CRC16 module therefore provides a bit reversed register pair for CRC16 operations to support both conventions.

15.3 CRC Checksum Generation

The CRC generator is first initialized by writing a 16-bit word (seed) to the CRC Initialization and Result (CRCINIRES) register. Any data that should be included into the CRC calculation must be written to the CRC Data Input (CRCDI or CRCDIRB) register in the same order that the original CRC signature was calculated. The actual signature can be read from the CRCINIRES register to compare the computed checksum with the expected checksum.

Signature generation describes a method of how the result of a signature operation can be calculated. The calculated signature, which is computed by an external tool, is called checksum in the following text. The checksum is stored in the product's memory and is used to check the correctness of the CRC operation result.

15.3.1 CRC Implementation

To allow parallel processing of the CRC, the linear feedback shift register (LFSR) functionality is implemented with an XOR tree. This implementation shows the identical behavior as the LFSR approach after 8 bits of data are shifted in when the LSB is 'shifted' in first. The generation of a signature calculation has to be started by writing a seed to the CRCINIRES register to initialize the register. Software or hardware (for example, the DMA) can transfer data to the CRCDI or CRCDIRB register (for example, from memory). The value in CRCDI or CRCDIRB is then included into the signature, and the result is available in the signature result registers at the next read access (CRCINIRES and CRCRESR). The signature can be generated using word or byte data.

If a word data is processed, the lower byte at the even address is used at the first clock (MCLK) cycle. During the second clock cycle, the higher byte is processed. Thus, it takes two clock cycles to process word data, while it takes only one clock (MCLK) cycle to process byte data.

Data bytes written to CRCDIRB in word mode or the data byte in byte mode are bit-wise reversed before the CRC engine adds them to the signature. The bits among each byte are reversed. Data bytes written to CRCDI in word mode or the data byte in byte mode are not bit reversed before use by the CRC engine.

If the checksum itself (with reversed bit order) is included into the CRC operation (as data written to CRCDI or CRCDIRB), the result in the CRCINIRES and CRCRESR registers must be zero.

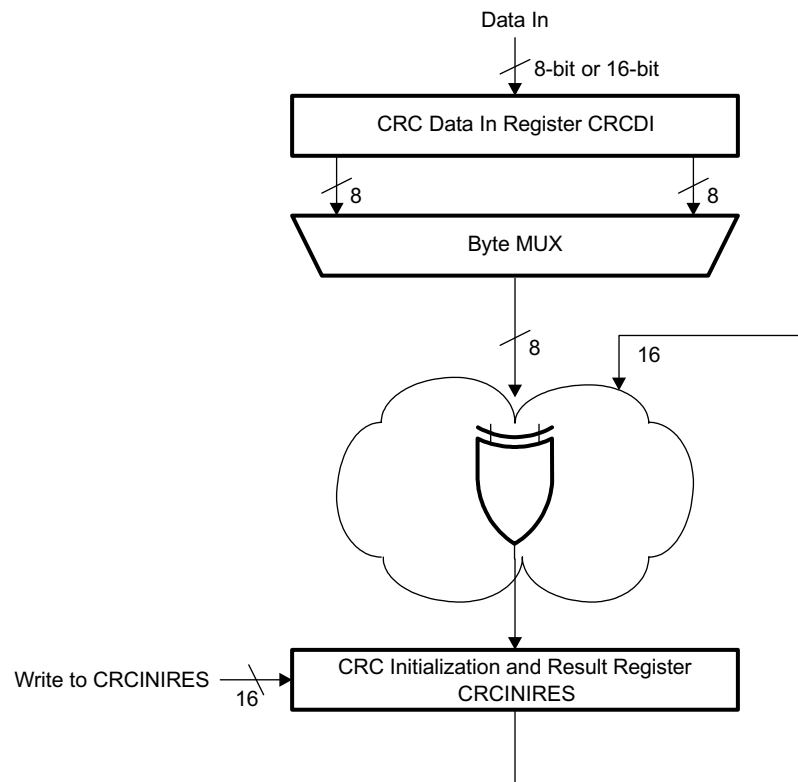


Figure 15-2. Implementation of CRC-CCITT Using the CRCDI and CRCINIRES Registers

15.3.2 Assembler Examples

[Example 15-1](#) demonstrates the operation of the on-chip CRC.

Example 15-1. General Assembler Example

```

...
PUSH    R4                ; Save registers
PUSH    R5
MOV     #StartAddress,R4  ; StartAddress < EndAddress
MOV     #EndAddress,R5
MOV     &INIT, &CRCINIRES ; INIT to CRCINIRES
L1 MOV   @R4+, &CRCDI      ; Item to Data In register
CMP     R5, R4             ; End address reached?
JLO     L1                 ; No
MOV     &Check_Sum, &CRCDI ; Yes, Include checksum
TST     &CRCINIRES         ; Result = 0?
JNZ     CRC_ERROR          ; No, CRCRES <> 0: error
...                          ; Yes, CRCRES=0:
                          ; information ok.
POP     R5                 ; Restore registers
POP     R4

```

The details of the implemented CRC algorithm are shown by the data sequences in [Example 15-2](#) using word or byte accesses and the CRC data-in as well as the CRC data-in reverse byte registers.

Example 15-2. Reference Data Sequence

```

...
mov    #0FFFFh,&CRCINIRES ; initialize CRC
mov.b  #00031h,&CRCDI_L   ; "1"
mov.b  #00032h,&CRCDI_L   ; "2"
mov.b  #00033h,&CRCDI_L   ; "3"
mov.b  #00034h,&CRCDI_L   ; "4"
mov.b  #00035h,&CRCDI_L   ; "5"
mov.b  #00036h,&CRCDI_L   ; "6"
mov.b  #00037h,&CRCDI_L   ; "7"
mov.b  #00038h,&CRCDI_L   ; "8"
mov.b  #00039h,&CRCDI_L   ; "9"

cmp     #089F6h,&CRCINIRES ; compare result
                        ; CRCRESR contains 06F91h
jeq     &Success           ; no error
br      &Error             ; to error handler

mov     #0FFFFh,&CRCINIRES ; initialize CRC
mov.w   #03231h,&CRCDI     ; "1" & "2"
mov.w   #03433h,&CRCDI     ; "3" & "4"
mov.w   #03635h,&CRCDI     ; "5" & "6"
mov.w   #03837h,&CRCDI     ; "7" & "8"
mov.b   #039h, &CRCDI_L    ; "9"

cmp     #089F6h,&CRCINIRES ; compare result
                        ; CRCRESR contains 06F91h
jeq     &Success           ; no error
br      &Error             ; to error handler

...
mov     #0FFFFh,&CRCINIRES ; initialize CRC
mov.b   #00031h,&CRCDIRB_L ; "1"
mov.b   #00032h,&CRCDIRB_L ; "2"
mov.b   #00033h,&CRCDIRB_L ; "3"
mov.b   #00034h,&CRCDIRB_L ; "4"
mov.b   #00035h,&CRCDIRB_L ; "5"
mov.b   #00036h,&CRCDIRB_L ; "6"
mov.b   #00037h,&CRCDIRB_L ; "7"
mov.b   #00038h,&CRCDIRB_L ; "8"
mov.b   #00039h,&CRCDIRB_L ; "9"

cmp     #029B1h,&CRCINIRES ; compare result
                        ; CRCRESR contains 08D94h
jeq     &Success           ; no error
br      &Error             ; to error handler

...
mov     #0FFFFh,&CRCINIRES ; initialize CRC
mov.w   #03231h,&CRCDIRB   ; "1" & "2"
mov.w   #03433h,&CRCDIRB   ; "3" & "4"
mov.w   #03635h,&CRCDIRB   ; "5" & "6"
mov.w   #03837h,&CRCDIRB   ; "7" & "8"
mov.b   #039h, &CRCDIRB_L ; "9"

cmp     #029B1h,&CRCINIRES ; compare result
                        ; CRCRESR contains 08D94h
jeq     &Success           ; no error
br      &Error             ; to error handler

```

15.4 CRC Registers

The CRC module registers are listed in [Table 15-1](#). The base address can be found in the device-specific data sheet. The address offset is given in [Table 15-1](#).

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 15-1. CRC Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	CRCDI	CRC Data In	Read/write	Word	0000h	Section 15.4.1
00h	CRCDI_L		Read/write	Byte	00h	
01h	CRCDI_H		Read/write	Byte	00h	
02h	CRCDIRB	CRC Data In Reverse Byte	Read/write	Word	0000h	Section 15.4.2
02h	CRCDIRB_L		Read/write	Byte	00h	
03h	CRCDIRB_H		Read/write	Byte	00h	
04h	CRCINIRES	CRC Initialization and Result	Read/write	Word	FFFFh	Section 15.4.3
04h	CRCINIRES_L		Read/write	Byte	FFh	
05h	CRCINIRES_H		Read/write	Byte	FFh	
06h	CRCRESR	CRC Result Reverse	Read only	Word	FFFFh	Section 15.4.4
06h	CRCRESR_L		Read/write	Byte	FFh	
07h	CRCRESR_H		Read/write	Byte	FFh	

15.4.1 CRCDI Register

CRC Data In Register

Figure 15-3. CRCDI Register

15	14	13	12	11	10	9	8
CRCDI							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CRCDI							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 15-2. CRCDI Register Description

Bit	Field	Type	Reset	Description
15-0	CRCDI	RW	0h	CRC data in. Data written to the CRCDI register is included to the present signature in the CRCNIREs register according to the CRC-CCITT standard.

15.4.2 CRC DIRB Register

CRC Data In Reverse Register

Figure 15-4. CRC DIRB Register

15	14	13	12	11	10	9	8
CRC DIRB							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CRC DIRB							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 15-3. CRC DIRB Register Description

Bit	Field	Type	Reset	Description
15-0	CRC DIRB	RW	0h	CRC data in reverse byte. Data written to the CRC DIRB register is included to the present signature in the CRCNIREs and CRCRESR registers according to the CRC-CCITT standard. Reading the register returns the register CRCDI content.

15.4.3 CRCINIRES Register

CRC Initialization and Result Register

Figure 15-5. CRCINIRES Register

15	14	13	12	11	10	9	8
CRCINIRES							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
7	6	5	4	3	2	1	0
CRCINIRES							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

Table 15-4. CRCINIRES Register Description

Bit	Field	Type	Reset	Description
15-0	CRCINIRES	RW	FFFFh	CRC initialization and result. This register holds the current CRC result (according to the CRC-CCITT standard). Writing to this register initializes the CRC calculation with the value written to it. The value just written can be read from CRCINIRES register.

15.4.4 CRCRESR Register

CRC Reverse Result Register

Figure 15-6. CRCRESR Register

15	14	13	12	11	10	9	8
CRCRESR							
r-1	r-1	r-1	r-1	r-1	r-1	r-1	r-1
7	6	5	4	3	2	1	0
CRCRESR							
r-1	r-1	r-1	r-1	r-1	r-1	r-1	r-1

Table 15-5. CRCRESR Register Description

Bit	Field	Type	Reset	Description
15-0	CRCRESR	R	FFFFh	CRC reverse result. This register holds the current CRC result (according to the CRC-CCITT standard). The order of bits is reverse (for example, CRCINIRES[15] = CRCRESR[0]) to the order of bits in the CRCINIRES register (see example code).

CRC32 Module

The 16-bit or 32-bit cyclic redundancy check (CRC32) module provides a signature for a given data sequence. This chapter describes the operation and use of the CRC32 module.

Topic	Page
16.1 Cyclic Redundancy Check (CRC32) Module Introduction	436
16.2 CRC Checksum Generation	436
16.3 CRC32 Register Descriptions	439

16.1 Cyclic Redundancy Check (CRC32) Module Introduction

The CRC module produces signatures for a given sequences of data values. These signatures are defined bit serial in various standard specifications. For CRC16-CCITT, a feedback path from data bits 4, 11, and 15 is generated (see Figure 16-1). This CRC signature is based on the polynomial given in the CRC-CCITT with $f(x)=x^{15}+x^{12}+x^5+1$.

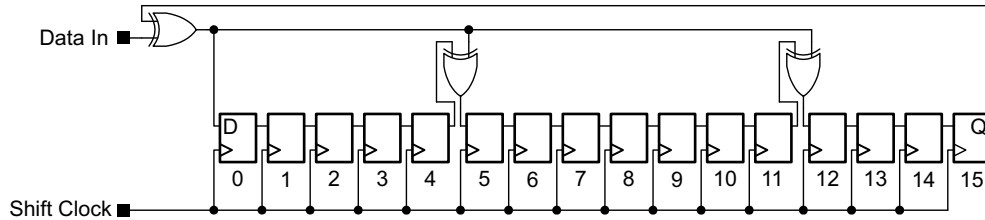


Figure 16-1. LFSR Implementation of CRC-CCITT as Defined in Standard (Bit 0 is MSB)

For CRC32-ISO3309 a feedback path from data bits 0, 1, 3, 4, 6, 7, 9, 10, 11, 15, 21, 22, 26 and 31 is generated (see Figure 16-2). This CRC signature is based on the polynomial given in the CRC32-ISO3309 with $f(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$.

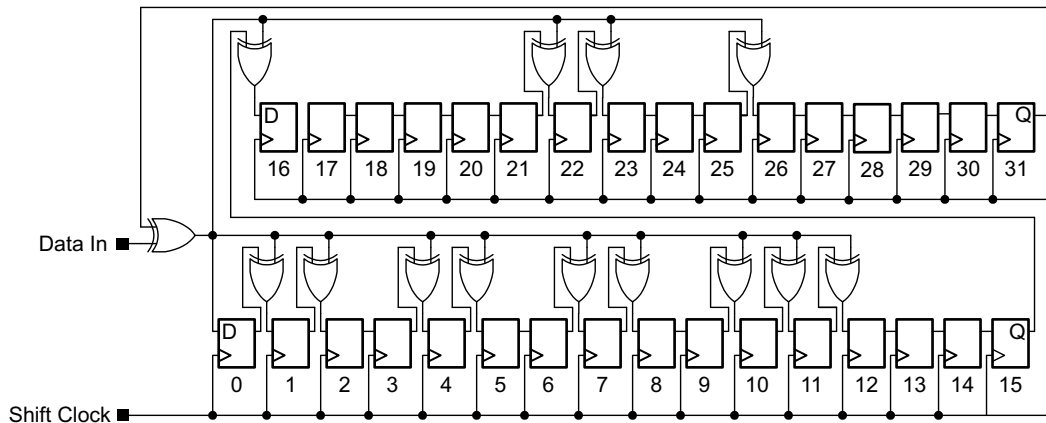


Figure 16-2. LFSR Implementation of CRC32-ISO3309 as Defined in Standard (Bit 0 is MSB)

Identical input data sequences result in identical signatures when the CRC is initialized with a fixed seed value. Different sequences of input data, in general, result in different signatures for a given CRC function.

The CRC32 module supports 16-bit and 32-bit CRC generation. They are independent from each other and supported by two dedicated register sets.

16.2 CRC Checksum Generation

The CRC generators are initialized by writing the "seed" to the CRC Initialization and Result (CRC16INIRES or CRC32INIRES) registers. Any data that should be included in the CRC calculation must be written to the CRC Data Input (CRC16DI or CRC32DI) registers in the same order that the original CRC signatures were calculated. The actual signature can be read from the CRC16INIRES or CRC32INIRES registers to compare the computed checksum with the expected checksum. Signature generation describes a method of how the result of a signature operation can be calculated. The calculated signature, which is computed by an external tool, is called checksum in the following text. The checksum is stored in the product's memory and is used to check the correctness of the CRC operation result.

16.2.1 CRC Standard and Bit Order

The various CRC standards were defined in the era of main frame computers. At that time, Bit 0 was treated as the MSB. Now, Bit 0 is typically the LSB (as the value of Bit $N = 2N$). In [Figure 16-1](#) and [Figure 16-2](#), the bit references are used as given in the original standards. The MSP430 microcontrollers treat Bit 0 as the LSB, as is typical in modern CPUs and MCUs.

This sometimes causes confusion, because Bit 0 has been treated as the LSB in some cases and as the MSB in other cases. Therefore, the CRC32 module provides a bit-reversed register pair for CRC16 and CRC32 operations to support both conventions.

16.2.2 CRC Implementation

To allow faster processing of the CRC, the linear feedback shift register (LFSR) functionality is implemented with a set of XOR trees. This implementation shows the identical behavior as the LFSR approach. After a set of 8, 16, or 32 bits is provided to the CRC32 module by writing to the CRC16DI or CRC32DI registers, a calculation for the whole set of input bits is performed. For this calculation, the CPU or the DMA can write to the memory-mapped data input registers. After the last value is written to CRC16DI or CRC32DIRB, the signature can be read from the CRC16INIRES or CRC32INIRES registers. The CRC16 and CRC32 generators accept byte- and word-wide access to the input registers CRC16DI and CRC32DI.

For bit-reversed conventions, write the data bytes to the CRC16DIRB or CRC32DIRB register.

Initialization is done by writing to the CRC, and CRC engine adds them to the signature. The bits among each byte are reversed. Data bytes written to CRCDI in word mode or the data byte in byte mode are not bit reversed before use by the CRC engine. If the checksum itself (with reversed bit order) is included in the CRC operation (as data written to CRCDI or CRCDIRB), the result in the CRCINIRES and CRCRESR register must be zero.

16.2.3 Assembler Examples

16.2.3.1 General Assembler Example

This example demonstrates the operation of the on-chip CRC.

```

        PUSH    R4                ; Save registers
        PUSH    R5
        MOV     #StartAddress,R4  ; StartAddress < EndAddress
        MOV     #EndAddress,R5
        MOV     &INIT,&CRCINIRES  ; INIT to CRCINIRES
L1:     MOV     @R4+,&CRCDI        ; Item to Data In register
        CMP     R5,R4             ; End address reached?
        JLO     L1               ; No
        MOV     &Check_Sum,&CRCDI ; Yes, Include checksum
        TST     &CRCINIRES        ; Result = 0?
        JNZ     CRC_ERROR        ; No, CRCRES <> 0: error
        ...                     ; Yes, CRCRES=0:
                                ; information ok.
        POP     R5                ; Restore registers
        POP     R4

```

16.2.3.2 Reference Data Sequence

The details of the implemented CRC algorithm is shown by the following data sequences using word or byte accesses and the CRC data-in as well as the CRC data-in reverse byte registers:

```

mov #0FFFFh,&CRCINIRES ; initialize CRC
mov.b #00031h,&CRCDI_L ; "1"
mov.b #00032h,&CRCDI_L ; "2"
mov.b #00033h,&CRCDI_L ; "3"
mov.b #00034h,&CRCDI_L ; "4"
mov.b #00035h,&CRCDI_L ; "5"
mov.b #00036h,&CRCDI_L ; "6"
mov.b #00037h,&CRCDI_L ; "7"
mov.b #00038h,&CRCDI_L ; "8"
mov.b #00039h,&CRCDI_L ; "9"
cmp #089F6h,&CRCINIRES ; compare result
; CRCRESR contains 06F91h
jeq Success ; no error
br &Error ; to error handler
...
mov #0FFFFh,&CRCINIRES ; initialize CRC
mov.w #03231h,&CRCDI ; "1" & "2"
mov.w #03433h,&CRCDI ; "3" & "4"
mov.w #03635h,&CRCDI ; "5" & "6"
mov.w #03837h,&CRCDI ; "7" & "8"
mov.b #039h, &CRCDI_L ; "9"
cmp #089F6h,&CRCINIRES ; compare result
; CRCRESR contains 06F91h
jeq Success ; no error
br &Error ; to error handler
...
mov #0FFFFh,&CRCINIRES ; initialize CRC
mov.b #00031h,&CRCDIRB_L ; "1"
mov.b #00032h,&CRCDIRB_L ; "2"
mov.b #00033h,&CRCDIRB_L ; "3"
mov.b #00034h,&CRCDIRB_L ; "4"
mov.b #00035h,&CRCDIRB_L ; "5"
mov.b #00036h,&CRCDIRB_L ; "6"
mov.b #00037h,&CRCDIRB_L ; "7"
mov.b #00038h,&CRCDIRB_L ; "8"
mov.b #00039h,&CRCDIRB_L ; "9"
cmp #029B1h,&CRCINIRES ; compare result
; CRCRESR contains 08D94h
jeq Success ; no error
br &Error ; to error handler
...
mov #0FFFFh,&CRCINIRES ; initialize CRC
mov.w #03231h,&CRCDIRB ; "1" & "2"
mov.w #03433h,&CRCDIRB ; "3" & "4"
mov.w #03635h,&CRCDIRB ; "5" & "6"
mov.w #03837h,&CRCDIRB ; "7" & "8"
mov.b #039h, &CRCDIRB_L ; "9"
cmp #029B1h,&CRCINIRES ; compare result
; CRCRESR contains 08D94h
jeq Success ; no error
br &Error ; to error handler
...

```


16.3 CRC32 Register Descriptions

16.3.1 CRC32 Registers

The CRC32 module registers with their address offsets are shown in [Table 16-1](#). The base address for the CRC32 module registers can be found in the device-specific data sheet.

Table 16-1. CRC32 Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
0002h	CRC32DIW1	CRC32 Data Input	read/write	word	0000h	Section 16.3.1.2
0000h	CRC32DIW0			word	0000h	Section 16.3.1.1
0000h	CRC32DIB0			byte	00h	
0004h	CRC32DIRBW1	CRC32 Data In Reverse	read/write	word	0000h	Section 16.3.1.4
0006h	CRC32DIRBW0			word	0000h	Section 16.3.1.3
0007h	CRC32DIRBB0			byte	00h	
000Ah	CRC32INIRESW1	CRC32 Initialization and Result	read/write	word	FFFFh	Section 16.3.1.6
0008h	CRC32INIRESW0			word	FFFFh	Section 16.3.1.5
000Bh	CRC32RESB3			byte	FFh	
000Ah	CRC32RESB2			byte	FFh	
0009h	CRC32RESB1			byte	FFh	
0008h	CRC32RESB0			byte	FFh	
000Ch	CRC32RESRW1	CRC32 Result Reverse	read/write	word	FFFFh	Section 16.3.1.8
000Eh	CRC32RESRW0			word	FFFFh	Section 16.3.1.7
000Ch	CRC32RESRB3			byte	FFh	
000Dh	CRC32RESRB2			byte	FFh	
000Eh	CRC32RESRB1			byte	FFh	
000Fh	CRC32RESRB0			byte	FFh	
0010h	CRC16DIW0	CRC16 Data Input	read/write	word	0000h	Section 16.3.1.9
0010h	CRC16DIB0			byte	00h	
0016h	CRC16DIRBW0	CRC16 Data In Reverse	read/write	word	0000h	Section 16.3.1.10
0017h	CRC16DIRBB0			byte	00h	
0018h	CRC16INIRESW0	CRC16 Init and Result	read/write	word	FFFFh	Section 16.3.1.11
0019h	CRC16INIRESB1			byte	FFh	
0018h	CRC16INIRESB0			byte	FFh	
001Eh	CRC16RESRW0	CRC16 Result Reverse	read/write	word	FFFFh	Section 16.3.1.12
001Fh	CRC16RESRB0			byte	FFh	
001Eh	CRC16RESRB1			byte	FFh	

16.3.1.1 CRC32DIW0 Register

Data Input Register Word_0 for 32-Bit CRCs

Data written to this register is taken into CRC32 signature calculations. This register can be accessed 8-bit wide and 16-bit wide.

Figure 16-3. CRC32DIW0 Register

15	14	13	12	11	10	9	8
CRC32DIW0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CRC32DIW0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 16-2. CRC32DIW0 Register Description

Bit	Field	Type	Reset	Description
15-0	CRC32DIW0	RW	0h	CRC32 data in word 0. Data written to the CRC32DIW0 register is included to the present signature in the CRC32INIR register according to the CRC32-ISO3309 standard.

16.3.1.2 CRC32DIW1 Register

Data Input Register Word_1 for 32-Bit CRCs

Data written to this register is taken into CRC32 signature calculations. This register can be accessed 8-bit wide and 16-bit wide.

Figure 16-4. CRC32DIW1 Register

15	14	13	12	11	10	9	8
CRC32DIW1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CRC32DIW1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 16-3. CRC32DIW1 Register Description

Bit	Field	Type	Reset	Description
15-0	CRC32DIW1	RW	0h	CRC32 data in word 1. Data written to the CRC32DIW1 register is included to the present signature in the CRC32INIR register according to the CRC32-ISO3309 standard.

16.3.1.3 CRC32DIRBW0 Register

Data In Register Word_0 with Reversed Bit Order for 32-Bit CRCs

Data written to this register is taken into CRC32 signature calculations. This register can be accessed 8-bit wide and 16-bit wide.

Figure 16-5. CRC32DIRBW0 Register

15	14	13	12	11	10	9	8
CRC32DIRBW0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CRC32DIRBW0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 16-4. CRC32DIRBW0 Register Description

Bit	Field	Type	Reset	Description
15-0	CRC32DIRBW0	RW	0h	CRC32 data in word 0 as bit reversed pattern. Data written to the CRC32DIRBW0 register is included to the present signature in the CRC32NIRE register according to the CRC32-ISO3309 standard.

16.3.1.4 CRC32DIRBW1 Register

Data In Register Word_1 with Reversed Bit Order for 32-Bit CRCs

Data written to this register is taken into CRC32 signature calculations. This register can be accessed 8-bit wide and 16-bit wide.

Figure 16-6. CRC32DIRBW1 Register

15	14	13	12	11	10	9	8
CRC32DIRBW1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CRC32DIRBW1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 16-5. CRC32DIRBW1 Register Description

Bit	Field	Type	Reset	Description
15-0	CRC32DIRBW1	RW	0h	CRC32 data in word1 as bit reversed pattern. Data written to the CRC32DIRBW1 register is included to the present signature in the CRC32NIRE register according to the CRC32-ISO3309 standard.

16.3.1.5 CRC32INIRESW0 Register

Data Initialization and Result Register Word_0 for 32-Bit CRCs

Data written to this register represents the seed for the CRC calculation. This register always reflects the latest signature of the values collected so far. This register can be accessed 8-bit wide and 16-bit wide.

Figure 16-7. CRC32INIRESW0 Register

15	14	13	12	11	10	9	8
CRC32INIRESW0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CRC32INIRESW0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 16-6. CRC32INIRESW0 Register Description

Bit	Field	Type	Reset	Description
15-0	CRC32INIRESW0	RW	0h	CRC32 data initialization and result word0. Data written to the CRC32INIRESW0 register is used as the seed for the CRC calculation according to the CRC32-ISO3309 standard. Reading this register returns the current result of the CRC calculation.

16.3.1.6 CRC32INIRESW1 Register

Data Initialization and Result Register Word_1 for 32-Bit CRCs

Data written to this register represents the seed for the CRC calculation. This register always reflects the latest signature of the values collected so far. This register can be accessed 8-bit wide and 16-bit wide.

Figure 16-8. CRC32INIRESW1 Register

15	14	13	12	11	10	9	8
CRC32INIRESW1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CRC32INIRESW1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 16-7. CRC32INIRESW1 Register Description

Bit	Field	Type	Reset	Description
15-0	CRC32INIRESW1	RW	0h	CRC32 data initialization and result word1. Data written to the CRC32INIRESW1 register is used as the seed for the CRC calculation according to the CRC32-ISO3309 standard. Reading this register returns the current result of the CRC calculation.

16.3.1.7 CRC32RESRW0 Register

Data Result Register Word_0 as Bit Reversed for 32-Bit CRCs

Data written to this register represents the seed for the CRC calculation. This register always reflects the latest signature of the values collected so far. This register can be accessed 8-bit wide and 16-bit wide.

Figure 16-9. CRC32RESRW0 Register

15	14	13	12	11	10	9	8
CRC32RESRW0							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
7	6	5	4	3	2	1	0
CRC32RESRW0							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

Table 16-8. CRC32RESRW0 Register Description

Bit	Field	Type	Reset	Description
15-0	CRC32RESRW0	RW	FFh	CRC32 bit reverse initialization and result word0. This register holds the current CRC32 result (according to the CRC32-ISO3309 standard). The order of bits is reverse to the order of bits in the CRC32INIRESW1 register.

16.3.1.8 CRC32RESRW1 Register

Data Result Register Word1 as Bit Reversed for 32-Bit CRCs

Data written to this register represents the seed for the CRC calculation. This register always reflects the latest signature of the values collected so far. This register can be accessed 8-bit wide and 16-bit wide.

Figure 16-10. CRC32RESRW1 Register

15	14	13	12	11	10	9	8
CRC32RESRW1							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
7	6	5	4	3	2	1	0
CRC32RESRW1							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

Table 16-9. CRC32RESRW1 Register Description

Bit	Field	Type	Reset	Description
15-0	CRC32RESRW1	RW	FFh	CRC32 bit reverse initialization and result word1. This register holds the current CRC32 result (according to the CRC32-ISO3309 standard). The order of bits is reverse to the order of bits in the CRC32INIRESW0 register.

16.3.1.9 CRC16DIW0 Register

Data In Register for 16-Bit CRCs

Data written to this register is taken into CRC16 signature calculations. This register can be accessed 8-bit wide and 16-bit wide.

Figure 16-11. CRC16DIW0 Register

15	14	13	12	11	10	9	8
CRC16DIW0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CRC16DIW0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 16-10. CRC16DIL0 Register Description

Bit	Field	Type	Reset	Description
15-0	CRC16DIW0	RW	0h	CRC16 data in. Data written to the CRC16DIW0 register is included to the present signature in the CRC16NIREs register according to the CRC16-CCITT standard.

16.3.1.10 CRC16DIRBW0 Register

Data In Register with Reversed Bit Order for 16-Bit CRCs

Data written to this register is taken into CRC16 signature calculations. This register can be accessed 8-bit wide and 16-bit wide.

Figure 16-12. CRC16DIRBW0 Register

15	14	13	12	11	10	9	8
CRC16DIRBW0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CRC16DIRBW0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 16-11. CRC16DIRBW0 Register Description

Bit	Field	Type	Reset	Description
15-0	CRC16DIRBW0	RW	0h	CRC16 data in reverse byte. Data written to the CRC16DIRBW0 register is included to the present signature in the CRC16NIREs and CRC16RESR registers according to the CRC-CCITT standard. Reading the register returns the register CRC16DI content.

16.3.1.11 CRC16INIRESW0 Register

Data Initialization and Result Register for 16-Bit CRCs

Data written to this register represents the seed for the CRC calculation. This register always reflects the latest signature of the values collected so far. This register can be accessed 8-bit wide and 16-bit wide.

Figure 16-13. CRC16INIRESW0 Register

15	14	13	12	11	10	9	8
CRC16INIRESW0							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
7	6	5	4	3	2	1	0
CRC16INIRESW0							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

Table 16-12. CRC16INIRESW0 Register Description

Bit	Field	Type	Reset	Description
15-0	CRC16INIRESW0	RW	FFh	CRC16 initialization and result. This register holds the current CRC16 result (according to the CRC16-CCITT standard). Writing to this register initializes the CRC16 calculation with the value written to it.

16.3.1.12 CRC16RESRW0 Register

Data Result Register with Reversed Bits for 16-Bit CRCs

Data written to this register represents the seed for the CRC calculation. This register always reflects the latest signature of the values collected so far. This register can be accessed 8-bit wide and 16-bit wide.

Figure 16-14. CRC16RESRW0 Register

15	14	13	12	11	10	9	8
CRC16RESRW0							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
CRC16RESRW0							
r0	r0	r0	r0	r0	r0	r0	r0

Table 16-13. CRC16RESRW0 Register Description

Bit	Field	Type	Reset	Description
15-0	CRC16RESRW0	RW	FFh	CRC16 reverse result. This register holds the current CRC16 result (according to the CRC16-CCITT standard). The order of bits is reverse to the order of bits in the CRC16INIRESW0 register.

Low-Energy Accelerator (LEA) for Signal Processing

The LEA (low-energy accelerator) module is an execution unit for vector-based signal processing operations. This chapter introduces the LEA.

Topic	Page
17.1 LEA Introduction	447
17.2 LEA Operation.....	447
17.3 LEA Registers	449

17.1 LEA Introduction

The LEA is a 32-bit hardware engine designed for operations that involve vector-based signal processing, such as FIR, IIR, and FFT without CPU intervention. The LEA supports multiple commands, which are issued by CPU. The LEA offers incomparable performance and energy consumption when performing vector-based digital signal processing computations; for performance benchmarks comparing LEA to using the CPU or other processors, see [Benchmarking the Signal Processing Capabilities of the Low-Energy Accelerator on MSP430™ MCUs](#).

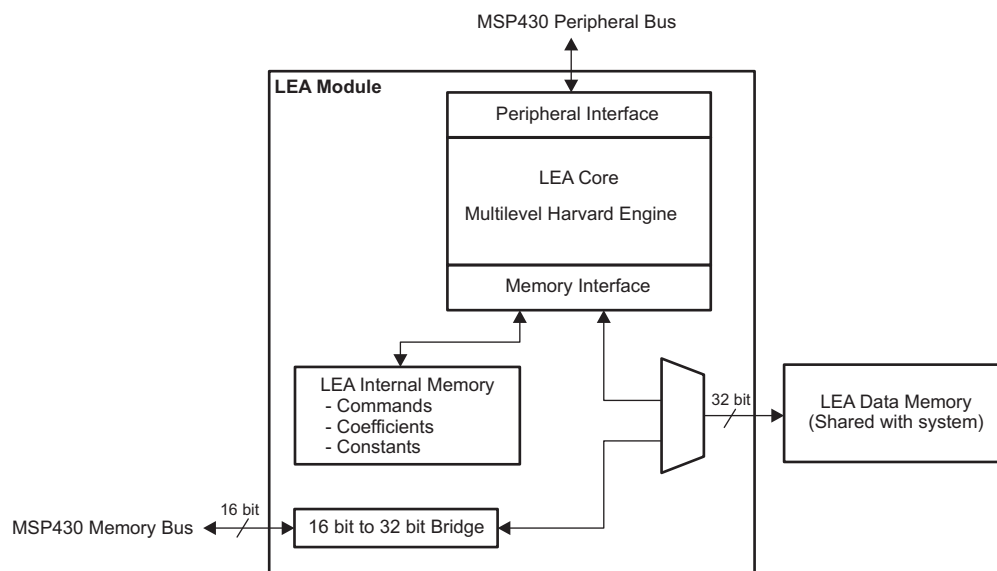


Figure 17-1. LEA System Block Diagram

17.2 LEA Operation

The LEA begins executing the selected operation when the CPU writes a LEA command to the LEA command register when the LEA is in idle mode. Before writing the command, the CPU must configure the LEA argument registers with the pointers to the parameter blocks for the designated operation. The LEA performs the operation without CPU intervention and triggers an interrupt when the operation is complete.

The LEA accesses the LEA data memory, which is used for input data, output data, and the parameter blocks. The LEA data memory is also accessible by the CPU and the DMA, so that the output data of the LEA operation can be moved to other memory location by the CPU or the DMA (see [Figure 17-1](#)). The CPU and the LEA can run simultaneously and independently unless they access the same system memory (RAM). See the device-specific data sheet for details about LEA availability and LEA data memory size.

The LEA supports a various of commands that are used to perform vector-based mathematical operations. [Table 17-1](#) lists the command groups that are available.

Table 17-1. LEA Command Groups

Group	Purpose or Use
Group 1	Basic pointwise vector and matrix operations
Group 2	Basic vector MAC operations (windowing, scaling, general)
Group 3	MAC, pointwise FIR, correlation, convolution
Group 4	Basic minimum/maximum vector search operations on 16-bit data
Group 5	Generic minimum/maximum search operations on 32-bit data
Group 6	Generic minimum/maximum search operations on dual 16-bit data and complex
Group 7	Block based FIR, correlation, convolution
Group 8	Taylor functions and operations on pointwise vectors and matrices
Group 9	FFT and iFFT bank filtering (DIT type)
Group 10	Bit-reversed carry propagated presort for DIT-FFTs
Group 11	FFT post operation for real points
Group 12	Vector and matrix deinterleave and sort functions
Group A	Programming structure and rearrange functions
Group B	Special functions for math, matrix, and DSP

17.2.1 Use the LEA in Programs

How the LEA operates is briefly described in [Section 17.2](#). It is not necessary to understand how the LEA works or the details of the LEA registers. The [Digital Signal Processing \(DSP\) Library for MSP Microcontrollers](#) offers easy-to-use APIs that use the functions of the LEA and provide a high-level environment to use the LEA in various applications. The DSP Library is a set of highly optimized API functions to perform many common signal processing operations on fixed-point numbers for MSP430 microcontrollers. The APIs automatically enable and use the LEA module if the LEA is available in the target device, and apply the optimal configurations to the LEA registers with the correct sequence. If the LEA is not available, the CPU is selected to perform the operations.

The full LEA commands are supported by the DSP Library APIs, which are listed in the [DSP Library API Guide](#), [Low-Energy Accelerator \(LEA\) Frequently Asked Questions \(FAQ\)](#), and [Benchmarking the Signal Processing Capabilities of the Low-Energy Accelerator on MSP430™ MCUs](#).

The following sequence of operations is an example of performing a vector-based algorithm using the LEA, DMA, ADC, and SPI.

1. The CPU sets up the DMA controller, ADC, and SPI.
2. A DMA channel collects samples from the ADC converter at a defined sampling rate and transfers data to the LEA memory.
3. After a block of data has been collected, the CPU enables one or a series of operations of the LEA using the APIs in the [Digital Signal Processing \(DSP\) Library for MSP Microcontrollers](#) to perform the required algorithm (for example, FIR, IIR, correlation, or FFT).
4. When the algorithm is complete, another DMA channel transfers the result of that algorithm to the SPI.
5. The SPI transfers the data to an external device.

17.2.2 Where to Get the DSP Library

The [Digital Signal Processing \(DSP\) Library for MSP Microcontrollers](#) can be downloaded separately, but TI recommends installing [MSPWare](#), which includes the DSP Library with user's guides, code examples, training, and other design resources for all MSP devices. [Table 17-2](#) lists the versions of the DSP Library and MSPWare that support the LEA.

Table 17-2. DSP Library and MSPWare Versions for the LEA

Link	Version
Digital Signal Processing (DSP) Library for MSP Microcontrollers	1.20.00.xx or later
MSP430Ware	3.60 or later

17.2.3 Where to Start

1. Read an introduction to LEA in [Setting a new standard for MCU performance while minimizing energy consumption](#).
2. See [Low-Energy Accelerator \(LEA\) Frequently Asked Questions \(FAQ\)](#) for the questions about the LEA.
3. See the [DSP Library API Guide](#) and [DSP Library Example Projects](#) for detailed information about the DSP Library APIs and examples.

17.3 LEA Registers

As mentioned in [Section 17.2.1](#), the DSP Library APIs are designed to apply the optimal configurations to the LEA registers for the selected operations. Direct access to LEA registers is not supported, and TI recommends using the [Digital Signal Processing \(DSP\) Library for MSP Microcontrollers](#) for the operations that the LEA module supports.

Ultrasonic Sensing Solution (USS, USS_A)

This chapter describes the operation of the USS and the USS_A module.

Topic	Page
18.1 Introduction	451
18.2 Operation of the USS Module	452
18.3 Debug Features	457

18.1 Introduction

The USS module is designed for analog-to-digital (ADC) converter based ultrasonic sensing technology in various measurement applications. [Figure 18-1](#) shows the block diagram of the USS module. The USS is a sophisticated system that consists of the following four submodules. USS_A is an extended variant of the original USS module and features SAPH_A instead of the SAPH module. Because USS_A is a superset of USS, USS_A is specifically named only when describing feature that differ between USS and USS_A.

- **UUPS:** Generates reference voltages, currents, and operation voltage for the USS module, and controls the power-up and power-down sequences of the USS module. See [UUPS](#) for details.
- **HSPLL:** Generates a low-jitter clock (68 MHz to 80 MHz) from the USSXT oscillator for the USS module. See [HSPLL](#) for details.
- **SAPH** (part of USS): Consists of a pulse generator, a low-impedance output driver, an input multiplexer, and an acquisition sequencer. See [SAPH](#) for details.
- **SAPH_A** (part of USS_A): Consists of an extended pulse generator, a low-impedance output driver, an input multiplexer, an acquisition sequencer and external bias terminals. See [SAPH_A](#) for details.
- **SDHS:** High-speed sigma-delta analog-to-digital converter with a dedicated data transfer controller. See [SDHS](#) for details

The submodules have different roles, and together they enable high-precision data acquisition for ultrasonic technology in various applications. The USS module has a dedicated power management module, UUPS, which generates operating voltage, reference voltages, and reference currents for other submodules. The USS module also has a dedicated clock module, HSPLL, which generates a very-low-jitter clock (68 MHz to 80 MHz) from the USSXT oscillator. The SAPH or SAPH_A module controls the signal flow including excitation pulse generation through a low-impedance driver (3-4 Ω , typical), internal bias voltages for Tx and Rx sides, and the input multiplexer. The SDHS is a high-speed analog-to-digital converter to sample the input signals and transfer the output data to the target memory location. The Low Energy Accelerator (LEA), if available, processes the data transferred by the SDHS.

The USS module supports three operation modes for data acquisition sequence: auto mode, register mode and ultra low power bias mode.

In auto mode, the entire measurement sequence is automatically controlled by the ASQ block. Control by ASQ enables ultra-low power consumption during the measurement and frees the CPU from data acquisition, so that ultrasonic application software can be executed in parallel with data acquisition with minimum intervention.

In register mode, the measurement sequence is fully controlled by user software. The register mode can be used during development and for special tasks like diagnostics.

In ultra low power bias mode the ASQ sequencer performs an extended power sequence dedicated for the ultra low power utility meter market.

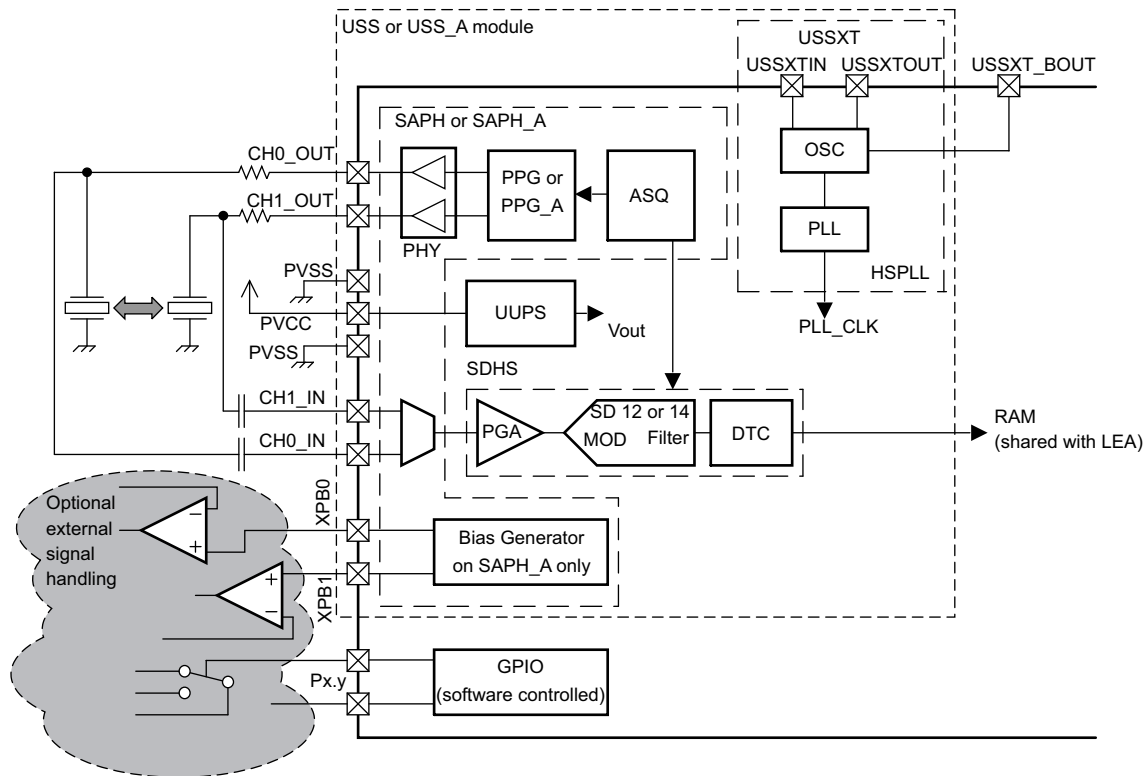


Figure 18-1. USS and USS_A Block Diagram

NOTE: Naming convention for register names and bit fields:

ModuleName.RegisterName or ModuleName.RegisterName.BitField

18.2 Operation of the USS Module

This section describes how the USS submodules are connected and controlled together. For details of each submodule, see the following chapters:

- [UUPS](#)
- [HSPLL](#)
- [SAPH or SAPH_A](#)
- [SDHS](#)

Figure 18-2 shows control signals that connect the submodules.

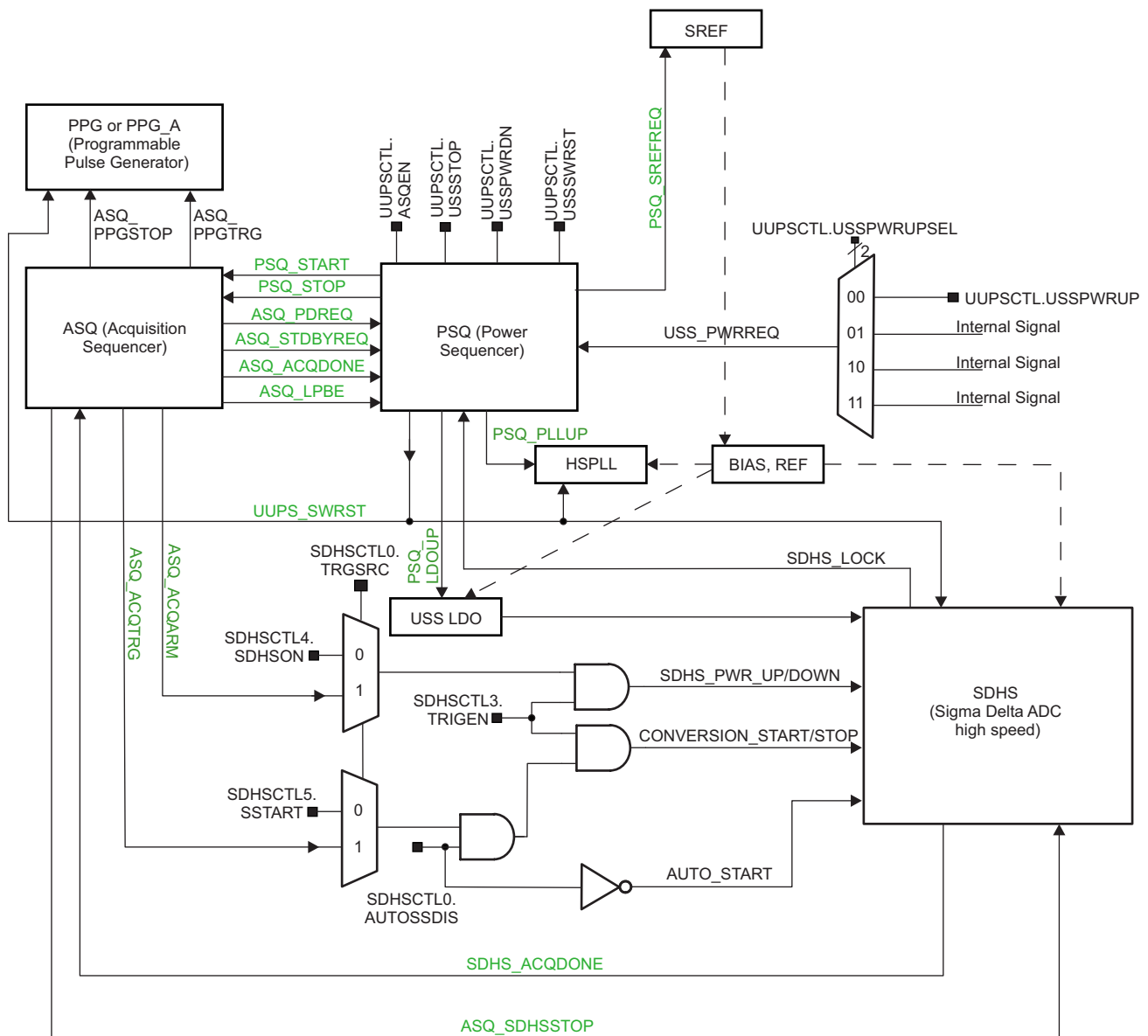


Figure 18-2. USS and USS_A Submodule Connections

NOTE: In Figure 18-2, the control signals between submodules are shown in green. The naming convention for these signal is SourceSubmoduleName_CommandToDestination.

Example 1: PSQ_START: The signal is from PSQ to tell the receiver submodule to START (start measurement)

Example 2: ASQ_ACQTRG: The signal is from ASQ to tell the receiver submodule to ACQTRG (acquisition trigger).

18.2.1 Auto Mode and Register Mode

The USS or USS_A module can perform the full data acquisition sequence with minimum involvement of CPU, which enables ultra-low power consumption during the measurement and frees the CPU from data acquisition, so that the ultrasonic application software can be executed in parallel to data acquisition with minimum intervention. The USS or USS_A module also support register mode, in which each measurement sequence is controlled by user software. Register mode can be used during development and for special tasks like diagnostics.

To enable the auto mode, apply the configurations in [Table 18-1](#) before turning on the USS or USS_A module.

Table 18-1. Auto Mode and Register Mode

Auto Mode Configuration	Action	Register Mode
UUPSCTL.SWRST = 0	Take USS submodules out of reset (once after reset)	UUPSCTL.SWRST = 0
SAPHCMCNF.LPBE = 0	Enable auto mode or register mode (on mode changes)	SAPHCMCNF.LPBE = 0
UUPSCTL.LBHDEL = 0, 1, 2, or 3	Set optimal start-up hold-off delay or leave at default	UUPSCTL.LBHDEL = 0, 1, 2, or 3
SAPHCMCNF.BIMP = 0, 1, 2, or 3	Set optimal bias impedance or leave at default	SAPHCMCNF.BIMP = 0, 1, 2, or 3
UUPSCTL.ASQEN = 1	PSQ to trigger ASQ when power is up (PSQ_START) (on mode changes)	UUPSCTL.ASQEN = 0
SAPHASCTL0.TRIGSEL = 1	ASQ is triggered by PSQ (on mode changes)	SAPHASCTL0.TRIGSEL = 0
SAPHOSEL.PCH0SEL = 1 SAPHOSEL.PCH1SEL = 1	Drive output drivers to GND (on mode changes)	SAPHOSEL.PCH0SEL = 0 SAPHOSEL.PCH1SEL = 0
SAPHBCTL.ASQBSW = 1	Tx bias, Rx bias control (on mode changes)	SAPHBCTL.ASQBSW = 0
SAPHPGCTL.PGSEL = 1	Select output channel in PPG (on mode changes)	SAPHPGCTL.PGSEL = 0
SAPHPGCTL.TRSEL = 1	Trigger PPG (on mode changes)	SAPHPGCTL.TRSEL = 0
SAPHICTL0.MUXCTL = 1	Input channel selection (on mode changes)	SAPHICTL0.MUXCTL = 0
SDHSCTL0.TRGSRC = 1	SDHS power up and conversion trigger source (on mode changes)	SDHSCTL0.TRGSRC = 0
SDHSCTL0.AUTOSSDIS = 1	SDHS conversion trigger (on mode changes)	SDHSCTL0.AUTOSSDIS = 0 or 1
SDHSCTL2.SMPCTLOFF = 0 (optional)	Total sample size is preprogrammed (on mode changes)	SDHSCTL2.SMPCTLOFF = 0 or 1
SDHSCTL2.DTCOFF = 0	Data transfer by DTC (on mode changes)	SDHSCTL2.DTCOFF = 0 or 1

18.2.1.1 Six Time Mark Events in Auto Mode

In auto mode, the six time mark registers generate important timing events during measurement. [Table 21-3](#) describes the time mark events.

Table 18-2. Time Mark Events

Time Mark	Action by ASQ
PSQ_START= 0 → 1	Apply Tx Bias
ASQ time counter = Value in SAPHATM_A register	Trigger PPG to generate excitation pulses and stop pulses
ASQ time counter = Value in SAPHATM_B register	Turn on the SDHS
ASQ time counter = Value in SAPHATM_C register	Apply Rx Bias
ASQ time counter = Value in SAPHATM_D register	Start sampling the input signal (trigger the SDHS)
ASQ time counter = Value in SAPHATM_E register	Restart the time counter for the next measurement
ASQ time counter = Value in SAPHATM_F register	Time-out

18.2.1.2 How to Start in Auto Mode

In auto mode, the entire measurement sequence is executed by the PSQ and the ASQ. The start-up sequence is simple but the following order must be used:

1. Applied desired configurations to USS registers (all submodules). Make sure to apply the auto mode (see [Table 18-1](#)).
2. Set the following bits to 1: SAPHPGCTL.PPGEN, SAPHASCTL0.ASQTEN, SAPHMCNF.LPBE and SDHSCTL3.TRIGEN
3. Assert the USS_PWRREQ signal (see [Table 18-3](#)).

Table 18-3. USS_PWRREQ Signal Source

UUPSCTL.USSPWRUPSEL	Selected Trigger Source	Comment
0	UUPSCTL.USSPWRUP = 1	Default, write only
1	Internal signal	See device-specific data sheet.
2	Internal signal	See device-specific data sheet.
3	Internal signal	See device-specific data sheet.

18.2.2 Control Signals

Start-up signal, USS_PWRREQ

USS_PWRREQ = 0 → 1 is the signal that powers up the USS module and starts a new measurement. If the USS module is already powered up, then the PSQ initiates a new measurement immediately. UUPSCTL.USSPWRUPSEL determines the source of the USS_PWRREQ signal. When the PSQ detects USS_PWRREQ signal, no additional event is accepted by the PSQ until the measurement is complete. UUPSCTL.USS_BUSY indicates whether or not the USS module is in a power transition or performing a measurement. While UUPSCTL.USS_BUSY = 1, the PSQ ignores the USS_PWRREQ signal (0 → 1).

Reset and Low Power Bias Mode control signals

On device Power Up the all USS submodules are kept in reset state. Set UUPSCTL.SWRST = 0 to release reset and put the module into operation mode.

The operation mode has to be selected before the USS is powered up. For Auto mode and Register Mode set SAPGMCNF.LPBE=0.

To select Low Power Bias Mode set SAPHMCNF.LPBE=1. SAPHMCNF.LPBE shall only be changed while the PSQ is in OFF state.

- **USS_SWRST:** SW reset signal fo all USS submodules.
- **ASQ_LPBE:** Enables Low Power Bias operation mode here the ASQ takes has full control over the input multiplexer and channel selection.

Power-up control signals

When USS_PWRREQ = 0 → 1 is detected, the PSQ starts the power-up sequence if the USS module has not been powered up (UUPSCTL.UPSTATE = 0). The PSQ requests the required reference voltage and currents, then enables the USS LDO and enables HSPLL. When the PLL is locked, generate the PSQ_START signal to ASQ.

- **PSQ_SREFREQ:** Request signal to the share reference module.
- **PSQ_LDOUP:** Enables the USS LDO after the reference voltage and currents from BIAS_REF module are settled.
- **PSQ_PLLUP:** Enables HSPLL when UUPSCTL.LDORDY = 1.
- **PSQ_START:** PSQ to ASQ to start a new measurement sequence. The PSQ asserts the signal when IREF_MOD, USS LDO, and HSPLL are fully settled and UUPSCTL.ASQEN bit = 1.
- **PSQ_STOP:** PSQ to ASQ to stop the existing measurement immediately. The PSQ asserts the signal when UUPSCTL.USSPWRDN = 1 or UUPSCTL.USSSTOP = 1.

Measurement control signals

When USS_START = 0 → 1 is detected, the ASQ starts a new measurement sequence based on the timing information in the time mark registers (SAPHATM_A to SAPHATM_F).

- **ASQ_PPGTRG:** ASQ to PPG to generate excitation pules.

- **ASQ_ACQARM:** ASQ to SDHS to power up the SDHS module.
- **ASQ_ACQTRG:** ASQ to SDHS to start data conversion and transfer output data to target memory.
- **SHDS_ACQDONE:** SDHS to ASQ to acknowledge that the preconfigured sample size has been captured.

Power-down control states and signals

When the desired measurement sequences finish execution, the power-down process starts. There are three power states that can be chosen:

- **READY:** The USS module is fully powered up (no changes in power state).
- **STANDBY:** The USS module is powered off, but required reference voltage is kept on for faster wakeup.
- **OFF:** The USS module is fully powered off.

The ASQ asserts ASQ_ACQDONE to acknowledge the PSQ that the measurement is complete. Depending on the selected power state, ASQ_PDREQ or ASQ_STBYREQ can be asserted along with the ASQ_ACQDONE signal.

- **ASQ_PDREQ:** Goes to OFF state.
- **ASQ_STDBYREQ:** Goes to STANDBY state.
- **ASQ_ASQDONE:** Indicates that the measurement is complete. If neither ASQ_PDREQ nor ASQ_STDBYREQ is asserted, then stay in READY state.

Emergency measurement stop control signals

While the USS module is active, the current measurement sequence can be stopped at any time:

- **PSQ_STOP:** PSQ to ASQ to stop the current measurement immediately (when UUPSCCTL.USSSTOP = 1 or UUPSCCTL.USSPWRDN = 1).
- **ASQ_PPGSTOP:** ASQ to PPG to stop pulse generation if the PPG is active.
- **ASQ_SDHSSTOP:** ASQ to SDHS to stop the data conversion and turn off the SDHS if the SDHS is active.

Table 18-4. Control Signals Among USS Submodules

Source	Signal	Receiver	Function	Condition to Generate the Signal
UUPSCCTL.SWRST	USS_SWRST	ASQ, PPG, SAPH, SDHS	Software reset to all USS submodules	UUPSCCTL.SWRST = 1
SAPHMCNF.LPBE	ASQ_LPBE	PSQ	Indication to enter low-power bias mode	SAPHMCNF.LPBE = 1
UUPSCCTL.USSPWRUP	USS_PWRREQ	PSQ	1. Power up the USS module	UUPSCCTL.USSPWRUP = 0 → 1
Internal signal			2. Assert PSQ_START if UUPSCCTL.ASQEN = 1	See device-specific data sheet
Internal signal			3. If UUPSCCTL.USS_BUSY = 1 or UUPSCCTL.UPSTATE = 2, the signal is ignored	See device-specific data sheet
Internal signal				See device-specific data sheet
PSQ	PSQ_SREFREQ	SREF	Shared reference request	USS_PWRREQ: 0 → 1 (valid)
	PSQ_LDOUP	USS LDO	Enable USS LDO	SREF is ready
	PSQ_PLLUP	HSPLL	Enable HSPLL	UUPSCCTL.LDORDY = 1
	PSQ_START	ASQ	Start a new measurement	UUPSCCTL.UPSTATE = 3
	PSQ_STOP		Stop the current measurement immediately	UUPSCCTL.USSPWRDN = 1, UUPSCCTL.USSSTOP = 1 or Enter debug mode

Table 18-4. Control Signals Among USS Submodules (continued)

Source	Signal	Receiver	Function	Condition to Generate the Signal
ASQ	ASQ_PPGTRG	PPG	Start pulse generation	SAPHATM_A register
	ASQ_PPGSTOP		Stop pulse generation immediately	PSQ_STOP = 1 and PPG is active
	ASQ_ACQARM	SDHS	Power up SDHS	SAPHATM_B register
	ASQ_ACQTRG		Conversion start	SAPHATM_D register
	ASQ_SDHSSTOP		Conversion stop and power off	PSQ_STOP = 1
	ASQ_PDREQ	PSQ	USS power down, PSQ can receive a new USS_PWRREQ signal	If SAPHASCTL1.ESOFF = 1 and SAPHASCTL1.STDBY = 0 when SDHS_ACQDONE = 1
	ASQ_STDBYREQ		USS standby, USS power down, PSQ can receive a new USS_PWRREQ signal	If SAPHASCTL1.ESOFF = 1 and SAPHASCTL1.STDBY = 1 when SDHS_ACQDONE = 1
	ASQ_ACQDONE		PSQ can receive a new USS_PWRREQ signal	SDHS_ACQDONE = 1
SDHS	SDHS_ACQDONE	ASQ	Signal to ASQ that SDHS data conversion is complete	When the preconfigured sample size has been captured

18.3 Debug Features

When the device enters debug mode, the USS module automatically stops the existing measurement, and all interrupts are suppressed. The debugger can read or write any USS registers, but SDHSCTL4.SDHSON and SDHSCTL5.SSTART are not functional. The PSQ ignores the USS_PWRREQ signal.

Universal USS Power Supply (UUPS)

This chapter describes the operation of the UUPS module.

Topic	Page
19.1 Introduction	459
19.2 USS Power-up Sequence.....	460
19.3 USS Power States.....	461
19.4 Interface to the ASQ (Acquisition Sequencer)	463
19.5 Interrupts.....	466
19.6 Debug Mode.....	466
19.7 UUPS Registers.....	467

19.1 Introduction

The Universal USS Power Supply (UUPS) is one of the submodules in the Ultrasonic Sensing Solution (USS) module. The USS module is designed for ADC-based ultrasonic sensing technology in various measurement applications. [Figure 19-2](#) shows the block diagram of the USS module.

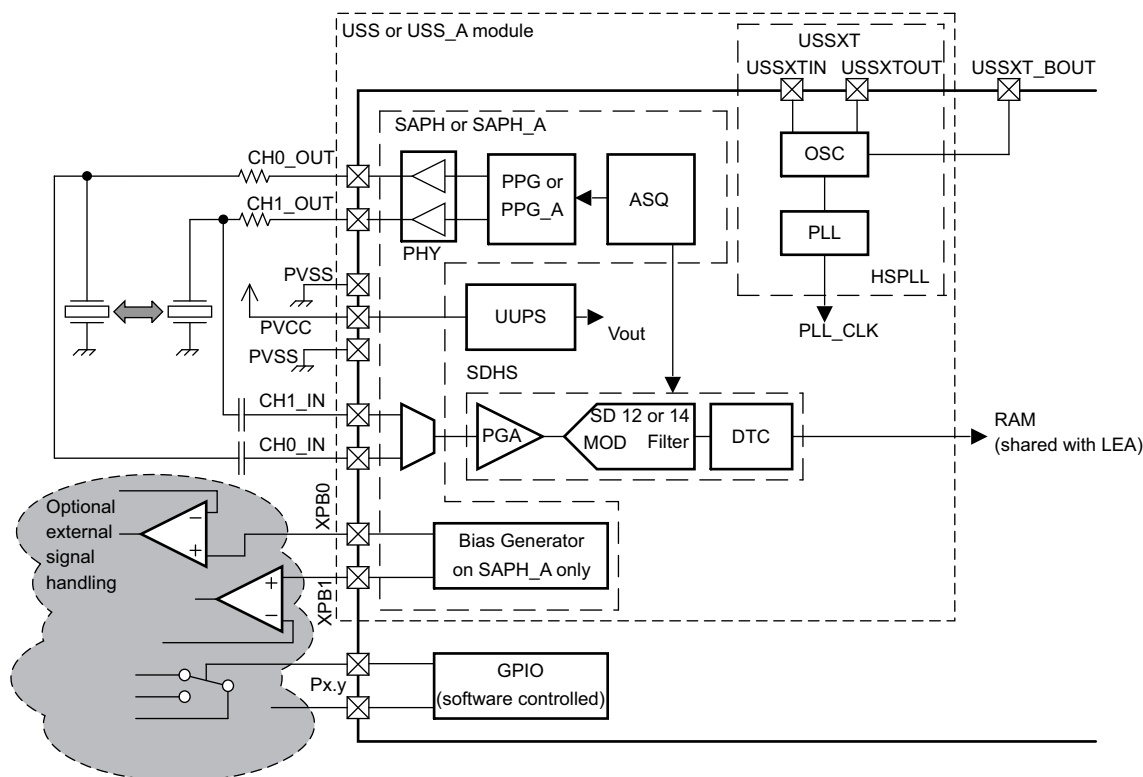


Figure 19-1. USS/USS_A Block Diagram

The UUPS module consists of three blocks: the power sequencer (PSQ), the reference generator (BIAS_REF), and the USS LDO (see [Figure 19-2](#)).

- PSQ block: Controls the power-up and power-down sequences of the USS module.
- BIAS_REF block: Generates required reference voltages and currents for the [SDHS](#), [HSPLL](#), and USS LDO.
- USS_LDO block: Generates regulated 1.6 V, which is used by the USS submodules.

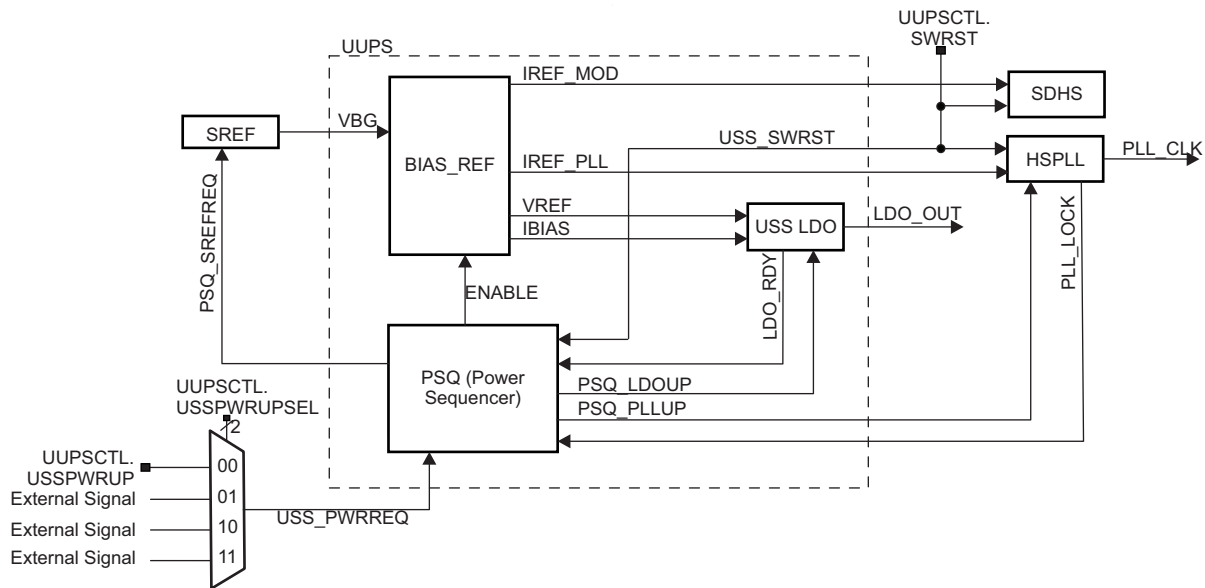


Figure 19-2. UUPS Block Diagram

The signal names in [Figure 19-2](#) are for information only to show how each block is connected inside the UUPS module. These signals are not under user control.

NOTE: Naming convention for register names and bit fields:

- UUPS registers: RegisterName or RegisterName.BitField
- Other module registers: ModuleNameRegisterName or ModuleNameRegisterName.BitField

19.2 USS Power-up Sequence

The Power Sequencer (PSQ) block controls the USS module power-up and power-down sequences. The PSQ powers up the USS module when the USS_PWRREQ signal is asserted (see [Figure 19-2](#)). The USS_PWRREQ signal can be generated from four different sources. When UUPSCTL.USSPWRUPSEL = 0, writing 1 to UUPSCTL.USSPWRUP generates the USS_PWRREQ signal and starts the USS power-up sequences. The other sources may or may not be available; see the device-specific data sheet for the internal signal sources (search for UUPSCTL.USSPWRUPSEL). The power states of the USS module can be monitored by reading UUPSCTL.UPSTATE.

The order of the USS power-up sequence is:

1. The USS_PWRREQ is asserted when the USS module is powered off (UUPSCTL.UPSTATE = 0, indicating that the USS module is in OFF state).
2. The PSQ sends a request to the shared reference (SREF) to generate VBG and starts an internal timer (UUPSCTL.UPSTATE = 2, indicating that the USS power state is in transition).
3. The PSQ enables the BIAS_REF block when the VBG is ready (UUPSCTL.UPSTATE = 2).
4. The PSQ turns on the USS_LDO when the required reference voltages and currents are ready (UUPSCTL.UPSTATE = 2).
5. The PSQ waits for the LDORDY, signal which can be monitored by reading UUPSCTL.LDORDY (UUPSCTL.UPSTATE = 2).
6. The PSQ enables the HSPLL module when LDORDY = 1 (UUPSCTL.UPSTATE = 2).
7. The PSQ waits for the PLL_LOCK signal from the HSPLL module. The PLL_LOCK can be monitored by reading HSPLLCTL.PLL_LOCK (UUPSCTL.UPSTATE = 2).
8. The PSQ sets UUPSCTL.UPSTATE = 3 to indicate that the USS is fully powered and ready to start a

new measurement.

9. If the PSQ internal timer reaches its time-out limit before the PLL_LOCK signal is asserted, UUPSRIS.PTMOUT is set to 1. The time-out is not programmable and can vary depending on the PLL output clock frequency. The maximum delay is 160 μ s plus the configured "holdoff time" (~300us max).

CAUTION

The application must turn on the USSXT oscillator (HSPLLUSSXTLCTL.OSCEN = 1) and wait for a sufficient time to let the oscillator start (this depends on the crystal or resonator characteristics) before powering up the USS module. The USSXT oscillator is not controlled by the PSQ. The PSQ assumes that the oscillator output is already available and stable in frequency and amplitude).

19.3 USS Power States

The following power states are supported by the UUPS module.

- **OFF:** The USS module is powered off (UUPSCTL.UPSTATE = 0).
- **TRANSITION:** The USS module is in transition state (UUPSCTL.UPSTATE = 2).
- **READY:** The USS module is fully powered up and ready (UUPSCTL.UPSTATE = 3).
- **STANDBY:** The USS module is powered off, but the SREF remains on for fast wakeup (UUPSCTL.UPSTATE = 1).
- **TIMEOUT:** The power-up sequence is taking more than expected time. The USS module goes back to OFF state and the PTMOUT (power time-out) interrupt is generated if enabled (UUPSCTL.UPSTATE bits = 0).

Table 19-1. USS Power State

Power State	State Register (UUPSCTL.UPSTATE)	Description
OFF	0	The USS module is fully powered off.
TRANSITION	2	In transition. A new trigger to the PSQ is ignored.
READY	3	The USS module is fully powered up.
STANDBY	1	The USS module is powered off, but SREF is enable for fast wakeup.
TIMEOUT	0	The USS power-up sequence has not been properly ended. The USS module goes back to OFF state and UUPSRIS.PTMOUT is set to 1. Note: A time-out should not happen during normal operating conditions. Make sure that the USSXT oscillator is enabled and working properly.

Figure 19-3 shows the USS module power states and their relationships to each other.

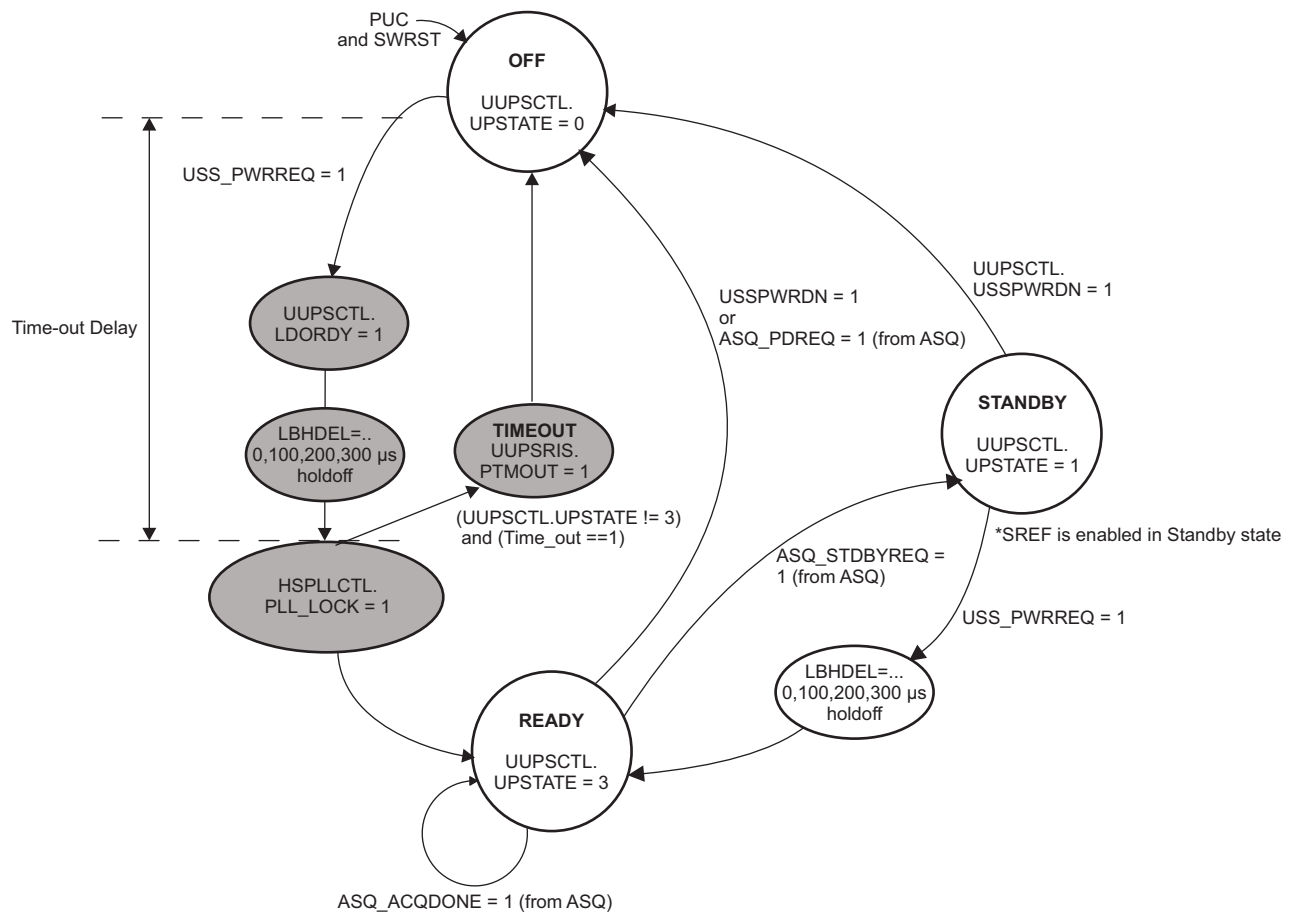


Figure 19-3. USS Power State Control Flow

Table 19-2 lists the signals that cause UUPS power state transitions.

Table 19-2. USS Power States and State Changes

Current Power State	Next State	UUPSCTL.UPSTATE	Trigger Signal
OFF	READY	0 → 2 → 3	USS_PWRREQ = 0 → 1
READY	STANDBY	3 → 2 → 1	ASQ_STDBYREQ = 0 → 1 (generated by ASQ)
READY	OFF	3 → 2 → 0	ASQ_PDREQ = 0 → 1 (generated by ASQ)
READY	OFF	3 → 2 → 0	UUPSCTL.USSPWRDN = 0 → 1
STANDBY	OFF	1 → 2 → 0	UUPSCTL.USSPWRDN = 0 → 1
STANDBY	READY	1 → 2 → 3	USS_PWRREQ = 0 → 1

The USS power state and the device power mode have a relationship as listed in Table 19-3. When the USS module is in READY state, keep the device in LPM0 or AM mode, because the USS module uses resources from the core domain of the device. If the device is in other low-power modes, the measurement performance degrades significantly.

The transition from STANDBY to READY can be extended by a user selectable holdoff delay. This method is used in ultra low bias mode to suppress the PLL operation while the system is still settling. Depending on the transducers and other external circuits some considerable power savings can be achieved.

Table 19-3. Device Power Modes and USS Power States

USS Power State Change	Device Power Mode
READY	LPM0 or higher
STANDBY	LPM3 or higher (PMMREGOFF = 0)

19.4 Interface to the ASQ (Acquisition Sequencer)

Figure 19-4 shows the interface signals between the PSQ and the ASQ (see Chapter 21 for details).

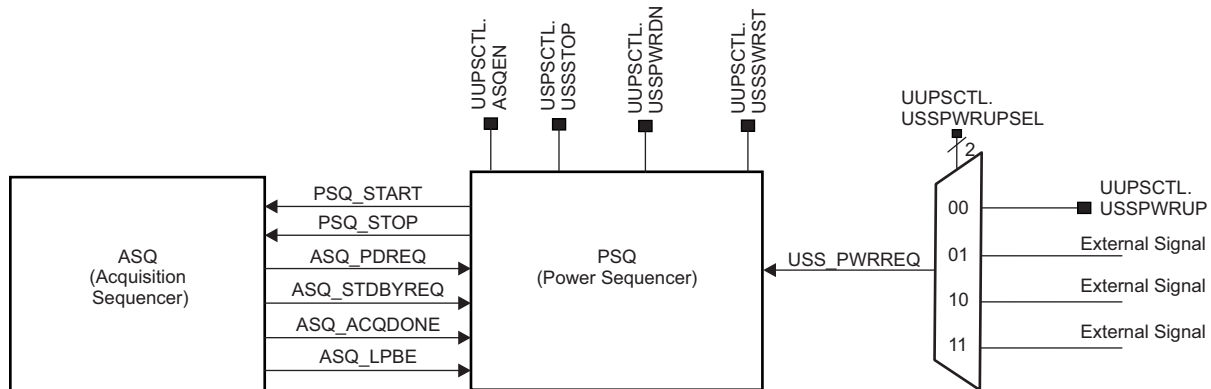


Figure 19-4. USS Power Control

NOTE: The control signals in Table 19-4 are internal only, and cannot be read or written by user software.

Table 19-4. Internal Control Signals

Control Signal	Description	Condition to be Triggered
USS_PWRREQ	PSQ to power up the USS module and generate PSQ_START to the ASQ if UUPSCCTL.ASQEN = 1	Trigger one of the sources selected by UUPSCCTL.USSPWRUPSEL
PSQ_START	ASQ to start a new measurement process if SAPHASCTL0.TRIGSEL = 1	UUPSCCTL.UPSTATE = 0 → 2 → 3 and UUPSCCTL.ASQEN = 1
PSQ_STOP	ASQ to stop the current measurement process	UUPSCCTL.USSSTOP = 0 → 1
ASQ_ACQDONE	Acknowledge PSQ that ASQ has completed the measurements	The measurements have been completed
ASQ_STDBYREQ	PSQ to enter STANDBY state	The measurements have been completed and SAPHASCTL1.ESOFF = 1 and SAPHASCTL1.STDBY = 1
ASQ_PDREQ	PSQ to power off the USS module	The measurements have been completed and SAPHASCTL1.ESOFF = 1 and SAPHASCTL1.STDBY = 0

19.4.1 Start New Measurements

If UUPSCCTL.ASQEN = 1, the PSQ block automatically sends the PSQ_START signal to the ASQ when the USS module is fully powered up. The ASQ starts new measurement sequences upon receiving the PSQ_START signal if SAPHASCTL0.TRIGSEL = 1. In this case, the full measurement sequence can be performed without any CPU intervention, and the USS_PWRREQ is the only trigger signal the USS module needs externally. The PSQ and ASQ must be configured independently (see Figure 19-4). The ASQ can be triggered by user software by writing SAPHASQTRIG.ASQTRIG = 1 when SAPHASCTL0.TRIGSEL = 0.

Figure 19-4 shows the interface signals between the PSQ and the ASQ (see Chapter 21 for details). If UUPSCTL.ASQEN = 1, the PSQ block sends the PSQ_START signal to the ASQ when the USS module reaches READY state from OFF state. Then, the ASQ starts new measurement sequences upon receiving the PSQ_START signal if SAPHASCTL0.TRIGSEL = 1. In this case, the full measurement sequences can be performed without any CPU intervention. The USS_PWRREQ is the only trigger signal the USS needs to take externally. The PSQ and ASQ must be configured independently (see Figure 19-4). The ASQ can be triggered manually by user software by writing SAPHASQTRIG.ASQTRIG = 1 when SAPHASCTL0.TRIGSEL = 0.

Table 19-5. ASQ Trigger

ASQ Auto Trigger Mode	PSQ Configuration	ASQ Configuration	How to Trigger ASQ
Yes	UUPSCTL.ASQEN = 1	SAPHASCTL0.TRIGSEL = 1	The PSQ sends a trigger signal (PSQ_START) to the ASQ when the USS gets to READY state
No	UUPSCTL.ASQEN = 0	SAPHASCTL0.TRIGSEL = 1	This configuration is not supported
No	UUPSCTL.ASQEN = 1	SAPHASCTL0.TRIGSEL = 0	The PSQ_START signal generated by the PSQ is ignored. The ASQ is triggered by writing 1 to SAPHASQTRIG.ASQTRIG.
No	UUPSCTL.ASQEN = 0	SAPHASCTL0.TRIGSEL = 0	The ASQ is triggered by writing 1 to SAPHASQTRIG.ASQTRIG.

19.4.2 Stop Measurement Before Completion

While the USS module is performing a measurement, the PSQ can stop the current measurement process using any of these methods:

- **Write 1 to UUPSCTL.USSSTOP:** The PSQ asserts the PSQ_STOP signal to the ASQ, then the ASQ starts the process to gracefully stop the current measurement. When the measurement is fully stopped, the ASQ asserts the ASQ_ACQDONE signal to the PSQ. No changes are made to the USS power state. The UUPSCTL.USSSTOP bit is cleared when the stop request has been completed. If this bit is set to 1 when the ASQ is idle, it is cleared immediately.
- **Enable Debug mode:** When debug entry is detected, the PSQ asserts, the PSQ asserts the PSQ_STOP signal to the ASQ, then the ASQ starts the process to stop the current measurement gracefully. When the measurement is fully stopped, the ASQ asserts the ASQ_ACQDONE signal to the PSQ. No changes to the USS power state.
- **Write 1 to UUPSCTL.USSPWRDN:** The PSQ asserts the PSQ_STOP signal to the ASQ, then the ASQ starts the process to gracefully stop the current measurement. When the measurement is fully stopped, the ASQ asserts the ASQ_ACQDONE signal, then the PSQ powers off the USS module. The UUPSCTL.USSPWRDN bit is cleared when the power down request has been completed. If this bit is set to 1 when the USS module is powered off (OFF state), it is cleared immediately.

19.4.3 USS Power State After Completion of Measurements

The USS power state after completion of the measurement can be programmed to one of the following states:

- **READY:** No changes to the USS power state. The USS module is ready for the next measurement.
- **STANDBY:** The USS module is powered off but, the SREF block is enabled for fast wakeup.
- **OFF:** The USS module is fully powered off.

Figure 19-4 lists the configuration to select one of the power states after measurement completion.

Table 19-6. Power States After Measurement Completion

Configuration Before Starting a New Measurement	Final USS Power State After Measurement Completion
SAPHASCTL1.ESOFF = 0	READY
SAPHASCTL1.ESOFF = 1 and SAPHASCTL1.STDBY = 1	STANDBY
SAPHASCTL1.ESOFF = 1 and SAPHASCTL1.STDBY = 0	OFF

19.4.4 UUPSCTL.USSPWRUP Bit and UUPSCTL.USS_BUSY Bit

Figure 19-2 shows that the USS_PWRREQ signal can be generated from a maximum of four different sources (see the device-specific data sheet for implementation). When UUPSCTL.USSPWRUPSEL = 0, UUPSCTL.USSPWRUP is selected. In the case, writing 1 to UUPSCTL.USSPWRUP asserts the USS_PWRREQ signal to the PSQ, then the PSQ starts the power-up sequences, and then the PSQ can trigger the PSQ to perform new measurements (if UUPSCTL.ASQEN = 1 and SAPHASCTL0.TRIGSEL = 1). Writing 1 to UUPSCTL.USSPWRUP is invalid and ignored if the power is in TRANSITION state (UUPSCTL.UPSTATE = 2) or the ASQ is performing a measurement sequence (UUPSCTL.USS_BUSY = 1), because the current power sequence or the measurement sequence should be completed before taking a new request.

The UUPSCTL.USS_BUSY bit can be used to monitor the USS module status. When a valid USS_PWRREQ signal is detected, the PSQ sets the UUPSCTL.USS_BUSY bit to 1 to indicate that the PSQ is in busy state and is not ready to take a new USS_PWRREQ signal. Do not to generate a new USS_PWRREQ while UUPSCTL.USS_BUSY = 1. The UUPSCTL.USS_BUSY is set to 1 when any one of the following conditions is satisfied:

- UUPSCTL.UPSTATE = 2 (power state = TRANSITION)
- The ASQ is performing a measurement sequence (the internal signal, ASQ_ACQDONE = 0) .
- The SDHS is sampling data (SDHSCTL5.SDHS_LCK = 1) even when the measurement is not under ASQ control (register mode).

19.5 Interrupts

The UUPS module supports the following interrupts:

- **STPBYDB interrupt:** The USS module has been interrupted by debug mode. This interrupt is reported when debug halt mode is entered when UUPSCTL.USS_BUSY = 1 or UUPSCTL.UPSTATE = 3. The interrupt indicates that any existing activities have been (or will be) stopped due to entering debug mode:
If UUPSCTL.USS_BUSY = 1 and UUPSRIS.STPBYDB = 1, then the USS module is stopping the current activities.
If UUPSCTL.USS_BUSY = 0 and UUPSRIS.STPBYDB = 1, then the USS module is idle.
- **PTMOUT interrupt:** This interrupt is reported when the power-up sequence takes more time than expected. The USS module is powered off and the UUPSRIS.PTMOUT is set to 1.
- **PREQIG interrupt:** This interrupt is reported when a new USS_PWRREQ is detected before completing the previous measurement. Two conditions of the USS_PWRREQ signal cannot be detected:
 1. After detecting a valid USS_PWRREQ signal, another USS_PWRREQ is applied within 3 MODOSC clock cycles.
 2. After UUPSRIS.PREQIG is cleared, a USS_PWRREQ signal is applied within 6 MODOSC clock cycles + 2 system clock cycles.

19.6 Debug Mode

The USS module stops activities when the device enters debug mode. The following actions are performed when entering debug mode.

- Assert the PSQ_STOP signal to the ASQ to stop the current measurement sequence.
- Set UUPSRIS.STPBYDB to 1 to indicate that the current measurement has been interrupted.
- Clear UUPSCTL.USS_BUSY bit to zero upon receiving the ASQ_ACQDONE from the ASQ.
- Ignore the USS_PWRREQ signal (writing 1 to UUPSCTL.USSPWRUP in debug mode has no effect).

19.7 UUPS Registers

[Table 19-7](#) lists the memory-mapped registers for the UUPS. All register offset addresses not listed in [Table 19-7](#) should be considered as reserved locations and the register contents should not be modified.

Table 19-7. UUPS Registers

Offset	Acronym	Register Name	Type	Reset	Section
0h	UUPSIIDX	Interrupt Index Register	read-only	0	Section 19.7.1
2h	UUPSMIS	Masked Interrupt Status Register	read-only	0h	Section 19.7.2
4h	UUPSRIS	Raw Interrupt Status Register	read-only	0h	Section 19.7.3
6h	UUPSIMSC	Interrupt Mask Register	read-write	0h	Section 19.7.4
8h	UUPSICR	Interrupt Clear Register.	write-only	0h	Section 19.7.5
Ah	UUPSISR	Interrupt Flag Set Register.	write-only	0h	Section 19.7.6
Ch	UUPSDESCLO	UUPS Descriptor Register L.	read-only	110h	Section 19.7.7
Eh	UUPSDESCHI	UUPS Descriptor Register H.	read-only	BA10h	Section 19.7.8
10h	UUPSCTL	UUPS Control	read-write	800h	Section 19.7.9

19.7.1 UUPSIIIDX Register (Offset = 0h) [reset = 0h]

UUPSIIIDX is shown in [Figure 19-5](#) and described in [Table 19-8](#).

Return to [Summary Table](#).

Interrupt Index Register.

Note: This register is word accessible. A byte access is also allowed but not recommended. Either high byte or low byte access alone can clear the pending interrupt flag.

Figure 19-5. UUPSIIIDX Register

15	14	13	12	11	10	9	8
IIDX							
R-							
7	6	5	4	3	2	1	0
IIDX							RESERVED
R-							R-

Table 19-8. UUPSIIIDX Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	IIDX	R		UUPS Interrupt Vector Value. Read only. It generates a value that can be used as address offset for fast interrupt service routine handling. On each read, only one interrupt is indicated. On a read, the current interrupt (highest priority) is automatically cleared by the hardware and the corresponding interrupt flag in RIS and MIS are cleared as well. After a read from the CPU (not from the debug interface), the register must be updated with the next highest priority interrupt, if none are pending, then it should display 0h. If the interrupt displayed by the IIDX register (highest priority pending interrupt) is cleared in the ICR through a software write of 1 in the corresponding bit field, the IIDX register shall be updated and the next priority interrupt (if any) be displayed. Reset type: PUC 0h (R) = No Interrupt pending 1h (R) = Interrupt Source: PTMOUT; Interrupt Priority: Highest 2h (R) = Interrupt Source: PREQIG 3h (R) = Interrupt Source: STPB YDB 4h (R) = Reserved; Interrupt Priority: Lowest
0	RESERVED	R	0h	Reserved

19.7.2 UUPSMIS Register (Offset = 2h) [reset = 0h]

UUPSMIS is shown in [Figure 19-6](#) and described in [Table 19-9](#).

Return to [Summary Table](#).

Masked Interrupt Status Register.

Implementation note: UUPSMIS = (RIS and IMSC) when read.

Figure 19-6. UUPSMIS Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					STPBYDB	PREQIG	PTMOUT
R-0h					R-0h	R-0h	R-0h

Table 19-9. UUPSMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2	STPBYDB	R	0h	USS has been interrupted by debug mode Masked Interrupt Status bit. Reset type: PUC 0h (R) = No interrupt pending 1h (R) = Interrupt pending
1	PREQIG	R	0h	UUPS Power Request Ignored Masked Interrupt Status bit. Reset type: PUC 0h (R) = No interrupt pending 1h (R) = Interrupt pending
0	PTMOUT	R	0h	UUPS Power Up Time Out Masked Interrupt Status bit. Reset type: PUC 0h (R) = No interrupt pending 1h (R) = Interrupt pending

19.7.3 UUPSRIS Register (Offset = 4h) [reset = 0h]

UUPSRIS is shown in [Figure 19-7](#) and described in [Table 19-10](#).

Return to [Summary Table](#).

Raw Interrupt Status Register. Read Only Register.

The UUPSRIS register allows the user to implement a poll scheme instead of an interrupt (as the interrupt does not need to be enabled)

Note that the UUPSRIS flag can be cleared by writing to the ICR register bit.

Figure 19-7. UUPSRIS Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				STPBYDB		PREQIG	PTMOUT
R-0h				R-0h		R-0h	R-0h

Table 19-10. UUPSRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2	STPBYDB	R	0h	USS has been interrupted by debug mode. Read Only. This interrupt flag is set when debug halt mode is entered when USS_BUSY = 1 or UPSTATE = 3. The interrupt indicates that any existing activity of USS will be or has been stopped due to entering debug halt. If USS_BUSY = 1 and STPBYDB = 1, then USS is in the middle of stopping the existing activities. If USS_BUSY = 0 and STPBYDB = 1, then USS is in idle state. Reset type: PUC 0h (R) = USS has not been interrupted by debug halt mode. 1h (R) = USS has been interrupted by debug halt mode.
1	PREQIG	R	0h	Power Request Ignored Raw Interrupt Status bit. Read Only. This interrupt flag is set when USS_PWRREQ is detected before completing the previous measurement. Note: There two conditions that a USS_PWRREQ signal cannot be detected. See below. 1) After detecting a valid USS_PWRREQ signal, another USS_PWRREQ is applied within 3 MODOSC clock period. 2) After RIS.PREQIG bit is cleared (either by ICR or reading IIDX register), a USS_PWRREQ signal is applied within 6 MODOSC clock + 2 System clock period. Reset type: PUC 0h (R) = No USS_PWRREQ signal has been ignored 1h (R) = USS_PWRREQ signal has been ignored
0	PTMOUT	R	0h	UUPS Power Up Time Out Raw Interrupt Status bit. Read Only Reset type: PUC 0h (R) = Time out during power up has not occurred 1h (R) = Time out during power up has occurred

19.7.4 UUPSIMSC Register (Offset = 6h) [reset = 0h]

UUPSIMSC is shown in [Figure 19-8](#) and described in [Table 19-11](#).

Return to [Summary Table](#).

Interrupt Mask Register.

This is a read and write register. On a read, it returns the current state of the mask on the relevant interrupt. On a write of 1 to a particular bit, it sets the corresponding mask of that interrupt. A write of 0 clears the mask.

Figure 19-8. UUPSIMSC Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				STPBYDB		PREQIG	PTMOUT
R-0h				R/W-0h		R/W-0h	R/W-0h

Table 19-11. UUPSIMSC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2	STPBYDB	R/W	0h	USS has been interrupted by debug mode Interrupt Mask bit. Reset type: PUC 0h (R) = STPBYDB Interrupt is disabled 1h (R) = STPBYDB Interrupt is enabled
1	PREQIG	R/W	0h	Power Request Ignored Interrupt Mask bit. Reset type: PUC 0h (R/W) = Power Request Ignore Interrupt is disabled. 1h (R/W) = Power Request Ignore Interrupt is enabled.
0	PTMOUT	R/W	0h	UUPS Power Up Time Out Interrupt Mask bit. Reset type: PUC 0h (R/W) = UUPS Power Up Time Out Interrupt is disabled. 1h (R/W) = UUPS Power Up Time Out Interrupt is enabled.

19.7.5 UUPSICR Register (Offset = 8h) [reset = 0h]

UUPSICR is shown in [Figure 19-9](#) and described in [Table 19-12](#).

Return to [Summary Table](#).

Interrupt Clear Register. Write 1 to clear the corresponding RIS bit. Read as zero.

Figure 19-9. UUPSICR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					STPBYDB	PREQIG	PTMOUT
R-0h					W-0h	W-0h	W-0h

Table 19-12. UUPSICR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2	STPBYDB	W	0h	USS has been interrupted by debug mode Interrupt Clear bit. Reset type: PUC
1	PREQIG	W	0h	Power Request Ignored Interrupt Clear bit. Write 1 to Clear RIS.PREQIG bit
0	PTMOUT	W	0h	UUPS Power Up Time Out Interrupt Clear bit. Write 1 to clear RIS.PTMOUT bit.

19.7.6 UUPSISR Register (Offset = Ah) [reset = 0h]

UUPSISR is shown in [Figure 19-10](#) and described in [Table 19-13](#).

Return to [Summary Table](#).

Interrupt Flag Set Register. Read as zero.

Figure 19-10. UUPSISR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					STPB YDB	PREQIG	PTMOUT
R-0h					W-0h	W-0h	W-0h

Table 19-13. UUPSISR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2	STPB YDB	W	0h	USS has been interrupted by debug mode Interrupt Set bit. Reset type: PUC
1	PREQIG	W	0h	Power Request Ignored Interrupt Set bit. Write 1 to set RIS.PREQIG bit
0	PTMOUT	W	0h	UUPS Power Up Time Out Interrupt Set bit. Write 1 to set RIS.PTMOUT bit Reset type: PUC

19.7.7 UUPSDESCLO Register (Offset = Ch) [reset = 110h]

UUPSDESCLO is shown in [Figure 19-11](#) and described in [Table 19-14](#).

Return to [Summary Table](#).

UUPS Descriptor Register L.

Figure 19-11. UUPSDESCLO Register

15	14	13	12	11	10	9	8
FEATUREVER				INSTNUM			
R-0h				R-1h			
7	6	5	4	3	2	1	0
MAJREV				MINREV			
R-1h				R-0h			

Table 19-14. UUPSDESCLO Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	FEATUREVER	R	0h	Feature Set for the module Reset type: PUC
11-8	INSTNUM	R	1h	Instance Number within the device. This will be a parameter to the RTL for modules that can have multiple instances
7-4	MAJREV	R	1h	Major Revision Reset type: PUC
3-0	MINREV	R	0h	Minor Revision Reset type: PUC

19.7.8 UUPSDESCHI Register (Offset = Eh) [reset = BA10h]

UUPSDESCHI is shown in [Figure 19-12](#) and described in [Table 19-15](#).

Return to [Summary Table](#).

UUPS Descriptor Register H.

Figure 19-12. UUPSDESCHI Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODULEID															
R-BA10h															

Table 19-15. UUPSDESCHI Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MODULEID	R	BA10h	Module Identifier. Reset type: PUC

19.7.9 UUPSCTL Register (Offset = 10h) [reset = 800h]

UUPSCTL is shown in [Figure 19-13](#) and described in [Table 19-16](#).

Return to [Summary Table](#).

UUPS Control Register

Figure 19-13. UUPSCTL Register

15	14	13	12	11	10	9	8
USSSTOP	USSPWRDN	LBHDEL		ASQEN		USSPWRUPSEL	USSPWRUP
R/W-0h	R/W-0h	R/W-0h		R/W-1h		R/W-0h	W-0h
7	6	5	4	3	2	1	0
USSSWRST		RESERVED		USS_BUSY		UPSTATE	LDORDY
R/W-0h		R-0h		R-0h		R-0h	R-0h

Table 19-16. UUPSCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	USSSTOP	R/W	0h	Force to stop the current measurement. This bit is self cleared. No change to the power state. This bit is cleared when the stop request has been completed. Note that if this bit is set to '1' when the ASQ is idle, it is cleared immediately. Reset type: PUC 0h (R/W) = No action 1h (R/W) = Stop the current measurement.
14	USSPWRDN	R/W	0h	Force to power down the USS module. This bit is self cleared. Writing '0' has no effect. This bit is cleared when the powerdown request has been completed. Note that if this bit is set to '1' when the USS module is in OFF state, it is cleared immediately. Reset type: PUC 0h (R/W) = No action 1h (R/W) = Stop the current measurement and power off the USS module.
13-12	LBHDEL	R/W	0h	Low power bias hold off delay. These bits define the duration of the hold off delay for the low power bias mode (SAPHMCNF.LPBE=1). The hold off delay is inserted from "OFF state" to "READY state" and from "STANDBY state" to "READY state". Set these bits to zero to avoid extra delay in Register Mode and Auto Mode. Reset type: PUC 0h (R/W) = no additional delay 1h (R/W) = additional hold off delay of ~100us (512 REFCLKs) 2h (R/W) = additional hold off delay of ~200us (1024 REFCLKs) 3h (R/W) = additional hold off delay of ~300us (1536 REFCLKs)
11	ASQEN	R/W	1h	Enable the PSQ_START signal to the ASQ. Note: This bit must be correctly configured before asserting the USS_PWRREQ signal. Reset type: PUC 0h (R/W) = Do not generate the PSQ_START signal event to ASQ. 1h (R/W) = Generate the PSQ_START signal event to the ASQ.
10-9	USSPWRUPSEL	R/W	0h	USS Power Up trigger source select. Reset type: PUC 0h (R/W) = CTL.USSPWRUP bit 1h (R/W) = Ext. trigger (see the device specific datasheet) 2h (R/W) = Ext. trigger (see the device-specific data sheet) 3h (R/W) = Ext. trigger (see the device-specific data sheet)

Table 19-16. UUPSCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	USSPWRUP	W	0h	Power up the USS module. This bit is self cleared. Writing '0' has no effect. Note: This bit is read as zero. Reset type: PUC 0h (R/W) = No action 1h (R/W) = Power up the USS module and generate the PSQ_START to the ASQ if CTL.ASQEN = 1. Note: This bit becomes invalid in debug mode.
7	USSSWRST	R/W	0h	Software reset Reset type: PUC 0h (R/W) = Disabled. USS (and sub modules) reset released for operation 1h (R/W) = Enabled. USS (and sub modules) logic held in reset state
6-4	RESERVED	R	0h	Reserved
3	USS_BUSY	R	0h	USS Busy bit. Read Only. This bit is set to '1' if one of the following conditions is met. 1) CTL.UPSTATE = 2 2) ASQ is in the middl of performing a measurement process 3) SDHS.CTL5.SDHS_LOCK = 1. Reset type: PUC 0h (R) = The USS module is not busy. 1h (R) = The USS module is busy.
2-1	UPSTATE	R	0h	USS Power state bits. Read Only Note: Due to the synchronization issue with 2 bits, there is a very short time window that an invalid value can be read while power transition. In case an accurarte state information is needed, a simple verification would be required - Read the resister twice in a row and check if the two values are the same. Reset type: PUC 0h (R) = USS is in OFF state 1h (R) = USS is in STANDBY state 2h (R) = USS power state is in transition. 3h (R) = USS is in READY state
0	LDORDY	R	0h	USS LDO is ready. Reset type: PUC 0h (R) = USS LDO is powered down or in transition state 1h (R) = USS LDO is powered on

High-Speed PLL (HSPLL)

This chapter describes the operation of the HSPLL module.

Topic	Page
20.1 Introduction	479
20.2 OSC Control Register (HSPLLUSXTCTL)	480
20.3 PLL Control (CTL) Register.....	481
20.4 Start-up Sequence of the USSXT Oscillator	481
20.5 Interrupts.....	482
20.6 HSPLL Registers	483

The High-Speed PLL (HSPLL) is one of the submodules in the Ultrasonic Sensing Solution (USS) module. The USS module is designed for analog-to-digital converter (ADC) based ultrasonic sensing technology in various measurement applications. [Figure 20-1](#) shows the block diagram of the USS module.

Figure 20-1. USS or USS_A Block Diagram

- **OSC (oscillator) block:** Generates a clock of 4 MHz or 8 MHz from USSXT (crystal or ceramic resonator).
- **PLL (phase-locked loop):** Generates a clock in the range of 68 MHz to 80 MHz from the output of the OSC.

Figure 20-2. HSPLL Block Diagram

The control bits of the OSC are grouped in the HSPLLUSSTCTL register, and the control bits of the PLL are grouped in the HSPLLCTL register.

NOTE: Naming conventions in this document for register names and bit fields:

- HSPLL registers: RegisterName or RegisterName.BitField
- Other module registers: ModuleNameRegisterName or ModuleNameRegisterName.BitField

20.2 OSC Control Register (HSPLLUSSTCTL)

As shown in [Figure 20-2](#), the PLL takes the input clock from OSC block. The OSC block supports both crystal resonators and ceramic resonators with a frequency range of 4 MHz to 8 MHz. The OSC does not support a bypass mode, so do not attempt to feed an external clock to the USSXTIN pin. Ensure that the oscillator is enabled and fully stable before enabling the USS module.

20.2.1 OSCEN Bit

The HSPLLUSSTCTL.OSCEN bit enables or disables the oscillator. The output of the oscillator is gated by default. When the oscillator is enabled, it drives the external resonator and waits for a predefined counter value before enabling the output of the oscillator to feed a stable clock to the PLL. One of the two predefined counter values can be selected by HSPLLUSSTCTL.OSCTYPE.

20.2.2 OSCTYPE Bit

The oscillator supports both crystal resonators and ceramic resonators. The resonators have different start-up times, so the correct setting must be applied before enabling the OSC (by writing 1 to HSPLLUSSTCTL.OSCEN). Ceramic resonators have faster wake-up time than crystal resonators. For a crystal resonator, TI recommends setting HSPLLUSSTCTL.OSCTYPE to 0 (gating counter = 4096). For a ceramic resonator, TI recommends setting HSPLLUSSTCTL.OSCTYPE to 1 (gating counter = 512).

20.2.3 OSCSTATE Bit

The HSPLLUSSTCTL.OSCSTATE bit is set after the predefined cycle count has been reached after the oscillator is enabled. This bit can be used to check whether the oscillator has started. The power-up sequence of the USS module is fully controlled by the PSQ (see [Chapter 19](#)). However, the oscillator must be enabled and stable before powering up the USS module. The application must turn on the USSXT oscillator (HSPLLUSSTCTL.OSCEN = 1) and wait until the oscillator output is available (HSPLLUSSTCTL.OSCSTATE = 1) before powering up the USS module. The OSCSTATE bit indicates sufficient signal strength, not signal quality of the oscillator output.

CAUTION

The application must turn on the USSXT oscillator (HSPLLUSSTCTL.OSCEN = 1) and wait for a sufficient time to let the oscillator start (this depends on the crystal and resonator characteristics) before powering up the USS module (see [Section 20.4.1](#)). The USSXT oscillator is not controlled by the PSQ. The PSQ assumes that the oscillator output is already available (see [Chapter 20](#)).

20.2.4 XTOUTOFF Bit

An application may be required to monitor the clock from the oscillator or to use the clock as the source of another subsystem. To meet these requirements, the buffered output clock from the oscillator can be enabled on the USSXT_BOUT pin by setting HSPLLUSSTCTL.XTOUTOFF = 0. The clock on the USSXT_BOUT pin can be monitored or used as a clock source. Never use the USSXTOUT pin for monitoring or for a clock source.

20.3 PLL Control (CTL) Register

The PLL block takes its input from the oscillator (4 MHz to 8 MHz) and generates the output clock in the range of 68 MHz to 80 MHz. The PLL block does not have a separate power control. It is fully controlled by the Power Sequencer (PSQ). When the USS module is in OFF state (UUPSCTL.UPSTATE = 0) or STANDBY state (UUPSCTL.UPSTATE = 1), the PLL block is turned off and does not consume power. See [Chapter 19](#) for details. The output of the PLL block is divided by half to keep a 50-50 duty cycle at the final output. The maximum frequency at the final output must be limited to 80 MHz.

20.3.1 PLLM[5:0] Bits

The PLL output frequency is determined as :

- PLL output clock frequency = input clock frequency $\times$ (PLLM + 1)
- The final output clock frequency = PLL output clock frequency / 2

The input clock to the PLL block must be 4 MHz to 8 MHz. Choose a HSPLLCTL.PLLM[5:0] value so that the final output clock is in the range of 68 MHz to 80 MHz. Set HSPLLCTL.PLLM[5:0] to the desired value before powering up the USS module and do not change the value while the USS module is on.

20.3.2 PLLINFREQ Bit

The PLL can be optimized for the best performance based on its input frequency. The HSPLLCTL.PLLINFREQ bit divides the input frequency range into two categories (≤ 6 MHz and > 6 MHz) and optimizes the PLL block for the specified input range and output frequency.

20.3.3 PLL_LOCK Bit

The HSPLLCTL.PLL_LOCK bit indicates whether or not the PLL output clock is stable. HSPLLCTL.PLL_LOCK is set to 1 when the PLL output frequency reaches the desired frequency and becomes stable. The lock signal is also used by the PSQ during its power-up sequence. When the PLL output is changed from locked to unlocked, the PLL unlock interrupt bit (HSPLLRIS.PLLUNLOCK) is set. When the PLL unlock interrupt occurs, TI recommends turning off the USS module, and then checking the USSXT oscillator to make sure it is operating correctly before enabling the USS module again.

20.3.4 USSXT Control Register

[Figure 20-2](#) show that the PLL has one input clock source, the USSXT oscillator. The oscillator supports both crystal resonators and ceramic resonators with the range of 4 MHz to 8 MHz. The oscillator must be enabled and fully stable before the PLL is enabled.

20.4 Start-up Sequence of the USSXT Oscillator

The PLL block is automatically enabled and disabled by the Power Sequencer (PSQ) during the power-up and power-down sequences of the USS module. The USSXT oscillator must be enabled and stabled before the application enables the USS module. The application must start the USS XT oscillator and wait until it is stable before powering up the USS module, as described in the following sequence:

1. Configure HSPLLUSSTLCTL.OSCTYPE bit correctly based on the resonator type (0 for a crystal resonator, 1 for a ceramic resonator).
2. Write 1 to the HSPLLUSSTLCTL.USSXTEN bit.
3. Wait for the start-up time. The device can enter a low-power mode while waiting for a TIMER interrupt.
4. Read the HSPLLUSSTLCTL.OSCSTATE bit to check if the USSXT started.
5. Now the USSXT is running. The USS module can be powered up.

NOTE: All of the USS submodules must be configured properly before powering up the USS module.

20.4.1 USSXT Start-up Behavior

The USSXT oscillator is designed to generate an even running output clock in frequency and amplitude while still allowing fast start-up. Therefore, the crystal or resonator is given more degrees of freedom during start-up. Some crystals and resonators use the distributed capacitance and inductance of the PCB traces and package as tank circuits and tend to oscillate on its harmonic frequencies during power up. Increasing the value of the serial resistance between USSXTOUT and crystal or resonator prevents such oscillation conditions at those higher frequencies. The maximum time for crystals and resonators is give in the data sheet. USSXT_BOUT can be used to monitor USSXT start-up and operation. Use USSXT_BOUT as system clock (for example, through HFXTIN) only after USSXT is completely started and has settled.

20.5 Interrupts

The HSPLL module supports one interrupt:

- **PLLUNLOCK interrupt:** When the PLL output status changes from locked to unlocked, HSPLLRI.SPLLUNLOCK is set to 1.

20.6 HSPLL Registers

[Table 20-1](#) lists the memory-mapped registers for the HSPLL. All register offset addresses not listed in [Table 20-1](#) should be considered as reserved locations and the register contents should not be modified.

Table 20-1. HSPLL Registers

Offset	Acronym	Register Name	Type	Reset	Section
0h	HSPLLIIDX	Interrupt Index Register	read-only	0	Section 20.6.1
2h	HSPLLMIS	Masked Interrupt Status Register.	read-only	0h	Section 20.6.2
4h	HSPLLRIS	Raw Interrupt Status Register	read-only	0h	Section 20.6.3
6h	HSPLLIMSC	Interrupt Mask Register	read-write	0h	Section 20.6.4
8h	HSPLLICR	Interrupt Flag Clear Register.	write-only	0h	Section 20.6.5
Ah	HSPLLISR	Interrupt Flag Set Register.	write-only	0h	Section 20.6.6
Ch	HSPLDESCLO	HSPLL Descriptor Register L.	read-only	110h	Section 20.6.7
Eh	HSPLDESCHI	HSPLL Descriptor Register H.	read-only	BD10h	Section 20.6.8
10h	HSPLLCTL	HSPLL Control Register	read-write	4000h	Section 20.6.9
12h	HSPLLUSSXTLCTL	USSXT Control Register	read-write	100h	Section 20.6.10

20.6.1 HSPLLIDX Register (Offset = 0h) [reset = 0h]

HSPLLIDX is shown in [Figure 20-3](#) and described in [Table 20-2](#).

Return to [Summary Table](#).

Interrupt Index Register.

Note: This register is word accessible. A byte access is also allowed but not recommended. Either high byte or low byte access alone can clear the pending interrupt flag.

Figure 20-3. HSPLLIDX Register

15	14	13	12	11	10	9	8
IIDX							
R-							
7	6	5	4	3	2	1	0
IIDX							RESERVED
R-							R-

Table 20-2. HSPLLIDX Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	IIDX	R		HSPLL Interrupt Vector Value. Read only. It generates a value that can be used as address offset for fast interrupt service routine handling. On each read, only one interrupt is indicated. On a read, the current interrupt (highest priority) is automatically cleared by the hardware and the corresponding interrupt flag in RIS and MISC are cleared as well. After a read from the CPU (not from the debug interface), the register must be updated with the next highest priority interrupt, if none are pending, then it should display 0h. If the interrupt displayed by the IIDX register (highest priority pending interrupt) is cleared in the MISC through a software write of 1 in the corresponding bit field, the IIDX register shall be updated and the next priority interrupt (if any) be displayed. Reset type: PUC 0h (R) = No Interrupt pending 1h (R) = Interrupt Source: PLLUNLOCK; Interrupt Priority: Highest 2h (R) = Reserved; Interrupt Priority: Lowest
0	RESERVED	R	0h	Reserved

20.6.2 HSPLLMIS Register (Offset = 2h) [reset = 0h]

HSPLLMIS is shown in [Figure 20-4](#) and described in [Table 20-3](#).

Return to [Summary Table](#).

Masked Interrupt Status Register.

Figure 20-4. HSPLLMIS Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							PLLUNLOCK
R-0h							R-0h

Table 20-3. HSPLLMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	PLLUNLOCK	R	0h	HSPLL Unlock Masked Interrupt Status bit. Reset type: PUC 0h (R) = No interrupt pending 1h (R) = Interrupt pending

20.6.3 HSPLLRIIS Register (Offset = 4h) [reset = 0h]

HSPLLRIIS is shown in [Figure 20-5](#) and described in [Table 20-4](#).

Return to [Summary Table](#).

Raw Interrupt Status Register. Read Only Register.

The HSPLLRIIS register allows the user to implement a poll scheme instead of an interrupt (as the interrupt does not need to be enabled) Note that the HSPLLRIIS flag can be cleared by writing to the MISC register bit even if the corresponding IM bit is not enabled.

Figure 20-5. HSPLLRIIS Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							PLLUNLOCK
R-0h							R-0h

Table 20-4. HSPLLRIIS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	PLLUNLOCK	R	0h	PLL Unlock Raw Interrupt Status bit. Read Only. This bit is set when PLL output changes from lock to unlock status. Reset type: PUC 0h (R) = PLL status has not been changed 1h (R) = PLL status has been changed from Lock to Unlock

20.6.4 HSPLLIMSC Register (Offset = 6h) [reset = 0h]

HSPLLIMSC is shown in [Figure 20-6](#) and described in [Table 20-5](#).

Return to [Summary Table](#).

Interrupt Mask Register. Note: writing '1' enables the corresponding interrupt.

Figure 20-6. HSPLLIMSC Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							PLLUNLOCK
R-0h							R/W-0h

Table 20-5. HSPLLIMSC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	PLLUNLOCK	R/W	0h	PLL Unlock Interrupt Mask bit. Reset type: PUC 0h (R/W) = PLL Unlock Interrupt is disabled 1h (R/W) = PLL Unlock Interrupt is enabled

20.6.5 HSPLLICR Register (Offset = 8h) [reset = 0h]

HSPLLICR is shown in [Figure 20-7](#) and described in [Table 20-6](#).

Return to [Summary Table](#).

Interrupt Flag Clear Register. Read as zero.

Figure 20-7. HSPLLICR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							PLLUNLOCK
R-0h							W-0h

Table 20-6. HSPLLICR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	PLLUNLOCK	W	0h	PLL Unlock Interrupt Clear bit. Write 1 to clear RIS.PLLUNLOCK bit Reset type: PUC

20.6.6 HSPLLISR Register (Offset = Ah) [reset = 0h]

HSPLLISR is shown in [Figure 20-8](#) and described in [Table 20-7](#).

Return to [Summary Table](#).

Interrupt Flag Set Register. Read as zero.

Figure 20-8. HSPLLISR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							PLLUNLOCK
R-0h							W-0h

Table 20-7. HSPLLISR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	PLLUNLOCK	W	0h	PLL Unlock Interrupt Set bit. Write 1 to set RIS.PLLUNLOCK bit Reset type: PUC

20.6.7 HSPLLDESCLO Register (Offset = Ch) [reset = 110h]

HSPLLDESCLO is shown in [Figure 20-9](#) and described in [Table 20-8](#).

Return to [Summary Table](#).

HSPLL Descriptor Register L.

Figure 20-9. HSPLLDESCLO Register

15	14	13	12	11	10	9	8
FEATUREVER				INSTNUM			
R-0h				R-1h			
7	6	5	4	3	2	1	0
MAJREV				MINREV			
R-1h				R-0h			

Table 20-8. HSPLLDESCLO Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	FEATUREVER	R	0h	Feature Set for the module Reset type: PUC
11-8	INSTNUM	R	1h	Instance Number within the device. This will be a parameter to the RTL for modules that can have multiple instances
7-4	MAJREV	R	1h	Major Revision Reset type: PUC
3-0	MINREV	R	0h	Minor Revision Reset type: PUC

20.6.8 HSPLLDESCHI Register (Offset = Eh) [reset = BD10h]

HSPLLDESCHI is shown in [Figure 20-10](#) and described in [Table 20-9](#).

Return to [Summary Table](#).

HSPLL Descriptor Register H.

Figure 20-10. HSPLLDESCHI Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODULEID															
R-BD10h															

Table 20-9. HSPLLDESCHI Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MODULEID	R	BD10h	Module Identifier. Reset type: PUC

20.6.9 HSPLLCTL Register (Offset = 10h) [reset = 4000h]

HSPLLCTL is shown in [Figure 20-11](#) and described in [Table 20-10](#).

Return to [Summary Table](#).

HSPLL Control Register

Figure 20-11. HSPLLCTL Register

15	14	13	12	11	10	9	8
PLLM						RESERVED	PLLINFREQ
R/W-10h						R-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED							PLL_LOCK
R-0h							R-0h

Table 20-10. HSPLLCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	PLLM	R/W	10h	PLL Multiplier. default value = 16. Valid data range: 16 ~ 39. The input clock to the PLL block must be 4MHz ~ 8MHz. Care needs to be taken to choose PLLM[5:0] value that the final output clock must be in range of 68MHz ~ 80MHz. Note that PLLM[5:0] needs to be configured with the desired value before powering up the USS module and must not be changed while the USS module is on. 10h (R/W) = 16 11h (R/W) = 17 12h (R/W) = 18 13h (R/W) = 19 14h (R/W) = 20 15h (R/W) = 21 16h (R/W) = 22 17h (R/W) = 23 18h (R/W) = 24 19h (R/W) = 25 1Ah (R/W) = 26 1Bh (R/W) = 27 1Ch (R/W) = 28 1Dh (R/W) = 29 1Eh (R/W) = 30 1Fh (R/W) = 31 20h (R/W) = 32 21h (R/W) = 33 22h (R/W) = 34 23h (R/W) = 35 24h (R/W) = 36 25h (R/W) = 37 26h (R/W) = 38 27h (R/W) = 39
9	RESERVED	R	0h	Reserved
8	PLLINFREQ	R/W	0h	PLL Input Frequency Selection. 0h (R/W) = Input frequency is equal to 6MHz or lower than 6MHz 1h (R/W) = Input frequency is higher than 6MHz
7-1	RESERVED	R	0h	Reserved

Table 20-10. HSPLLCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	PLL_LOCK	R	0h	PLL Lock Status. Note: When PLL output is changed from locked status to unlock status, PLL Unlock Interrupt bit (RIS.PLLUNLOCK) is set. It is recommended to enable the interrupt while performing a measurement. Reset type: PUC 0h (R) = PLL is not running or not locked 1h (R) = PLL is locked

20.6.10 HSPLLUSSXTLCTL Register (Offset = 12h) [reset = 100h]

HSPLLUSSXTLCTL is shown in [Figure 20-12](#) and described in [Table 20-11](#).

Return to [Summary Table](#).

USSXT Control Register.

HSPLL has a dedicated XTAL which generates the input frequency of the HSPLL.

Figure 20-12. HSPLLUSSXTLCTL Register

15	14	13	12	11	10	9	8
RESERVED						OSCTYPE	XTOUTOFF
R-0h						R/W-0h	R/W-1h
7	6	5	4	3	2	1	0
RESERVED						OSCSTATE	USSXTEN
R-0h						R-0h	R/W-0h

Table 20-11. HSPLLUSSXTLCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9	OSCTYPE	R/W	0h	Oscillator Type. The oscillator output clock is gated until it is fully stabilized after power up in order to provide a stable clock frequency to HSPLL. 0h (R) = XTAL : Gating Counter Length: 4096. It is recommended to use this configuration for crystal resonators. Note: the counter counts the oscillator clock, so total time can be calculated as Time = 4096 x 1/Oscillator Clock Frequency. 1h (R) = CERAMIC : Gating Counter Length: 512. It is recommended to use this configuration for ceramic resonators. Note: the counter counts the oscillator clock, so total time can be calculated as Time = 512x 1/Oscillator Clock Frequency.
8	XTOUTOFF	R/W	1h	USSXT Buffered Output OFF 0h (R/W) = Enable USSXT buffered output 1h (R/W) = Disable USSXT buffered output. Default.
7-2	RESERVED	R	0h	Reserved
1	OSCSTATE	R	0h	Oscillator start indication. This bit indicates that the oscillator started and has sufficient signal strength (not signal quality) 0h (R) = Oscillator is either not enabled or in the middle of start-up transition. 1h (R) = Oscillator has started but is not stable yet. Wait for sufficient time for stabilization.
0	USSXTEN	R/W	0h	USSXT Enable. Reset type: PUC 0h (R) = Disable USSXT Oscillator 1h (R) = Enable USSXT Oscillator

Sequencer for Acquisition, Programmable Pulse Generator, and Physical Interface (SAPH, SAPH_A)

This chapter describes the operation of the SAPH module.

Topic	Page
21.1 Introduction	496
21.2 Programmable Pulse Generator (PPG or PPG_A) Block	497
21.3 Physical Interface (PHY) Block	504
21.4 Acquisition Sequencer (ASQ).....	510
21.5 Ultra-Low-Power Bias Mode	514
21.6 Interrupts Triggers	515
21.7 DMA Triggers	515
21.8 SAPH and SAPH_A Registers	516

21.1 Introduction

The sequencer for acquisition, programmable pulse generator, and physical interface (SAPH or SAPH_A) is one of the submodules in the Ultrasonic Sensing Solution (USS or USS_A) module. The USS module is designed for analog-to-digital converter (ADC) based ultrasonic sensing technology in various measurement applications. See [Chapter 18](#) for details.

The SAPH module features the PPG generator, and the SAPH_A module features the PPG_A generator. The SAPH_A is a superset of SAPH; therefore, SAPH_A is specifically named only when necessary to distinguish from the SAPH module.

The SAPH or SAPH_A consist of three blocks (see [Figure 21-1](#)):

- Acquisition Sequencer (ASQ)
- Programmable Pulse Generator (PPG or PPG_A)
- Physical Interface (PHY)

SAPH_A provides additional bias voltages for optional external signal conditioning like low-noise amplifiers (LNAs) and booster amplifiers.

- PPG or PPG_A block: Generates pulses at different frequencies.
- PHY block: Controls output channels and input channels of the USS module.
- ASQ block: The entire measurement sequence can be controlled by user software (called register mode) or by ASQ without any CPU intervention (called auto mode). The auto mode helps reduce the measurement power consumption, because the CPU can stay in LPM0 during measurement.
- External bias generator on SAPH_A

NOTE: Naming convention for register names and bit fields:

- SAPH registers: RegisterName or RegisterName.BitField
 - Other module registers: ModuleNameRegisterName or ModuleNameRegisterName.BitField
 - Code written for USS and SAPH modules is also accepted by USS_A and SAPH_A modules. The register names are resolved automatically.
-

NOTE: Before writing to any SAPH register with an offset address greater than 0x0F, unlock the registers by writing 0x45B to the KEY register. The unlock must be performed only once. Writing any other value to the KEY register locks the registers. Read accesses are always allowed.



Figure 21-2 shows the conceptual block diagram of the programmable pulse generator (PPG). In *italics* the extended elements of PPG A.



21.2.1 Pulse Generation

The PPG offers flexible pulse generation profiles for single tone generation at different frequencies. The PPG_A is an extended variant of the PPG and extends the profiles with dual tone, trill tone, and multi tone (chirp).

21.2.2 Single Tone Generation

The output pulses consists of four different phases: *Pause*, *Excitation*, *Stop*, and *Pause* again. A state machine in PPG and PPG_A controls the flow. On PPG_A set SAPH_XPGCTL.XMOD = 0 for single tone generation. When the PPG or PPG_A is triggered, it leaves the Pause phase and generates excitation pulses followed by stop pulses, then goes to Pause phase again (see Figure 21-3, Figure 21-4 and Figure 21-5). The stop pulses have a 180° phase shift compared to the excitation pulses. The PPG generates up to 127 excitation pulses and up to 15 stop pulses, which are controlled by the SAPHPGC.EPULS and SAPHPGC.SPULS bits, respectively. The pulse polarity is programmable in the SAPHPGC.PPOL bit. The signal polarity for Pause can be programmed to be logical high, logical low, or high impedance through the SAPHPGC.PLEV and SAPHPGC.PHIZ bits.

The PPG can be triggered by writing 1 to the SAPHPPGTRIG.PPGTRIG bit when SAPHPGCTL.TRSEL = 0 (register mode) or by the acquisition sequencer (ASQ) when SAPHPGCTL.TRSEL = 1 (auto mode). To avoid unintended pulse outputs, keep SAPHPGCTL.PPGEN = 0 while updating the PPG registers. After the PPG registers are updated, write 1 to the SAPHPGCTL.PPGEN bit before triggering the PPG. The SAPHPGCTL.PPGEN bit must be set before triggering the PPG. The output channel is determined by the SAPHPGCTL.PPGCHSEL bit when SAPHPGCTL.PGSEL = 0 (register mode) or by the acquisition sequencer (ASQ) when SAPHPGCTL.PGSEL = 1 (auto mode). Another layer of output control is inside the PHY, so both blocks must be configured properly (see Section 21.3). The PPG or PPG_A automatically stops when it completes generating the pulses. To stop generating pulses before completion, regardless of operation mode, write 1 to SAPHPGCTL.STOP. The PPG immediately stops generating pulses. The SAPHPGCTL.STOP bit is automatically cleared to zero.

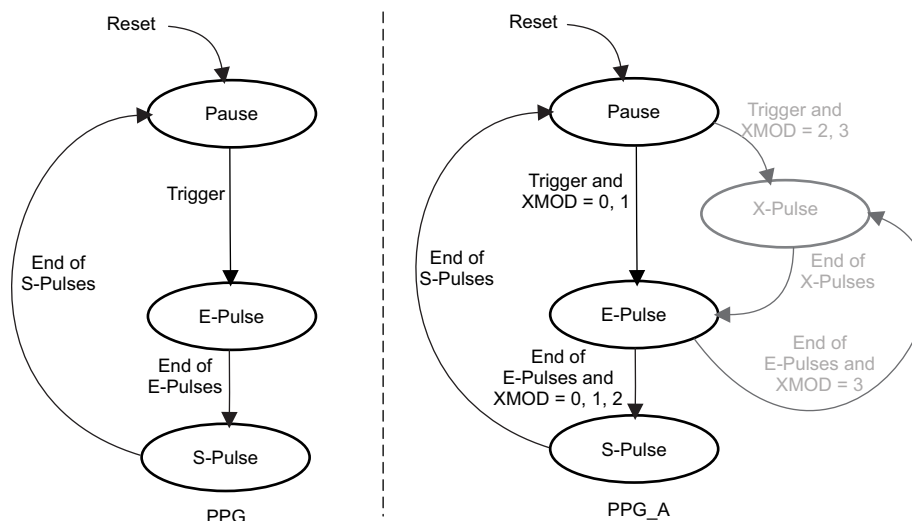


Figure 21-3. PPG or PPG_A Internal State Diagrams for Single Tone

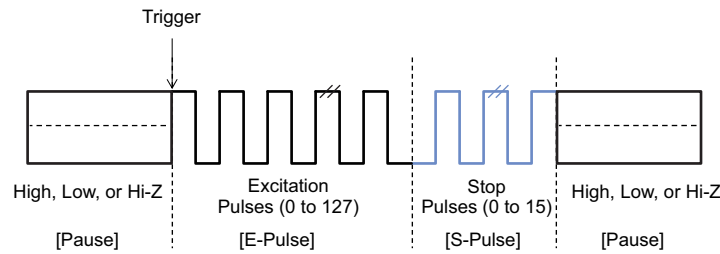


Figure 21-4. PPG Single Tone Generation With SAPHPGC.PPOL = 0 (Starts With High Polarity)

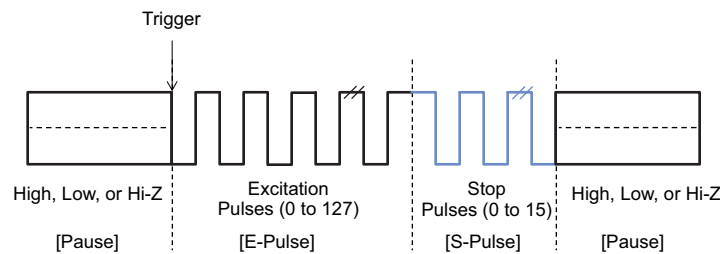
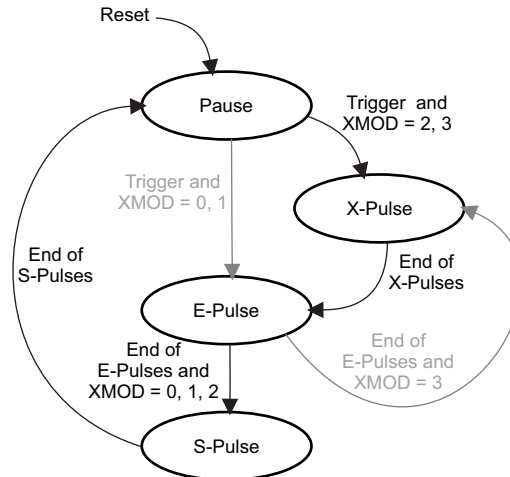
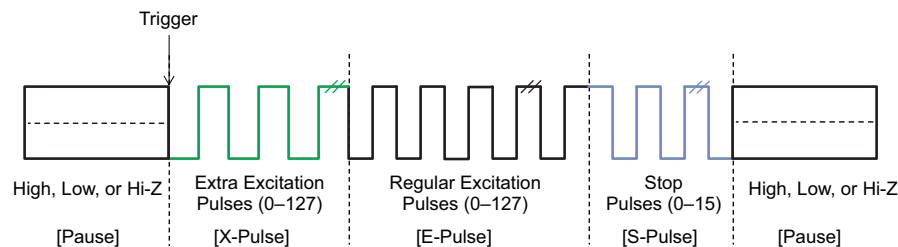


Figure 21-5. PPG Single Tone Generation With SAPHPGC.PPOL = 1 (Starts With Low Polarity)

21.2.3 Dual Tone Generation

The output pulses consists of five different phases: *Pause*, *Extra_Excitation*, *Regular_Excitation*, *Stop*, and *Pause* again. A state machine in PPG_A controls the flow. Set SAPH_AXPGCTL.XMOD = 2 for dual tone generation. When the PPG_A is triggered, it leaves the Pause phase, generates excitation pulses with a frequency defined by SAPH_AXPGHPER/SAPH_AXPGLPER, then excitation pulses with a frequency defined by SAPH_APGHPER/SAPH_APGLPER followed by stop pulses, then goes to Pause phase again (see Figure 21-6 and Figure 21-7). The stop pulses have a 180° phase shift compared to the regular excitation pulses. The stop pulses have same frequency as the regular excitation pulses. The PPG generates up to 127 extra excitation pulses, up to 127 regular excitation pulses and up to 15 stop pulses, which are controlled by the SAPH_AXPGCTL.XPULS, SAPH_APGC.EPULS and SAPH_APGC.SPULS bits, respectively. The pulse polarity is programmable in the SAPH_APGC.PPOL bit. The signal polarity of Pause can be programmed to be logical high, logical low, or high impedance through the SAPH_APGC.PLEV and SAPH_APGC.PHIZ bits.

The PPG_A can be triggered by writing 1 to the SAPH_APPGTRIG.PPGTRIG bit when SAPH_APGCTL.TRSEL = 0 (register mode) or by the acquisition sequencer (ASQ) when SAPH_APGCTL.TRSEL = 1 (auto mode). To avoid unintended pulse outputs, keep SAPH_APGCTL.PPGEN = 0 while updating the PPG_A registers. After the PPG_A registers are updated, write 1 to the SAPH_APGCTL.PPGEN bit before triggering the PPG_A. The SAPH_APGCTL.PPGEN bit must be set before triggering the PPG_A. The output channel is determined by the SAPH_APGCTL.PPGCHSEL bit when SAPH_APGCTL.PGSEL = 0 (register mode) or by the acquisition sequencer (ASQ) when SAPH_APGCTL.PGSEL = 1 (auto mode). Another layer of output control is inside the PHY, so both blocks must be configured properly (see Section 21.3). The PPG_A automatically stops when it completes generating the pulses. To stop generating pulses before completion, regardless of operation mode, write 1 to SAPH_APGCTL.STOP. The PPG_A immediately stops generating pulses. The SAPH_APGCTL.STOP bit is automatically cleared to zero.


Figure 21-6. PPG_A State Diagram for Dual Tone

Figure 21-7. PPG_A Dual Tone Generation With SAPHPGC.PPOL = 1 (Starts With Low Polarity)

21.2.4 Trill Tone Generation

The output pulses consists of multiple phases: *Pause*, a software defined set of *Extra_Excitation* and *Regular_Excitation* phases, *Stop*, and *Pause* again. A state machine in PPG_A controls the flow. Set `SAPH_AXPGCTL.XMOD = 3` for trill tone generation. When the PPG_A is triggered, it leaves *Pause* phase and generates extra excitation pulses with a frequency defined by `SAPH_AXPGHPER/SAPH_AXGPLPER`, then regular excitation pulses with a frequency defined by `SAPH_APGHPER/SAPH_APGLPER`. While `XMOD = 3` the extra excitation pulses followed by the regular excitation pulses is repeated. Software is required to change `SAPH_AXPGCTL.XMOD = 2` to terminate the trill. The last regular excitation pulses are followed by stop pulses. Then PPG_A goes to *Pause* phase again (see [Figure 21-8](#) and [Figure 21-9](#)). The stop pulses have a 180° phase shift compared to the last regular excitation pulses. The stop pulses have same frequency as the last regular excitation pulses. The PPG generates up to 127 extra excitation pulses, up to 127 regular excitation pulses and up to 15 stop pulses, which are controlled by the `SAPH_AXPGCTL.XPULS`, `SAPH_APGC.EPULS` and `SAPH_APGC.SPULS` bits, respectively. The pulse polarity is programmable in the `SAPH_APGC.PPOL` bit. The signal polarity of *Pause* can be programmed to be logical high, logical low, or high impedance through the `SAPH_APGC.PLEV` and `SAPH_APGC.PHIZ` bits.

The PPG_A can be triggered by writing 1 to the `SAPH_APPGTRIG.PPGTRIG` bit when `SAPH_APGCTL.TRSEL = 0` (register mode) or by the acquisition sequencer (ASQ) when `SAPH_APGCTL.TRSEL = 1` (auto mode). To avoid unintended pulse outputs, keep `SAPH_APGCTL.PPGEN = 0` when preparing the PPG_A registers. After the PPG_A registers are prepared, write 1 to the `SAPH_APGCTL.PPGEN` bit before triggering the PPG_A. The `SAPH_APGCTL.PPGEN` bit must be set before triggering the PPG_A. The output channel is determined by the `SAPH_APGCTL.PPGCHSEL` bit when `SAPH_APGCTL.PGSEL = 0` (register mode) or by the acquisition sequencer (ASQ) when `SAPH_APGCTL.PGSEL = 1` (auto mode). Another layer of output control is inside the PHY, so both blocks must be configured properly (see [Section 21.3](#)). After software sets `XMOD = 2` by directly writing or using the DMA, the PPG_A automatically stops when it completes

generating the pulses. Setting XMOD = 2 must be done when entering the last X-Pulse state. The current PPG_A state is indicated in SAPH_AXPGCTL.XSTAT. Leave at least 6 CPU clocks time margin for synchronizing. To stop generating pulses before completion, regardless of operation mode, write 1 to SAPH_APGCTL.STOP. The PPG_A immediately stops generating pulses. The SAPH_APGCTL.STOP bit is automatically cleared to zero.

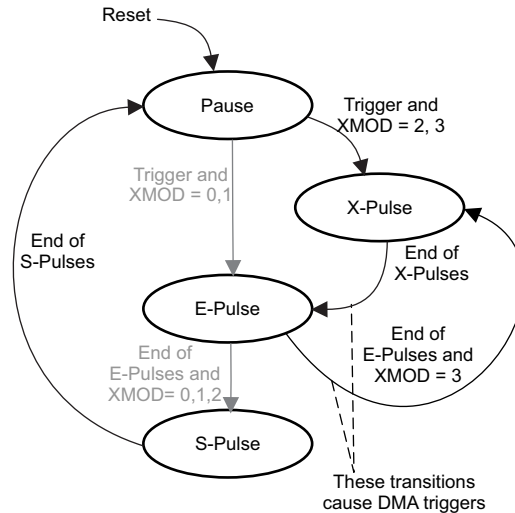


Figure 21-8. PPG_A State Diagram for Trill Tone

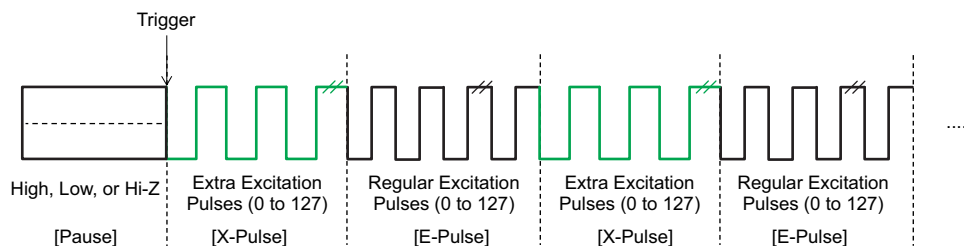


Figure 21-9. PPG_A Trill Tone Generation With SAPHPGC.PPOL = 1 (Starts With Low Polarity)

21.2.5 Multi Tone Generation

The output pulses consists of multiple phases: *Pause*, a software defined set of *Extra_Excitation* and *Regular_Excitation* phases, *Stop*, and *Pause* again. A state machine in PPG_A controls the flow. Set SAPH_AXPGCTL.XMOD = 3 like for trill tone generation. When the PPG_A is triggered, it leaves the Pause phase, generates extra excitation pulses with a frequency defined by SAPH_AXPGHPER/SAPH_AXPGGLPER, then regular excitation pulses with a frequency defined by SAPH_APGHPER/SAPH_APGLPER. While XMOD = 3 the extra excitation pulses followed by the regular excitation pulses is repeated. Software is required to change the following:

SAPH_AXPGHPER, SAPH_AXPGGLPER, SAPH_AXPGCTL.XPULS,
SAPHPGHPER, SAPHPGGLPER, SAPHPGCTL.EPULS, SAPHPGCTL.SPULS,

in a well timed manner. This can be done by direct register access or by DMA accesses.

Finally setting SAPH_AXPGCTL.XMOD = 2 to terminates the multi tone. The last regular excitation pulses are followed by stop pulses. Then PPG_A goes to Pause phase again (see Figure 21-10 and Figure 21-9). The stop pulses have a 180° phase shift compared to the last regular excitation pulses. The stop pulses have same frequency as the last regular excitation pulses. The PPG generates up to 127 extra excitation pulses, up to 127 regular excitation pulses and up to 15 stop pulses, which are controlled by the SAPH_AXPGCTL.XPULS, SAPH_APGC.EPULS and SAPH_APGC.SPULS bits, respectively. The pulse polarity is programmable in the SAPH_APGC.PPOL bit. The signal polarity of Pause can be programmed to be logical high, logical low, or high impedance through the SAPH_APGC.PLEV and SAPH_APGC.PHIZ bits.

The PPG_A can be triggered by writing 1 to the SAPH_APPGTRIG.PPGTRIG bit when SAPH_APGCTL.TRSEL = 0 (register mode) or by the acquisition sequencer (ASQ) when SAPH_APGCTL.TRSEL = 1 (auto mode). To avoid unintended pulse outputs, keep SAPH_APGCTL.PPGEN = 0 while preparing the PPG_A registers. After the PPG_A registers are prepared, write 1 to the SAPH_APGCTL.PPGEN bit before triggering the PPG_A. The SAPH_APGCTL.PPGEN bit must be set before triggering the PPG_A. The output channel is determined by the SAPH_APGCTL.PPGCHSEL bit when SAPH_APGCTL.PGSEL = 0 (register mode) or by the acquisition sequencer (ASQ) when SAPH_APGCTL.PGSEL = 1 (auto mode). Another layer of output control is inside the PHY, so both blocks must be configured properly (see [Section 21.3](#)). After software sets XMOD = 2 by directly writing or using the DMA, the PPG_A automatically stops when it completes generating the pulses. Setting XMOD = 2 must be done when entering the last X-Pulse state. The current PPG_A state is indicated in SAPH_AXPBGCTL.XSTAT Leave at least 6 CPU clocks time margin for synchronizing. To stop generating pulses before completion, regardless of operation mode, write 1 to SAPH_APGCTL.STOP. The PPG_A immediately stops generating pulses. The SAPH_APGCTL.STOP bit is automatically cleared to zero.

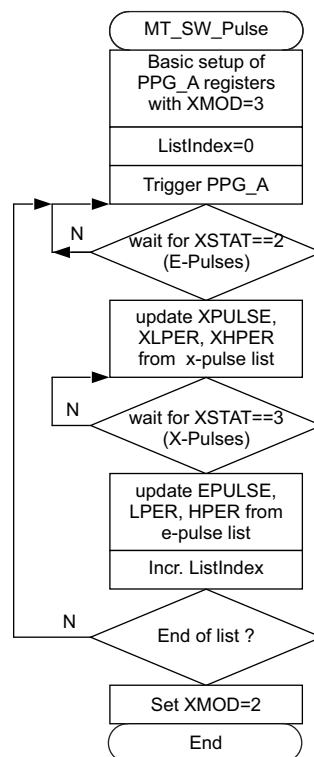


Figure 21-10. PPG_A Software Flow Chart for Multi Tone

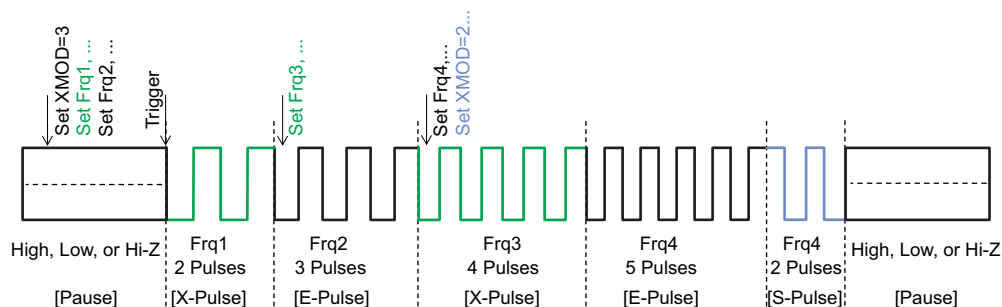


Figure 21-11. PPG_A Multi Tone Generation With SAPHPGC.PPOL = 1 (Starts With Low Polarity)

21.2.6 Excitation Pulse Frequency on PPG or PPG_A

The excitation pulse frequency is determined by the PPG clock frequency and the SAPHPGLPER.LPER and SAPHPGHPER.HPER bits. The high duration of a regular excitation pulse and low duration of a regular excitation pulse are determined by SAPHPGHPER.HPER (8 bits) and SAPHPGLPER.LPER (8 bits), respectively. Thus:

- Excitation pulse frequency = HSPLL frequency / (SAPHPGHPER.HPER + SAPHPGLPER.LPER)

21.2.7 Extra Excitation Pulse Frequency on PPG_A

The pulse frequency for the X-Pulses is determined by the PPG_A clock frequency and the SAPH_AXPGHPER.XHPER and SAPH_AXPGGLPER.XLPER bits. The high duration of an extra excitation pulse and low duration of an extra excitation pulse are determined by SAPH_AXPGHPER.XHPER (8 bits) and SAPH_AXPGGLPER.XLPER (8 bits), respectively. Thus:

- X-Pulse frequency = HSPLL frequency / (SAPH_AXPGHPER.XHPER + SAPH_AXPGGLPER.XLPER)

21.2.8 Test Tone Generation

To start the PPG test tone generation, write SAPHPGCTL.TONE = 1. While SAPHPGCTL.TONE = 1, the PPG consistently generates excitation pulses (no stop pulse). In this case, SAPHPGC.EPULSE and SAPHPCG.SPULSE are ignored. To terminate the PPG test tone generation, write SAPHPGCTL.TONE = 0.

21.3 Physical Interface (PHY) Block

The PHY controls the input and output pins of the USS module. The USS module has dedicated pins (CH0_OUT, CH0_IN, CH1_OUT, CH1_IN, two PVSS, and PVCC), which are controlled by the USS module, not by the digital I/O module.

21.3.1 Output Channels (CH0_OUT and CH1_OUT)

Figure 21-12 shows the functional block diagram for the output channels, CH0_OUT and CH1_OUT.

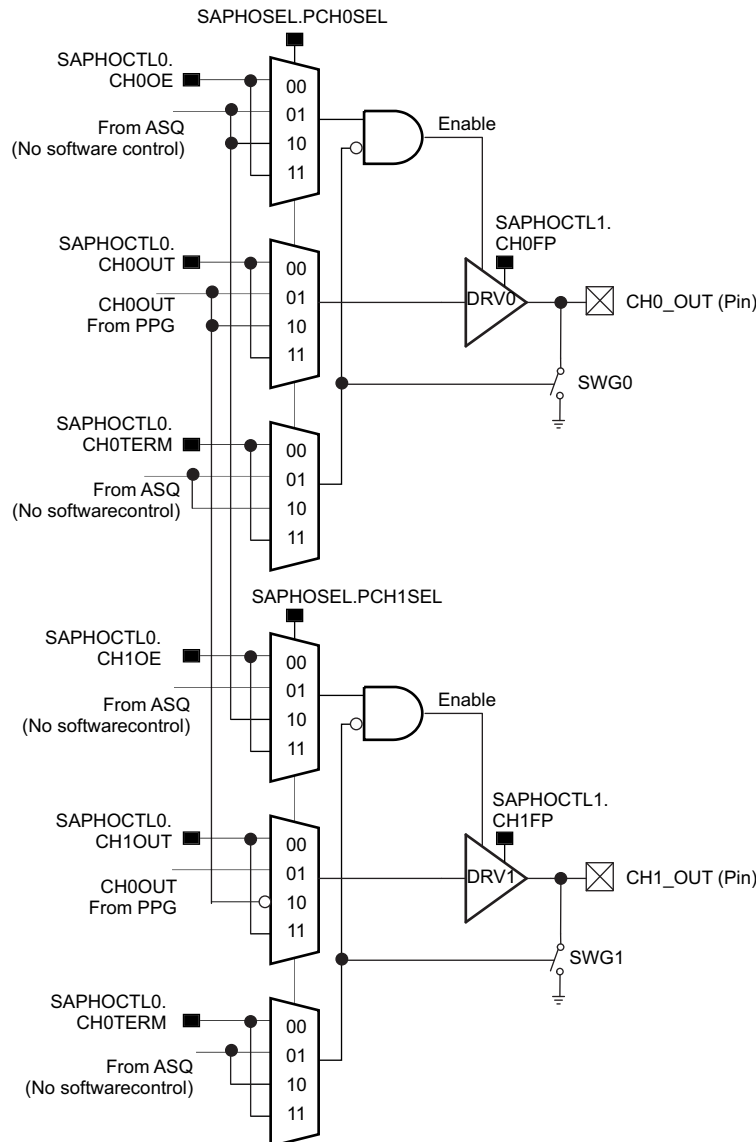


Figure 21-12. PHY Output Pins

The output pins (CH0_OUT and CH1_OUT) can be used as general I/O pins when SAPHOSEL.PCH0SEL = 0 or 3. In this case, the polarity of the two pins is controlled by SAPHOCTL0.CH0OUT and SAPHOCTL0.CH1OUT, and the pins are enabled or disabled (Hi-Z) by SAPHOCTL0.CH0OE and SAPHOCTL0.CH1OE. The pins can be connected to GND through the SWG0 and SWG1 switches by SAPHOCTL0.CH0TERM and SAPHOCTL0.CH1TERM, respectively.

When SAPHOSEL.PCH0SEL = 1, the output pins work as single output drivers. The pulses generated by PPG are output on the selected pin. This is the typical use case for the flow measurement application.

When SAPHOSEL.PCH0SEL = 2, the two output pins working as differential output drivers. The pulses generated by PPG are output on both pins differentially.

When SAPHOSEL.PCH0SEL = 1 or 2, the two pins are controlled by the PPG block and ASQ block. The pins are automatically enabled when PPG generates pulses. The SWG0 and SWG1 switches are controlled by ASQ as part of measurement sequence (see [Section 21.4](#)).

The drive strength of DRV0 and DRV1 are maximized when SAPHOCTL1.CH0FP = 1 and SAPHOCTL1.CH1FP = 1, respectively. The drive strength of the pins are determined by SAPHCH0PUT, SAPHCH0PDT, SAPHCH1PUT, and SAPHCH1PDT when SAPHOCTL1.CH0FP = 0 and SAPHOCTL1.CH1FP = 0, respectively (see [Section 21.3.2](#)).

When SAPHOSEL.PCH0SEL = 1 or 2, the two pins are controlled by the PPG block and ASQ block. The pins are automatically enabled when PPG generates pulses. The SWG0 and SWG1 switches are controlled by ASQ as part of measurement sequence (see [Section 21.4](#)).

21.3.2 Trim Registers for the Output Drivers and Termination Resistors

The output channels (CH0_OUT and CH1_OUT) have low-impedance output drivers (DRV0 and DRV1) and termination switches (SWG0 and SWG1). To support the impedance-matching requirement in application environments using ultrasonic technology, programmability is offered to the output drive strength of the drivers (DRV0 and DRV1) and the termination switches (SWG0 and SWG1). The drivers are based on inverter architecture, which consists of PMOS and NMOS. Thus, three trim registers are offered for each channel (see [Table 21-1](#) for details).

During manufacturing, optimal impedance of the drivers and termination switches are determined and their trim values are stored to the device boot data memory (not user accessible). The output impedance of DRV0 and DRV1 are trimmed to match each other (with the lowest possible value), and the termination switches (SWG0 and SWG1) are trimmed to match the impedance of each driver. During the boot process, the trim values are written to the trim registers by bootcode. The default trim values may be different from device to device. Programmability is offered if different impedance values are preferred in a specific application environment.

Table 21-1. Trim Registers

Register	Description	Trim Range (Typical) ⁽¹⁾
SAPHCH0PUT.CH0PUT	PMOS trim bits (4 bits) for the channel 0 output driver	15 = lowest ($\approx 2.5 \Omega$) 0 = highest Each step $\approx 3\%$
SAPHCH0PDT.CH0PDT	NMOS trim bits (4 bits) for the channel 0 output driver	15 = lowest ($\approx 2.5 \Omega$) 0 = highest Each step $\approx 3\%$
SAPHCH0TT	Termination switch trim bits (4 bits) for channel 0	15 = lowest ($\approx 2.5 \Omega$) 0 = highest Each step $\approx 3\%$
SAPHCH1PUT.CH1PUT	PMOS trim bits (4 bits) for the channel 1 output driver	15 = lowest ($\approx 2.5 \Omega$) 0 = highest Each step $\approx 3\%$
SAPHCH1PDT.CH1PDT	NMOS trim bits (4 bits) for the channel 1 output driver	15 = lowest ($\approx 2.5 \Omega$) 0 = highest Each step $\approx 3\%$
SAPHCH1TT	Termination switch trim bits (4 bits) for channel 1	15 = lowest ($\approx 2.5 \Omega$) 0 = highest Each step $\approx 3\%$

⁽¹⁾ See the device-specific data sheet for details.

The trim registers are written with the default values during every boot up; thus, if different trim values are preferred, the values must be written to the trim registers by software after every boot.

NOTE: To avoid unintended writes to the trim registers, the SAPHTACTL.UNLOCK bit can block write access to the trim registers. The trim registers are locked when SAPHTACTL.UNLOCK = 0.

21.3.3 Input Channels (CH0_IN and CH1_IN)

Figure 21-13 shows the functional block diagram for the input channels, CH0_IN and CH1_IN.

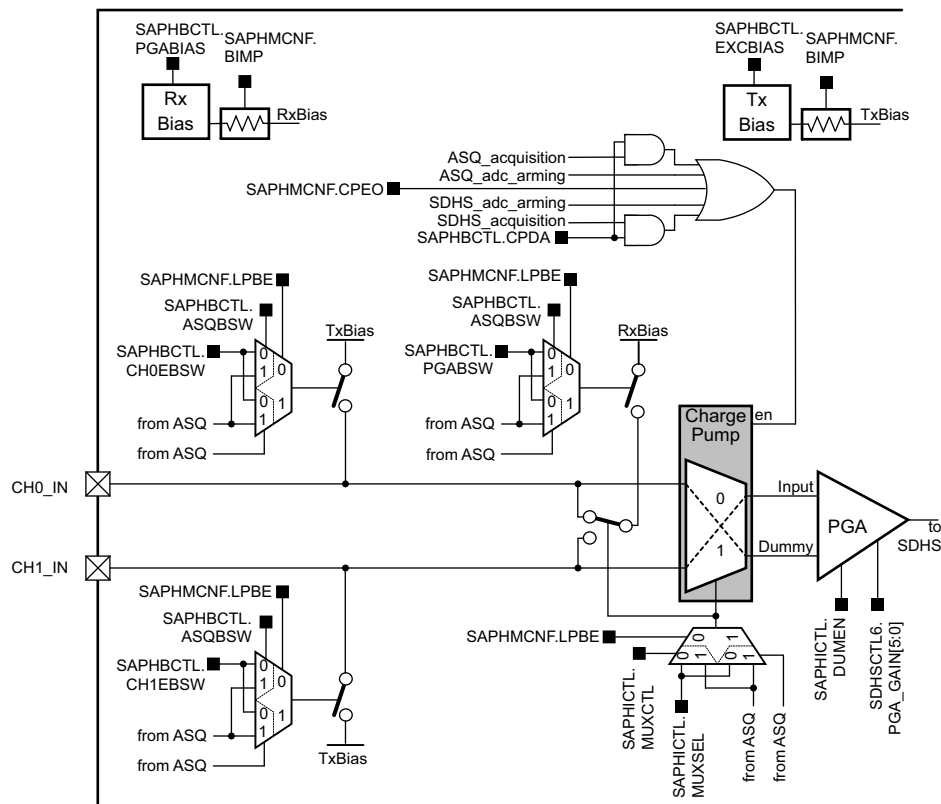


Figure 21-13. SAPH or SAPH_A Analog Input Signal Chain

The PGA uses only one input channel at a time, and the input channel is selected by SAPHICTL0.MUXSEL when SAPHICTL0.MUXCTL = 0 and SAPHMCNF.LPBE = 0 (register mode) or by the acquisition sequencer (ASQ) when SAPHICTL0.MUXCTL = 1 and SAPHMCNF.LPBE = 0 (auto mode). Register mode uses the SAPHOCTL0, SAPHOCTL1, SAPHOSEL, and SAPHBCTL registers to control the signal chain. Auto mode uses the SAPHASCTL0, SAPHASCTL1, SAPHAPOL, SAPHAPLEV and SAPHAHIZ registers to control the signal chain.

In Ultra Low Power Bias mode (SAPHMCNF=1) both register sets are and the multiplexer selection is controlled by the ASQ.

The PGA has two input channels, but one of the channels is a dummy input, which has the same input impedance of the real input. The dummy input is designed to meet the impedance match requirement between transmit mode and receive mode of one channel. The dummy impedance is enabled when SAPHICTL0.DUMEN = 1 (default).

The input multiplexer is powered by a charge pump, which generates 3.2 V from the USS LDO voltage (1.6 V). The charge pump is turned on while the SAPHMCNF.CPEO=1 and during the preparation/arming for SDHS acquisition. If desired, the charge pump is turned off while capturing the Rx signal through the SDHS to reduce noise (SAPHBCTL.CPDA = 0); however, if signal capture is more than 300 μ s, the charge pump should remain on (SAPHBCTL.CPDA = 1). Configure SAPHBCTL.CPDA before starting a measurement sequence. The clock of the charge pump is generated from a divided SDHS modulator frequency to maintain coherence during acquisition.

The impedance of the bias voltages TxBias and RxBias can be programmed with SAPHMCNF.BIMP. This allows to find the best fit for the reactant behavior of transducers on bias voltage changes (shorter ringing).

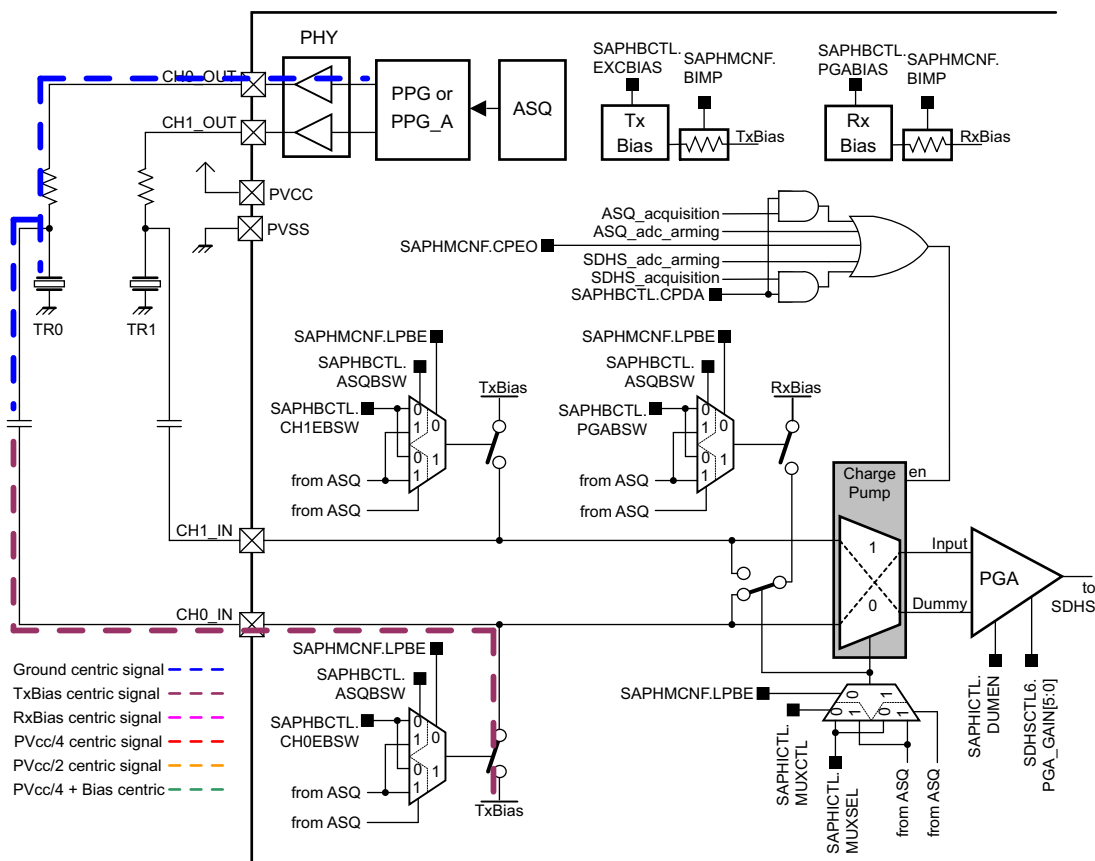
Table 21-2. Supply to the Rx Multiplexer

Register	Description	Recommended Data Capture Time (Typical)
SAPHBCTL.CPDA = 0 (default)	The charge pump is automatically off when data capture begins.	<300 μ s
SAPHBCTL.CPDA = 1	The charge pump remains on during data capture.	$\geq$ 300 μ s

21.3.3.1 Tx Bias and Rx Bias

During a measurement sequence, bias voltages must be applied appropriately to the transmit signal and receive signal. In auto mode, the entire measurement sequence including applying appropriate bias voltages is fully controlled by the ASQ. In register mode, the application must apply the correct bias voltages during a measurement sequence. Figure 21-13 shows the two bias voltage sources, Tx bias and Rx bias. The Tx bias must be applied to the output channel before generating excitation pulses. The Rx bias must be applied to the input channel before capturing the input signal. A typical measurement sequence is as follows [assuming that excitation (Tx) at CH0 and reception (Rx) at CH1]:

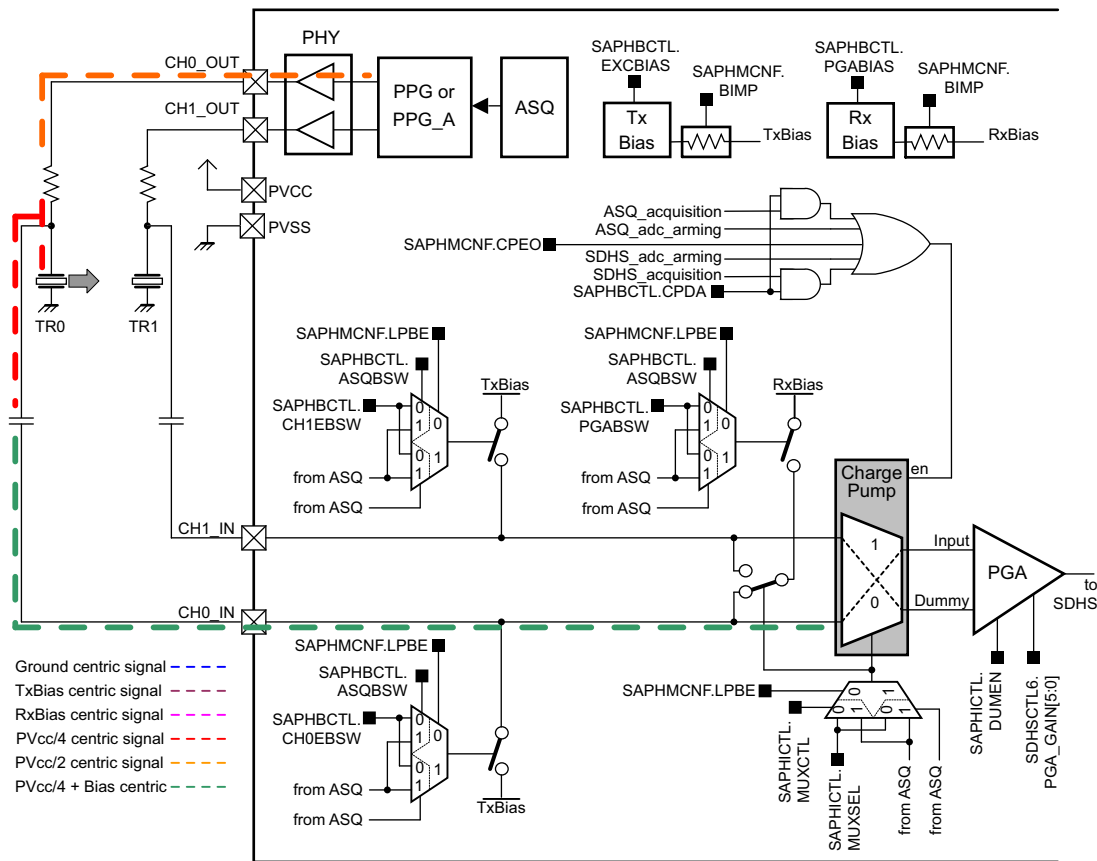
1. Before generating excitation pulses and stop pulses, apply the Tx bias to CH0_IN while CH0_OUT is connected to GND through the SWG0 switch (see Figure 21-14 and Figure 21-12).



NOTE: Apply Tx bias to CH0_IN while CH0_OUT is connected to GND through SWG0.

Figure 21-14. Before Excitation

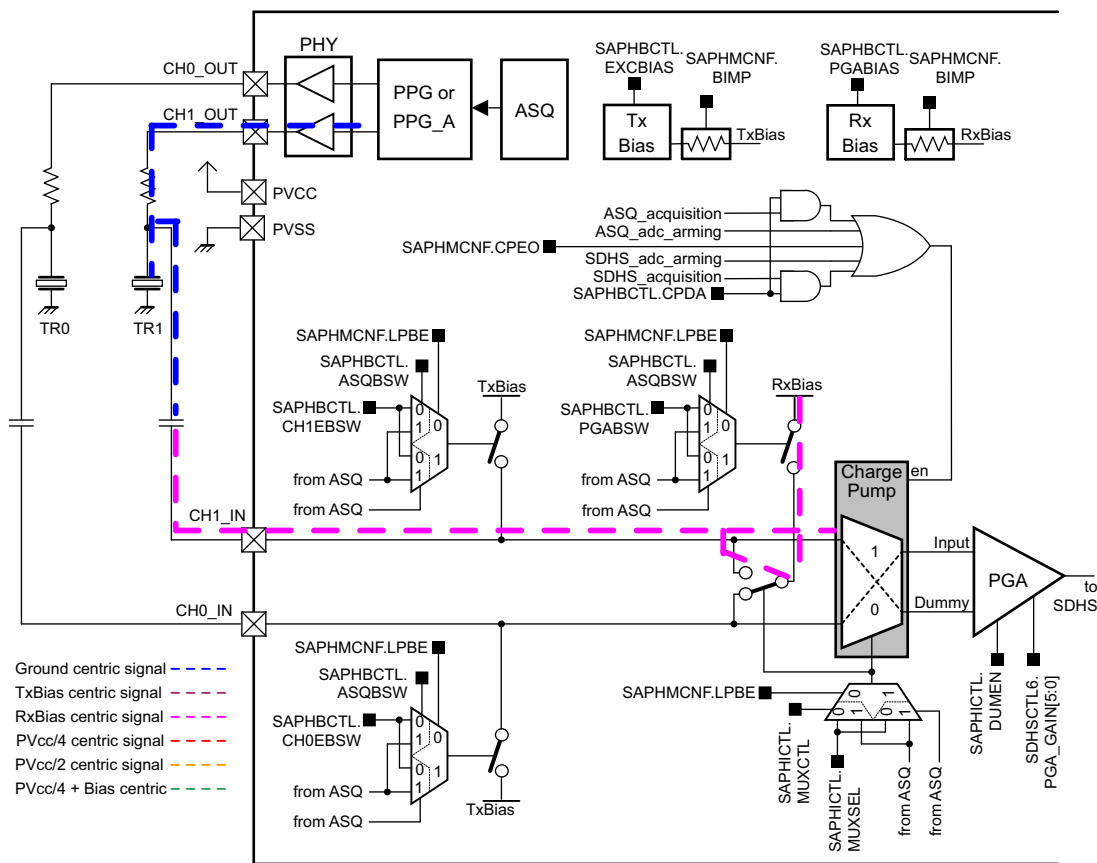
2. Generate excitation pulses and stop pulses through CH0_OUT, and CH0_IN is connected to PGA dummy load to meet the impedance-matching requirement. The pulses are driven to the transducer 0 (TR0) (see Figure 21-15).



NOTE: Excitation pulses are driven on CH0_OUT, and CH0_IN is connected to the PGA dummy input.

Figure 21-15. Excitation

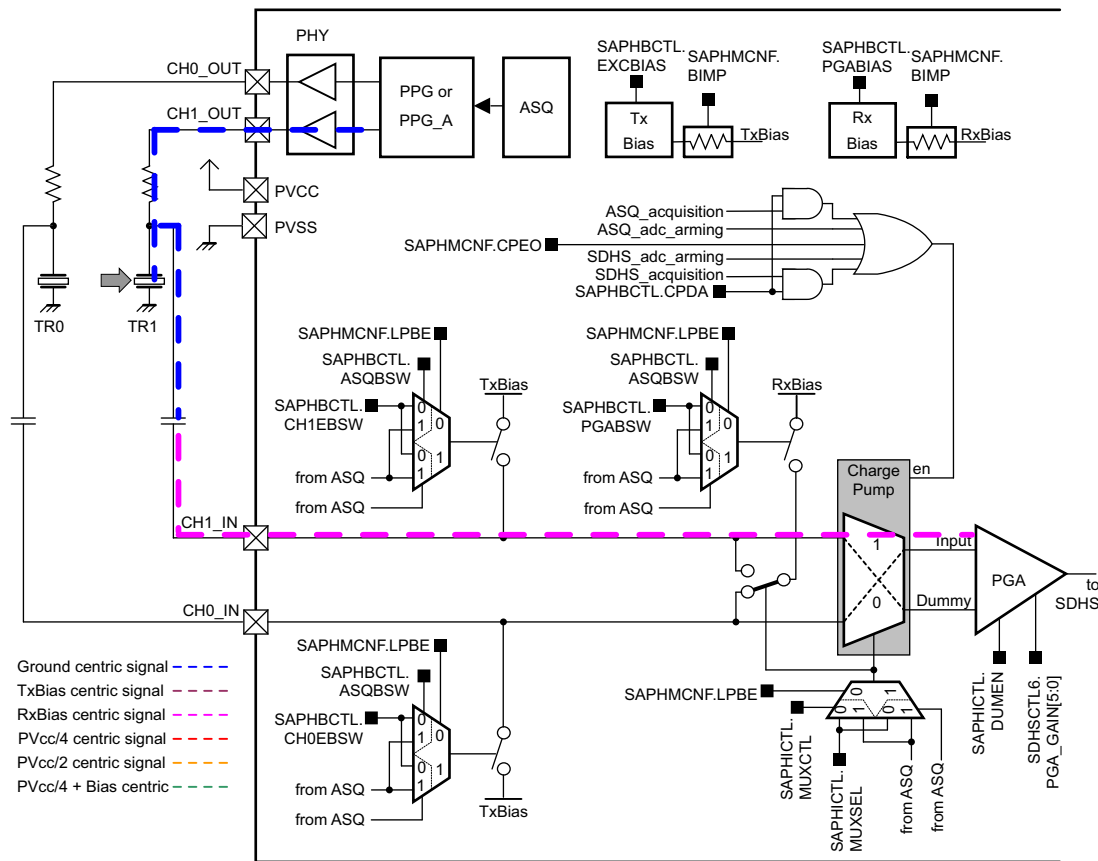
- Wait for almost the complete estimated time of flight (TOF) between the transducer 0 (TR0) and the transducer 1 (TR1). The delay must be less than TOF to capture the beginning of the arriving pulse train.
- Before the signal arrives at the transducer 1 (TR1), power on the SDHS ADC so that the SDHS is fully settled before sampling data.
- Before the signal arrives at the transducer 1 (TR1), apply the Rx bias to CH1_IN while CH1_OUT is connected to GND through the SWG1 switch (see [Figure 21-16](#) and [Figure 21-12](#)).



NOTE: Apply Rx bias to CH1_IN while CH1_OUT is connect to GND through SWG1.

Figure 21-16. Before Reception

6. Trigger the SDHS to sample the arrived signal at the transducer 1 (TR1): CH1_IN is connected to PGA input for data capturing. CH1_OUT is connected to GND through the SWG1 switch (see [Figure 21-17](#)).



NOTE: CH1_IN is connected to the PGA input, and CH1_OUT is connected to GND through the SWG1 switch.

Figure 21-17. Reception

The Tx bias voltage and the Rx bias voltage are programmable independently by SAPHBCTL.EXCBIAS and SAPHBCTL.PGABIAS, respectively.

21.3.4 External Bias (XPB0 and XPB1 on SAPH_A)

The SAPH_A module features external bias terminals XPB0 and XPB1. Set SAPH_AICTL0.XPB0FEN=1 to enable the analog bias function on the XPB0 terminal. Use SAPH_AICTL0.XPBSW0 to turn the bias switch on or off. If SAPH_AICTL0.XPB0FEN=0 the terminal is used as a regular digital port pin. On SAPH_AICTL0.XPB0FEN=1 and SAPH_AICTL0.XPBSW0=1 the PGA bias voltage is provided on XPB0. XPB1 works the same way with its corresponding control bits.

The XPB0 and XPB1 terminals provide auxiliary voltages for optional external analog circuits with high impedance inputs.

Note:!! The XPB0 and XPB1 control logic requires the HSPLL clock to be running for operation.

21.4 Acquisition Sequencer (ASQ)

The entire measurement sequence can be controlled by user software (called register mode), by ASQ without any CPU intervention (called auto mode) or by ASQ without any CPU intervention (called ultra low power bias mode). The auto mode helps reduce the measurement power consumption, because the CPU can stay in LPM0 during the measurement sequences. The ASQ is a state machine that can be used to control the entire measurement sequence. Figure 21-17 shows the ASQ block diagram. In auto mode a mixed operation of register control and ASQ control is possible for various signal chain elements. In ultra low power bias mode all control is obtained by the ASQ.

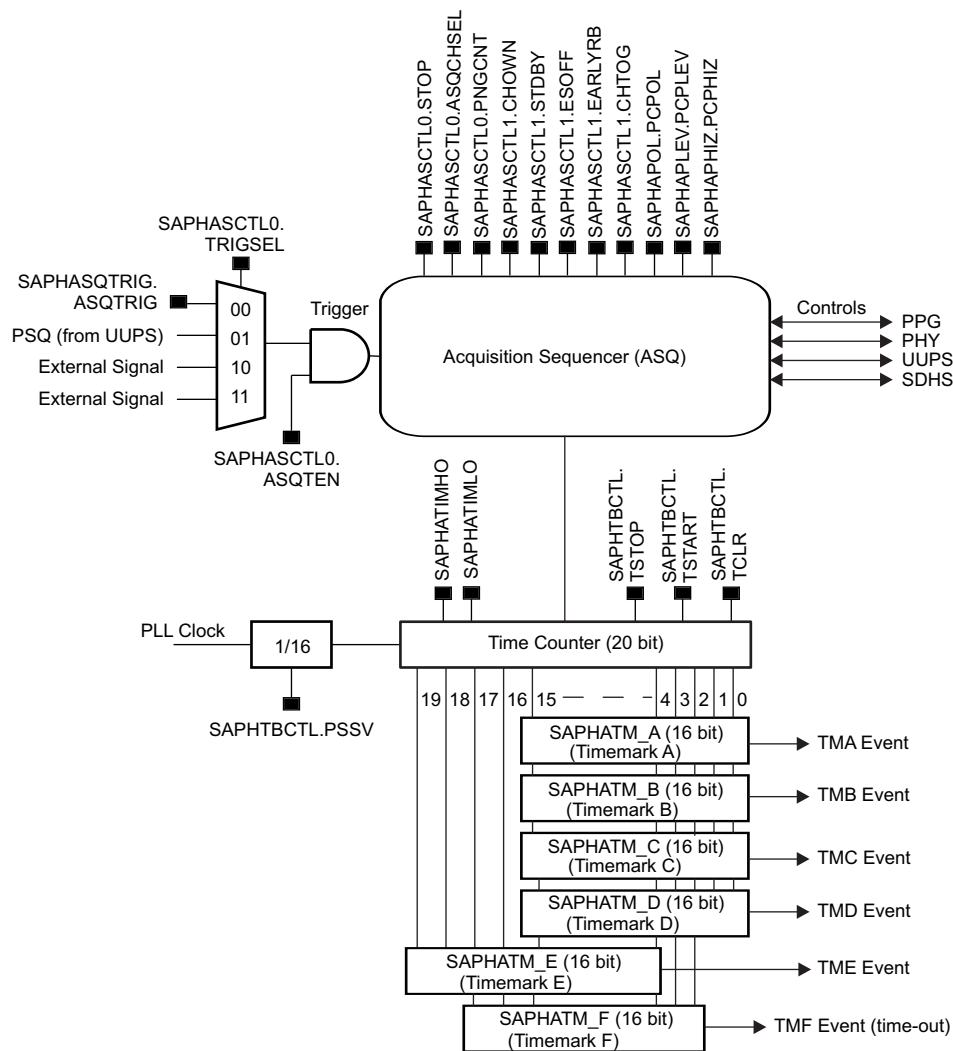


Figure 21-18. ASQ Block Diagram

21.4.1 Time Counter

Figure 21-17 shows that the ASQ has a 20-bit time counter running at 1/16 of the PLL clock. The counter value can be read from the read-only registers, SAPHATIMLO (low 16 bits) and SAPHATIMHI (high 4 bits). The time counter is controlled by the ASQ during measurement in auto mode; however, the time counter can be forced to stop (SAPHATBCTL.TSTOP = 1), reset (SAPHATBCTL.TCLR = 1), and start (SAPHATBCTL.TSTART = 1) by user software if necessary. Controlling the time counter by user software is not recommended while ASQ is actively controlling the measurement sequences, because these changes interfere with the measurement timings.

21.4.2 Six Time Mark Events

The six time mark registers are to generate important timing events during measurement in auto mode. See [Table 21-3](#) for details about the time mark events.

Table 21-3. Time Mark Events

Register	Time Counter Bits	Event Name	Action Done by ASQ
SAPHASQTRIG.ASQTRIG or signal from PSQ	N/A	ASQ Trigger	Apply Rx bias
SAPHATM_A	[15: 0]	TMA	Trigger PPG to generate excitation pulses and stop pulses
SAPHATM_B	[15: 0]	TMB	Turn on the SDHS
SAPHATM_C	[15: 0]	TMC	Apply Rx bias
SAPHATM_D	[15: 0]	TMD	Start sampling the input signal (trigger the SDHS)
SAPHATM_E	[19: 4]	TME	Restart the time counter for the next measurement
SAPHATM_F	[17: 2]	TMF	Time-out

The order of the time mark events is determined by the counter values written to the SAPHATM_x registers.

- Time mark event period = PLL clock $\times$ 1/16 $\times$ value in the time mark registers (for TMA, TMB, TMC, and TMD)
- TME event period = PLL clock $\times$ 1/16 $\times$ value in the time mark register $\times$ 16
- TMF event period = PLL clock $\times$ 1/16 $\times$ value in the time mark register $\times$ 4

21.4.3 Triggering the ASQ

There are two ways to start the ASQ. One is user software trigger by writing 1 to SAPHASQTRIG.ASQTRIG, and the other is auto trigger by the PSQ from the UUPS module after the USS power up is complete. When SAPHASCTL0.TRIGSEL = 0, SAPHASQTRIG.ASQTRIG is selected as the trigger source of the ASQ. When SAPHASCTL0.TRIGSEL = 1, the signal from the PSQ is selected as the trigger source of the ASQ. In this case, configure all USS registers before turning on the USS module, so that the entire measurement process can be fully automated by the PSQ and ASQ.

To avoid undesired ASQ triggers, keep SAPHASCTL0.ASQTEN = 0 while configuring ASQ registers. After updating the ASQ registers, write SAPHASCTL0.ASQTEN = 1 before triggering the ASQ. The SAPHASCTL0.ASQTEN bit must be set before triggering the ASQ.

21.4.4 Auto Mode and Register Mode

The entire measurement sequence can be controlled by user software (called register mode) or by the ASQ without any CPU intervention (called auto mode). See [Table 21-4](#) for differences. Configuration of each submodule is the same and is not listed in [Table 21-4](#).

Table 21-4. Auto Mode and Register Mode

Action	Auto Mode	Control Register in Auto Mode	Register Mode	Control Register in Register Mode
Take USS submodules out of reset (once after reset)	UUPSCTL.SWRST = 0	UUPSCTL.SWRST	UUPSCTL.SWRST = 0	UUPSCTL.SWRST
Enable auto mode or register mode (on mode changes)	SAPHCMCNF.LPBE = 0	SAPHCMCNF.LPBE	SAPHCMCNF.LPBE = 0	SAPHCMCNF.LPBE
Set TxBias and RxBias impedance	SAPHCMCNF.BIMP = 0, 1, 2, 3	SAPHCMCNF.BIMP	SAPHCMCNF.BIMP = 0, 1, 2, 3	SAPHCMCNF.BIMP
Drive output drivers to GND	SAPHOSEL.PCH0SEL = 1	N/A (automatic)	SAPHOSEL.PCH0SEL = 0	SAPHOCTL0.CH0TERM, SAPHOCTL0.CH1TERM
Tx bias	SAPHBCTL.ASQBSW = 1	N/A (automatic)	SAPHBCTL.ASQBSW = 0	SAPHBCTL.CH0EBSW, SAPHBCTL.CH1EBSW
Select output channel in PPG	SAPHPGCTL.PGSEL = 1	SAPHASCTL0.ASQCHSEL, SAPHASCTL1.CHTOG	SAPHPGCTL.PGSEL = 0	SAPHPGCTL.PPGCHSEL
Trigger PPG	SAPHPGCTL.TRSEL = 1	SAPHATM_A	SAPHPGCTL.TRSEL = 0	SAPHPGTRIG.PPGTRIG
Input channel selection	SAPHICTL0.MUXCTL = 1	SAPHASCTL0.ASQCHSEL, SAPHASCTL1.CHTOG, SAPHASCTL1.CHOWN	SAPHICTL0.MUXCTL = 0	SAPHICTL0.MUXSEL
Turn on the SDHS	SDHSCTL0.TRGSRC = 1	SAPHATM_B	SDHSCTL0.TRGSRC = 0	SDHSCTL4.SDHSON
Rx bias	SAPHBCTL.ASQBSW = 1	SAPHATM_C or SAPHASCTL1.EARLYRB	SAPHBCTL.ASQBSW = 0	SAPHBCTL.PGABSW, SAPHICTL0.MUXSEL
SDHS conversion start	SDHSCTL0.TRGSRC = 1	SAPHATM_D	SDHSCTL0.TRGSRC = 0	SDHSCTL5.SSTART
The number of measurement	Triggering ASQ	SAPHASCTL0.PNGCNT (up to 4)	No ASQ trigger	N/A

The order of the time mark events is determined by the counter values written to the SAPHATM_x registers.

21.5 Ultra-Low-Power Bias Mode

The ultra-low bias power mode suppresses the PLL operation during the hold-off delay phase on USS startup. Because the PLL and its clock tree operate at a high frequency, considerable energy savings can be achieved. Common utility flow meters that enter LPM3 or LPM3.5 between the measurements with turned off USSXT benefit in particular from that mode.

Figure 21-19 shows a typical use case with register mode or auto mode. The timing axis and the amplitudes in the diagram are not to scale. Following a very short summary of that theoretical use cases...

After an RTC event at (a), the system wakes up from LPMx, runs boot code, configures the USS trimming, and enters application code. This turns on USSXT, prepares time-out on TimerA that is running from LFXCLK, then the CPU enters LPMx.

At (b), the TimerA service routine starts the UUPS, and the CPU enters LPMx again.

At (c), the transducers and external circuits may show ringing due to power up and bias voltage enable. The signals settle then.

At (d) the HSPLL generates a clock with almost no remaining phase variations. Near TA, the transducers have settled and the regular sequence is started by generating pulses. At TB, well before the arrival of the signal, the SDHS and its clock is enabled. At TD, the data acquisition starts.

At (e), the sequence is completed. The interrupt service routine performs calculations and sets the system to LPMx again.

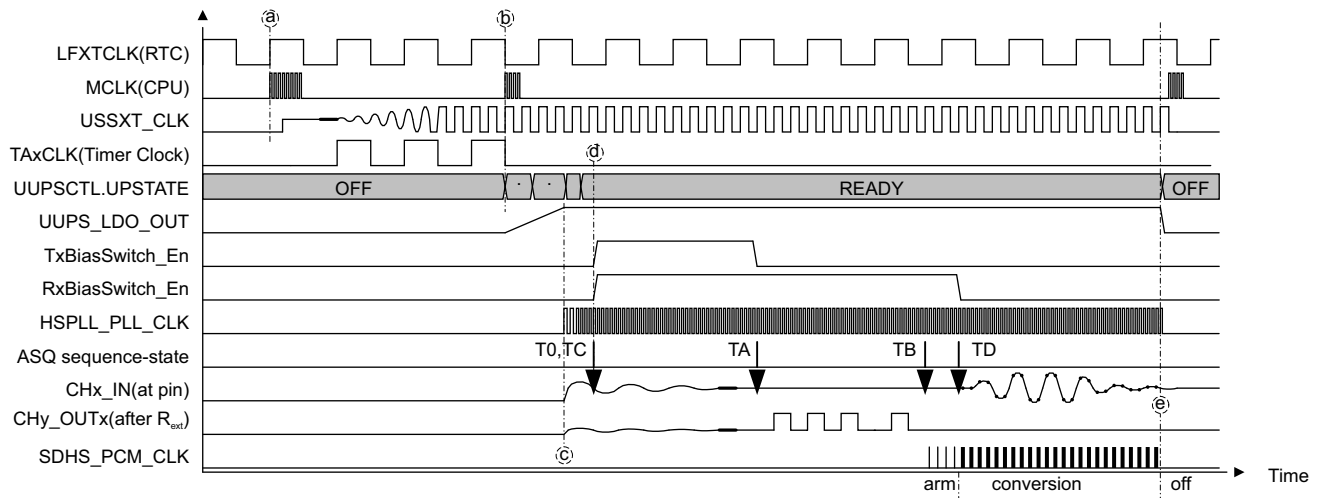


Figure 21-19. Auto Mode and Register Mode Example

Figure 21-20 shows a the use case with ultra-low-power bias mode. The timing axis and the amplitudes in the diagram are not to scale. The following is a short summary of that theoretical use case:

After an RTC event at (a), the system wakes up from LPMx, runs boot code, configures the USS trimming, and enters application code. This turns on USSXT, prepares time-out on TimerA that is running from LFXCLK, and then the CPU enters LPMx.

At (b), the TimerA service routine starts the UUPS with hold-off delay, and the CPU enters LPMx again. The UUPS state machine starts up with suppressed HSPLL.

At (c), the transducers and external circuits may show ringing due to power up and bias voltages. The signals settle then.

At (d), the HSPLL generates a clock with almost no remaining phase variations. Near TA, the transducers have settled and the regular sequence is started by generating pulses. At TB, well before the arrival of the signal, the SDHS and its clock are enabled. At TD, the data acquisition starts.

At (e), the sequence is completed. The interrupt service routine perform calculations and sets the system to LPMx again.

The ASQ.mux control signal switches between both control register sets during power up. Both register sets require same values of the corresponding control bits for a smooth bias generation and handover.

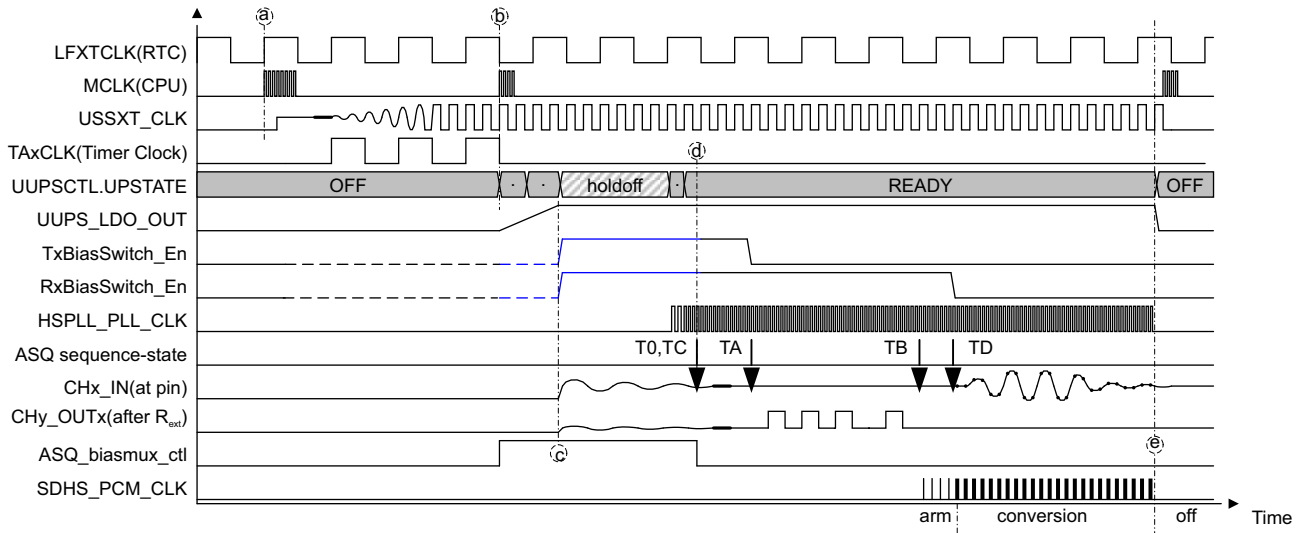


Figure 21-20. Ultra-Low-Power Bias Mode Example

21.6 Interrupts Triggers

The SAPH or SAPH_A module supports four interrupts:

- **PNGDN interrupt:** The interrupt occurs when the PPG completes pulse generation.
- **SEQDN interrupt:** The interrupt occurs when ASQ completes all of the measurements programmed in SAPHASCTL0.PNGCNT. For example, when SAPHASCTL0.PNGCNT = 3, four measurements are performed. The interrupt indicates that all four measurements have been completed.

NOTE: After ASQ is triggered, if the current measurement is interrupted by entering debug halt mode (UUPSRIS.STPBYDB = 1), the SEQDN interrupt is also reported.

- **TMFTO interrupt:** This interrupt is valid when ASQ is active (auto mode). The interrupt indicates that the time counter in ASQ has reached to TIMEMARK_F (time-out).
- **DATAERR interrupt:** This interrupt indicates that either WINHI interrupt or WINLO interrupt has occurred in SDHS. See [Chapter 22](#) for details.

21.7 DMA Triggers

The SAPH_A module supports a DMA trigger. The trigger is fired when the PPG_A state machine transitions from S_Pulse to X-Pulse and when the PPG_A state machine transitions from X-Pulse to S-Pulse.

21.8 SAPH and SAPH_A Registers

Table 21-5 lists the memory-mapped registers for SAPH/SAPH_A. All register offset addresses not listed in Table 21-5 should be considered as reserved locations and the register contents should not be modified.

Table 21-5. SAPH Registers

Offset	Acronym	Register Name	Type	Reset	Section
0h	SAPHIIDX SAPH_AIIDX	Interrupt Index	read-only	0h	Section 21.8.1
2h	SAPHMIS SAPH_AMIS	Masked Interrupt Satus	read-only	0h	Section 21.8.2
4h	SAPHRIS SAPH_ARIS	Raw Interrupt Status	read-only	0h	Section 21.8.3
6h	SAPHIMSC SAPH_AIMSC	Interrupt Mask	read-write	0h	Section 21.8.4
8h	SAPHICR SAPH_AICR	Interrupt Clear	write-only	0h	Section 21.8.5
Ah	SAPHISR SAPH_AISR	Interrupt Set	write-only	0h	Section 21.8.6
Ch	SAPHDESCLO SAPH_ADESCLO	Module-Descriptor Low Word	read-only	10h	Section 21.8.7
Eh	SAPHDESCHI SAPH_ADESCHI	Module-Descriptor High Word	read-only	5553h	Section 21.8.8
10h	SAPHKEY SAPH_AKEY	Key	write-only	0h	Section 21.8.9
12h	SAPHOCTL0 SAPH_AOCTL0	Physical Interface Output Control #0	read-write	0h	Section 21.8.10
14h	SAPHOCTL1 SAPH_AOCTL1	Physical Interface Output Control #1	read-write	0h	Section 21.8.11
16h	SAPHOSEL SAPH_AOSEL	Physical Interface Output Function Select	read-write	5h	Section 21.8.12
20h	SAPHCH0PUT SAPH_ACH0PUT	Channel 0 Pull UpTrim Register	read-write	0h	Section 21.8.13
22h	SAPHCH0PDT SAPH_ACH0PDT	Channel 0 Pull DownTrim Register	read-write	0h	Section 21.8.14
24h	SAPHCH0TT SAPH_ACH0TT	Channel 0 Termination Trim	read-write	0h	Section 21.8.15
26h	SAPHCH1PUT SAPH_ACH1PUT	Channel 1 Pull UpTrim	read-write	0h	Section 21.8.16
28h	SAPHCH1PDT SAPH_ACH1PDT	Channel 1 Pull DownTrim	read-write	0h	Section 21.8.17
2Ah	SAPHCH1TT SAPH_ACH1TT	Channel 1 Termination Trim	read-write	0h	Section 21.8.18
2Ch	SAPHMCNF SAPH_AMCNF	Mode Configuration Register	read-write	2h	Section 21.8.19
2Eh	SAPHTACTL SAPH_ATACTL	Trim Access Control	read-write	0h	Section 21.8.20
30h	SAPHICTL0 SAPH_AICTL0	Physical Interface Input Control #0	read-write	90h	Section 21.8.21
34h	SAPH_ABCTL SAPHBCTL	Bias Control	read-write	A1h	Section 21.8.22
40h	SAPHPGC SAPH_APGC	PPG Count	read-write	0h	Section 21.8.23
42h	SAPHPLPER SAPH_APLPER	Pulse Generator Low Period	read-write	0h	Section 21.8.24
44h	SAPHPLHPR SAPH_APLHPR	Pulse Generator High Period	read-write	0h	Section 21.8.25

Table 21-5. SAPH Registers (continued)

Offset	Acronym	Register Name	Type	Reset	Section
46h	SAPHPGCTL SAPH_APGCT	PPG Control	read-write	11h	Section 21.8.26
48h	SAPHPPGTRIG SAPH_APPGTRIG	PPG Software Trigger	write-only	0h	Section 21.8.27
4Ah	SAPH_AXPCTL	Extended Pulse Control Register	read-write	0h	Section 21.8.28
4Ch	SAPH_AXPLPER	Extra Pulse Low Period Register	read-write	0h	Section 21.8.29
4Eh	SAPH_AXPHPER	Extra Pulse High Period Register	read-write	0h	Section 21.8.30
60h	SAPHASCTL0 SAPH_AASCTL0	A-SEQ control register 0	read-write	0h	Section 21.8.31
62h	SAPHASCTL1 SAPH_AASCTL1	A-SEQ control register 1	read-write	0h	Section 21.8.32
64h	SAPHASQTRIG APH_AASQTRIG	ASQ Software Trigger	write-only	0h	Section 21.8.33
66h	SAPHAPOL SAPH_AAPOL	ASQ ping output polarity	read-write	0h	Section 21.8.34
68h	SAPHAPLEV SAPH_AAPLEV	ASQ ping pause level	read-write	0h	Section 21.8.35
6Ah	SAPHAPHIZ SAPH_AAPHIZ	ASQ ping pause impedance	read-write	0h	Section 21.8.36
6Eh	SAPHATM_A SAPH_AATM_A	A-SEQ start to 1st ping	read-write	0h	Section 21.8.37
70h	SAPHATM_B SAPH_AATM_B	ASQ start to ADC arm	read-write	0h	Section 21.8.38
72h	SAPHATM_C SAPH_AATM_C	Count for the TIMEMARK C Event	read-write	0h	Section 21.8.39
74h	SAPHATM_D SAPH_AATM_D	ASQ start to ADC trig	read-write	0h	Section 21.8.40
76h	SAPHATM_E SAPH_AATM_E	ASQ start to restart	read-write	0h	Section 21.8.41
78h	SAPHATM_F SAPH_AATM_F	ASQ start to timeout	read-write	0h	Section 21.8.42
7Ah	SAPHTBCTL SAPH_ATBCTL	Time Base Control	read-write	0h	Section 21.8.43
7Ch	SAPHATIMLO APH_AATIMLO	Acquisition Timer Low Part	read-only	0h	Section 21.8.44
7Eh	SAPHATIMHI SAPH_AATIMHI	Acquisition Timer High Part	read-only	0h	Section 21.8.45

21.8.1 SAPHIIDX/SAPH_AIIDX Register (Offset = 0h) [reset = 0h]

SAPHIIDX/SAPH_AIIDX is shown in [Figure 21-21](#) and described in [Table 21-6](#).

Return to [Summary Table](#).

Interrupt Index Register

Figure 21-21. SAPHIIDX/SAPH_AIIDX Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				IIDX		RESERVED	
R-0h				RH-0h		R-0h	

Table 21-6. SAPHIIDX/SAPH_AIIDX Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	
3-1	IIDX	RH	0h	This register provides the highest priority enabled interrupt index. Reset type: PUC 0h (R/W) = NONE : no interrupts pending 1h (R/W) = DATAERR : This interrupt indicates that either WINHI interrupt or WINLO interrupt has occurred in SDHS. 2h (R/W) = TMFTO : This interrupt is valid when ASQ is activcve (auto mode). The interrupt indicates that the time counter in ASQ has reached to TIMEMARK_F (timeout). 3h (R/W) = SEQDN : This interrupt is valid when ASQ is activcve (auto mode). The interrupt occurs when ASQ completes all of the measurements programmed in SAPHASCTL0.PNGCNT. For example, when SAPHASCTL0.PNGCNT = 3, total four measurements are performed. The interrupt indicates that all of the four measurements have been completed. 4h (R/W) = PNGDN : This interrupt is valid when ASQ is active (auto mode). The interrupt occurs when ASQ completes one measurement sequence. For example, when SAPHASCTL0.PNGCNT = 3, total four measurements are performed. The interrupt indicates that one measurement has been completed.
0	RESERVED	R	0h	

21.8.2 SAPHMIS/SAPH_AMIS Register (Offset = 2h) [reset = 0h]

SAPHMIS/SAPH_AMIS is shown in [Figure 21-22](#) and described in [Table 21-7](#).

Return to [Summary Table](#).

Masked Interrupt Status Register

Figure 21-22. SAPHMIS/SAPH_AMIS Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				PNGDN	SEQDN	TMFTO	DATAERR
R-0h				RH-0h	RH-0h	RH-0h	RH-0h

Table 21-7. SAPHMIS/SAPH_AMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	
3	PNGDN	RH	0h	This interrupt is valid when ASQ is active (auto mode). The interrupt occurs when ASQ completes one measurement sequence. For example, when SAPHASCTL0.PNGCNT = 3, total four measurements are performed. The interrupt indicates that one measurement has been completed. Reset type: PUC
2	SEQDN	RH	0h	This interrupt is valid when ASQ is active (auto mode). The interrupt occurs when ASQ completes all of the measurements programmed in SAPHASCTL0.PNGCNT. For example, when SAPHASCTL0.PNGCNT = 3, total four measurements are performed. The interrupt indicates that all of the four measurements have been completed. Reset type: PUC
1	TMFTO	RH	0h	This bit indicates a TIMEMARK F (timeout) event has happened. Reset type: PUC
0	DATAERR	RH	0h	This interrupt indicates that either WINHI interrupt or WINLO interrupt has occurred in SDHS. Reset type: PUC

21.8.3 SAPHRIS/SAPH_ARIS Register (Offset = 4h) [reset = 0h]

SAPHRIS/SAPH_ARIS is shown in [Figure 21-23](#) and described in [Table 21-8](#).

Return to [Summary Table](#).

Raw Interrupt Status Register

Figure 21-23. SAPHRIS/SAPH_ARIS Register

15	14	13	12	11	10	9	8
RESERVED	DAV	RESERVED					
R-0h	RH-0h	R-0h					
7	6	5	4	3	2	1	0
RESERVED				PNGDN	SEQDN	TMFTO	DATAERR
R-0h				RH-0h	RH-0h	RH-0h	R-0h

Table 21-8. SAPHRIS/SAPH_ARIS Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	
14	DAV	RH	0h	This bit indicates a DMA access violation Reset type: PUC
13-4	RESERVED	R	0h	
3	PNGDN	RH	0h	This interrupt is valid when ASQ is active (auto mode). The interrupt occurs when ASQ completes one measurement sequence. For example, when SAPHASCTL0.PNGCNT = 3, total four measurements are performed. The interrupt indicates that one measurement has been completed. Reset type: PUC
2	SEQDN	RH	0h	This interrupt is valid when ASQ is activcve (auto mode). The interrupt occurs when ASQ completes all of the measurements programmed in SAPHASCTL0.PNGCNT. For example, when SAPHASCTL0.PNGCNT = 3, total four measurements are performed. The interrupt indicates that all of the four measurements have been completed. Reset type: PUC
1	TMFTO	RH	0h	This bit indicates a TIMEMARK F (timeout) event has happened. Reset type: PUC
0	DATAERR	R	0h	This interrupt indicates that either WINHI interrupt or WINLO interrupt has occurred in SDHS. Reset type: PUC

21.8.4 SAPHIMSC/SAPH_AIMSC Register (Offset = 6h) [reset = 0h]

SAPHIMSC/SAPH_AIMSC is shown in [Figure 21-24](#) and described in [Table 21-9](#).

Return to [Summary Table](#).

Interrupt Mask Register

Figure 21-24. SAPHIMSC/SAPH_AIMSC Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED			DAV	PNGDN	SEQDN	TMFTO	DATAERR
R/W-0h			RH-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 21-9. SAPHIMSC/SAPH_AIMSC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	RESERVED	R/W	0h	
4	DAV	RH	0h	This bit indicates a DMA access violation Reset type: PUC
3	PNGDN	R/W	0h	This bit enables the PNGDN interrupt Reset type: PUC
2	SEQDN	R/W	0h	This bit enables the SEQDN interrupt Reset type: PUC
1	TMFTO	R/W	0h	This bit enables the TIMEMARK F (timeout) interrupt. Reset type: PUC
0	DATAERR	R/W	0h	This bit enables the DATAERR interrupt. Reset type: PUC

21.8.5 SAPHICR/SAPH_AICR Register (Offset = 8h) [reset = 0h]

SAPHICR/SAPH_AICR is shown in [Figure 21-25](#) and described in [Table 21-10](#).

Return to [Summary Table](#).

Interrupt Clear Register

Figure 21-25. SAPHICR/SAPH_AICR Register

15	14	13	12	11	10	9	8
RESERVED	DAV	RESERVED					
W-0h	HW1C-0h	W-0h					
7	6	5	4	3	2	1	0
RESERVED				PNGDN	SEQDN	TMFTO	DATAERR
W-0h				HW1C-0h	HW1C-0h	HW1C-0h	HW1C-0h

Table 21-10. SAPHICR/SAPH_AICR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	W	0h	
14	DAV	HW1C	0h	Writing one this bit to clear any previous DMA access violations
13-4	RESERVED	W	0h	
3	PNGDN	HW1C	0h	Writing one this bit to clear the pending PNGDN interrupt. Reset type: PUC
2	SEQDN	HW1C	0h	Writing one this bit to clear the pending SEQDN interrupt. Reset type: PUC
1	TMFTO	HW1C	0h	Writing one this bit to clear the pending TIMEMARK F (timeout) interrupt Reset type: PUC
0	DATAERR	HW1C	0h	Writing one this bit to clear the pending DATAERR interrupt. Reset type: PUC

21.8.6 SAPHISR/SAPH_AISR Register (Offset = Ah) [reset = 0h]

SAPHISR/SAPH_AISR is shown in [Figure 21-26](#) and described in [Table 21-11](#).

Return to [Summary Table](#).

Interrupt Set Register

Figure 21-26. SAPHISR/SAPH_AISR Register

15	14	13	12	11	10	9	8
RESERVED	DAV	RESERVED					
W-0h	HW1S-0h	W-0h					
7	6	5	4	3	2	1	0
RESERVED				PNGDN	SEQDN	TMFTO	DATAERR
W-0h				HW1S-0h	HW1S-0h	HW1S-0h	HW1S-0h

Table 21-11. SAPHISR/SAPH_AISR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	W	0h	
14	DAV	HW1S	0h	Writing one this bit to set RIS.DAV by software. Reset type: PUC
13-4	RESERVED	W	0h	
3	PNGDN	HW1S	0h	Writing one this bit to generate a PNGDN interrupt by software. Reset type: PUC
2	SEQDN	HW1S	0h	Writing one this bit to generate a SEQDN interrupt by software. Reset type: PUC
1	TMFTO	HW1S	0h	Writing one this bit to generate a TIMEMARK F (timeout) interrupt by software. Reset type: PUC
0	DATAERR	HW1S	0h	Writing one this bit generates a DATAERR interrupt by software. Reset type: PUC

21.8.7 SAPHDESCLO/SAPH_ADESCLO Register (Offset = Ch) [reset = 10h]

SAPHDESCLO/SAPH_ADESCLO is shown in [Figure 21-27](#) and described in [Table 21-12](#).

Return to [Summary Table](#).

Module-Descriptor Low Word

Figure 21-27. SAPHDESCLO/SAPH_ADESCLO Register

15	14	13	12	11	10	9	8
FEATUREVER				INSTNUM			
R-0h				R-0h			
7	6	5	4	3	2	1	0
MAJREV				MINREV			
R-1h				R-0h			

Table 21-12. SAPHDESCLO/SAPH_ADESCLO Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	FEATUREVER	R	0h	Feature Version
11-8	INSTNUM	R	0h	Instance Number
7-4	MAJREV	R	1h	Major Revision
3-0	MINREV	R	0h	Minor Revision

21.8.8 SAPHDESCHI/SAPH_ADESCHI Register (Offset = Eh) [reset = 5553h]

SAPHDESCHI/SAPH_ADESCHI is shown in [Figure 21-28](#) and described in [Table 21-13](#).

Return to [Summary Table](#).

Module Descriptor High Word

Figure 21-28. SAPHDESCHI/SAPH_ADESCHI Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ModuleID															
R-5553h															

Table 21-13. SAPHDESCHI/SAPH_ADESCHI Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	ModuleID	R	5553h	SAPH's Identifier

21.8.9 SAPHKEY/SAPH_AKEY Register (Offset = 10h) [reset = 0h]

SAPHKEY/SAPH_AKEY is shown in [Figure 21-29](#) and described in [Table 21-14](#).

Return to [Summary Table](#).

SAPH Key register

Figure 21-29. SAPHKEY/SAPH_AKEY Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY															
W-0h															

Table 21-14. SAPHKEY/SAPH_AKEY Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	KEY	W	0h	This field must be written with 0x454B using a word write to unlock the SAPH registers for write accesses. Writing a value other than 0x454B locks the SAPH registers again. The KEY register only locks/unlocks the SAPH registers with offset address of 0xF or higher. SAPHIIDX, SAPHMIS, SAPHRIS, SAPHIMSC, SAPHICR, SAPHISR, SAPHDESCLO, and SAPHDESCHI registers are not affected and the registers are not locked. Note: This register is write only. Reading always returns with zero. 454Bh (W) = 0x454B

21.8.10 SAPHOCTL0/SAPH_AOCTL0 Register (Offset = 12h) [reset = 0h]

SAPHOCTL0/SAPH_AOCTL0 is shown in [Figure 21-30](#) and described in [Table 21-15](#).

Return to [Summary Table](#).

Physical Interface Output Control #0

Figure 21-30. SAPHOCTL0/SAPH_AOCTL0 Register

15	14	13	12	11	10	9	8
RESERVED						CH1OUT	CH0OUT
R/W-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED						CH1OE	CH0OE
R/W-0h						R/W-0h	R/W-0h

Table 21-15. SAPHOCTL0/SAPH_AOCTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R/W	0h	
9	CH1OUT	R/W	0h	CH1_OUT Value. When SAPHOSEL.PCH1SEL =0 and SAPHOCTL0.CH1OE=1, this bit represents the logical value on the CH1 terminal. 0 = low 1 = high Reset type: PUC 0h (R/W) = Ch1 is set to low signal 1h (R/W) = Ch1 is set to high signal
8	CH0OUT	R/W	0h	CH0_OUT Value. When SAPHOSEL.PCH0SEL =0 and SAPHOCTL0.CH0OE=1, this bit represents the logical value on the CH0 terminal. 0 = low 1 = high Reset type: PUC 0h (R/W) = Ch0 is set to low signal 1h (R/W) = Ch0 is set to high signal
7-2	RESERVED	R/W	0h	
1	CH1OE	R/W	0h	CH1_OUT Enable. When SAPHOSEL.PCH1SEL =0, this bit enables the output CH1 when set to 1. When SAPHOSEL.PCH1SEL != 0, this bit is invalid. Reset type: PUC 0h (R/W) = Ch1 Output is HiZ 1h (R/W) = CH1 Output is driving
0	CH0OE	R/W	0h	CH0_OUT Enable. When SAPHOSEL.PCH0SEL =0, this bit enables the output CH0 when set to 1. When SAPHOSEL.PCH0SEL != 0, this bit is invalid. Reset type: PUC 0h (R/W) = Ch0 Output is HiZ 1h (R/W) = CH0 Output is driving

21.8.11 SAPHOCTL1/SAPH_AOCTL1 Register (Offset = 14h) [reset = 0h]

SAPHOCTL1/SAPH_AOCTL1 is shown in [Figure 21-31](#) and described in [Table 21-16](#).

Return to [Summary Table](#).

Physical Interface Output Control #1

Figure 21-31. SAPHOCTL1/SAPH_AOCTL1 Register

15	14	13	12	11	10	9	8
RESERVED						CH1FP	CH0FP
R/W-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED						CH1TERM	CH0TERM
R/W-0h						R/W-0h	R/W-0h

Table 21-16. SAPHOCTL1/SAPH_AOCTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R/W	0h	
9	CH1FP	R/W	0h	DRV1 (output driver on the CH1_OUT) full strength enable. 0 = The DRV1 output impedance is determined by SAPHCH1PUT and SAPHCH1PDT registers. 1 = The DRV1 has lowest output impedance. Reset type: PUC 0h (R/W) = Ch1 Output is set to normal strength 1h (R/W) = Ch1 Output is set to maximum strength
8	CH0FP	R/W	0h	DRV0 (output driver on the CH0_OUT) full strength enable. 0 = The DRV0 output impedance is determined by SAPHCH0PUT and SAPHCH0PDT registers. 1 = The DRV0 has lowest output impedance. Reset type: PUC 0h (R/W) = Ch0 Output is set to normal strength 1h (R/W) = Ch0 Output is set to maximum strength
7-2	RESERVED	R/W	0h	
1	CH1TERM	R/W	0h	CH1 termination switch (SWG1) enable. When SAPHOSEL.PCH1SEL =0, this bit controls the SWG1 switch. 0 = SWG1 is off. 1 = SWG1 is on. The CH1_OUT is disabled and connected to PVSS via SWG1 switch. No need to change SAPHOCTL0.CH1OE status. Reset type: PUC 0h (R/W) = CH1 Output is defined by SAPHOCTL0.CH1OUT and SAPHOCTL0.CH1OE 1h (R/W) = CH1 Output is set low with termination strength
0	CH0TERM	R/W	0h	CH0 termination switch (SWG0) enable. When SAPHOSEL.PCH0SEL =0, this bit controls the SWG0 switch. 0 = SWG0 is off. 1 = SWG0 is on. The CH0_OUT is disabled and connected to PVSS via SWG0 switch. No need to change SAPHOCTL0.CH0OE status. Reset type: PUC 0h (R/W) = CH0 Output is defined by SAPHOCTL0.CH0OUT and SAPHOCTL0.CH0OE 1h (R/W) = CH0 Output is set low with termination strength

21.8.12 SAPHOSEL/SAPH_AOSEL Register (Offset = 16h) [reset = 5h]

SAPHOCTL1/SAPH_AOSEL is shown in [Figure 21-32](#) and described in [Table 21-17](#).

Return to [Summary Table](#).

Physical Interface Output Function Select

Figure 21-32. SAPHOSEL/SAPH_AOSEL Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				PCH1SEL		PCH0SEL	
R/W-0h				R/W-1h		R/W-1h	

Table 21-17. SAPHOSEL/SAPH_AOSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R/W	0h	
3-2	PCH1SEL	R/W	1h	Output functional select for CH0_OUT. Reset type: PUC 0h (R/W) = GPIO : CH1_OUT is used as a GPO pin. It is controlled by SAPHOCTL0.CH1OUT and SAPHOCTL0.CH1OE. 1h (R/W) = PPGSE : CH1_OUT is driven by the PPG. 2h (R/W) = CH1_OUT is driven by the PPG as differential output along with CH0_OUT. (CH0_OUT and CH1_OUT are always opposite polarity) 3h (R/W) = CH1_OUT is used as a GPO pin. It is controlled by SAPHOCTL0.CH1OUT and SAPHOCTL0.CH1OE.
1-0	PCH0SEL	R/W	1h	Output functional select for CH0_OUT. Reset type: PUC 0h (R/W) = GPIO : CH0_OUT is used as a GPO pin. It is controlled by SAPHOCTL0.CH0OUT and SAPHOCTL0.CH0OE. 1h (R/W) = PPGSE : CH0_OUT is driven by the PPG. 2h (R/W) = CH0_OUT is driven by the PPG as differential output along with CH1_OUT. (CH0_OUT and CH1_OUT are always opposite polarity) 3h (R/W) = CH0_OUT is used as a GPO pin. It is controlled by SAPHOCTL0.CH0OUT and SAPHOCTL0.CH0OE.

21.8.13 SAPHCH0PUT/SAPH_ACH0PUT Register (Offset = 20h) [reset = 0h]

SAPHCH0PUT/SAPH_ACH0PUT is shown in [Figure 21-33](#) and described in [Table 21-18](#).

Return to [Summary Table](#).

DRV0 (CH0_OUT driver) Trim Register for pull-up.

Figure 21-33. SAPHCH0PUT/SAPH_ACH0PUT Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				CH0PUT			
R/W-0h				R/W-0h			

Table 21-18. SAPHCH0PUT/SAPH_ACH0PUT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R/W	0h	
3-0	CH0PUT	R/W	0h	DRV0 pull up trim register. Write access is allowed only when SAPHTACTL.UNLOCK=1. For secure the trim value, it is recommended to keep SAPHTACTL.UNLOCK=0 during normal operation. Reset type: BOR

21.8.14 SAPHCH0PDT/SAPH_ACH0PDT Register (Offset = 22h) [reset = 0h]

SAPHCH0PDT/SAPH_ACH0PDT is shown in [Figure 21-34](#) and described in [Table 21-19](#).

Return to [Summary Table](#).

DRV0 (CH0_OUT driver) Trim Register for pull-down.

Figure 21-34. SAPHCH0PDT/SAPH_ACH0PDT Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				CH0PDT			
R/W-0h				R/W-0h			

Table 21-19. SAPHCH0PDT/SAPH_ACH0PDT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R/W	0h	
3-0	CH0PDT	R/W	0h	DRV0 pull down trim register. Write access is allowed only when SAPHTACTL.UNLOCK=1. For secure the trim value, it is recommended to keep SAPHTACTL.UNLOCK=0 during normal operation. Reset type: BOR

21.8.15 SAPHCH0TT/SAPH_ACH0TT Register (Offset = 24h) [reset = 0h]

SAPHCH0TT/SAPH_ACH0TT is shown in [Figure 21-35](#) and described in [Table 21-20](#).

Return to [Summary Table](#).

SWG0 (CH0_OUT termination switch) Trim Register.

Figure 21-35. SAPHCH0TT/SAPH_ACH0TT Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				CH0TT			
R/W-0h				R/W-0h			

Table 21-20. SAPHCH0TT/SAPH_ACH0TT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R/W	0h	
3-0	CH0TT	R/W	0h	SWG0 trim register. Write access is allowed only when SAPHTACTL.UNLOCK=1. For secure the trim value, it is recommended to keep SAPHTACTL.UNLOCK=0 during normal operation. Reset type: BOR

21.8.16 SAPHCH1PUT /SAPH_ACH1PUT Register (Offset = 26h) [reset = 0h]

SAPHCH1PUT/SAPH_ACH1PUT is shown in [Figure 21-36](#) and described in [Table 21-21](#).

Return to [Summary Table](#).

DRV1 (CH1_OUT driver) Trim Register for pull-up.

Figure 21-36. SAPHCH1PUT/SAPH_ACH1PUT Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				CH1PUT			
R/W-0h				R/W-0h			

Table 21-21. SAPHCH1PUT/SAPH_ACH1PUT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R/W	0h	
3-0	CH1PUT	R/W	0h	DRV1 pull up trim register. Write access is allowed only when SAPHTACTL.UNLOCK=1. For secure the trim value, it is recommended to keep SAPHTACTL.UNLOCK=0 during normal operation. Reset type: BOR

21.8.17 SAPHCH1PDT/SAPH_ACH1PDT Register (Offset = 28h) [reset = 0h]

SAPHCH1PDT/SAPH_ACH1PDT is shown in [Figure 21-37](#) and described in [Table 21-22](#).

Return to [Summary Table](#).

DRV1 (CH1_OUT driver) Trim Register for pull-down.

Figure 21-37. SAPHCH1PDT/SAPH_ACH1PDT Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				CH1PDT			
R/W-0h				R/W-0h			

Table 21-22. SAPHCH1PDT/SAPH_ACH1PDT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R/W	0h	
3-0	CH1PDT	R/W	0h	DRV1 pull down trim register. Write access is allowed only when SAPHTACTL.UNLOCK=1. For secure the trim value, it is recommended to keep SAPHTACTL.UNLOCK=0 during normal operation. Reset type: BOR

21.8.18 SAPHCH1TT/SAPH_ACH1TT Register (Offset = 2Ah) [reset = 0h]

SAPHCH1TT/SAPH_ACH1TT is shown in [Figure 21-38](#) and described in [Table 21-23](#).

Return to [Summary Table](#).

SWG1 (CH1_OUT termination switch) Trim Register.

Figure 21-38. SAPHCH1TT/SAPH_ACH1TT Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				CH1TT			
R/W-0h				R/W-0h			

Table 21-23. SAPHCH1TT/SAPH_ACH1TT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R/W	0h	
3-0	CH1TT	R/W	0h	SWG1 trim register. Write access is allowed only when SAPHTACTL.UNLOCK=1. For secure the trim value, it is recommended to keep SAPHTACTLUNLOCK=0 during normal operation. Reset type: BOR

21.8.19 SAPHMCNF/SAPH_AMCNF Register (Offset = 2Ch) [reset = 2h]

SAPHMCNF/SAPH_AMCNF is shown in [Figure 21-39](#) and described in [Table 21-24](#).

Return to [Summary Table](#).

Mode Configuration Register. This bit needs to be unlocked via SAPH_AKEY for changes

Figure 21-39. SAPHMCNF/SAPH_AMCNF Register

15	14	13	12	11	10	9	8
RESERVED				LPBE	RSV1		CPEO
R/W-0h				R/W-0h	R/W-0h		R/W-0h
7	6	5	4	3	2	1	0
RESERVED				RSV0		BIMP	
R/W-0h				R/W-0h		R/W-2h	

Table 21-24. SAPHMCNF/SAPH_AMCNF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R/W	0h	
11	LPBE	R/W	0h	LPBE, low power bias mode enable. This bit enables the low power bias operation mode. The selection of the operation mode shall only be changed while the PSQ is in OFF state (changes during other states of the PSQ causes corrupt measurement results and irregular triggers of sub modules ba ASQ) Reset type: BOR 0h (R/W) = For manual bias mode and regular ASQ bias mode. In this configuration the user controls by the ASQBSW has full control over the TxBias and RxBias switches. 1h (R/W) = Low power bias mode. In this mode the ASQ uses the CHxESW and PGABSW as auxiliary values to achieve faster channel setting on reactive input loads. The ASQ has full controls over the bias switch multiplexer.
10-9	RSV1	R/W	0h	Reserved for future use Reset type: BOR
8	CPEO	R/W	0h	Charge pump explicitly on. This bit enables the charge pump of the input multiplexer. Reset type: BOR 0h (R/W) = Charge pump is turned on by SDHS and ASQ related requests only. 1h (R/W) = Charge pump is turned on regardless of SDHS and ASQ related charge pump requests.
7-4	RESERVED	R/W	0h	
3-2	RSV0	R/W	0h	Reserved for future use Reset type: BOR
1-0	BIMP	R/W	2h	Bias generator Impedance configuration. These bits define the impedance of the buffers for RxBias and TxBias. While for resistive loads the lowest impedance shows the fastest settling; is this not the case for reactive loads. Reset type: BOR 0h (R/W) = Buffer impedance 500 ohms for RxBias and 450 ohms for TxBias 1h (R/W) = Buffer impedance 900 ohms for RxBias and 850 ohms for TxBias 2h (R/W) = Buffer impedance 1500 ohms for RxBias and 1450 ohms for TxBias 3h (R/W) = Buffer impedance 2950 Ohms for RxBias and 2900 Ohms for TxBias

21.8.20 SAPHTACTL/SAPH_ATACTL Register (Offset = 2Eh) [reset = 0h]

SAPHTACTL/SAPH_ATACTL is shown in [Figure 21-40](#) and described in [Table 21-25](#).

Return to [Summary Table](#).

Trim Access Control Register

Figure 21-40. SAPHTACTL/SAPH_ATACTL Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				RESERVED		UNLOCK	
R/W-0h				R-0h		R/W-0h	

Table 21-25. SAPHTACTL/SAPH_ATACTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R/W	0h	
2-1	RESERVED	R	0h	Reserved
0	UNLOCK	R/W	0h	When UNLOCK = 1, the trim registers are allowed to be updated (SAPHCH0PUT, SAPHCH0PDT, SAPHCH0TT, SAPHCH1PUT, SAPHCH1PDT, and SAPHCH1TT). Reset type: PUC

21.8.21 SAPHICTL0 /SAPH_AICTL0 Register (Offset = 30h) [reset = 90h]

SAPHICTL0/SAPH_AICTL0 is shown in [Figure 21-41](#) and described in [Table 21-26](#).

Return to [Summary Table](#).

Physical Interface Input Control #0

Figure 21-41. SAPHICTL0/SAPH_AICTL0 Register

15	14	13	12	11	10	9	8
RESERVED		XPBSW1	XPBSW0	RESERVED		XPB1FEN	XPB0FEN
R/W-0h		R/W-0h	R/W-0h	R/W-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DUMEN	RESERVED		MUXCTL	MUXSEL			
R/W-1h	R/W-0h		R/W-1h	R/W-0h			

Table 21-26. SAPHICTL0/SAPH_AICTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	RESERVED	R/W	0h	
13	XPBSW1	R/W	0h	External PGA Bias Switch 1 control (only on SAPH_A) Reset type: PUC 0h (R/W) = External bias switch is open (no bias driven) 1h (R/W) = External bias switch is closed (bias is driven)
12	XPBSW0	R/W	0h	External PGA Bias Switch 0 control (only on SAPH_A) Reset type: PUC 0h (R/W) = External bias switch is open (no bias driven) 1h (R/W) = External bias switch is closed (bias is driven)
11-10	RESERVED	R/W	0h	
9	XPB1FEN	R/W	0h	XPB1 Pin Function Enable(only on SAPH_A) Reset type: PUC 0h (R/W) = XPB1 pin function disabled 1h (R/W) = XPB1 pin function enabled
8	XPB0FEN	R/W	0h	XPB0 Pin Function Enable(only on SAPH_A) Reset type: PUC 0h (R/W) = XPB0 pin function disabled 1h (R/W) = XPB0 pin function enabled
7	DUMEN	R/W	1h	PGA dummy load enable on the deselected multiplexer inputs. Reset type: PUC 0h (R/W) = PGA dummy input load is Hi-Z. 1h (R/W) = PGA dummy input load matches the PGA input impedance.
6-5	RESERVED	R/W	0h	
4	MUXCTL	R/W	1h	Input Multiplexer Control source Reset type: PUC 0h (R/W) = The input multiplexer is controlled by SAPHICTL0.MUXSEL (register mode) 1h (R/W) = The input multiplexer is controlled by ASQ (auto mode)

Table 21-26. SAPHCTL0/SAPH_AICTL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	MUXSEL	R/W	0h	Input Multiplexer Channel Select Reset type: PUC 0h (R/W) = CH0IN : Channel 0 is selected for input 1h (R/W) = CH1IN : Channel 1 is selected for input 2h (R/W) = reserved for future channels 3h (R/W) = reserved for future channels 4h (R/W) = reserved for future channels 5h (R/W) = reserved for future channels 6h (R/W) = reserved for future channels 7h (R/W) = reserved for future channels 8h (R/W) = no channel is selected 9h (R/W) = no channel is selected Ah (R/W) = no channel is selected Bh (R/W) = no channel is selected Ch (R/W) = no channel is selected Dh (R/W) = no channel is selected Eh (R/W) = no channel is selected Fh (R/W) = no channel is selected

21.8.22 SAPHBCTL/SAPH_ABCTL Register (Offset = 34h) [reset = A1h]

SAPHBCTL/SAPH_ABCTL is shown in [Figure 21-42](#) and described in [Table 21-27](#).

Return to [Summary Table](#).

Bias Control Register

Figure 21-42. SAPHBCTL/SAPH_ABCTL Register

15	14	13	12	11	10	9	8
RESERVED						CH1EBSW	CH0EBSW
R/W-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
PGABIAS		EXCBIAS		CPDA	LILC	PGABSW	ASQBSC
R/W-2h		R/W-2h		R/W-0h	R/W-0h	R/W-0h	R/W-1h

Table 21-27. SAPHBCTL/SAPH_ABCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R/W	0h	
9	CH1EBSW	R/W	0h	Channel 1 Tx bias Switch Control. Note that the Tx bias voltage is determined by SAPHBCTL.EXCBIAS. Reset type: PUC 0h (R/W) = Tx bias switch to CH1 is open. 1h (R/W) = Tx bias switch to CH1 is closed (enabled).
8	CH0EBSW	R/W	0h	Channel 0 Tx bias Switch Control. Note that the Tx bias voltage is determined by SAPHBCTL.EXCBIAS. Reset type: PUC 0h (R/W) = Tx bias switch to CH0 is open. 1h (R/W) = Tx bias switch to CH0 is closed (enabled).
7-6	PGABIAS	R/W	2h	PGA bias (Rx bias) Voltage Select Reset type: PUC 0h (R/W) = 0.75V nominal 1h (R/W) = 0.8V nominal 2h (R/W) = 0.9V nominal 3h (R/W) = 0.95V nominal
5-4	EXCBIAS	R/W	2h	Excitation bias (Rx bias) Voltage Select Reset type: PUC 0h (R/W) = 0.2V nominal 1h (R/W) = 0.3V nominal 2h (R/W) = 0.4V nominal 3h (R/W) = 0.6V nominal
3	CPDA	R/W	0h	Enable the power supply (Charge Pump) for the the input multiplexer during data acquisition. Reset type: PUC 0h (R/W) = Turn off the charge pump during data acquisition. 1h (R/W) = Keep turning on the charge pump during data acquisition.
2	LILC	R/W	0h	LILC test feature. Leave at 0 for proper operation Reset type: PUC 0h (R/W) = LILC is disabled. 1h (R/W) = LILC is enabled.
1	PGABSW	R/W	0h	Rx bias (PGA bias) switch control. Note that the channel to apply the Rx bias is determined by SAPHICTL0.MUXSEL. Reset type: PUC 0h (R/W) = Rx bias switch is open. 1h (R/W) = Rx bias switch is closed (enabled).

Table 21-27. SAPHBCTL/SAPH_ABCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	ASQBSC	R/W	1h	Tx bias and Rx bias switches control source select Reset type: PUC 0h (R/W) = Bias switches are controlled by SAPHBCTL.CH0EBSW, SAPHBCTL.CH1EBSW, SAPHBCTL.PGABSW bits (register mode). 1h (R/W) = Bias switches are controlled by ASQ (auto mode)

21.8.23 SAPHPGC/SAPH_APGC Register (Offset = 40h) [reset = 0h]

SAPHPGC/SAPH_APGC is shown in [Figure 21-43](#) and described in [Table 21-28](#).

Return to [Summary Table](#).

PPG Count Register

Figure 21-43. SAPHPGC/SAPH_APGC Register

15	14	13	12	11	10	9	8
PHIZ	PLEV	PPOL	RESERVED			SPULS	
R/W-0h	R/W-0h	R/W-0h	R/W-0h			R/W-0h	
7	6	5	4	3	2	1	0
RESERVED				EPULS			
R/W-0h				R/W-0h			

Table 21-28. SAPHPGC/SAPH_APGC Register Field Descriptions

Bit	Field	Type	Reset	Description
15	PHIZ	R/W	0h	Hi-Z enable to PPG output during inactive. Reset type: PUC 0h (R/W) = PPG output during inactive is determined by SAPHPGC.PLEV bit. 1h (R/W) = PPG output is in Hi-Z during inactive regardless of SAPHPGC.PLEV bit.
14	PLEV	R/W	0h	PPG output level during inactive. This bit affects the status of PPG output before and after excitations. Note that this bit is only valid when SAPHPGC.PHIZ = 0. Reset type: PUC 0h (R/W) = PPG output is low during inactive 1h (R/W) = PPG output is high during inactive
13	PPOL	R/W	0h	Pulse Polarity. This bit defines the polarity of the first excitation pulse. Reset type: PUC 0h (R/W) = The excitation begins with logical high phase. The stop begins with logical low phase. 1h (R/W) = The excitation begins with logical low phase. The stop begins with logical high phase.
12	RESERVED	R/W	0h	
11-8	SPULS	R/W	0h	Stop Pulse Count; This bit field defines the number of stop pulses. Minimum value is zero. Stop pulses have the inverted polarity of excitation pulses. Reset type: PUC
7	RESERVED	R/W	0h	
6-0	EPULS	R/W	0h	Excitation Pulse Count. This bit field defines the number of excitation pulses. Minimum value is zero. Reset type: PUC

21.8.24 SAPHGLPER/SAPH_APGLPER Register (Offset = 42h) [reset = 0h]

SAPHGLPER/SAPH_APGLPER is shown in [Figure 21-44](#) and described in [Table 21-29](#).

Return to [Summary Table](#).

PPG output pulse low period.

Figure 21-44. SAPHGLPER/SAPH_APGLPER Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
LPER							
R/W-0h							

Table 21-29. SAPHGLPER/SAPH_APGLPER Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	
7-0	LPER	R/W	0h	Low phase period of PPG excitation pulses. This value defines the length of the low phase of the pulses. The minimum count is two regardless of the value set in this register. Reset type: PUC

21.8.25 SAPHPGHPER/SAPH_APGHPER Register (Offset = 44h) [reset = 0h]

SAPHPGHPER/SAPH_APGHPER is shown in [Figure 21-45](#) and described in [Table 21-30](#).

Return to [Summary Table](#).

PPG output pulse high period.

Figure 21-45. SAPHPGHPER/SAPH_APGHPER Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
HPER							
R/W-0h							

Table 21-30. SAPHPGHPER/SAPH_APGHPER Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	
7-0	HPER	R/W	0h	High phase period of PPG excitation pulses. This value defines the length of the high phase of the pulses. The minimum count is two regardless of the value set in this register. Reset type: PUC

21.8.26 SAPHPGCTL/SAPH_APGCTL Register (Offset = 46h) [reset = 11h]

SAPHPGCTL/SAPH_APGCTL is shown in [Figure 21-46](#) and described in [Table 21-31](#).

Return to [Summary Table](#).

PPG Control Register

Figure 21-46. SAPHPGCTL/SAPH_APGCTL Register

15	14	13	12	11	10	9	8
STOP	TONE	PSCEN	RESERVED	RESERVED	PPGEN	RESERVED	
R/W-0h	RH/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
RESERVED	TRSEL	RESERVED	PPGCHSEL	PGSEL			
R/W-0h	R/W-1h	R/W-0h	R/W-0h	R/W-0h	R/W-1h		

Table 21-31. SAPHPGCTL/SAPH_APGCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	STOP	R/W	0h	Writing one to this bit stops the PPG to generate pulses. This bit is cleared automatically. Reset type: PUC
14	TONE	RH/W	0h	Tone generation enable. The frequency of the test tone is determined by LPER and HPER. The test tone persists while SAPHPGCTL.TONE = 1 & SAPHPGCTL.STOP=0. Either writing 0 to SAPHPGCTL.TONE or writing 1 to SAPHPGCTL.STOP stops tone generation. Reset type: PUC 0h (R/W) = ENABLE : Test tone generation is enabled.
13	PSCEN	R/W	0h	Test feature: Keep this bit zero for proper operation. Reset type: PUC 0h (R/W) = disabled 1h (R/W) = enabled; PPG is clocked with 1/4 of PLL_CLK without phase control
12	RESERVED	R/W	0h	
11-10	RESERVED	R	0h	Reserved
9	PPGEN	R/W	0h	PPGEN PPG trigger enable. This bit can be used to indicates that the configuration of the pulse generator is complete. The PPG can only be started when this bit is set to 1 regardless of its trigger source. It is recommended to keep this bit zero while updating PPG registers. Reset type: PUC 0h (R/W) = PPG trigger is disabled. 1h (R/W) = PPG trigger is enabled.
8-6	RESERVED	R/W	0h	
5-4	TRSEL	R/W	1h	PPG Trigger source select. Reset type: PUC 0h (R/W) = Writing 1 to PPGTRIG.PPGTRIG to trigger the PPG (start pulse generation). 1h (R/W) = PPG trigger is controlled by the ASQ. 2h (R/W) = Ext. Signal (See device specific datasheet) 3h (R/W) = Ext. Signal (See device specific datasheet)
3-2	RESERVED	R/W	0h	
1	PPGCHSEL	R/W	0h	PPG output channel select when SAPHPGCTL.PGSEL = 0. Reset type: PUC 0h (R/W) = CH0 is selected 1h (R/W) = CH1 is selected

Table 21-31. SAPHPGCTL/SAPH_APGCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	PGSEL	R/W	1h	PPG output channel select source. Reset type: PUC 0h (R/W) = PPG output channel is selected by SAPHPGCTL.PPGCHSEL bit (register mode). 1h (R/W) = PPG output channel is selected by ASQ (auto mode).

21.8.27 SAPHPPGTRIG/SAPH_APPGTRIG Register (Offset = 48h) [reset = 0h]

SAPHPPGTRIG/SAPH_APPGTRIG is shown in [Figure 21-47](#) and described in [Table 21-32](#).

Return to [Summary Table](#).

PPG Software Trigger Register

Figure 21-47. SAPHPPGTRIG/SAPH_APPGTRIG Register

15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED							PPGTRIG
W-0h							W-0h

Table 21-32. SAPHPPGTRIG/SAPH_APPGTRIG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	W	0h	
0	PPGTRIG	W	0h	Writing '1' to this bit triggers the PPG to generate pulses when SAPHPGCTL.TRSEL =0. Note: This bit is write only. Reading always returns with zero.

21.8.28 SAPH_AXPGCTL Register (Offset = 4Ah) [reset = 0h]

SAPH_AXPGCTL is shown in [Figure 21-48](#) and described in [Table 21-33](#).

Return to [Summary Table](#).

Extended Pulse Control Register

Figure 21-48. SAPH_AXPGCTL Register

15	14	13	12	11	10	9	8
RESERVED	ETY	XMOD	RESERVED	XSTAT			
R/W-0h	R/W-0h	R/W-0h	R/W-0h	RH-0h			
7	6	5	4	3	2	1	0
RESERVED	XPULS						
R/W-0h	R/W-0h						

Table 21-33. SAPH_AXPGCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R/W	0h	
14	ETY	R/W	0h	Event type. Selects the type of the event generated in SAPH_AXPGCTL.XMOD=3 Reset type: PUC 0h (R/W) = Timer count event 1h (R/W) = DMA trigger event
13-12	XMOD	R/W	0h	Extended Mode Select Reset type: PUC 0h (R/W) = Single tone generation (Pause-E-S-Pause) 1h (R/W) = reserved 2h (R/W) = Dual Tone Generation (Pause-X-E-S-Pause) 3h (R/W) = Dual Tone Loop (...X-E-X-E... triggers events)
11-10	RESERVED	R/W	0h	
9-8	XSTAT	RH	0h	Phase Status of PPG_A Reset type: PUC 0h (R) = PPG_A in Pause Phase 1h (R) = PPG_A in Stop Phase 2h (R) = PPG_A in Regular Excitation phase 3h (R) = PPG_A in Extra Excitation Phase
7	RESERVED	R/W	0h	
6-0	XPULS	R/W	0h	XPULS Extra Pulse Count; This bit field defines the count of extra excitation pulses excited. If set to zero no extra excitation pulses are generated. Directly after the start the regular excitation phase is entered. Any other number will generate up the number of excitation pulses set by this field . Reset type: PUC

21.8.29 SAPH_AXPGGLPER Register (Offset = 4Ch) [reset = 0h]

SAPH_AXPGGLPER is shown in [Figure 21-49](#) and described in [Table 21-34](#).

Return to [Summary Table](#).

Extra Pulse Low Period Register

Figure 21-49. SAPH_AXPGGLPER Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
XLPER							
R/W-0h							

Table 21-34. SAPH_AXPGGLPER Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	
7-0	XLPER	R/W	0h	XLPER low phase period of the extra pulses. This value defines the length of the low phase of the extra pulses in units of high speed clocks. The minimum count is two regardless of the value set in this register. Reset type: PUC

21.8.30 SAPH_AXPGHPER Register (Offset = 4Eh) [reset = 0h]

SAPH_AXPGHPER is shown in [Figure 21-50](#) and described in [Table 21-35](#).

[Return to Summary Table.](#)

Extra Pulse High Period Register

Figure 21-50. SAPH_AXPGHPER Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
XHPER							
R/W-0h							

Table 21-35. SAPH_AXPGHPER Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	
7-0	XHPER	R/W	0h	XHPER high phase period of the extra pulses. This value defines the length of the high phase of the pulses in units of high speed clocks. The minimum count is two regardless of the value set in this register. Reset type: PUC

21.8.31 SAPHASCTL0/SAPH_AASCTL0 Register (Offset = 60h) [reset = 0h]

SAPHASCTL0/SAPH_AASCTL0 is shown in [Figure 21-51](#) and described in [Table 21-36](#).

Return to [Summary Table](#).

A-SEQ control register 0

Figure 21-51. SAPHASCTL0/SAPH_AASCTL0 Register

15	14	13	12	11	10	9	8
RESERVED	ERABRT	RESERVED	TRIGSEL	ASQTEN	RESERVED		
R-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
ASQSTOP	RESERVED	ASQCHSEL	RESERVED	PNGCNT			
R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	

Table 21-36. SAPHASCTL0/SAPH_AASCTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	RESERVED	R	0h	Reserved
13	ERABRT	R/W	0h	Stop ASQ when the DATAERR interrupt occurs when the measurements controlled by ASQ (auto mode). Note that it is not possible to resume the measurement from where it was stopped. When ASQ is triggered again, the measurement sequence starts from the beginning. Reset type: PUC 0h (R/W) = Continue the measurements until completion regardless of the DATAERR interrupt. 1h (R/W) = Stop the ASQ upon the DATAERR interrupt.
12	RESERVED	R	0h	Reserved
11-10	TRIGSEL	R/W	0h	ASQ trigger select. Reset type: PUC 0h (R/W) = SWTRIG : Writing '1' to ASQTRIG.ASQTRIG 1h (R/W) = PSQ : The PSQ is selected to start the ASQ. 2h (R/W) = Timer : Ext. Signal (See device specific datasheet) 3h (R/W) = Ext. Signal (See device specific datasheet)
9	ASQTEN	R/W	0h	ASQ Trigger Enable. This bit can be used to indicate that the configuration of the ASQ is complete. The ASQ can only be triggered when this bit is set to '1' regardless of its trigger source. It is recommended to keep this bit zero while updating ASQ registers. Reset type: PUC 0h (R/W) = ASQ trigger is disabled. 1h (R/W) = ASQ trigger is enabled.
8	RESERVED	R/W	0h	
7	ASQSTOP	R/W	0h	Stop the ASQ. Writing '1' to this bit stops the measurement sequence controlled by the ASQ. This bit is self cleared. Reset type: PUC
6-5	RESERVED	R	0h	Reserved

Table 21-36. SAPHASCTL0/SAPH_AASCTL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	ASQCHSEL	R/W	0h	This bit selects the channel to start with when ASQ controls the measurement sequences. Reset type: PUC 0h (R/W) = CH0 is selected to start with. If SAPHASCTL0.PNGCNT = 3 and SAPHASCTL1.CHTOG =1, then the channel selection would be CH0 -> CH1 -> CH0 -> CH1. If SAPHASCTL0.PNGCNT = 3 and SAPHASCTL1.CHTOG =0, then the channel selection would be CH0 -> CH0 -> CH0 -> CH0. 1h (R/W) = CH1 is selected to start with. If SAPHASCTL0.PNGCNT = 3 and SAPHASCTL1.CHTOG =0, then the channel selection would be CH1 -> CH1 -> CH1 -> CH1. If SAPHASCTL0.PNGCNT = 3 and SAPHASCTL1.CHTOG =1, then the channel selection would be CH1 -> CH0 -> CH1 -> CH0.
3-2	RESERVED	R/W	0h	
1-0	PNGCNT	R/W	0h	The total number of measurements to be performed. 0 = 1 measurement will be performed (Min) 1 = 2 measurements will be performed 2 = 3 measurements will be performed 3 = 4 measurements will be performed (Max) Note: This bit field is static, does not reflect the currently remaining measurement numbers. Reset type: PUC

21.8.32 SAPHASCTL1/SAPH_AASCTL1 Register (Offset = 62h) [reset = 0h]

SAPHASCTL1/SAPH_AASCTL1 is shown in [Figure 21-52](#) and described in [Table 21-37](#).

Return to [Summary Table](#).

A-SEQ control register 1

Figure 21-52. SAPHASCTL1/SAPH_AASCTL1 Register

15	14	13	12	11	10	9	8
RESERVED				CHOWN	STDBY	RESERVED	ESOFF
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
EARLYRB	RESERVED		CHACT	RESERVED	RESERVED	RESERVED	CHTOG
R/W-0h	R-0h		R-0h	R/W-0h	R-0h	R-0h	R/W-0h

Table 21-37. SAPHASCTL1/SAPH_AASCTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	CHOWN	R/W	0h	In general, if CH0 is selected for Tx, CH1 is selected for Rx. However, when this bit is set to '1', the Tx channel and Rx channel are identical. This bit can be used for debugging purpose. Reset type: PUC 0h (R/W) = Tx channel and Rx channel are not the same (This is the typical configuration). 1h (R/W) = Tx channel and Rx channel are identical.
10	STDBY	R/W	0h	ASQ can send a request signal to the PSQ (Power Sequencer) of the USS module when the OFF request is received (See SAPHASCTL1.ESOFF and the UUPS module). Reset type: PUC 0h (R/W) = PWROFF : The ASQ sends a power down request to the PSQ (Power Sequencer) when the OFF request is received. 1h (R/W) = STDBY : The ASQ sends a standby request to the PSQ (Power Sequencer) when the OFF request is received.
9	RESERVED	R/W	0h	
8	ESOFF	R/W	0h	Enable the OFF request when ASQ completes all of the measurement sequences. Reset type: PUC 0h (R/W) = OFF request is disabled. The ASQ does not send a request about USS power mode to the PSQ. 1h (R/W) = OFF request is generated after sequence
7	EARLYRB	R/W	0h	Early Receive Bias Control. The Rx bias is applied at the TIMEMARK C, but when this bit is set to '1', the Rx bias is applied at the TIMEMARK A. Reset type: PUC 0h (R/W) = Rx bias is applied to the Rx channel by the TIMEMARK C 1h (R/W) = Rx bias is applied to the Rx channel by the TIMEMARK A
6-5	RESERVED	R	0h	Reserved
4	CHACT	R	0h	Read Only bit. This bit indicates the currently selected Tx channel. Reset type: PUC
3	RESERVED	R/W	0h	
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved

Table 21-37. SAPHASCTL1/SAPH_AASCTL1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	CHTOG	R/W	0h	Channel toggle enable at each PNGDN interrupt. Reset type: PUC 0h (R/W) = Channel toggle is disabled. 1h (R/W) = Channel toggle is enabled at each PNGDN interrupt.

21.8.33 SAPHASQTRIG/SAPH_AASQTRIG Register (Offset = 64h) [reset = 0h]

SAPHASQTRIG/SAPH_AASQTRIG is shown in [Figure 21-53](#) and described in [Table 21-38](#).

Return to [Summary Table](#).

ASQ Software Trigger Register

Figure 21-53. SAPHASQTRIG/SAPH_AASQTRIG Register

7	6	5	4	3	2	1	0
RESERVED							ASQTRIG
W-0h							W-0h

Table 21-38. SAPHASQTRIG/SAPH_AASQTRIG Register Field Descriptions

Bit	Field	Type	Reset	Description
7-1	RESERVED	W	0h	
0	ASQTRIG	W	0h	Writing '1' to this bit trigger the ASQ when SAPHASCTL0.TRIGSEL = 0. Note: This bit is write only. Reading always returns with zero.

21.8.34 SAPHAPOL/SAPH_AAPOL Register (Offset = 66h) [reset = 0h]

SAPHAPOL/SAPH_AAPOL is shown in [Figure 21-54](#) and described in [Table 21-39](#).

Return to [Summary Table](#).

The PPG output pulse polarity when ASQ is controlling the measurement.

Figure 21-54. SAPHAPOL/SAPH_AAPOL Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				PCPOL			
R/W-0h				R/W-0h			

Table 21-39. SAPHAPOL/SAPH_AAPOL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R/W	0h	
3-0	PCPOL	R/W	0h	Bit 0 defines the PPG pulse polarity for the first measurement. Bit 1 defines the PPG pulse polarity for the second measurement. Bit 2 defines the PPG pulse polarity for the third measurement. Bit 3 defines the PPG pulse polarity for the fourth measurement. 0 = PPG output pulses starts with logical high polarity. 1 = PPG output pulses starts with logical low polarity. Reset type: PUC

21.8.35 SAPHAPLEV /SAPH_AAPLEV Register (Offset = 68h) [reset = 0h]

SAPHAPLEV/SAPH_AAPLEV is shown in [Figure 21-55](#) and described in [Table 21-40](#).

Return to [Summary Table](#).

The PPG output level at pause when ASQ is controlling the measurement.

Figure 21-55. SAPHAPLEV/SAPH_AAPLEV Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				PCPLEV			
R/W-0h				R/W-0h			

Table 21-40. SAPHAPLEV/SAPH_AAPLEV Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R/W	0h	
3-0	PCPLEV	R/W	0h	Bit 0 defines the PPG output level at pause for the first measurement when PCPHIZ bit 0 = 0. Bit 1 defines the PPG output level at pause for the second measurement when PCPHIZ bit 1 = 0. Bit 2 defines the PPG output level at pause for the third measurement when PCPHIZ bit 2 = 0. Bit 3 defines the PPG output level at pause for the fourth measurement when PCPHIZ bit 3 = 0. 0 = Logical Low. 1 = Logical High. Reset type: PUC

21.8.36 SAPHAPHIZ /SAPH_AAPHIZ Register (Offset = 6Ah) [reset = 0h]

SAPHAPHIZ/SAPH_AAPHIZ is shown in [Figure 21-56](#) and described in [Table 21-41](#).

Return to [Summary Table](#).

The PPG output status at pause when ASQ is controlling the measurement.

Figure 21-56. SAPHAPHIZ/SAPH_AAPHIZ Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				PCPHIZ			
R/W-0h				R/W-0h			

Table 21-41. SAPHAPHIZ/SAPH_AAPHIZ Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R/W	0h	
3-0	PCPHIZ	R/W	0h	Bit 0 defines the PPG output status at pause for the first measurement. Bit 1 defines the PPG output status at pause for the second measurement. Bit 2 defines the PPG output status at pause for the third measurement. Bit 3 defines the PPG output status at pause for the fourth measurement. 0 = PPG output level is determined by SAPHAPLEV.PCPLEV bits 1 = Hi-z. regardless of SAPHAPLEV.PCPLEV bits Reset type: PUC

21.8.37 SAPHATM_A/SAPH_AATM_A Register (Offset = 6Eh) [reset = 0h]

SAPHATM_A/SAPH_AATM_A is shown in [Figure 21-57](#) and described in [Table 21-42](#).

Return to [Summary Table](#).

Count for the TIMEMARK A Event

Figure 21-57. SAPHATM_A/SAPH_AATM_A Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEMARK_A															
R/W-0h															

Table 21-42. SAPHATM_A/SAPH_AATM_A Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TIMEMARK_A	R/W	0h	This value is used to generate the TIMEMARK A event by comparing with the ASQ timer counter value. TIMEMARK_A value [15:0] are compared to ASQ Timer Counter [15:0] TIMEMARK A event is to trigger the PPG. Before this event, the Tx bias is applied to the transmit channel, so make sure to provide enough time to settle the Tx bias. The minimum value is two. Reset type: PUC

21.8.38 SAPHATM_B/SAPH_AATM_B Register (Offset = 70h) [reset = 0h]

SAPHATM_B/SAPH_AATM_B is shown in [Figure 21-58](#) and described in [Table 21-43](#).

Return to [Summary Table](#).

Count for the TIMEMARK B Event

Figure 21-58. SAPHATM_B/SAPH_AATM_B Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEMARK_B															
R/W-0h															

Table 21-43. SAPHATM_B/SAPH_AATM_B Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TIMEMARK_B	R/W	0h	This value is used to generate the TIMEMARK B event by comparing with the ASQ timer counter value. TIMEMARK_B value [15:0] are compared to ASQ Timer Counter [15:0] TIMEMARK B event is to turn on the SDHS (ADC converter). The SDHS starts sampling input signal at the TIMEMARK D event. The time between B and D is for the SDHS settling time. The minimum value is two. Reset type: PUC

21.8.39 SAPHATM_C/SAPH_AATM_C Register (Offset = 72h) [reset = 0h]

SAPHATM_C/SAPH_AATM_C is shown in [Figure 21-59](#) and described in [Table 21-44](#).

Return to [Summary Table](#).

Count for the TIMEMARK C Event

Figure 21-59. SAPHATM_C/SAPH_AATM_C Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEMARK_C															
R/W-0h															

Table 21-44. SAPHATM_C/SAPH_AATM_C Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TIMEMARK_C	R/W	0h	This value is used to generate the TIMEMARK C event by comparing with the ASQ timer counter value. TIMEMARK_C value [15:0] are compared to ASQ Timer Counter [15:0] TIMEMARK C event is to apply Rx bias to the PGA input channel. The Rx bias should be applied before the signal arrives at the input channel. The time between C and D is for the Rx bias settling time. The minimum value is two. Reset type: PUC

21.8.40 SAPHATM_D/SAPH_AATM_D Register (Offset = 74h) [reset = 0h]

SAPHATM_D/SAPH_AATM_D is shown in [Figure 21-60](#) and described in [Table 21-45](#).

Return to [Summary Table](#).

Count for the TIMEMARK D Event

Figure 21-60. SAPHATM_D/SAPH_AATM_D Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEMARK_D															
R/W-0h															

Table 21-45. SAPHATM_D/SAPH_AATM_D Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TIMEMARK_D	R/W	0h	This value is used to generate the TIMEMARK D event by comparing with the ASQ timer counter value. TIMEMARK_D value [15:0] are compared to ASQ Timer Counter [15:0] TIMEMARK D event is to trigger the SDHS to start sampling on the input signal. The time between D and E is for the total data sampling time + transfer time via DTC. The minimum value is two. Reset type: PUC

21.8.41 SAPHATM_E/SAPH_AATM_E Register (Offset = 76h) [reset = 0h]

SAPHATM_E/SAPH_AATM_E is shown in [Figure 21-61](#) and described in [Table 21-46](#).

Return to [Summary Table](#).

Count for the TIMEMARK E Event

Figure 21-61. SAPHATM_E/SAPH_AATM_E Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEMARK_E															
R/W-0h															

Table 21-46. SAPHATM_E/SAPH_AATM_E Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TIMEMARK_E	R/W	0h	This value is used to generate the TIMEMARK E event by comparing with the ASQ timer counter value. TIMEMARK_E value [15:0] are compared to ASQ Timer Counter [19:4] TIMEMARK E event is to resume the measurement sequence. The ASQ time counter is reset. The minimum value is two. Reset type: PUC

21.8.42 SAPHATM_F/SAPH_AATM_F Register (Offset = 78h) [reset = 0h]

SAPHATM_F/SAPH_AATM_F is shown in [Figure 21-62](#) and described in [Table 21-47](#).

Return to [Summary Table](#).

Count for the TIMEMARK F Event

Figure 21-62. SAPHATM_F/SAPH_AATM_F Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEMARK_F															
R/W-0h															

Table 21-47. SAPHATM_F/SAPH_AATM_F Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TIMEMARK_F	R/W	0h	This value is used to generate the TIMEMARK F event by comparing with the ASQ timer counter value. TIMEMARK_F value [15:0] are compared to ASQ Timer Counter [17:2] TIMEMARK F event is to check out if the measurement is completed within this time limit. The ASQ time counter is reset. The TMFTO interrupt is generated if an measurement has not been completed until the event occurs. The event will abort all of the pre-programmed measurment sequences. The minimum value is two. Note: TIMEMAR E should be programmed to be longer than TIMEMARK F. Reset type: PUC

21.8.43 SAPHTBCTL/SAPH_ATBCTL Register (Offset = 7Ah) [reset = 0h]

SAPHTBCTL/SAPH_ATBCTL is shown in [Figure 21-63](#) and described in [Table 21-48](#).

Return to [Summary Table](#).

Time Base Control Register

Figure 21-63. SAPHTBCTL/SAPH_ATBCTL Register

15	14	13	12	11	10	9	8
PSTAT		RESERVED				RESERVED	RESERVED
R-0h		R/W-0h				R-0h	R-0h
7	6	5	4	3	2	1	0
PSSV				RESERVED	TSTOP	TSTART	TCLR
R/W-0h				R/W-0h	W-0h	W-0h	W-0h

Table 21-48. SAPHTBCTL/SAPH_ATBCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	PSTAT	R	0h	These bits reflect the current power state of the USS module. These bits are also reflected in "energy trace" of the debug tools. As simple unidirectional measurement sequence would show the sequence: "0,1,2,1,3,1,0" Reset type: PUC 0h (R/W) = ASQ inactive or waiting for trigger (ASQ timer not running) 1h (R/W) = ASQ timer running (ASQ timer tuning) 2h (R/W) = PPG during ping process 3h (R/W) = ADC during acquisition process
13-10	RESERVED	R/W	0h	
9	RESERVED	R	0h	Reserved
8	RESERVED	R	0h	Reserved
7-4	PSSV	R/W	0h	ASQ pre-scaler shift. The value written to the PSSV bits shifts the start point of the ASQ's pre-scaler. Note that the value only affects the first cycle of the pre-scaler. 0 = No shift 1 = The pre-scaler starts 1 clock later 2 = The pre-scaler starts 2 clocks later ... 15 = The pre-scaler starts 15 clocks later Reset type: PUC
3	RESERVED	R/W	0h	
2	TSTOP	W	0h	The ASQ time counter stop. Writing '1' to this bit stops the counter. This bit is self cleared. TSTOP, TSTART, and TCLR bits are offered for only debugging purpose. It is not recommend to use this bit while ASQ is active. Note: This bit is write only. Reading always returns with zero.
1	TSTART	W	0h	The ASQ time counter start. Writing '1' to this bit starts the counter. This bit is self cleared. TSTOP, TSTART, and TCLR bits are offered for only debugging purpose. It is not recommend to use this bit while ASQ is active. Note: This bit is write only. Reading always returns with zero.
0	TCLR	W	0h	The ASQ time counter clear. Writing '1' to this bit clears the the counter value. The counter must be stopped prior to be cleared. This bit is self cleared. TSTOP, TSTART, and TCLR bits are offered for only debugging purpose. It is not recommend to use this bit while ASQ is active. Note: This bit is write only. Reading always returns with zero.

21.8.44 SAPHATIMLO/SAPH_AATIMLO Register (Offset = 7Ch) [reset = 0h]

SAPHATIMLO/SAPH_AATIMLO is shown in [Figure 21-64](#) and described in [Table 21-49](#).

Return to [Summary Table](#).

ASQ Time Counter Low Part [15:0]

Figure 21-64. SAPHATIMLO/SAPH_AATIMLO Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATIMLO															
R-0h															

Table 21-49. SAPHATIMLO/SAPH_AATIMLO Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	ATIMLO	R	0h	ASQ Timer Counter low part. The reading this register returns the counter value [15:0]. Reset type: PUC

21.8.45 SAPHATIMHI/SAPH_AATIMHI Register (Offset = 7Eh) [reset = 0h]

SAPHATIMHI/SAPH_AATIMHI is shown in [Figure 21-65](#) and described in [Table 21-50](#).

Return to [Summary Table](#).

ASQ Time Counter High Part[19:16]

Figure 21-65. SAPHATIMHI/SAPH_AATIMHI Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				ATIMHI			
R-0h				R-0h			

Table 21-50. SAPHATIMHI /SAPH_AATIMHI Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	
3-0	ATIMHI	R	0h	ASQ Timer Counter high part. The reading this register returns the counter value [19:16]. Reset type: PUC

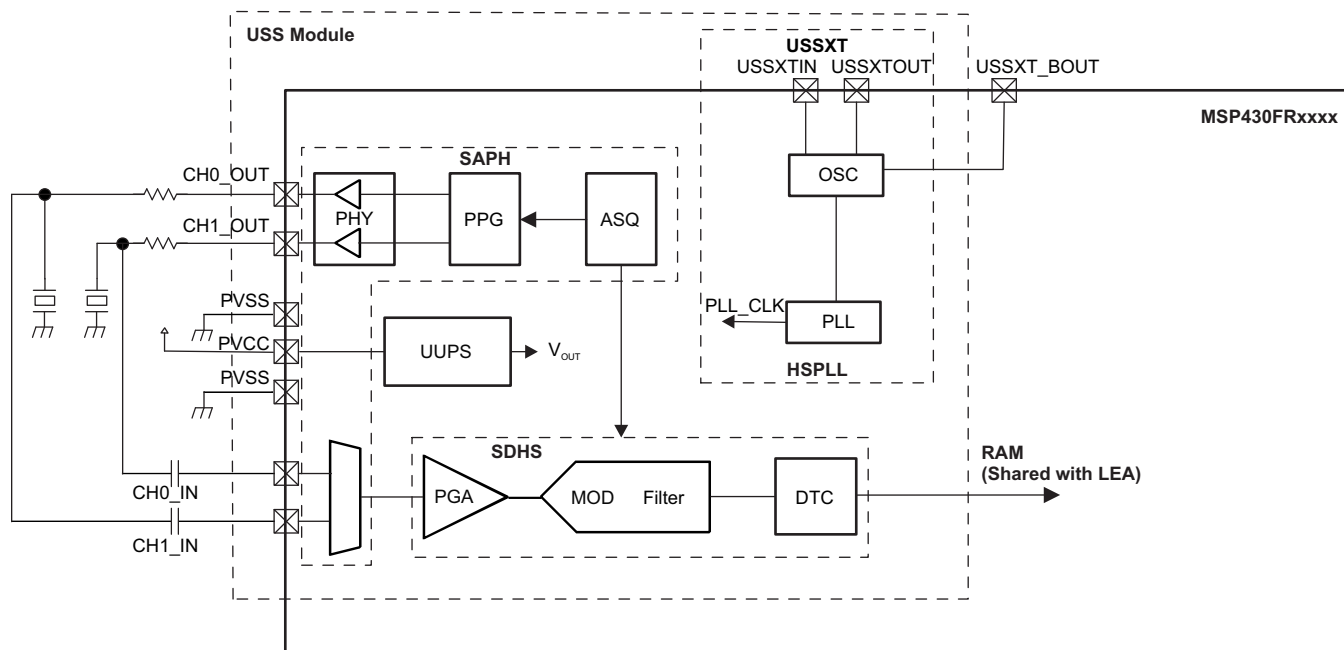
Sigma-Delta High Speed (SDHS)

This chapter describes the operation of the SDHS module.

Topic	Page
22.1 Introduction	569
22.2 SDHS Functional Operation	569
22.3 Interrupts.....	594
22.4 Debug Mode.....	594
22.5 SDHS Registers.....	595

22.1 Introduction

The Sigma-Delta High Speed (SDHS) is a high-performance high-speed 12-bit analog-to-digital converter (ADC). The SDHS is one of the submodules in the Ultrasonic Sensing Solution (USS) module. The USS module is designed for ADC-based ultrasonic sensing technology in various measurement applications. Figure 22-1 shows the block diagram of the USS module.



Copyright © 2017, Texas Instruments Incorporated

Figure 22-1. USS Block Diagram

The SDHS module consists of three blocks: the programmable gain amplifier (PGA), the sigma-delta high speed (SDHS), and the data transfer controller (DTC) (see Figure 22-1).

- PGA block: Applies a gain to the input signal before the SDHS.
- SDHS block: ADC converts that converts the input signal to digital data at the programmed sampling rate.
- DTC block: Transfers the output data from the SDHS to the LEA RAM for data processing.

NOTE: Naming convention for register names and bit fields:

- SDHS registers: RegisterName or RegisterName.BitField
- Other module registers: ModuleNameRegisterName or ModuleNameRegisterName.BitField

22.2 SDHS Functional Operation

The SDHS is a high-performance high-speed ADC that supports output data rates up to 8 Msps. Figure 22-2 shows the SDHS block diagram. The converter consists of a third-order sigma-delta modulator and digital decimation filters. The decimation filters are CIC filters with selectable oversampling ratios of 10, 20, 40, 80, or 160.

Features of the SDHS include:

- Third-order sigma-delta architecture
- High-speed data conversion rate up to 8 Msps
- Low-pass filter with selectable oversampling ratios of 10, 20, 40, 80, or 160

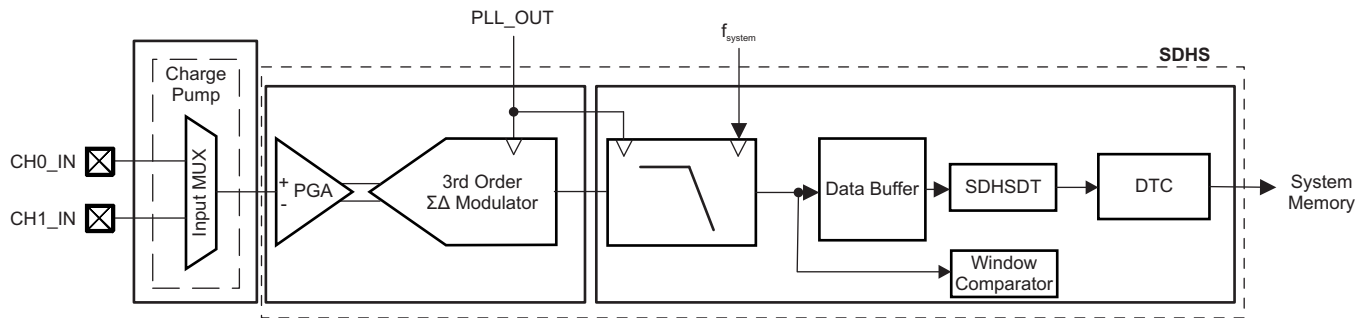


Figure 22-2. SDHS Block Diagram

22.2.1 Input Multiplexer

The input multiplexer is mentioned for completeness here, but is part of the physical interface (PHY). See the PHY module (the SAPH module) for details on how to select the input channels, how bias is provided, and what other features are available on a given device.

22.2.2 Third-Order Modulator

The sigma-delta ADC consists of two parts: modulator and digital filter. A third-order modulator is used in SDHS. The modulator performs oversampling up to 80 MHz against the analog input signal and feeds the digital filter a pulse code modulated (PCM) data stream. The digital filter averages the bitstreams from the modulator over a given number of bits and generates output data at a reduced sampling rate, specified by the oversampling rate (OSR), for further data processing.

Averaging can be used to increase the signal-to-noise performance of a conversion (see [Figure 22-3 a](#) and [b](#)). With a conventional ADC, each factor-of-4 oversampling can improve the SNR by approximately 6 dB or 1 bit. To achieve a 16-bit resolution out of a simple 1-bit ADC would require an impractical oversampling rate of $4^{15} = 1\,073\,741\,824$. To overcome this limitation, the sigma-delta modulator implements a technique called noise shaping. Using a feedback loop and integrators, the quantization noise is pushed to higher frequencies and, thus, much lower oversampling rates are sufficient to achieve high resolutions (see [Figure 22-3 c](#)).

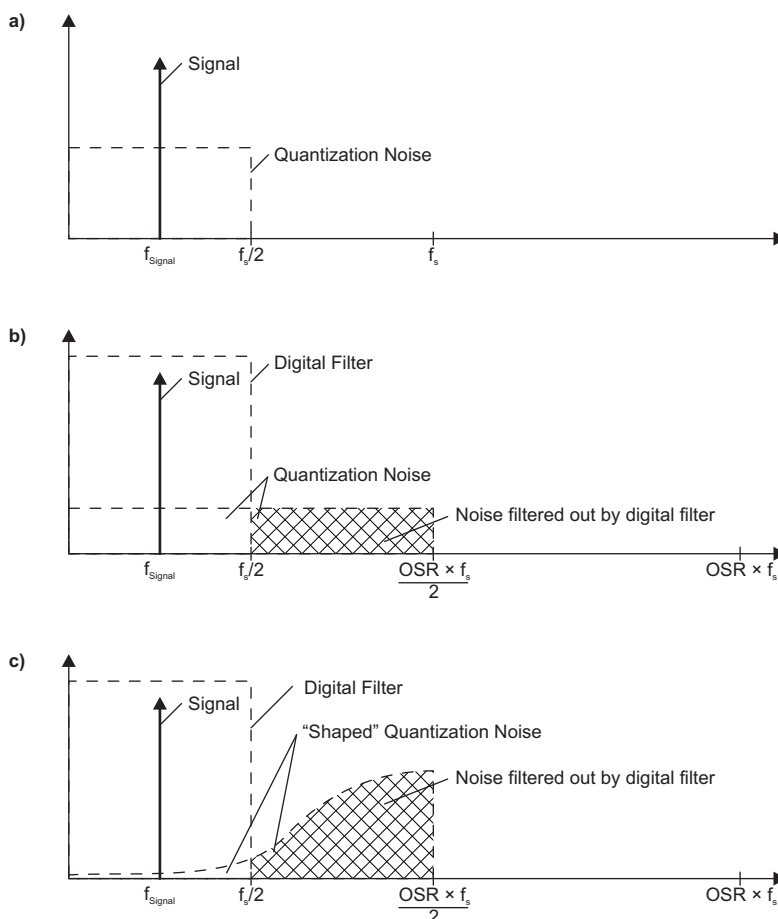


Figure 22-3. Sigma-Delta Principle

22.2.3 Digital Output

The SDHS digital filter processes the data stream from the modulator. The filter consists of two stages. The first stage is a 7th-order CIC filter (CIC⁷), which is always enabled and its decimation ratio is 10. The second-stage filter is a 1st-order CIC filter (CIC¹) which is automatically enabled when the OSR ratio is greater than 10. The second-stage filter can have decimation ratio of 2, 4, 8, or 16. With an OSR ratio greater than 10, both of the CIC filters are enabled and cascaded. See [Figure 22-7](#) for details.

22.2.3.1 CIC⁷ Filter

[Figure 22-4](#) shows the structure of a CIC⁷ filter, which is selected when SDHSCTL1.OSR = 10.

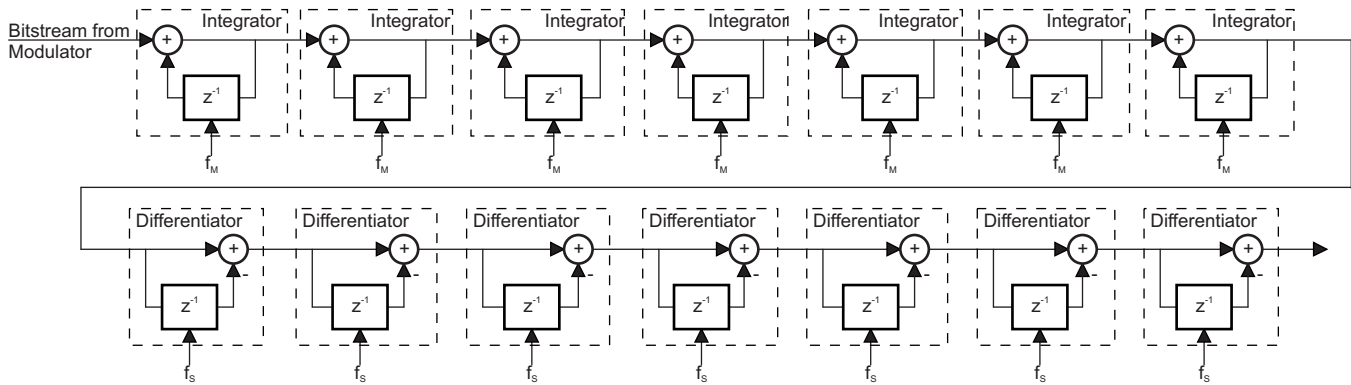


Figure 22-4. CIC⁷ Filter Structure

The transfer function is described in the z-domain by Equation 11.

$$H(z) = \left(\frac{1}{\text{OSR}} \times \frac{1 - z^{-\text{OSR}}}{1 - z^{-1}} \right)^7 \quad (11)$$

The transfer function is described in the frequency domain by Equation 12.

$$H(f) = \left(\frac{\text{sinc}\left(\text{OSR}\pi \frac{f}{f_M}\right)}{\text{sinc}\left(\pi \frac{f}{f_M}\right)} \right)^7 = \left(\frac{1}{\text{OSR}} \times \frac{\sin\left(\text{OSR} \times \pi \times \frac{f}{f_M}\right)}{\sin\left(\pi \times \frac{f}{f_M}\right)} \right)^7$$

where

- OSR = The ratio of the modulator frequency f_M to the sample frequency f_S (12)

Figure 22-5 shows the filter frequency response beyond f_S (normalized), and Figure 22-6 shows the filter frequency response within f_S . The first filter notch is always at $f_S = f_M / \text{OSR}$. The digital filter for the SDHS converter completes the decimation of the digital bit stream and outputs the new conversion result to the SDHSDT register at the sample frequency, f_S .

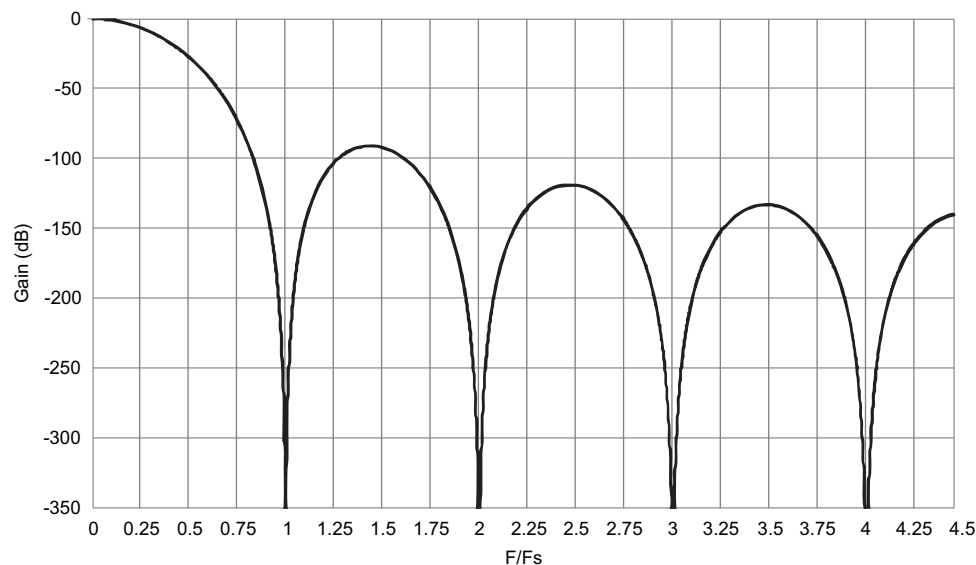


Figure 22-5. SDHS Filter Frequency Response, SDHCTL1.OSR = 10

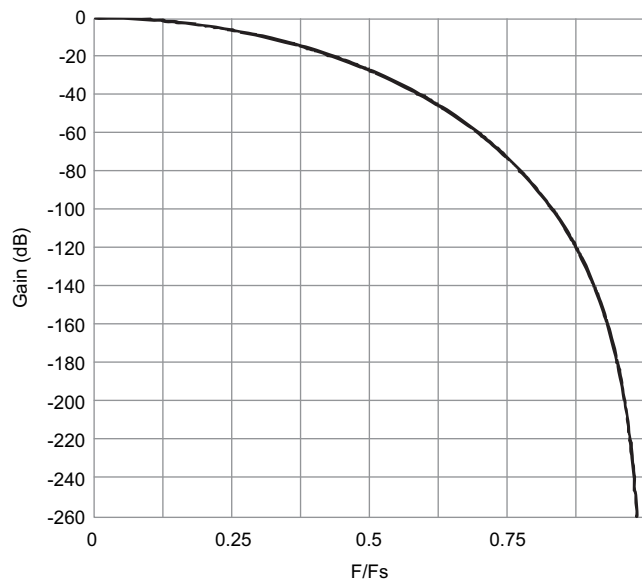


Figure 22-6. SDHS Filter Frequency Response Within f_s , SDHSCTL1.OSR = 10

22.2.3.2 CIC¹ Filter

The second-stage filter is a 1st-order CIC filter, which is automatically enabled when the OSR ratio is greater than 10. The second-stage filter can have decimation ratio of 2, 4, 8, or 16. By cascading the CIC⁷ and CIC¹ filters, the OSR ratio can be 10, 20, 40, 80, or 160.

22.2.3.3 Digital Filter Output

Figure 22-7 shows the block diagram of the SDHS digital filter. The CIC⁷ is a 7th-order CIC filter with decimation of 10. The CIC¹ is a 1st-order CIC filter with programmable decimation of 2, 4, 8, or 16. By cascading the two filters, OSR of 10, 20, 40, 80, or 160 is offered. The OSR ratio is configured by the SDHSCTL1.OSR bits. Each output of the CIC⁷ and CIC¹ is truncated to 14 bits. The final output data is 16 bit, but the effective number of bits and data format are programmable through the SDHSCTL0.DALGN, SDHSCTL0.OBR, SDHSCTL0.SHIFT, and SDHSCTL0.DFMSEL bits.

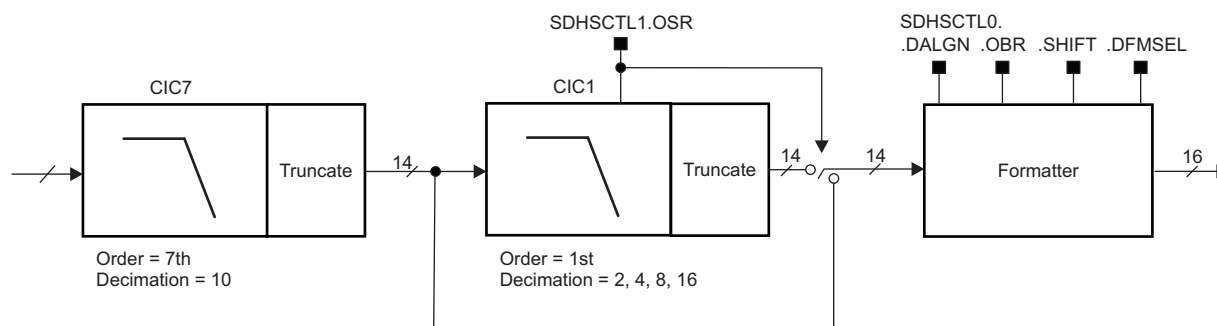


Figure 22-7. Digital Filter Block Diagram

The final frequency response of the SDHS digital filter is determined by SDHSCTL1.OSR bit. Figure 22-8 shows the frequency response of cascading CIC⁷ and CIC¹ beyond f_s (normalized) when SDHSCTL1.OSR = 20.

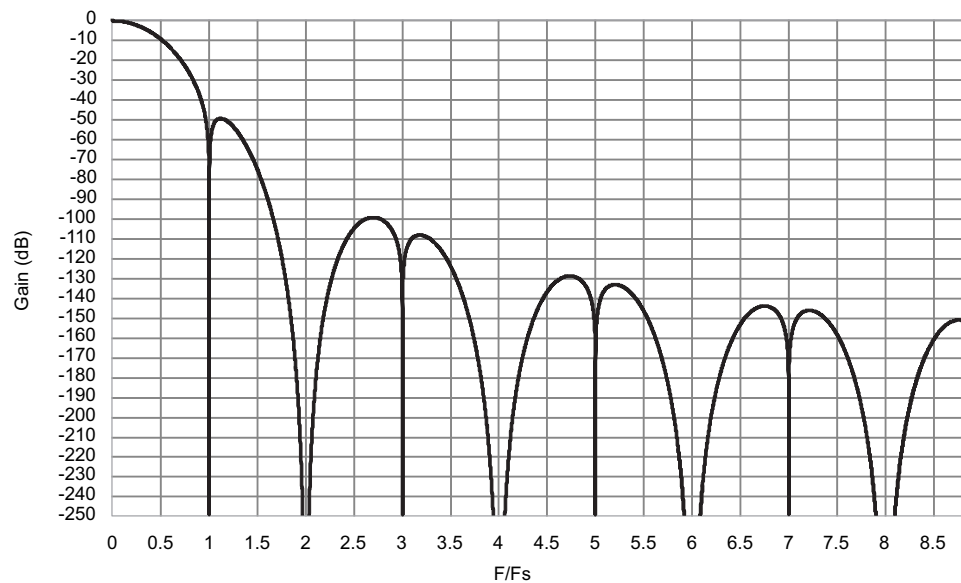


Figure 22-8. SDHS Filter Frequency Response, SDHSCTL1.OSR = 20

Figure 22-9 shows the frequency response of cascading CIC^7 and CIC^1 within f_s (normalized) when SDHSCTL1.OSR = 20.

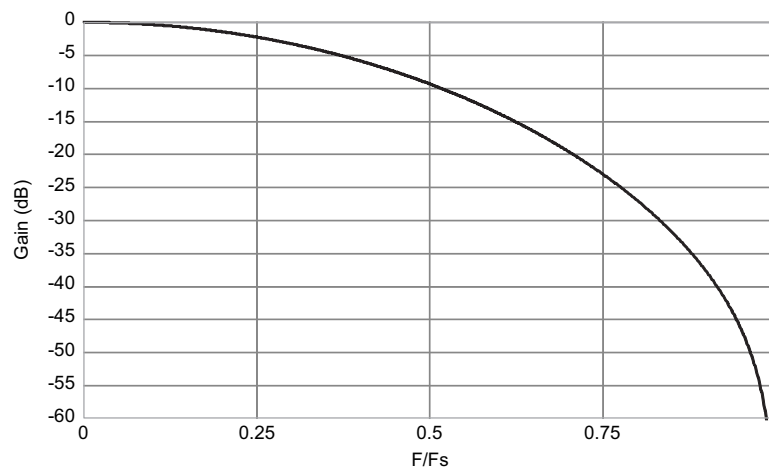


Figure 22-9. SDHS Filter Frequency Response Within f_s , SDHSCTL1.OSR = 20

Figure 22-10 shows the frequency response of cascading CIC^7 and CIC^1 within f_s (normalized) when SDHSCTL1.OSR = 40.

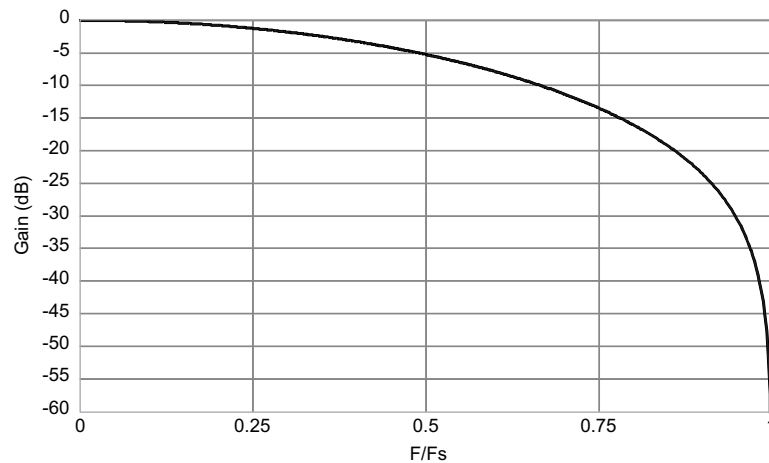


Figure 22-10. SDHS Filter Frequency Response within f_s , SDHSCTL1.OSR = 40

Figure 22-11 shows the frequency response of cascading CIC^7 and CIC^1 within f_s (normalized) when SDHSCTL1.OSR = 80.

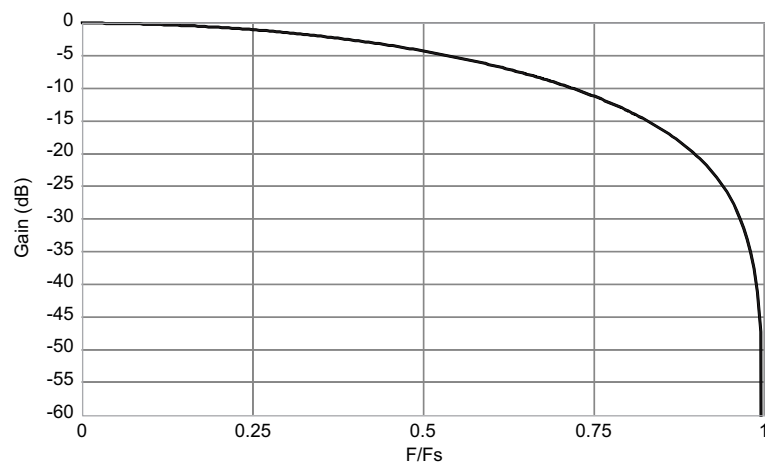


Figure 22-11. SDHS Filter Frequency Response within f_s , SDHSCTL1.OSR = 80

Figure 22-12 shows the frequency response of cascading CIC^7 and CIC^1 within f_s (normalized) when SDHSCTL1.OSR = 160.

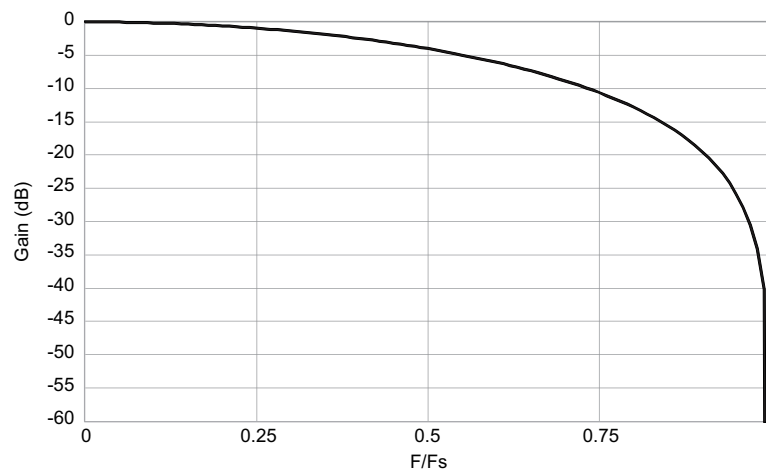


Figure 22-12. SDHS Filter Frequency Response within f_s , SDHSCTL1.OSR = 160

22.2.3.4 Filter Output Data Formats

SDHS filter output can be configured as offset binary format (SDHSCTL0.DFMSEL = 1) or 2s complement format (SDHSCTL0.DFMSEL = 0). The output values range is between 0 to FS when offset binary format is selected, between $-f_s/2$ and $+f_s/2$ when 2s complement format is selected. See [Table 22-1](#) for the data range based on the configuration of SDHSCTL0.DFMSEL, SDHSCTL0.DALGN, and SDHSCTL0.OBR bits.

Table 22-1. Data Format

Format (SDHSCTL0.DFMSEL Bit)	Alignment (SDHSCTL0.DALGN Bit)	Analog Input	Data Output (SDHSCTL0.OBR = 0): 12 Bit	Data Output (SDHSCTL0.OBR = 1): 13 Bit	Data Output (SDHSCTL0.OBR = 2): 14 Bit
2s Complement (0)	Right Aligned (0)	$+f_s/2$	0x7FF	0xFFF	0x1FFF
		0	0	0	0
		$-f_s/2$	0x800	0x1000	0x2000
Offset Binary (1)	Right Aligned (0)	$+f_s/2$	0xFFF	0x1FFF	0x3FFF
		0	0x800	0x1000	0x2000
		$-f_s/2$	0	0	0

The effective number of bits in the output data format is determined by SDHSCTL0.OBR and SDHSCTL0.SHIFT bits. [Figure 22-13](#) shows bits selection from the filter output to the SDHSDT register with left alignment mode (SDHSCTL0.DALGN = 0). [Figure 22-14](#) shows bits selection from the filter output to the SDHSDT register with right alignment mode (SDHSCTL0.DALGN = 1). Increasing the SDHSCTL0.SHIFT value is equivalent of multiplying the output data by 2 at every shift. Take care when using SDHSCTL0.SHIFT bits and ensure that signal overflow never happens after shifting.

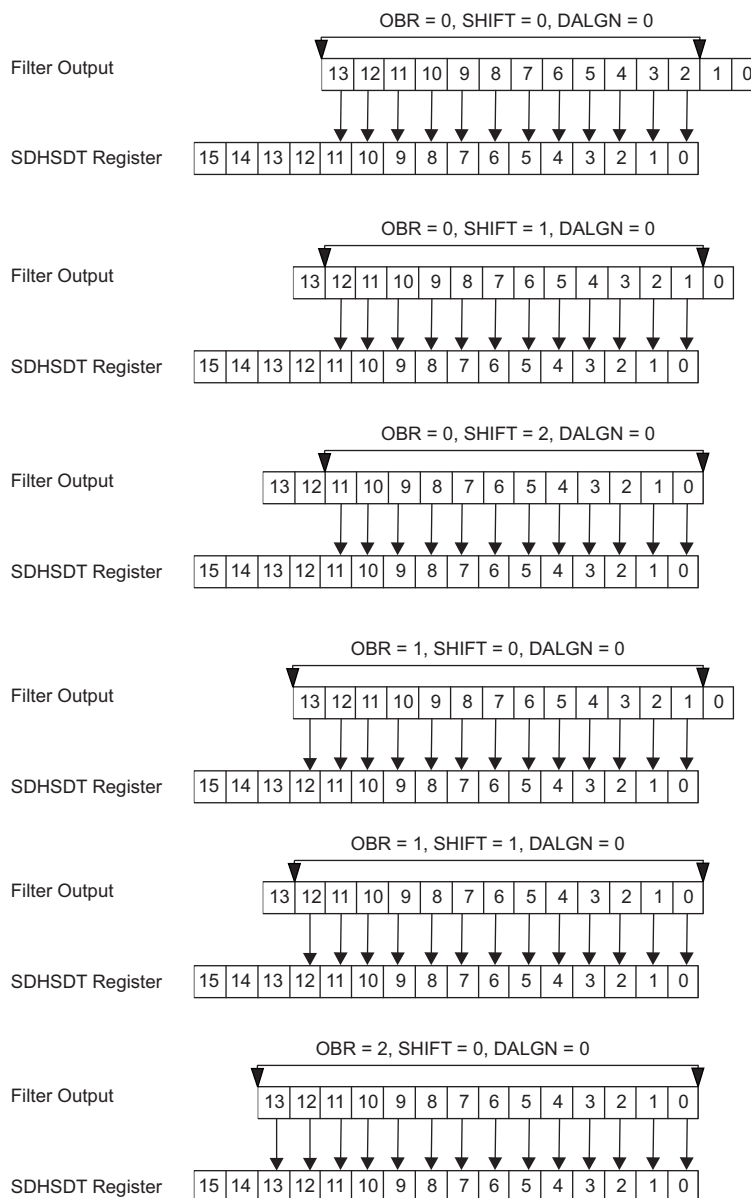


Figure 22-13. Bits Selection From Filter to the Data Register (SDHSCTL0.DALGN = 0)

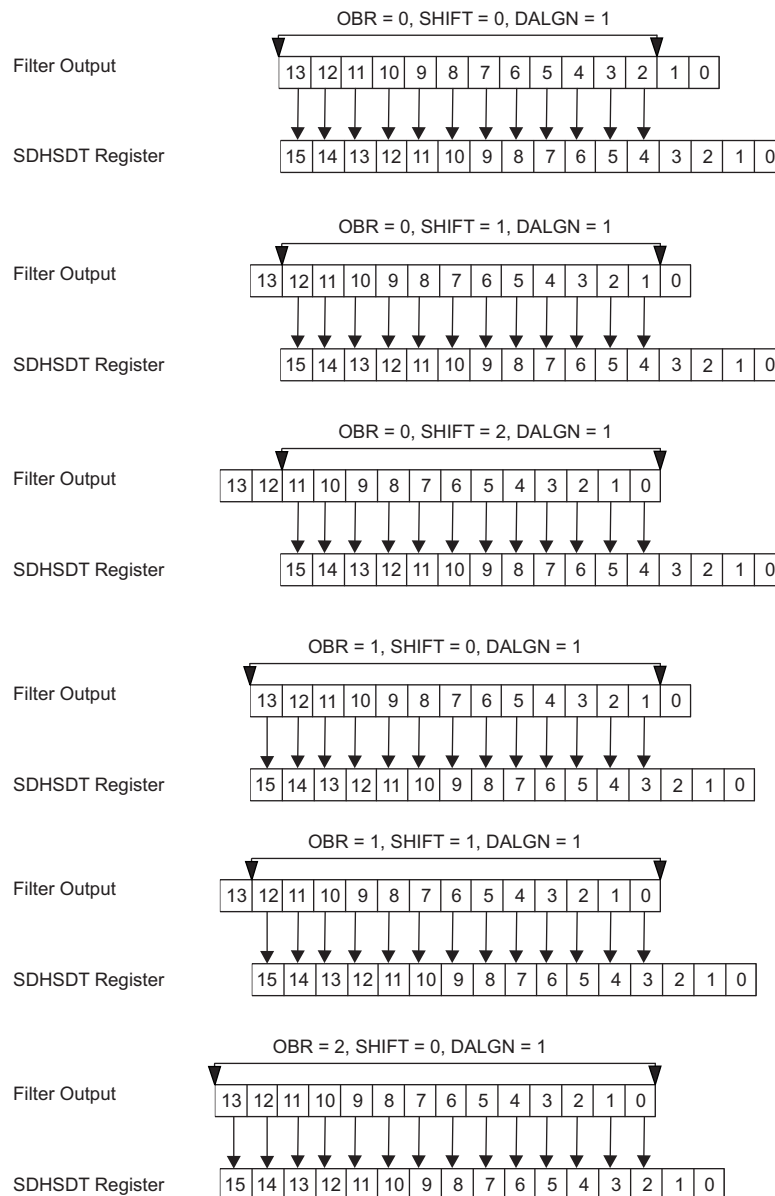


Figure 22-14. Bits Selection From Filter to the Data Register (SDHSCTL0.DALGN = 1)

22.2.4 Data Transfer Controller (DTC) and Internal Data Buffer

The SDHS supports output data rates up to 8 Msps, which is faster than the system DMA can support, so the SDHS has a dedicated Data Transfer Controller (DTC) and an internal data buffer to support up to 8-MHz data transfer speed to the target memory.

Figure 22-14 shows the block diagram of the output data path. A conversion result from digital filter goes first to the internal data buffer. The buffer has 64-word depth. As soon as a new data is available in the buffer, the data is latched with the system clock (called synchronization to the system clock) and is written to SDHSDT register. Then the DTC reads the data from SDHSDT register and transfers to the destination memory location.

The DTC may require more than one sample period to transfer the data to system memory. Thus, the buffer depth is selected to achieve the 8-MHz data transfer speed when the system clock is equal to or higher than the SDHS output data rate. Take care when selecting SDHSCTL1.OSR bits or the system clock frequency. The system clock frequency must be equal to or greater than the SDHS output data rate, or a data overflow may occur.

- System clock frequency $\geq$ SDHS output data rate
- SDHS output data rate = PLL output frequency / SDHSCTL1.OSR

The DTC is enabled by default but can be disabled when SDHSCTL2.DTCOFF = 1. When the DTC is disabled, the data in the SDHSDT register must be read by CPU. If the SDHSDT register has not been read by CPU over 64 sample periods, the internal buffer becomes full and does not take any more new data. In the case, newly generated data is lost and causes the overflow interrupt (SDHSRIS.OVF).

NOTE: If data conversion is performed with SDHSCTL2.DTCOFF = 1, the data buffer may not be empty when the conversion stops. The CPU can continue to read the data until the buffer is empty. While the CPU is reading the data from the buffer after conversion is stopped, data format configurations (SDHSCTL0.DFMSEL, SDHSCTL0.DALGN, and SDHSCTL0.OBR bits) must not be changed.

The destination system memory is the LEA RAM, which is part of system memory. The DTC automatically recognize the base address of the LEA RAM in the target device. Only offset address needs to be configured to SDHSDTCDA register.

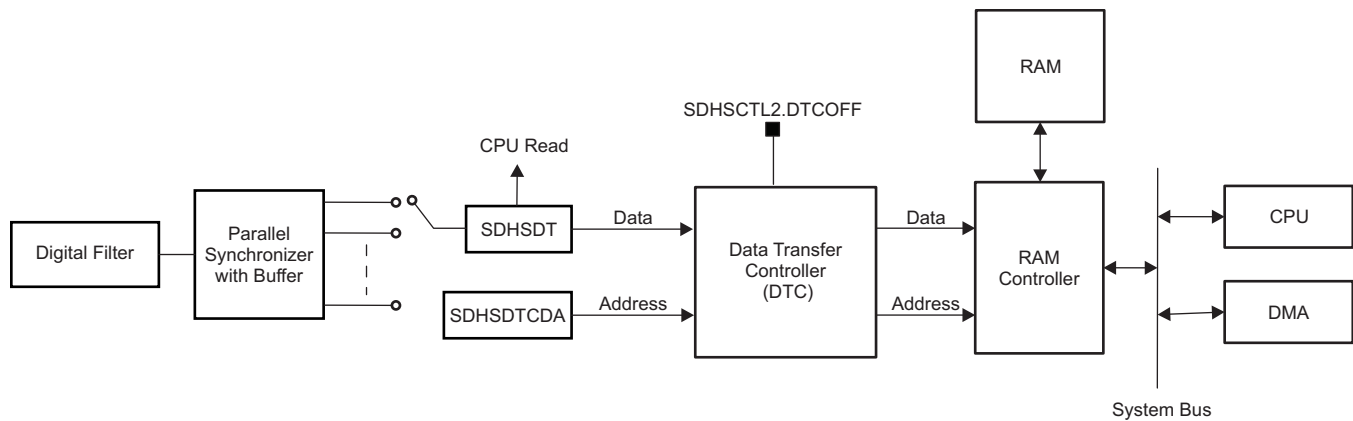
- Destination address = LEA RAM base address + offset address
- Offset address = SDHSDTCDA register value x 2 (for example, if SDHSDTCDA = 2, then the destination address = LEA base address + 4)

The SDHSDTCCA register value automatically increases by 1 at every data transfer, so the current target address can be monitored by reading SDHSDTCDA register. The maximum offset address is 64KB. The SDHSDTCDA register value resets when the offset address reaches the 64KB limit and starts from zero again.

NOTE: The DTC block supports up to 64KB of memory size; however, the available memory (LEA RAM) could be smaller than 64KB (see the device-specific data sheet). If the SDHSDTCDA register goes beyond the available memory space, data transferred to the outside of the available memory is lost. In this case, the DTC block does not overwrite data in other memory areas, because it accesses only the LEA RAM.

Take care not to go beyond the available memory address range. The DTC cannot detect if the offset address goes beyond the available LEA RAM. TI recommends using SDHSCTL2.SMPSZ[9:0], which controls the total number of samples.

NOTE: While the DTC is transferring data to the destination memory, the memory is blocked from being accessed by CPU or DMA. If the CPU or DMA attempts to access the same memory, 0x3FFF is returned and an NMI is generated (DACCESSIFG).


Figure 22-15. Data Output Path

22.2.5 PGA Gain Control

The programmable gain amplifier (PGA) allows adjusting the input signal amplitude to satisfy the input range of the SDHS modulator. The input gain can be adjusted by SDHSCTL6.PGA_GAIN[5:0]. See [Table 22-2](#) for the PGA gain table.

Table 22-2. PGA Gain Table

SDHSCTL6.PGA_GAIN[5:0] Bits	PGA Gain (Typical) (dB)	Tolerance	Shift (Entire Table)
00xxxxb	−6.5	±1 dB	±1 dB
010000b	−6.5	±1 dB	±1 dB
010001b	−6.5	±1 dB	±1 dB
010010b	−5.5	±1 dB	±1 dB
010011b	−4.6	±1 dB	±1 dB
010100b	−4.1	±1 dB	±1 dB
010101b	−3.3	±1 dB	±1 dB
010110b	−2.3	±1 dB	±1 dB
010111b	−1.4	±1 dB	±1 dB
011000b	−0.8	±1 dB	±1 dB
011001b	0.1	±1 dB	±1 dB
011010b	1.0	±1 dB	±1 dB
011011b	1.9	±1 dB	±1 dB
011100b	2.6	±1 dB	±1 dB
011101b	3.5	±1 dB	±1 dB
011110b	4.4	±1 dB	±1 dB
011111b	5.2	±1 dB	±1 dB
100000b	6.0	±1 dB	±1 dB
100001b	6.8	±1 dB	±1 dB
100010b	7.7	±1 dB	±1 dB
100011b	8.7	±1 dB	±1 dB
100100b	9.0	±1 dB	±1 dB
100101b	9.8	±1 dB	±1 dB
100110b	10.7	±1 dB	±1 dB
100111b	11.7	±1 dB	±1 dB
101000b	12.2	±1 dB	±1 dB

Table 22-2. PGA Gain Table (continued)

SDHSCTL6.PGA_GAIN[5:0] Bits	PGA Gain (Typical) (dB)	Tolerance	Shift (Entire Table)
101001b	13.0	±1 dB	±1 dB
101010b	13.9	±1 dB	±1 dB
101011b	14.9	±1 dB	±1 dB
101100b	15.5	±1 dB	±1 dB
101101b	16.3	±1 dB	±1 dB
101110b	17.2	±1 dB	±1 dB
101111b	18.2	±1 dB	±1 dB
110000b	18.8	±1 dB	±1 dB
110001b	19.6	±1 dB	±1 dB
110010b	20.5	±1 dB	±1 dB
110011b	21.5	±1 dB	±1 dB
110100b	22.0	±1 dB	±1 dB
110101b	22.8	±1 dB	±1 dB
110110b	23.6	±1 dB	±1 dB
110111b	24.6	±1 dB	±1 dB
111000b	25.0	±1 dB	±1 dB
111001b	25.8	±1 dB	±1 dB
111010b	26.7	±1 dB	±1 dB
111011b	27.7	±1 dB	±1 dB
111100b	28.1	±1 dB	±1 dB
111101b	28.9	±1 dB	±1 dB
111110b	29.8	±1 dB	±1 dB
111111b	30.8	±1 dB	±1 dB

22.2.6 SDHS Power and Conversion Control

The SDHS has two trigger sources for power and conversion. Figure 22-16 shows the connections of the power and conversion trigger signals to the SDHS. Table 22-3 lists the use cases of the trigger signals.

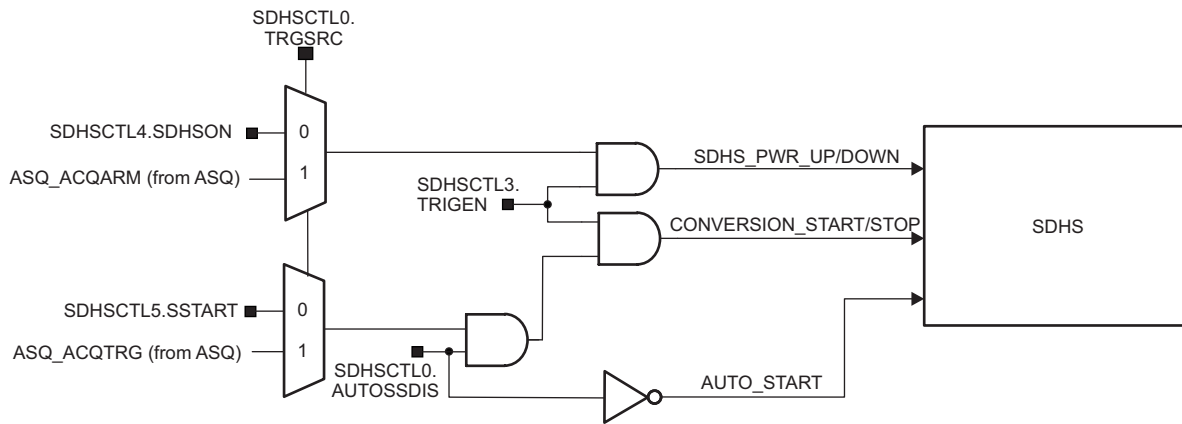


Figure 22-16. SDHS Power and Conversion Trigger Source

Table 22-3. Control Signals for Power and Conversion

SDHSCTL3.TRIGEN	SDHSCTL0.TRG SRC	SDHSCTL0.AUT OSSDIS	Trigger Signal			
			Power Up	Power Down	Conversion Start	Conversion Stop
1	0	0	SDHSCTL4.SDHSON = 0 → 1	SDHSCTL4.SDHSON = 1 → 0	Start automatically after power up	See Section 22.2.12
1	0	1	SDHSCTL4.SDHSON = 0 → 1	SDHSCTL4.SDHSON = 1 → 0	SDHSCTL5.SSTART = 0 → 1	See Section 22.2.12
1	1	0	Not supported	Not supported	Not supported	Not supported
1	1	1	ASQ_ACQARM = 0 → 1 (from ASQ)	ASQ_ACQARM = 1 → 0 (from ASQ)	ASQ_ACQTRIG = 0 → 1 (from ASQ)	See Section 22.2.12
0	x	x	Not supported	Not supported	Not supported	Not supported

22.2.6.1 SDHS in Auto Mode and Register Mode

The SDHS is one of the submodules in the USS module. The USS module is designed for ADC-based ultrasonic sensing technology in various measurement applications. The USS module has its own power module (see [USS](#)) to provide the required voltages to the USS submodules including the SDHS. The USS module support two control modes, register mode and auto mode. In the register mode, the SDHS is controlled by its own registers. In the auto mode, the SDHS is controlled by the ASQ. See [SAPH](#) and [USS](#) for details.

Table 22-4. USS Auto Mode and Register Mode

USS Control Mode	Configuration for SDHS	Control by: SDHS Power up	Control by: SDHS Conversion Start
Register	SDHSCTL0.TRG SRC = 0	SDHSCTL4.SDHSON	SDHSCTL5.SSTART
Auto	SDHSCTL0.TRG SRC = 1	ASQ (ASQ_ACQARM signal)	ASQ (ASQ_ACQTRIG signal)

22.2.6.1.1 SDHS Configuration in Register Mode

In register mode (SDHSCTL0.TRGSRC = 0), the SDHS can be used as an stand-alone ADC converter. See the following sequence for the correct SDHS configuration:

1. Turn on the USS module and wait for UUPCTL.UPSTATE = 3.
2. Configure all registers except the SDHSCTL3, SDHSCTL4, and SDHSCTL5 registers (no specific order is required, SDHSCTL0.TRGSRC = 0).
3. Enable trigger sources: Set SDHSCTL3.TRIGEN = 1.
4. Power up the SDHS: Set SDHSCTL4.SDHSON = 1 (If SDHSCTL0.AUTOSSDIS = 1, then the remaining steps are not required).
5. Wait for the SDHS settling time (see [SDHS](#)).
6. Start conversion: Set SDHSCTL5.SSTART = 1.

[Figure 22-17](#) shows this configuration sequence.

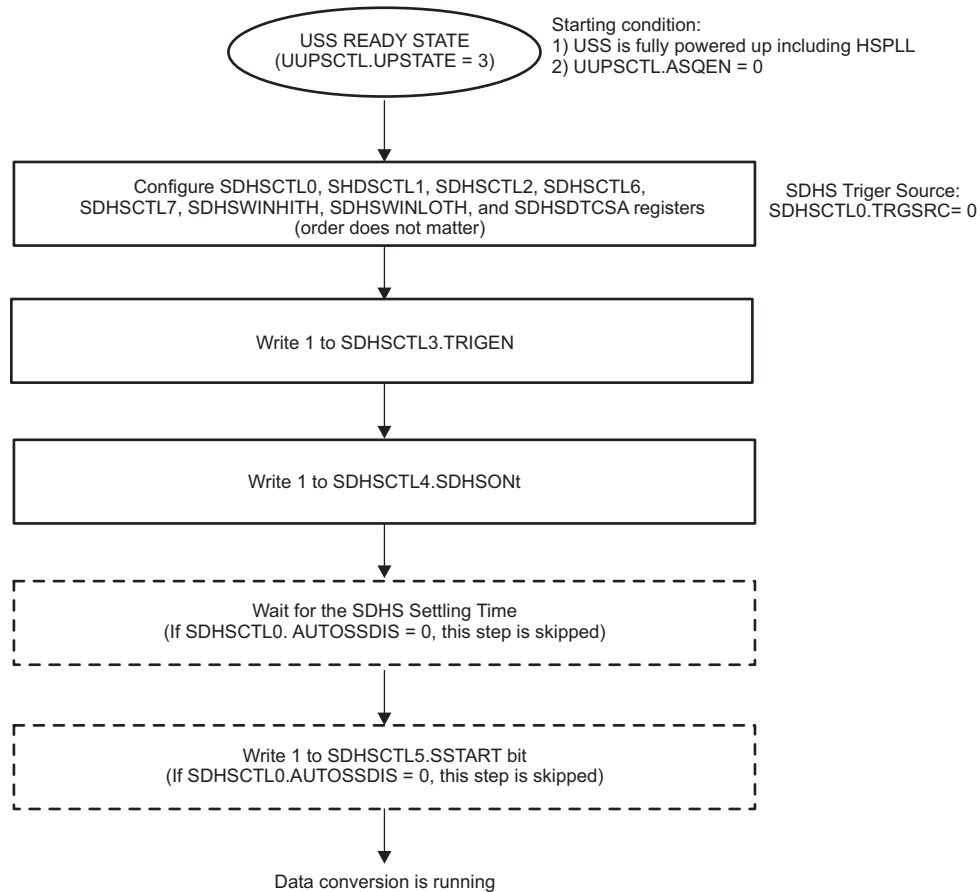


Figure 22-17. SDHS Operation in Register Mode (SDHSCTL0.TRGSRC = 0)

22.2.6.1.2 SDHS Configuration in Auto Mode

In auto mode (SDHSCTL0.TRGSRC = 1), the SDHS is fully controlled by the ASQ. See the following sequence for the correct SDHS configuration:

1. Configure all registers except SDHSCTL3, SDHSCTL4, and SDHSCTL5 (no specific order is required, SDHSCTL0.TRGSRC = 1).
2. Enable trigger sources: Set SDHSCTL3.TRIGEN = 1.
3. Turn on the USS module.

[Figure 22-18](#) shows an example of USS measurement in auto mode (see [USS](#) for details).

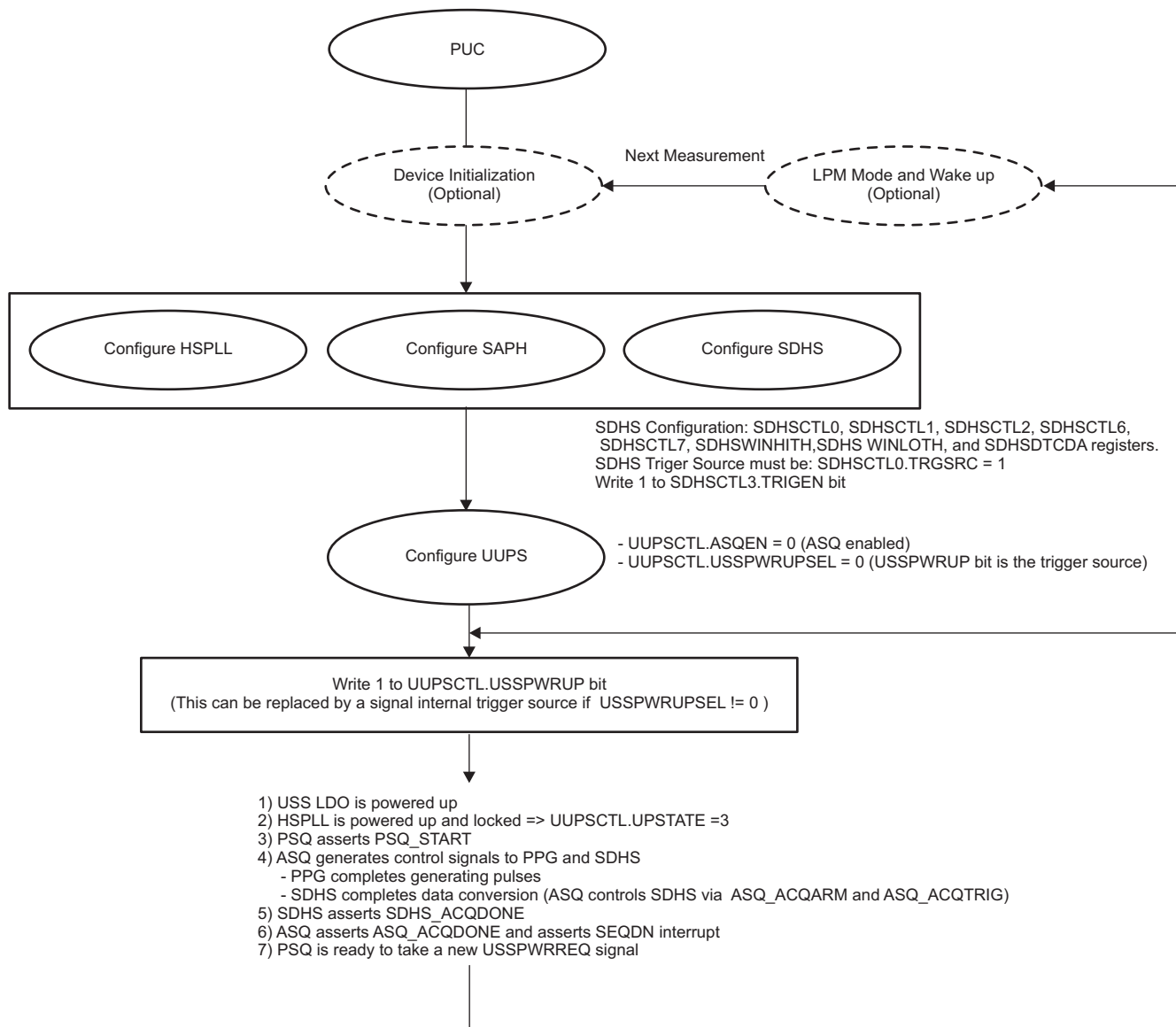


Figure 22-18. SDHS Operation as Part of USS Measurement (SDHSCTL0.TRGSRC = 1)

22.2.7 TRIGEN Bit and SDHS_LOCK Bit

SDHSCTL3.TRIGEN has two functional roles:

- The first role is to enable the SDHS to receive the power trigger and conversion trigger signals (see [Figure 22-16](#)). SDHSCTL3.TRIGEN must be set to 1 before applying the trigger signals to the SDHS.
- The second role is to lock the SDHSCTL0, SDHSCTL1, SDHSCTL2, SDHSCTL7, SDHSWINHITH, SDHSWINLOTH, and SDHSDTCDA registers.

When SDHSCTL3.TRIGEN = 1, those registers cannot be modified. Thus, SDHSCTL3.TRIGEN must be set to 1 after updating the SDHS registers (except SDHSCTL3, SDHSCTL4, and SDHSCTL5) and before powering up the SDHS. See [Section 22.2.6.1.1](#) for an example of the SDHS register configuration sequence. SDHSCTL3.TRIGEN can be used as the power-up trigger signal (see [Figure 22-19](#) and [Figure 22-19](#)). However, this is not recommended when SDHSCTL0.AUTOSDIS = 1, because the SDHS may not be fully settled before data conversion starts.

After the SDHS is triggered for power up, SDHSCTL5.SDHS_LOCK is automatically set to 1. When SDHSCTL5.SDHS_LOCK = 1, the SDHSCTL3 register is locked. Both SDHSCTL3.TRIGEN and SDHS_LOCK protect the SDHS registers from inadvertent modifications while the SDHS is active. The PGA gain registers (SDHSCTL6) are not locked even after SDHS is powered up; however, take care when updating the PGA gain while SDHS is performing data conversion. Expect a transition period before the new gain is applied (see the device-specific data sheet for the PGA gain settling time).

Table 22-5. SDHSCTL3.TRIGEN Bit and SDHSCTL5.SDHS_LOCK Bit

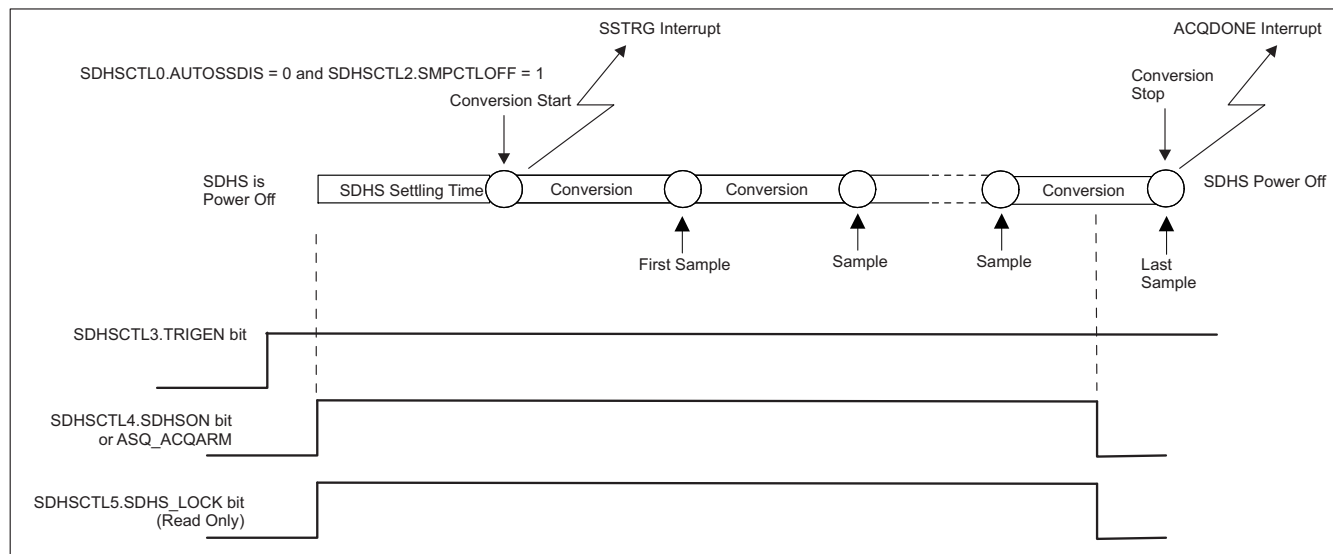
Control Bit	Type	How to Set the Control Bit	Registers Locked
SDHSCTL3.TRIGEN	Read/Write	Write 1 to SDHSCTL3.TRIGEN bit	SDHSCTL0, SDHSCTL1, SDHSCTL2, SDHSCTL7, SDHSWINHITH, SDHSWINLOTH, and SDHSDTCDA
SDHSCTL5.SDHS_LOCK	Read Only	When SDHS_PWR_UP is asserted	SDHSCTL3

Table 22-6. Timing of the SDHS_LOCK bit

Type	SDHSCTL0.TRGSRC = 0	SDHSCTL0.TRGSRC = 1
SDHS power trigger	SDHSCTL4.SDHSON = 1 (by software)	ASQ_ACQARM = 1 (from ASQ)
Time to set SDHS_LOCK bit after the trigger	No delay	Delay = 4 × system clock period + 4 × (PLL clock period × 10)

To update the SDHS registers, the SDHS must be powered off first, then write 0 to SDHSCTL3.TRIGEN. When the SDHS is powered off, SDHSCTL5.SDHS_LOCK is automatically cleared to 0.

SDHS power up is synchronized to the trigger source



SDHS power up is synchronized to SDHSCTL3.TRIGEN bit

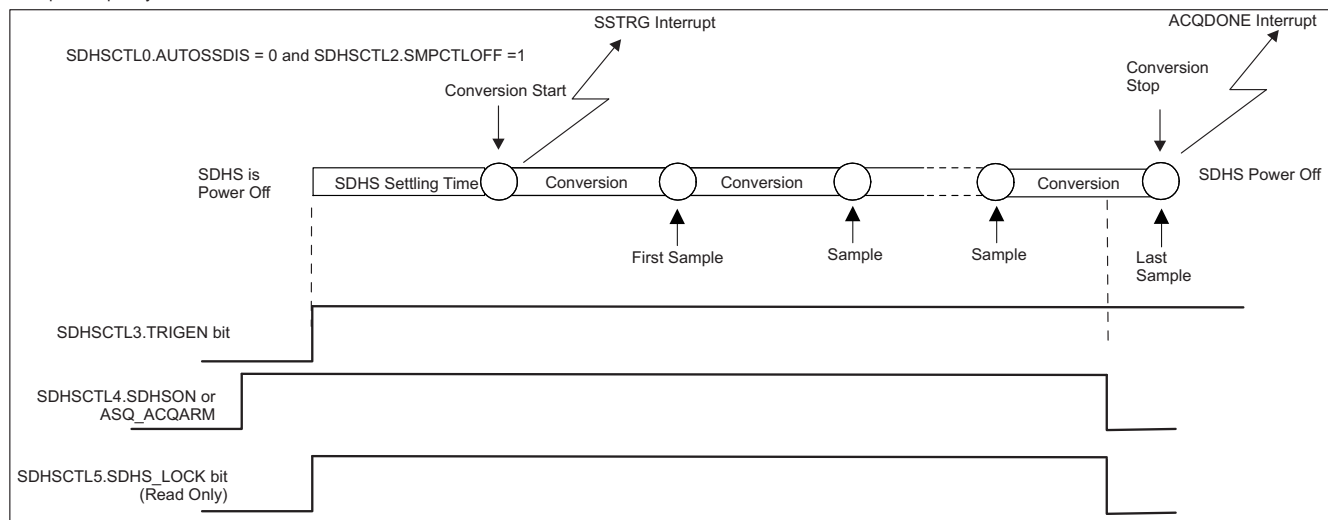
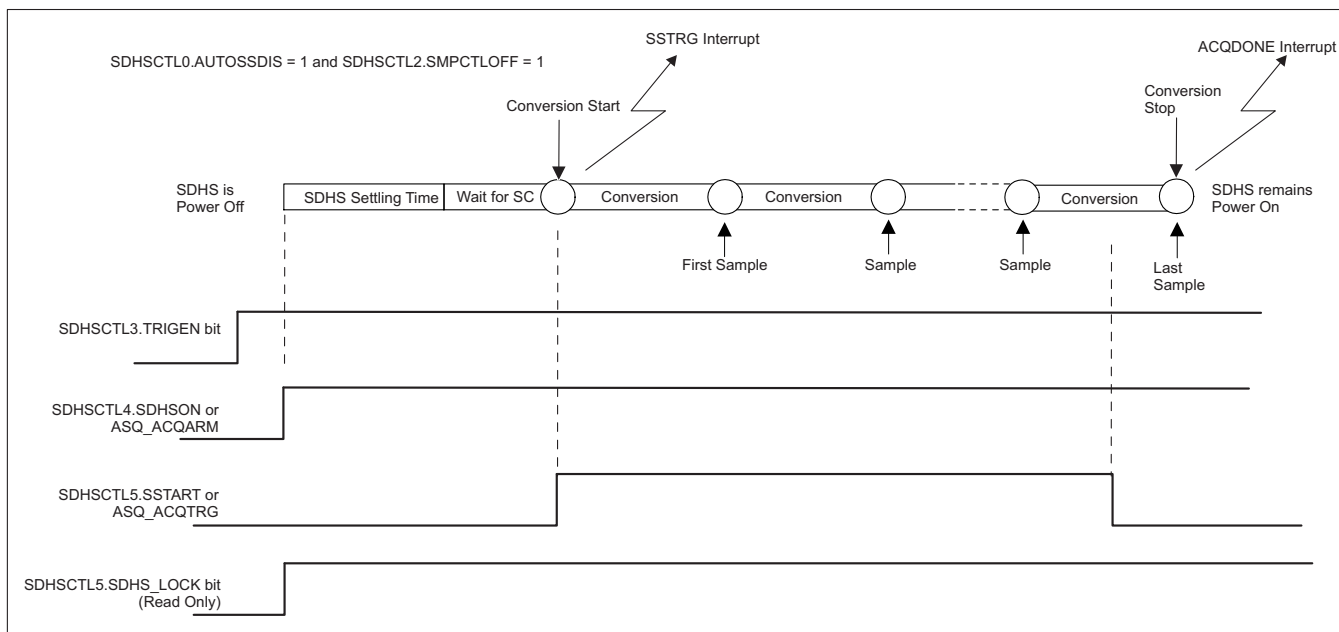


Figure 22-19. Example Using SDHSCTL3.TRIGEN Bit (SDHSCTL0.AUTOSSDIS = 0)

SDHS power up is synchronized to the trigger source



SDHS power up is synchronized to SDHSCTL3.TRIGEN bit (not recommended)

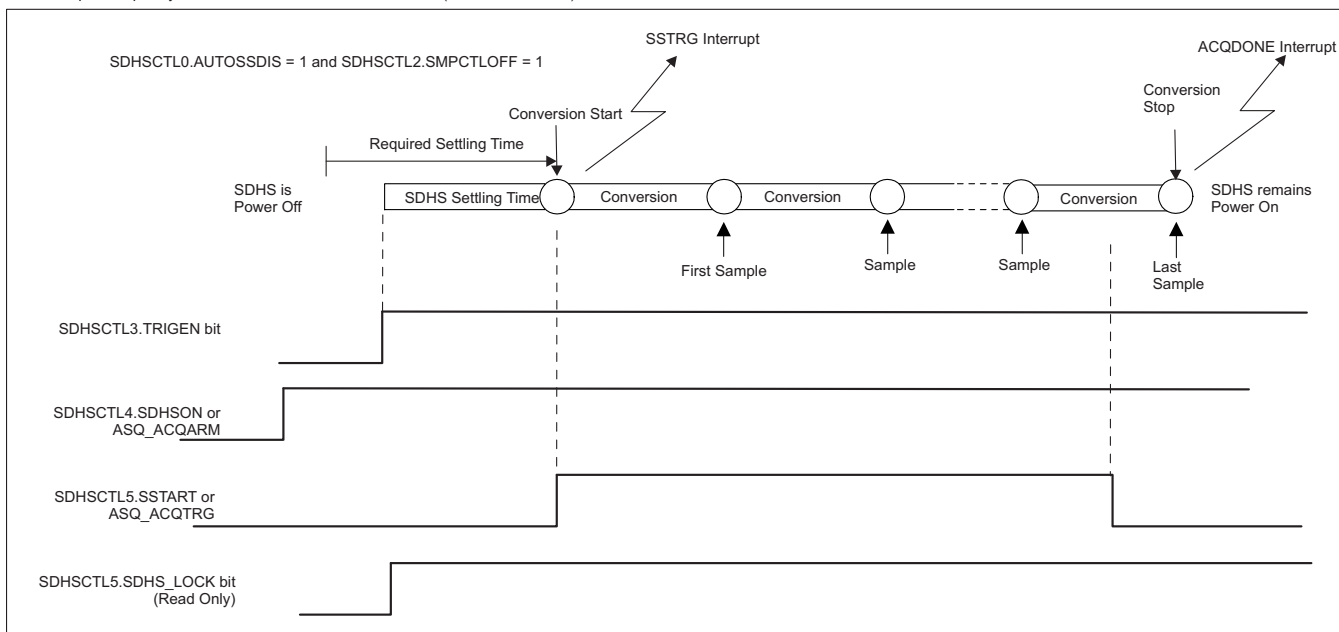


Figure 22-20. Example Using SDHSCTL3.TRIGEN bit (SDHSCTL0.AUTOSSDIS = 1)

NOTE: In the following sections, it is assumed that SDHSCTL3.TRIGEN is set to 1 before powering up the SDHS.

22.2.8 AUTOSSDIS (Auto Conversion Start Disable) Bit

The SDHS supports two different ways to start data conversion after power up (see [Table 22-7](#)).

Table 22-7. Conversion Control Mode

Mode	Control	Power up SDHS		Conversion Start		Operation
	SDHSCTL0.AUTOSSDIS	When SDHSCTL0.TR GSRC = 0	When SDHSCTL0.TR GSRC = 1	When SDHSCTL0.TR GSRC = 0	When SDHSCTL0.TR GSRC = 1	
Auto conversion start: disabled	1	SDHSCTL4.SDHSON = 1	ASQ_ACQARM = 1 (from ASQ)	SDHSCTL5.SS TART = 1	ASQ_ACQTRG = 1 (from ASQ)	Power up and conversion start are controlled separately. The SDHS settling time must be satisfied before asserting the conversion start signal.
Auto conversion start: enabled	0	SDHSCTL4.SDHSON = 1	ASQ_ACQARM = 1 (from ASQ)	Not Required	Not Required	Data conversion automatically begins after power up. The SDHS settling time is automatically applied.

When SDHSCTL0.AUTOSSDIS = 0, automatic conversion start is enabled. Conversion automatically starts after the SDHS is powered on. During power up, the SDHS settling time is monitored and when the SDHS is settled, conversion automatically begins. See [Figure 22-21](#) for the sequence of conversion start and stop in this mode.

When SDHSCTL0.AUTOSSDIS = 1, auto conversion start is disabled (default after reset). In this mode, power control and conversion control are separate (see [Figure 22-22](#)). This mode can be used when data conversion start time needs to be precisely controlled. It is important that the SDHS settling time must be satisfied before data conversion start signal is applied. If the data conversion start trigger signal is applied during the settling time, the SDHS data conversion is automatically delayed until the settling time has passed. SDHS settling time is The greater of the PGA settling time and SDHS modulator settling time. See the device-specific data sheet for the settling times.

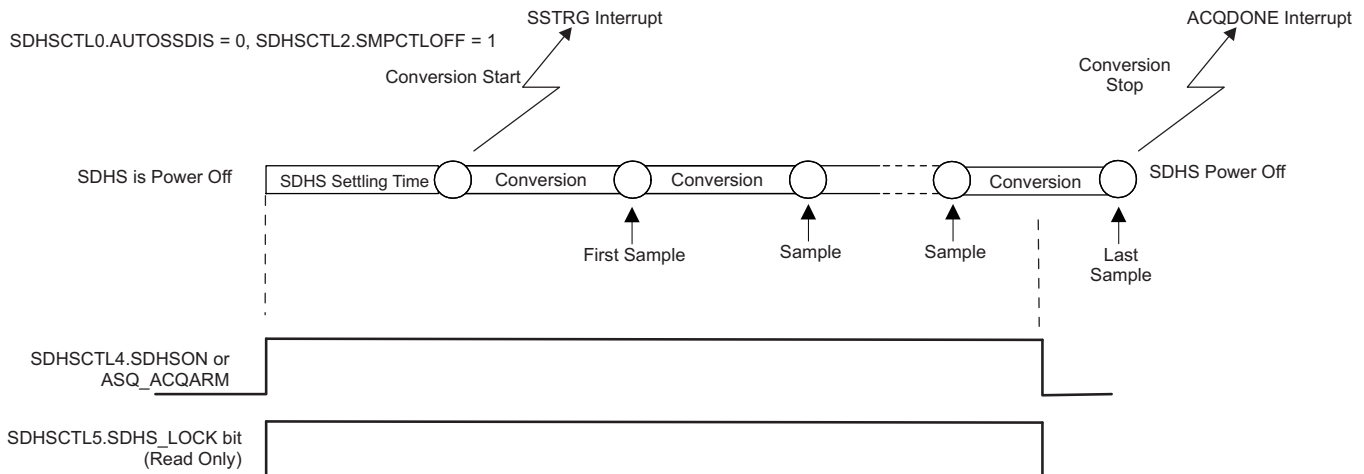


Figure 22-21. Conversion Start and Stop When SDHSCTL0.AUTOSSDIS = 0

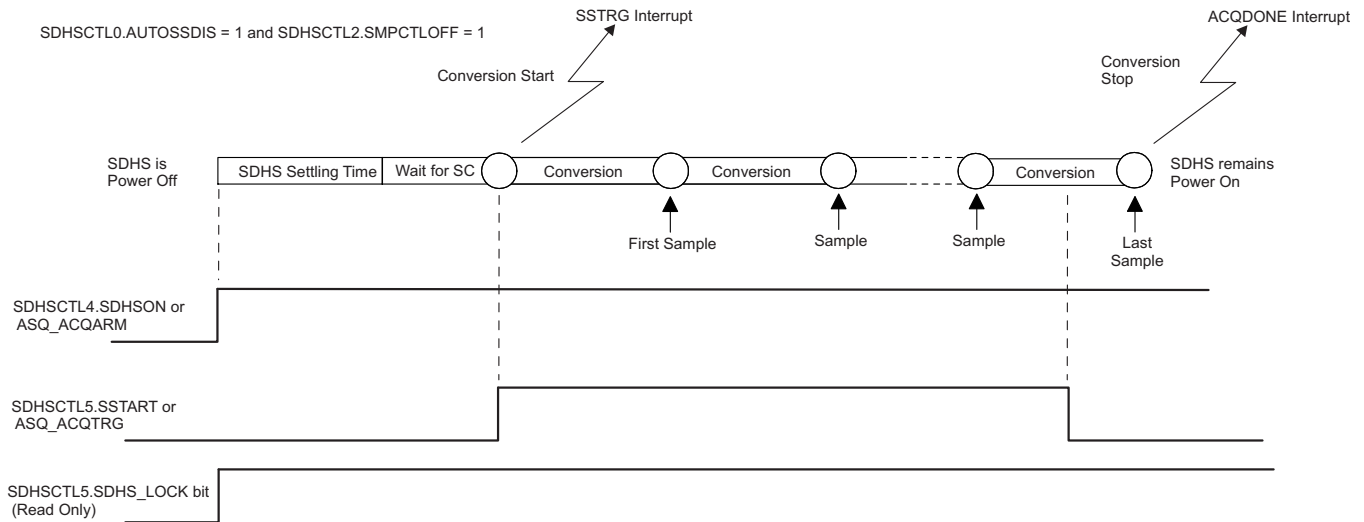


Figure 22-22. Conversion Start and Stop When SDHSCTL0.AUTOSSDIS = 1

22.2.9 INTDLY (Interrupt Delay) bits

After conversion start, the position of the first output data to the internal data buffer and the first `SDHSRIS.DTRDY` interrupt can be adjusted by the `SDHSCTL0.INTDLY` delay. Any skipped data is permanently lost. The delay is applied each time conversion starts. The `SDHSRIS.OVF` (overflow) interrupt is not enabled for the selected number of delay samples. Figure 22-23 shows the first interrupt position when `SDHSCTL0.INTDLY = 2`. The window comparator feature is not applied to the skipped samples (see Section 22.2.11 for the window comparator).

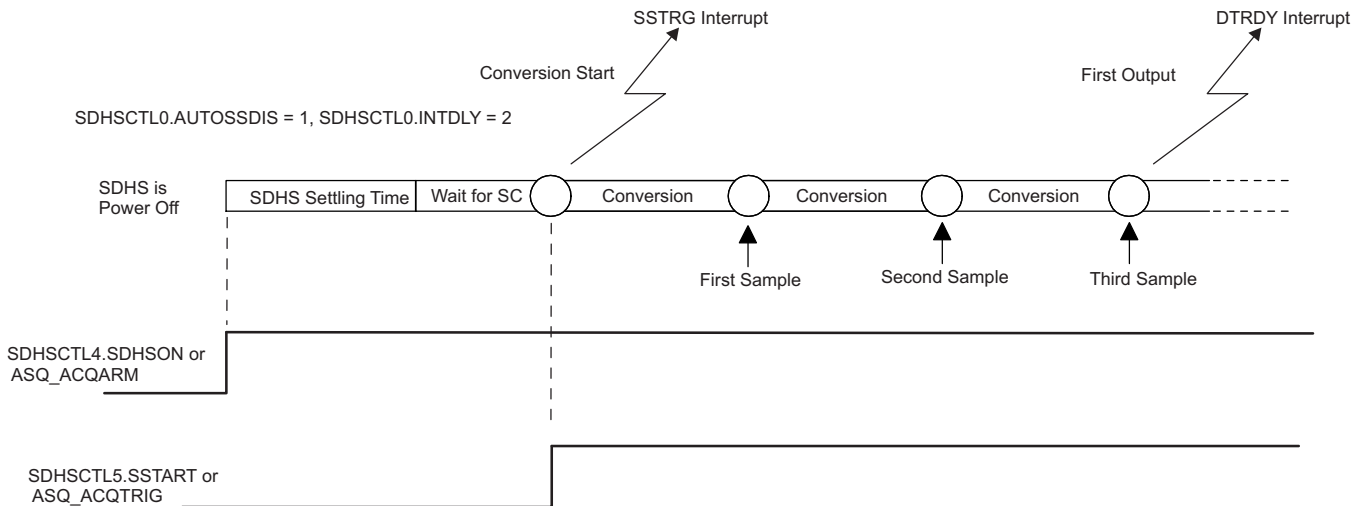


Figure 22-23. First Interrupt Position With SDHSCTL0.INTDLY = 2

By the nature of sigma-delta ADC converters, if a steep and abrupt input level change (like a step function) is applied, a few samples are needed before the full input level is reached at the output. The `INTDLY` can be used if the unsettled output data should be skipped. This skipping is not required for most applications. See Table 22-8 for the output data settling time.

Table 22-8. Conversion Control Mode

Input Change	SDHSCTL1.OSR Bit	Fully Settled Output Sample From the Time a Step Function is Applied	SDHSCTL0.INTDLY Y Value
Synchronous to f_s	10	5th sample	≤ 4
	20	3th sample	≤ 2
	40	2th sample	≤ 1
	80	1st sample	0
	160	1st sample	0
Asynchronous to f_s	10	6th sample	≤ 5
	20	4th sample	≤ 3
	40	3rd sample	≤ 2
	80	2nd sample	≤ 1
	160	2nd sample	≤ 1

22.2.10 Total Sample Size

The total number of samples that the SDHS generates can be predefined by SDHSCTL2.SMPSZ when SDHSCTL2.SMPCTLOFF = 0. The value written to SDHSCTL2.SMPSZ includes the samples skipped by SDHSCTL0.INTDLY:

- Total number of samples SDHS generates = SMPSZ + 1 (when SDHSCTL2.SMPCTLOFF = 0)
- The number of samples that can be read or transferred = SMPSZ – INTDLY + 1 (when SDHSCTL2.SMPCTLOFF = 0)

When SDHSCTL2.SMPCTLOFF = 0, the SDHS automatically stops data conversion after completing the number of samples configured in SDHSCTL2.SMPSZ. The SDHSCTL2.SMPSZ bit is ignored when SDHSCTL2.SMPCTLOFF = 1. In this case, the SDHS continues conversion until it is stopped by the trigger source. Take care when writing a value to SDHSCTL2.SMPSZ. If SDHSCTL2.SMPSZ – SDHSCTL0.INTDLY + 1 $\leq$ 0, no output data is generated. For example:

If SDHSCTL2.SMPSZ = 10 and SDHSCTL0.INTDLY = 2, then the number of available samples would be 10 – 2 + 1 = 9.

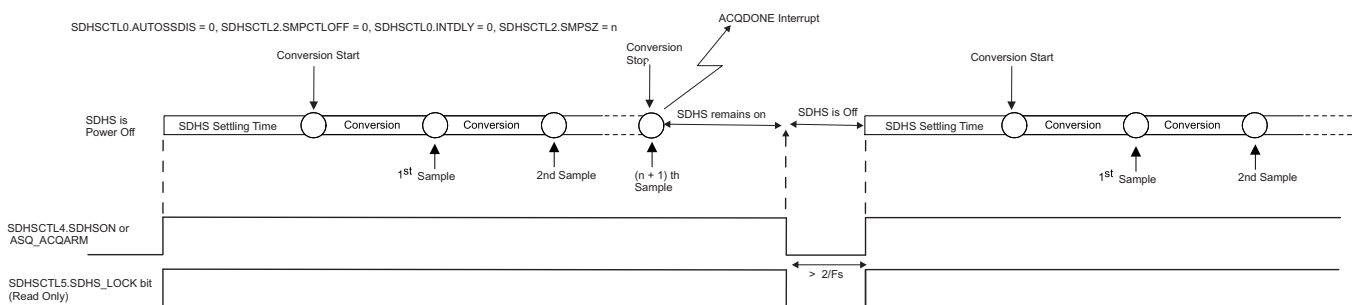


Figure 22-24. SDHSCTL0.AUTOSDIS = 0, SDHSCTL2.SMPCTLOFF = 0, SDHSCTL0.INTDLY = 0, Total Sample Size is Controlled by SDHSCTL2.SMPSZ

SDHSCTL0.AUTOSSDIS = 1, SDHSCTL2.SMPCTLOFF = 0, SDHSCTL0.INTDLY = 0, SDHSCTL2.SMPSZ = n:

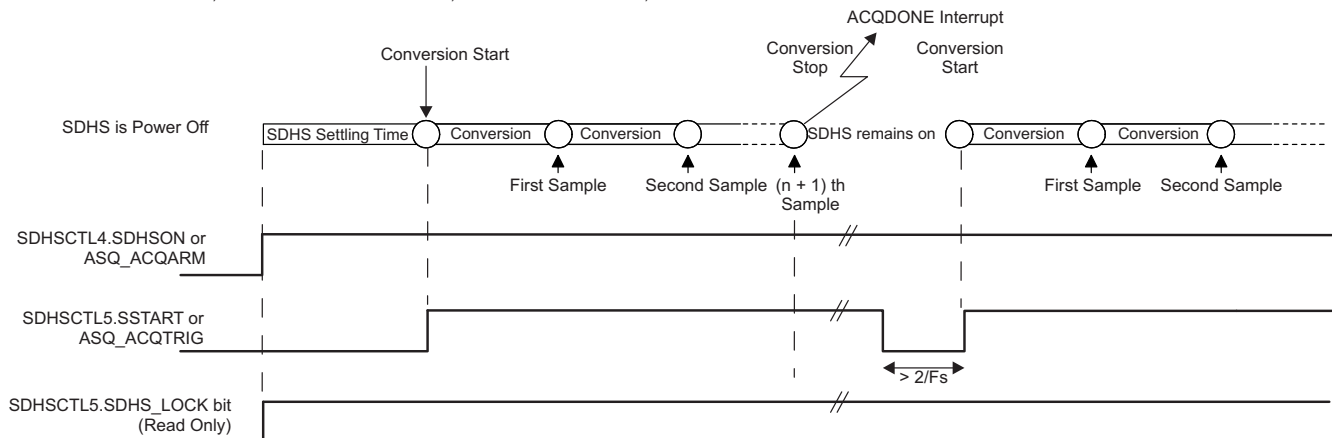


Figure 22-25. SDHSCTL0.AUTOSSDIS = 1, SDHSCTL2.SMPCTLOFF = 0, SDHSCTL0.INTDLY = 0, Total Sample Size is Controlled by SDHSCTL2.SMPSZ

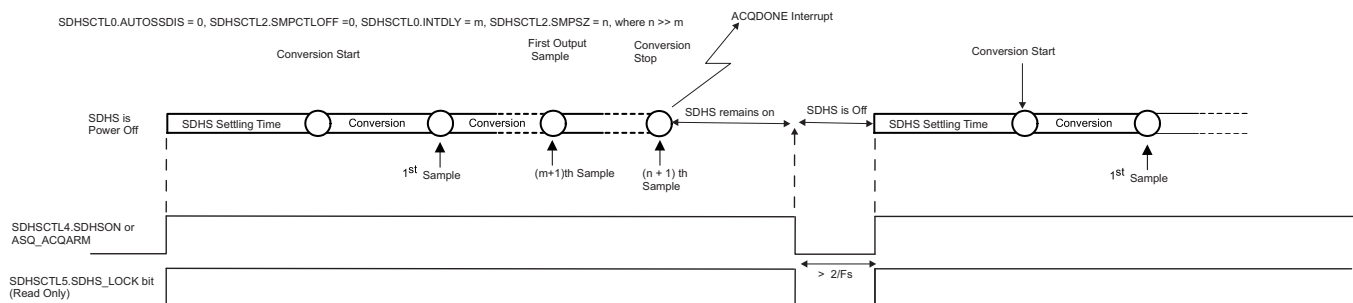


Figure 22-26. SDHSCTL0.AUTOSSDIS = 1, SDHSCTL2.SMPCTLOFF = 0, SDHSCTL0.INTDLY = m, Total Sample Size is Controlled by SDHSCTL2.SMPSZ

When data conversion has stopped or SDHS has been powered down, 2 sample periods must pass before starting a new data conversion or turning SDHS on (see [Figure 22-24](#) and [Figure 22-25](#)).

- SDHS output sampling period = $\text{SDHSCTL1.OSR} / \text{PLL output clock frequency}$ (= SDHS modulator sampling frequency).
- SDHS output sampling frequency = $\text{PLL output clock frequency}$ (= SDHS modulator sampling frequency) / SDHSCTL1.OSR .

22.2.11 Window Comparator

The window comparator can monitor data range without CPU interventions. The window comparator is enabled by the SDHSCTL2.WINDMPEN bit. The comparator compares the latest conversion result against the value in the SDHSWINHITH register and the SDHSWINLOTH register, then asserts SDHSRIS.WINHI (window high interrupt flag) if the conversion result is higher than the value in SDHSWINHITH register, or asserts SDHSRIS.WINLO (window low interrupt flag) if the conversion result is lower than the value in SDHSWINLOTH register. The window comparison is always performed with the latest output data before it goes to the internal buffer (see [Section 22.2.4](#) for details about the internal buffer). The window comparison is functional even when the internal buffer is full; however, the internal buffer stops taking new data, so the sample that caused either the SDHSRIS.WINHI or SDHSRIS.WINLO interrupt cannot be read by the SDHSDT register. The application must ensure that the values in the SDHSWINHITH and SDHSWINLOTH registers are in the correct data format. The interrupt flags (WINHI and WINLO) must be reset by user software.

22.2.12 Conditions to Stop Data Conversion

Table 22-9 lists the conditions that stop SDHS data conversion. The SDHSRIS.ACQDONE bit is set to 1 to indicate that data conversion has been stopped (either incomplete or complete). When the previous data conversion has been forced to stopped before completion, the SDHSRIS.ISTOP bit is set to 1 to indicate that data conversion has been interrupted (incomplete).

Table 22-9. SDHS Conversion Stop Conditions

Stop Condition	SDHSCTL0. TRGSRC	SDHSCTL0. AUTOSSDIS	SDHSCTL2. SMPCTLOFF	Result	
Number of samples = SDHSCTL2.SMP SZ + 1	Don't care	Don't care	0	SDHS power	No change
				Data conversion	Stop
				SDHSRIS.ISTOP	No change
				SDHS_ACQDONE (to ASQ)	Assert
				SDHSRIS.ACQDONE	Assert
	Don't care	Don't care	1	Not supported	
The ASQ stops the SDHS operation (ACQ_SDHSSTOP: 0 → 1) caused by: • UUPSCCTL.USSSTOP = 0 → 1 • UUPSCCTL.USSPWRDN = 0 → 1 • SAPHASCTL0.STOP = 0 → 1 • Enter debug mode while the SDHS is performing data conversion	Don't care	Don't care	Don't care	SDHS power	No change
				Data conversion	Stop
				SDHSRIS.ISTOP	Assert
				SDHS_ACQDONE (to ASQ)	Assert
				SDHSRIS.ACQDONE	Assert
SDHSCTL4.SDHSON: 1 → 0	0	Don't care	0	SDHS power	Power off
				Data conversion	Stop
				SDHSRIS.ISTOP	Assert
				SDHS_ACQDONE (to ASQ)	Assert
				SDHSRIS.ACQDONE	Assert
			1	SDHS power	Power off
				Data conversion	Stop
				SDHSRIS.ISTOP	No change
				SDHS_ACQDONE (to ASQ)	Assert
				SDHSRIS.ACQDONE	Assert
	1	Don't care	Don't care	Not supported	
ASQ_ACQARM: 1 → 0 (from ASQ)	0	Don't care	Don't care	Not supported	
	1	Don't care	0	SDHS power	Power off
				Data conversion	Stop
				SDHSRIS.ISTOP	Assert
				SDHS_ACQDONE (to ASQ)	Assert
				SDHSRIS.ACQDONE	Assert
				SDHS power	Power off
				Data conversion	Stop
				SDHSRIS.ISTOP	No change
				SDHS_ACQDONE (to ASQ)	Assert
				SDHSRIS.ACQDONE	Assert

Table 22-9. SDHS Conversion Stop Conditions (continued)

Stop Condition	SDHSCTL0. TRGSRC	SDHSCTL0. AUTOSSDIS	SDHSCTL2. SMPCTLOFF	Result	
SDHSCTL5.SSTART: 1 → 0	0	0	Don't care	Not supported	
		1	0	SDHS power	No change
				Data conversion	Stop
				SDHSRIS.ISTOP	Assert
				SDHS_ACQDONE (to ASQ)	Assert
				SDHSRIS.ACQDONE	Assert
			1	SDHS power	No change
				Data conversion	Stop
				SDHSRIS.ISTOP	No change
				SDHS_ACQDONE (to ASQ)	Assert
				SDHSRIS.ACQDONE	Assert
	1	Don't care	Don't care	Not supported	
ASQ_ACQTRG: 1 → 0 (from ASQ)	0	Don't care	Don't care	Not supported	
	1	0	Don't care	Not supported	
		1	0	SDHS power	No change
				Data conversion	Stop
				SDHSRIS.ISTOP	Assert
				SDHS_ACQDONE (to ASQ)	Assert
				SDHSRIS.ACQDONE	Assert
			1	SDHS power	No change
				Data conversion	Stop
				SDHSRIS.ISTOP	No change
				SDHS_ACQDONE (to ASQ)	Assert
				SDHSRIS.ACQDONE	Assert

The stop conditions listed in [Table 22-9](#) can occur when data conversion has already completed or has not started. [Table 22-10](#) summarizes how the SDHS responds to the signals when no data conversion is ongoing.

Table 22-10. SDHS Response to Conversion Stop Signals When Data Conversion is Not Running

Action	Conditions	Result	
SDHSCTL4.SDHSON: 1 → 0 (SDHSCTL0.TRGSRC = 0) or ASQ_ACQARM: 1 → 0 (SDHSCTL0.TRGSRC = 1)	SDHS is not performing data conversion	SDHS Power	Power off
		SDHSRIS.ISTOP bit	No change
		SDHS_ACQDONE (to ASQ)	No change
		RIS.ACQDONE	No change
The ASQ stops the SDHS operation (ACQ_SDHSSTOP: 0 → 1)	SDHS is not performing data conversion	SDHS power	No change
		SDHSRIS.ISTOP bit	No change
		SDHS_ACQDONE (to ASQ)	Assert
		SDHSRIS.ACQDONE	No change

22.3 Interrupts

The SDHS support the following interrupt sources:

- **SDHSRIS.OVF (data overflow interrupt):** When the internal buffer overflows, the SDHSRIS.OVF bit is asserted.
- **SDHSRIS.ACQDONE (acquisition done interrupt):** The SDHSRIS.ACQDONE is asserted when data conversion has been finished (either complete or incomplete).
If SDHSCTL2.DTOFF = 0, then SDHSRIS.ACQDONE is asserted when the data buffer is empty (that is, the DTC completes the data transfer).
If SDHSCTL2.DTCOFF = 1, then SDHSRIS.ACQDONE is asserted as the data conversion stops regardless of the data buffer status. In this case, user can continuously read SDHSDT register until the data buffer is empty. While reading the SDHSDT register, the data format configuration must not be changed (SDHSCTL0.DFMSEL, SDHSCTL0.DALGN, and SDHSCTL0.OBR).
- **SDHSRIS.SSTRG (start sampling trigger interrupt):** This bit indicates that the SDHS has started data conversion.
- **SDHSRIS.DTRDY (data ready interrupt):** This bit is asserted when a new data is available in the data buffer and remains asserted as long as the data buffer is not empty. The data read by CPU or DTC is removed from the data buffer. The SDHSRIS.DTRDY bit is deasserted when the data buffer becomes empty.
- **SDHSRIS.WINHI (window high interrupt):** This bit is asserted when a new output data is higher than the value in the SDHSWINHITH register.
- **SDHSRIS.WINHL (window low interrupt):** This bit is asserted when a new output data is lower than the value in the SDHSWINLOTH register.

In addition, the SDHSRIS.ISTOP (incomplete stop status) bit is asserted when the data conversion has been interrupted without completion. This bit is not an interrupt flag. It can be used as a status bit, not an interrupt flag.

22.3.1 IIDX, Interrupt Vector Generator

All SDHS interrupt sources are prioritized and combined to source a single interrupt vector. The SDHSIIDX register is used to determine which enabled SDHS interrupt sources have requested an interrupt. The SDHSIIDX register generates a value that can be used as address offset for fast interrupt service routine handling. On each read, only one interrupt is indicated. On a read, the current interrupt (highest priority) is automatically cleared by the hardware and the corresponding interrupt flag in SDHSRIS and SDHSMISC are also cleared. After a read from the CPU (not from the debug interface), the register must be updated with the next highest priority interrupt, if none are pending, then it reads as 0h. If the interrupt reported by the SDHSIIDX register (highest priority pending interrupt) is cleared in the SDHSICR by a software write of 1 in the corresponding bit field, the SDHSIIDX register is updated and the next priority interrupt (if any) is available.

22.4 Debug Mode

When the device is in debug mode, the SDHS cannot be enabled. Writes to the SDHSCTL4.SDHSON and SDHSCTL5.SSTART bits are prohibited. If the SDHS is already performing data conversion, the data conversion is stopped automatically and the SDHSRIS.ISTOP bit is asserted (see [Table 22-9](#)).

22.5 SDHS Registers

[Table 22-11](#) lists the memory-mapped registers for the SDHS. All register offset addresses not listed in [Table 22-11](#) should be considered as reserved locations and the register contents should not be modified.

Note 1: When SDHSCTL3.TRIGEN = 1, SDHSCTL0, SDHSCTL1, SDHSCTL2, SDHSCTL7, SDHSWINHITH, SDHSWINLOTH, and SDHSDTCDA registers are locked. In other words, an attempt to update those registers will be ignored.

Note 2: When SDHSCTL5.SDHS_LOCK = 1, SDHSCTL3 register is locked.

Note 3: SDHSCTL3.TRIGEN bit is a read-write bit, which is controlled by user program, whereas SDHSCTL5.SDHS_LOCK bit is a read-only bit, which is a status bit that indicates whether or not the SDHS is powered-up.

Note 4: SDHSCTL3.TRIGEN bit must be set to 1 before applying a power-up signal to SDHS (SDHSON bit or an external SDHS PWR UP signal)

Note 5:

When SDHSCTL0.TRGSRC = 0:

Once SDHSCTL4.SDHSON bit is written as 1, SDHSCTL5.SDHS_LOCK bit is set immediately.

In order to update SDHS registers, clear SDHSCTL4.SDHSON bit first, and then SDHSCTL3.TRIGEN bit needs to be cleared.

When SDHSCTL0.TRGSRC = 1:

It takes up to 4 system clock cycles to set SDHSCTL5.SDHS_LOCK bit after detecting an external SDHS PWR UP signal.

In order to update SDHS registers, the SDHS_PWR_UP signal should be de-asserted first, then SDHSCTL3.TRIGEN bit needs to be cleared to be zero.

Table 22-11. SDHS Registers

Offset	Acronym	Register Name	Type	Reset	Section
0h	SDHSIIDX	Interrupt Index Register	read-only	0h	Section 22.5.1
2h	SDHSMIS	Masked Interrupt Status and Clear Register	read-only	0h	Section 22.5.2
4h	SDHSRIS	Raw Interrupt Status Register	read-only	0h	Section 22.5.3
6h	SDHSIMSC	Interrupt Mask Register	read-write	0h	Section 22.5.4
8h	SDHSICR	Interrupt Clear Register.	write-only	0h	Section 22.5.5
Ah	SDHSISR	Interrupt Set Register.	write-only	0h	Section 22.5.6
Ch	SDHSDESCLO	SDHS Descriptor Register L.	read-only	110h	Section 22.5.7
Eh	SDHSDESCHI	SDHS Descriptor Register H.	read-only	BB10h	Section 22.5.8
10h	SDHSCTL0	SDHS Control Register 0	read-write	8001h	Section 22.5.9
12h	SDHSCTL1	SDHS Control Register 1	read-write	0h	Section 22.5.10
14h	SDHSCTL2	SDHS Control Register 2	read-write	0h	Section 22.5.11
16h	SDHSCTL3	SDHS Control Register 3	read-write	0h	Section 22.5.12
18h	SDHSCTL4	SDHS Control Register 4	read-write	0h	Section 22.5.13
1Ah	SDHSCTL5	SDHS Control Register 5	read-write	0h	Section 22.5.14
1Ch	SDHSCTL6	SDHS Control Register 6	read-write	19h	Section 22.5.15
1Eh	SDHSCTL7	SDHS Control Register 7	read-write	Fh	Section 22.5.16
22h	SDHSDT	SDHS Data Conversion Register	read-only	0h	Section 22.5.17
24h	SDHSWINHITH	SDHS Window Comparator High Threshold Register.	read-write	0h	Section 22.5.18
26h	SDHSWINLOTH	SDHS Window Comparator Low Threshold Register.	read-write	0h	Section 22.5.19
28h	SDHSDTCDA	DTC destination address register	read-write	0h	Section 22.5.20

22.5.1 SDHSIIDX Register (Offset = 0h) [reset = 0h]

SDHSIIDX is shown in [Figure 22-27](#) and described in [Table 22-12](#).

Return to [Summary Table](#).

Interrupt Index Register.

Note: This register is word accessible. A byte access is also allowed but not recommended. Either high byte or low byte access alone can clear the pending interrupt flag.

Figure 22-27. SDHSIIDX Register

15	14	13	12	11	10	9	8
IIDX							
R-0h							
7	6	5	4	3	2	1	0
IIDX							Reserved
R-0h							R-0h

Table 22-12. SDHSIIDX Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	IIDX	R	0h	SDHS Interrupt Vector Value. Read only. It generates a value that can be used as address offset for fast interrupt service routine handling. On each read, only one interrupt is indicated. On a read, the current interrupt (highest priority) is automatically de-asserted by the hardware and the corresponding bit in RIS and MISC are de-asserted as well. After a read from the CPU (not from the debug interface), the register is updated with the next highest priority interrupt, if none are pending, then it is read as zero. If the interrupt displayed by the SDHSIIDX register (highest priority pending interrupt) is cleared by writing '1' to a corresponding bit in the ICR register, the SDHSIIDX register shall be updated and the next priority interrupt (if any) is read. Reset type: PUC 0h (R) = No Interrupt pending. 1h (R) = Interrupt Source: SDHSRIS.OVF; Interrupt Priority: Highest 2h (R) = Interrupt Source: SDHSRIS.ACQDONE 3h (R) = Interrupt Source: SDHSRIS.SSTRG 4h (R) = Interrupt Source: SDHSRIS.DTRDY 5h (R) = Interrupt Source: SDHSRIS.WINHI 6h (R) = Interrupt Source: SDHSRIS.WINLO 7h (R) = Reserved; Interrupt 8h (R) = Reserved; Interrupt Priority: Lowest
0	Reserved	R	0h	Reserved. Always reads as 0.

22.5.2 SDHSMIS Register (Offset = 2h) [reset = 0h]

SDHSMIS is shown in [Figure 22-28](#) and described in [Table 22-13](#).

Return to [Summary Table](#).

Masked Interrupt Status Register.

Figure 22-28. SDHSMIS Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							
7	6	5	4	3	2	1	0
Reserved		WINLO	WINHI	DTRDY	SSTRG	ACQDONE	OVF
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

Table 22-13. SDHSMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved. Always reads as 0.
5	WINLO	R	0h	SDHS Window Low Masked Interrupt Status bit. Reset type: PUC 0h (R) = No interrupt pending 1h (R) = Interrupt pending
4	WINHI	R	0h	SDHS Window High Masked Interrupt Status bit. Reset type: PUC 0h (R) = No interrupt pending 1h (R) = Interrupt pending
3	DTRDY	R	0h	SDHS Data Ready Masked Interrupt Status bit. Reset type: PUC 0h (R) = No interrupt pending 1h (R) = Interrupt pending
2	SSTRG	R	0h	SDHS Conversion Start Trigger Masked Interrupt Status bit. Reset type: PUC 0h (R) = No interrupt pending 1h (R) = Interrupt pending
1	ACQDONE	R	0h	Acquisition Done Masked Interrupt Status bit. Reset type: PUC 0h (R) = No interrupt pending 1h (R) = Interrupt pending
0	OVF	R	0h	SDHS Data Overflow Masked Interrupt Status bit. Reset type: PUC 0h (R) = No interrupt pending 1h (R) = Interrupt pending

22.5.3 SDHSRIS Register (Offset = 4h) [reset = 0h]

SDHSRIS is shown in [Figure 22-29](#) and described in [Table 22-14](#).

Return to [Summary Table](#).

Raw Interrupt Status Register. Read Only Register.

Figure 22-29. SDHSRIS Register

15	14	13	12	11	10	9	8
ISTOP	Reserved						
R-0h	R-0h						
7	6	5	4	3	2	1	0
Reserved		WINLO	WINHI	DTRDY	SSTRG	ACQDONE	OVF
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

Table 22-14. SDHSRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
15	ISTOP	R	0h	Incomplete Stop Raw Interrupt Status bit. Read Only. This bit is asserted when data conversion has been interrupted and stopped before completing the number of samples defined in SDHSCTL2.SAMPSZ. This bit is offered only for polling the event by reading this bit. Interrupt is not available for this event. This bit must be de-asserted by writing '1' to SDHSICR.ISTOP bit. Reset type: PUC 0h (R) = No ISTOP event 1h (R) = Conversion has been interrupted and stopped before completing the number of samples defined in SDHSCTL2.SAMPSZ.
14-6	Reserved	R	0h	Reserved. Always reads as 0.
5	WINLO	R	0h	SDHS Window Low Raw Interrupt Status bit. Read Only. This bit is asserted when output data value is lower than the value in the SDHSWINLOTH register. Note: 1) The window comparator is only enabled when SDHSCTL2.WINCMPEN = 1. 2) Note: It takes 4 system clock cycles + 4 sampling periods to update SDHSRIS.WINLO bit after the condition is detected. Reset type: PUC 0h (R) = No new data is lower than the value in the SDHSWINLOTH register 1h (R) = New data is low than the value in the SDHSWINLOTH register
4	WINHI	R	0h	SDHS Window High Raw Interrupt Status bit. Read Only. This bit is asserted when the output data value is higher than the value in the SDHSWINHITH register. Note: 1) The window comparator is only enabled when SDHSCTL2.WINCMPEN = 1. 2) It takes 4 system clock cycles + 4 sampling periods to update SDHSRIS.WINHI after the condition is detected. Reset type: PUC 0h (R) = No WINHI event 1h (R) = The output data value is higher than the value in the SDHSWINHITH register

Table 22-14. SDHSRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	DTRDY	R	0h	SDHS Data Ready Raw Interrupt Status bit. Read Only. This bit is asserted when a new conversion data is available in the data buffer and remains set as long as the buffer is not empty regardless of the SDHS data conversion status. Note: the data buffer is automatically cleared when the data buffer becomes empty. Following two methods can be used to empty the data buffer after completing data conversion if necessary: 1) When the DTC is enabled, no additional action is required. The DTC reads the data buffer until the data buffer becomes empty. 2) When the DTC is disabled, then either read the SDHSDT register until this bit is cleared or enable the DTC so that the DTC empties the buffer. Either way, this bit will be cleared when the buffer becomes empty. Reset type: PUC 0h (R) = No DTRDY event 1h (R) = The data buffer has become empty.
2	SSTRG	R	0h	SDHS Conversion Start Trigger Raw Interrupt Status bit. Read Only. Reset type: PUC 0h (R) = No SSTRG event 1h (R) = Conversion Start signal has been asserted
1	ACQDONE	R	0h	Acquisition Done Raw Interrupt Status bit. Read Only. This bit is not de-asserted by hardware. This bit is asserted when data conversion is ended (either complete or incomplete). If SDHSCTL2.DTOFF = 0, then this bit is asserted when data buffer becomes empty (i.e. when DTC completes the data transfer). If SDHSCTL2.DTCOFF = 1, then this bit is asserted immediately when data conversion stops regardless of the data buffer status. In this case, CPU can continuously read the SDHSDT register until the data buffer becomes empty. Reset type: PUC 0h (R) = No ACQDONE event 1h (R) = Data conversion has been finished (either complete or incomplete).
0	OVF	R	0h	SDHS Data Overflow Raw Interrupt Status bit. Read Only. This bit is not automatically de-asserted by hardware. Reset type: PUC 0h (R) = No OVF event 1h (R) = When DTC is enabled (SDHSCTL2.DTCOFF = 0), DTC has dropped at least one sample. This indicates that the system clock needs to be increased. When DTC is disabled (SDHSCTL2.DTCOFF = 1), At least one new sample has been overwritten to SDHSDT register before the previous value is read.

22.5.4 SDHSIMSC Register (Offset = 6h) [reset = 0h]

SDHSIMSC is shown in [Figure 22-30](#) and described in [Table 22-15](#).

Return to [Summary Table](#).

Interrupt Mask Register.

Figure 22-30. SDHSIMSC Register

15	14	13	12	11	10	9	8
ISTOP	Reserved						
R-0h	R-0h						
7	6	5	4	3	2	1	0
Reserved	WINLO	WINHI	DTRDY	SSTRG	ACQDONE	OVF	
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 22-15. SDHSIMSC Register Field Descriptions

Bit	Field	Type	Reset	Description
15	ISTOP	R	0h	Incomplete Stop Interrupt Mask bit. Read Only. Note that this interrupt is always disabled. No interrupt will be generated.
14-6	Reserved	R	0h	Reserved. Always reads as 0.
5	WINLO	R/W	0h	SDHS Window Low Interrupt Mask bit. Reset type: PUC 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
4	WINHI	R/W	0h	SDHS Window High Interrupt Mask bit. Reset type: PUC 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
3	DTRDY	R/W	0h	SDHS Data Ready Interrupt Mask bit. Reset type: PUC 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
2	SSTRG	R/W	0h	SDHS Start Conversion Trigger Interrupt Mask bit. Reset type: PUC 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
1	ACQDONE	R/W	0h	Acquisition Done Interrupt Mask bit. Reset type: PUC 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
0	OVF	R/W	0h	SDHS Data Overflow Interrupt Mask bit. Reset type: PUC 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled

22.5.5 SDHSICR Register (Offset = 8h) [reset = 0h]

SDHSICR is shown in [Figure 22-31](#) and described in [Table 22-16](#).

Return to [Summary Table](#).

Interrupt Clear Register. Writing '1' to clear the corresponding bit in SDHSRIS register. Read as zero.

Note: This register can be used to clear an interrupt source without reading the SDHSIIDX register.

Figure 22-31. SDHSICR Register

15	14	13	12	11	10	9	8
ISTOP	Reserved						
W-0h	R-0h						
7	6	5	4	3	2	1	0
Reserved	WINLO	WINHI	DTRDY	SSTRG	ACQDONE	OVF	
R-0h	W-0h	W-0h	W1S-0h	W-0h	W-0h	W-0h	W-0h

Table 22-16. SDHSICR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	ISTOP	W	0h	Incomplete Stop Interrupt Clear bit.
14-6	Reserved	R	0h	Reserved. Always reads as 0.
5	WINLO	W	0h	SDHS Window Low Interrupt Clear bit.
4	WINHI	W	0h	SDHS Window High Interrupt Clear bit. Reset type: PUC
3	DTRDY	W1S	0h	SDHS Data Ready Interrupt Clear bit. Note: SDHSRIS.DTRDY is automatically cleared by hardware when the data buffer is empty. This bit can be used to de-assert SDHSRIS.DTRDY only when SDHSRIS.DTRDY is asserted by writing '1' to SDHSISR.DTRDY and the data buffer is empty. Reset type: PUC
2	SSTRG	W	0h	SDHS Conversion Start Trigger Interrupt Clear bit.
1	ACQDONE	W	0h	Acquisition Done Interrupt Clear bit.
0	OVF	W	0h	SDHS Data Overflow Interrupt Clear bit.

22.5.6 SDHSISR Register (Offset = Ah) [reset = 0h]

SDHSISR is shown in [Figure 22-32](#) and described in [Table 22-17](#).

Return to [Summary Table](#).

Interrupt Set Register. Writing '1' to assert the corresponding bit in SDHSRIS register. Read as zero.

Note: This register can be used for debugging purpose to generate an interrupt manually.

Figure 22-32. SDHSISR Register

15	14	13	12	11	10	9	8
ISTOP	Reserved						
W-0h	R-0h						
7	6	5	4	3	2	1	0
Reserved	WINLO		WINHI	DTRDY	SSTRG	ACQDONE	OVF
R-0h	W-0h		W-0h	W1S-0h	W-0h	W-0h	W-0h

Table 22-17. SDHSISR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	ISTOP	W	0h	Incomplete Stop Interrupt Set bit.
14-6	Reserved	R	0h	Reserved. Always reads as 0.
5	WINLO	W	0h	SDHS Window Low Interrupt Set bit.
4	WINHI	W	0h	SDHS Window High Interrupt Set bit. Reset type: PUC
3	DTRDY	W1S	0h	SDHS Data Ready Interrupt Set bit. Write 1 to set SDHSRIS.DTRDY bit. Note: This bit can be used to test the interrupt when the data buffer is empty. In the case, SDHSRIS.DTRDY bit does not indicate actual status of the data buffer. Once SDHSRIS.DTRDY is asserted by SDHSISR.DTRDY, then it can be de-asserted by SDHSICR.DTRDY bit as long as the data buffer is empty. Reset type: PUC
2	SSTRG	W	0h	SDHS Start Conversion Trigger Interrupt Set bit.
1	ACQDONE	W	0h	Acquisition Done Interrupt Set bit.
0	OVF	W	0h	SDHS Data Overflow Interrupt Set bit.

22.5.7 SDHSDDESCLO Register (Offset = Ch) [reset = 110h]

SDHSDDESCLO is shown in [Figure 22-33](#) and described in [Table 22-18](#).

Return to [Summary Table](#).

SDHS Descriptor Register L.

Figure 22-33. SDHSDDESCLO Register

15	14	13	12	11	10	9	8
FEATUREVER				INSTNUM			
R-0h				R-1h			
7	6	5	4	3	2	1	0
MAJREV				MINREV			
R-1h				R-0h			

Table 22-18. SDHSDDESCLO Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	FEATUREVER	R	0h	Feature Set for the module Reset type: PUC
11-8	INSTNUM	R	1h	Instance Number within the device. This will be a parameter to the RTL for modules that can have multiple instances
7-4	MAJREV	R	1h	Major Revision Reset type: PUC
3-0	MINREV	R	0h	Minor Revision Reset type: PUC

22.5.8 SDHSDESCHI Register (Offset = Eh) [reset = BB10h]

SDHSDESCHI is shown in [Figure 22-34](#) and described in [Table 22-19](#).

Return to [Summary Table](#).

SDHS Descriptor Register H.

Figure 22-34. SDHSDESCHI Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODULEID															
R-BB10h															

Table 22-19. SDHSDESCHI Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MODULEID	R	BB10h	Module Identifier. Reset type: PUC

22.5.9 SDHSCTL0 Register (Offset = 10h) [reset = 8001h]

SDHSCTL0 is shown in [Figure 22-35](#) and described in [Table 22-20](#).

Return to [Summary Table](#).

SDHS Control Register 0.

When SDHSCTL3.TRGGEN bit = 1 or SDHSCTL5.SDHS_LOCK bit = 1, this register is locked. In that case, an attempt to update this registers will be ignored.

Figure 22-35. SDHSCTL0 Register

15	14	13	12	11	10	9	8
TRGSRC	Reserved	SHIFT		OBR		DFMSEL	
R/W-1h	R-0h	R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
DALGN	Reserved				INTDLY		AUTOSSDIS
R/W-0h		R-0h			R/W-0h		R/W-1h

Table 22-20. SDHSCTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	TRGSRC	R/W	1h	SDHS trigger source select. Reset type: PUC 0h (R/W) = Register control mode: - SDHSCTL4.SDHSOEN is the source of the SHDS_PWR_UP/DOWN signal - SDHSCTL5.SSTART is the source of the CONVERSION_START/STOP signal 1h (R/W) = ASQ control mode: The SDHS is controlled by the ASQ. - ASQ_ACQARM signal from the ASQ is the source of the SHDS_PWR_UP/DOWN signal - ASQ_ACQTRIG signal from the ASQ is the source of the CONVERSION_START/STOP signal
14	Reserved	R	0h	Reserved. Always reads as 0.
13-12	SHIFT	R/W	0h	MSB Shift. Reset type: PUC 0h (R/W) = No Shift, MSB. 1h (R/W) = MSB - 1 (Shift left by 1 from filter out). If SDHSCTL0.OBR = 2, then this configuration is invalid. No shift is performed. 2h (R/W) = MSB - 2 (Shift left by 2 from filter out). If SDHSCTL0.OBR = 1, then this configuration is invalid. No shift is performed. 3h (R/W) = Reserved (No shift)
11-10	OBR	R/W	0h	Output Bit Resolution. Reset type: PUC 0h (R/W) = 12-bit 1h (R/W) = 13-bit 2h (R/W) = 14-bit 3h (R/W) = Reserved (default: 12-bit)
9-8	DFMSEL	R/W	0h	Data format. Reset type: PUC 0h (R/W) = 2's complement 1h (R/W) = Offset binary 2h (R/W) = Reserved (defaults to 0, 2s complement) 3h (R/W) = Reserved (defaults to 0, 2s complement)

Table 22-20. SDHSCTL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	DALGN	R/W	0h	Data alignment. Reset type: PUC 0h (R/W) = Right-aligned. 1h (R/W) = Left-aligned.
6-4	Reserved	R	0h	Reserved. Always reads as 0.
3-1	INTDLY	R/W	0h	DTRDY Interrupt delay select. This register can be used to discard up to 7 samples after conversion start. Note that the skipped samples will be lost. Reset type: PUC 0h (R/W) = No delay 1h (R/W) = 1 sample delay, 2nd sample is the first interrupt 2h (R/W) = 2 samples delay, 3rd sample is the first interrupt 3h (R/W) = 3 samples delay, 4th sample is the first interrupt 4h (R/W) = 4 samples delay, 5th sample is the first interrupt 5h (R/W) = 5 samples delay, 6th sample is the first interrupt 6h (R/W) = 6 samples delay, 7th sample is the first interrupt 7h (R/W) = 7 samples delay, 8th sample is the first interrupt
0	AUTOSSDIS	R/W	1h	SDHS Auto Sample Start Disable. Reset type: PUC 0h (R/W) = Auto Sample start enabled. SDHS is powered up when the SHDS_PWR_UP applied, then data conversion is automatically started once the SDHS is fully powered up. 1h (R/W) = Auto Sample start disabled. (This configuration must be used when the ASQ controls the measurement sequences) - SHDS_PWR_UP signal to turn on the SDHS - CONVERSION_START signal to start data conversion

22.5.10 SDHSCTL1 Register (Offset = 12h) [reset = 0h]

SDHSCTL1 is shown in [Figure 22-36](#) and described in [Table 22-21](#).

Return to [Summary Table](#).

SDHS Control Register 1

When SDHSCTL3.TRGEN = 1 or SDHSCTL5.SDHS_LOCK = 1, this register is locked. In that case, an attempt to update this registers will be ignored.

Figure 22-36. SDHSCTL1 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							
7	6	5	4	3	2	1	0
Reserved				OSR			
R-0h				R/W-0h			

Table 22-21. SDHSCTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	Reserved	R	0h	Reserved. Always reads as 0.
3-0	OSR	R/W	0h	Over Sampling Rate. Output Data Rate = Input Clock Frequency / OSR. Note: values not shown below are reserved. Reset type: PUC 0h (R/W) = 10 1h (R/W) = 20 2h (R/W) = 40 3h (R/W) = 80 4h (R/W) = 160

22.5.11 SDHSCTL2 Register (Offset = 14h) [reset = 0h]

SDHSCTL2 is shown in [Figure 22-37](#) and described in [Table 22-22](#).

Return to [Summary Table](#).

SDHS Control Register 2

When SDHSCTL3.TRGGEN = 1 or SDHSCTL5.SDHS_LOCK = 1, this register is locked. In that case, an attempt to update this registers will be ignored.

Figure 22-37. SDHSCTL2 Register

15	14	13	12	11	10	9	8
DTCOFF	WINCMPEN	Reserved			SMPCTLOFF	SMPSZ	
R/W-0h	R/W-0h	R-0h			R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
SMPSZ							
R/W-0h							

Table 22-22. SDHSCTL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	DTCOFF	R/W	0h	Data Transfer Controller (DTC) Off. Reset type: PUC 0h (R/W) = DTC enabled. The DTC automatically transfers the data from the SDHSDT register to the address specified in the SDHSDTCDA register. 1h (R/W) = DTC disabled. The data in the SDHSDT register must be read by CPU, otherwise the overflow interrupt flag (SDHSRIS.OVF) will eventually be asserted.
14	WINCMPEN	R/W	0h	Window Comparator Enable. Note: - For the samples skipped by SDHSCTL0.INTDLY, window comparison is not applied. - Window comparison is performed with the latest conversion result before it is pushed to the internal buffer. - Window comparison is still functional when the internal buffer is full. Reset type: PUC 0h (R/W) = Window Comparator is disabled 1h (R/W) = Window Comparator is enabled
13-11	Reserved	R	0h	Reserved. Always reads as 0.
10	SMPCTLOFF	R/W	0h	Disable sampling size counting. Reset type: PUC 0h (R/W) = Total sampling size is determined by SDHSCTL2.SMPSZ bits. The SDHS automatically stops data conversion. 1h (R/W) = SDHSCTL2.SMPSZ bits are ignored. Conversion does not stop until the trigger source selected by TRGSRC bits is deasserted.
9-0	SMPSZ	R/W	0h	Total Sample Size. Total Sample Size = SMPSZ + 1. Note that SDHSCTL2.SMPSZ includes the samples skipped by SDHSCTL0.INTDLY: - The total number of samples SDHS generates = SMPSZ + 1. - The number of samples SDHS generates via SDHSDT register = SMPSZ - INTDLY + 1. Care must be taken when writing a value to SDHSCTL2.SMPSZ. If SDHSCTL2.SMPSZ - SDHSCTL0.INTDLY + 1 <= 0, then no data output to SDHSDT register. Reset type: PUC

22.5.12 SDHSCTL3 Register (Offset = 16h) [reset = 0h]

SDHSCTL3 is shown in [Figure 22-38](#) and described in [Table 22-23](#).

Return to [Summary Table](#).

SDHS Control Register 3

Figure 22-38. SDHSCTL3 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							TRIGEN
R-0h							R/W-0h

Table 22-23. SDHSCTL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved. Always reads as 0.
0	TRIGEN	R/W	0h	SDHS Trigger Enable bit. This bit enables SDHS Trigger Source and lock SDHS registers. Note: This bit is used to inform SDHS that register configuration has been completed. This bit must be written as 1 before applying SDHS_PWR_UP signal. Once this bit is asserted, SDHSCTL0, SDHSCTL1, SDHSCTL2, SDHSCTL7, SDHSWINHITH, SDHSWINLOTH, and SDHSDTCDA registers are locked (not allowed to be modified). This bit is locked once a SDHS_PWR_UP signal is applied. See SDHSCTL5.SDHS_LOCK for the sequence of SDHS register configuration. Reset type: PUC 0h (R/W) = SDHS Trigger is disabled. Once this bit is de-asserted, SDHSCTL0, SDHSCTL1, SDHSCTL2, SDHSCTL7, SDHSWINHITH, SDHSWINLOTH, and SDHSDTCDA registers are unlocked (allowed to be modified). 1h (R/W) = SDHS Trigger is enabled. Once this bit is asserted, SDHSCTL0, SDHSCTL1, SDHSCTL2, SDHSCTL7, SDHSWINHITH, SDHSWINLOTH, and SDHSDTCDA registers are locked (not allowed to be modified).

22.5.13 SDHSCTL4 Register (Offset = 18h) [reset = 0h]

SDHSCTL4 is shown in [Figure 22-39](#) and described in [Table 22-24](#).

Return to [Summary Table](#).

SDHS Control Register 4

Figure 22-39. SDHSCTL4 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							SDHSON
R-0h							R/W-0h

Table 22-24. SDHSCTL4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved. Always reads as 0.
0	SDHSON	R/W	0h	Turn on the SDHS module. When SDHSCTL0.TRGSRC =0, SDHS Power-up/down bit. Note: - When SDHSCTL0.AUTOSSDIS = 0 and SDHSCTL0.TRGSRC =0, the SDHSON bit only powers up the SDHS, does not start sampling. - When SDHSCTL0.AUTOSSDIS = 1 and SDHSCTL0.TRGSRC =0, the SDHSON powers up the SDHS and starts sampling when the SDHS is fully powered up. - When SDHSCTL0.TRGSRC = 1, this bit is invalid. Reset type: PUC 0h (R/W) = Power down the SDHS module 1h (R/W) = Power on the SDHS module

22.5.14 SDHSCTL5 Register (Offset = 1Ah) [reset = 0h]

SDHSCTL5 is shown in [Figure 22-40](#) and described in [Table 22-25](#).

Return to [Summary Table](#).

SDHS Control Register 5

Figure 22-40. SDHSCTL5 Register

15	14	13	12	11	10	9	8
Reserved							SDHS_LOCK
R-0h							R-0h
7	6	5	4	3	2	1	0
Reserved							SSTART
R-0h							R/W-0h

Table 22-25. SDHSCTL5 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0h	Reserved. Always reads as 0.
8	SDHS_LOCK	R	0h	SDHS register lock status bit. This bit is asserted when the SDHS has received the SDHS_PWR_UP signal. When this bit is asserted, SDHSCTL3 register is locked. Note: 1)When SDHSCTL0.TRGSRC = 0: Once SDHSCTL4.SDHSON is written as 1, SDHSCTL5.SDHS_LOCK is asserted immediately. In order to update SDHS registers, clear SDHSCTL4.SDHSON first, and then SDHSCTL3.TRIGEN needs to be cleared. 2) When SDHSCTL0.TRGSRC = 1: It takes up to 4 system clock cycles to assert SDHSCTL5.SDHS_LOCK after detecting the SDHS_PWR_UP signal (ASQ_ACQARM from the ASQ). In order to update SDHS registers, the SDHS_PWR_UP signal should be de-asserted first by the ASQ, then SDHSCTL3.TRIGEN needs to be cleared. 3) SDHS register configuration sequence: a) After reset (or device power up), configure SDHSCTL0, SDHSCTL1, SDHSCTL2, SDHSCTL7, SDHSWINHITH, SDHSWINLOTH, and SDHSDTCDA registers b) write '1' to SDHSCTL3.TRIGEN bit => SDHSCTL0, SDHSCTL1, SDHSCTL2, SDHSCTL7, SDHSWINHITH, SDHSWINLOTH, and SDHSDTCDA registers are locked c) Apply a SDHS_PWR_UP signal => SDHSCTL3 register is locked d) Apply a CONVERSION_START signal if necessary Reset type: PUC 0h (R) = SDHSCTL3 register is unlocked. 1h (R) = SDHSCTL3 register is locked as well as SDHSCTL0, SDHSCTL1, SDHSCTL2, SDHSCTL7, SDHSWINHITH, SDHSWINLOTH, and SDHSDTCDA registers. Only read is allowed.
7-1	Reserved	R	0h	Reserved. Always reads as 0.

Table 22-25. SDHSCTL5 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	SSTART	R/W	0h	Start data conversion. Note: - When SDHSCTL0.AUTOSDIS = 0 and SDHSCTL0.TRGSRC = 0, the SDHSON powers up the SDHS, the SSTART triggers data conversion. It is very important to wait for the SDHS settling time (34 usec) before asserting SSTART from the time of SDHSON = 0 -> 1. - When SDHSCTL0.AUTOSDIS = 1 and SDHSCTL0.TRGSRC = 0, this bit is invalid. - When SDHSCTL0.TRGSRC = 1, this bit is invalid. Reset type: PUC 0h (R/W) = Stop conversion 1h (R/W) = Start conversion

22.5.15 SDHSCTL6 Register (Offset = 1Ch) [reset = 19h]

SDHSCTL6 is shown in [Figure 22-41](#) and described in [Table 22-26](#).

Return to [Summary Table](#).

SDHS Control Register 6

Figure 22-41. SDHSCTL6 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							
7	6	5	4	3	2	1	0
Reserved		PGA_GAIN					
R-0h		R/W-19h					

Table 22-26. SDHSCTL6 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved. Always reads as 0.
5-0	PGA_GAIN	R/W	19h	PGA Gain Control bits. These bits control the Gain range of the analog amplifier. See PGA Gain Table for details. Reset type: PUC

22.5.16 SDHSCTL7 Register (Offset = 1Eh) [reset = Fh]

SDHSCTL7 is shown in [Figure 22-42](#) and described in [Table 22-27](#).

Return to [Summary Table](#).

SDHS Control Register 7

Figure 22-42. SDHSCTL7 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				MODOPTI			
R-0h				R/W-Fh			

Table 22-27. SDHSCTL7 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved. Always reads as 0.
4-0	MODOPTI	R/W	Fh	SDHS Modulator Optimization bits. In order to get the maximum performance of SDHS, it is recommended to configure this bits based on the PLL output frequency. See below for details: PLL output frequency (=Fmod) : MODOPTI bits 77MHz <= Fmod <= 80MHz: 0xC 74MHz <= Fmod < 77MHz: 0xD 71MHz <= Fmod < 74MHz: 0xE 68MHz <= Fmod < 71MHz: 0xF (Reset) Where Fmod = PLL output frequency Reset type: PUC

22.5.17 SDHSDT Register (Offset = 22h) [reset = 0h]

SDHSDT is shown in [Figure 22-43](#) and described in [Table 22-28](#).

Return to [Summary Table](#).

SDHS Data Conversion Register

Figure 22-43. SDHSDT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDHSDT															
R-0h															

Table 22-28. SDHSDT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SDHSDT	R	0h	Conversion Data. Reset type: PUC

22.5.18 SDHSWINHITH Register (Offset = 24h) [reset = 0h]

SDHSWINHITH is shown in [Figure 22-44](#) and described in [Table 22-29](#).

Return to [Summary Table](#).

SDHS Window Comparator High Threshold Register.

When SDHSCTL3.TRGEN = 1 or SDHSCTL5.SDHS_LOCK = 1, this register is locked. In that case, an attempt to update this registers will be ignored.

Figure 22-44. SDHSWINHITH Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WINHITH															
R/W-0h															

Table 22-29. SDHSWINHITH Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	WINHITH	R/W	0h	SDHS Window Comparator High Threshold Register. The user always needs to ensure that the values in this register is in the correct data format. When the conversion data is higher than the value specified in this register, WINHI interrupt flag is asserted. It only works during SDHS conversion is on-going. Note: Once the condition is detected, it takes 4 system clock cycles + 4 sampling periods to update SDHSRIS.WINHITH due to synchronization requirement. Reset type: PUC

22.5.19 SDHSWINLOTH Register (Offset = 26h) [reset = 0h]

SDHSWINLOTH is shown in [Figure 22-45](#) and described in [Table 22-30](#).

Return to [Summary Table](#).

SDHS Window Comparator Low Threshold Register.

When SDHSCTL3.TRGEN bit = 1 or SDHSCTL5.SDHS_LOCK bit = 1, this register is locked. In that case, an attempt to update this registers will be ignored.

Figure 22-45. SDHSWINLOTH Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WINLOTH															
R/W-0h															

Table 22-30. SDHSWINLOTH Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	WINLOTH	R/W	0h	SDHS Window Comparator Low Threshold Register. The user always needs to ensure that the values in this register is in the correct data format. When the conversion data is lower than the value specified in this register, WINLO interrupt flag is asserted. It only works during SDHS conversion is on-going. Note: Note: Once the condition is detected, it takes 4 system clock cycles + 4 sampling periods to update RIS.WINLO due to synchronization requirement. Reset type: PUC

22.5.20 SDHSDTCDA Register (Offset = 28h) [reset = 0h]

SDHSDTCDA is shown in [Figure 22-46](#) and described in [Table 22-31](#).

Return to [Summary Table](#).

DTC destination offset address. As DTC transfers data, this register value increases by 1 at every data transfer.

Note: When SDHSCTL3.TRGEN bit = 1 or SDHSCTL5.SDHS_LOCK bit = 1, this register is locked. In that case, an attempt to update this registers will be ignored.

Figure 22-46. SDHSDTCDA Register

15	14	13	12	11	10	9	8
Reserved	DTCDA						
R-0h	R/W-0h						
7	6	5	4	3	2	1	0
DTCDA							
R/W-0h							

Table 22-31. SDHSDTCDA Register Field Descriptions

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved. Always reads as 0.
14-0	DTCDA	R/W	0h	DTC destination offset address. Destination location = base address + DTCDA x 2. The address offset for DTC destination memory. The size of the destination memory is up to 64KB. The address register value increases by 1 at every data transfer. Note: Care must be taken not to go beyond the available memory address range (specified by the device datasheet). The DTC is able to accesses to the LEA RAM only.

Metering Test Interface (MTIF)

This chapter describes the MSP430 Metering Test Interface (MTIF) module.

Topic	Page
23.1 MTIF Introduction	620
23.2 MTIF Operation.....	621
23.3 MTIF Block Diagram.....	624
23.4 MTIF Registers	625

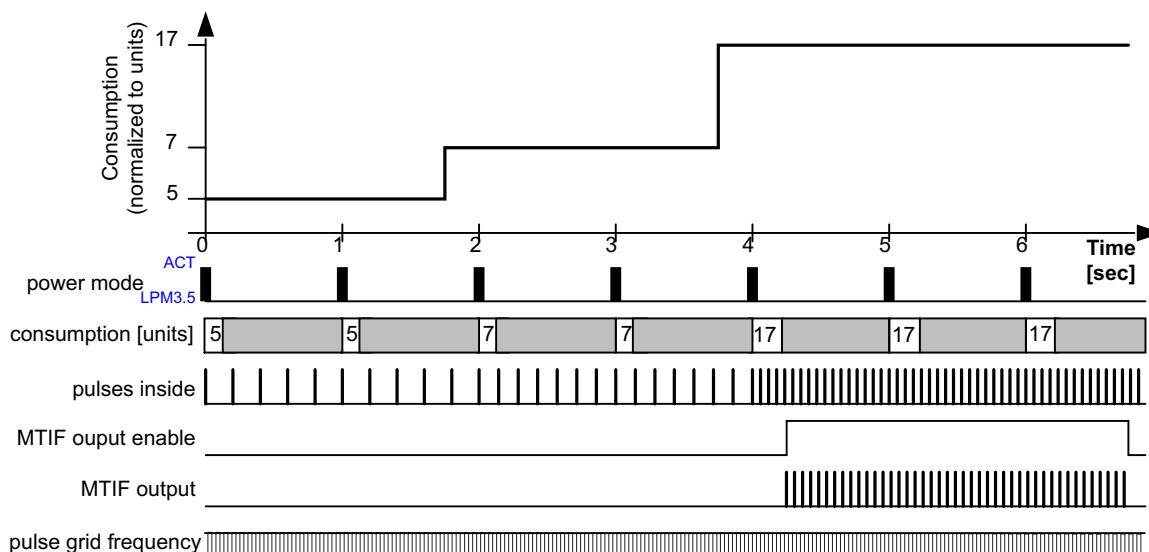


Figure 23-2. MTIF Pulse Diagram

23.2 MTIF Operation

The MTIF module can be configured in active mode like any other MSP430 module. The inner core of MTIF operates as the RTC from a power domain that is still active in LPM3.5.

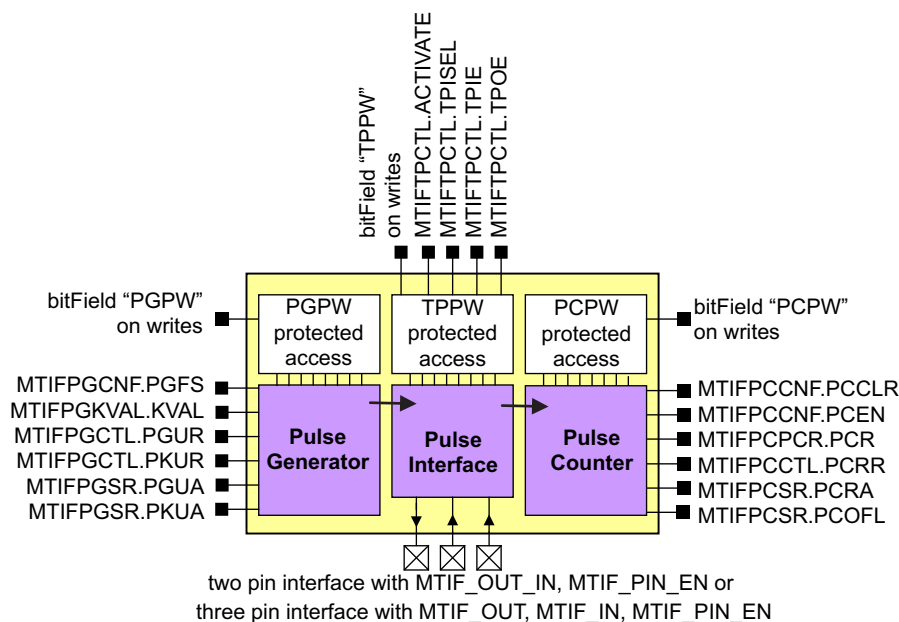


Figure 23-3. MTIF Internal Interfaces

23.2.1 MTIF and RTC C

MTIF and RTC share the same clock source, power, and resets. MTIF can therefore be kept synchronous with the RTC. With an RTC operated from 32.768 kHz, the MTIF can generate pulse rates from 8 to 1024 pulses per second.

23.2.2 Initialization of the MTIF

Before configuring the MTIF, the RTC module must be configured for operation. Chose the maximum pulse frequency for operation and select PGFS accordingly (see [Table 23-1](#)). In the following initialization example, a grid frequency of 256 Hz is chosen. The grid frequency is the frequency at which the inner state machines of the pulse generator and the pulse counter run. The width of pulses and the gaps are derived from this frequency. A frequency of 256 Hz allows a pulse rate of 128 pulses per seconds. The frame duration is 1 second. Within this second, up to 127 pulses can be generated. No pulse is generated in the update time slot.

[Table 23-2](#) summarizes the process of initializing MTIF. The MTIF requires a running RTC, therefore the RTC is initialized first. Next, the MTIF pulse generator, pins, and counter are configured. Optionally, the pulse counter can be cleared (regulations in some countries require that the counter can be cleared only once in a meter's lifetime).

Table 23-1. PGFS Values

PGFS	Pulses Rate	Frame Duration	Update Time Slot	Grid Frequency
0	8	16 s	62 ms before frame end	16 Hz
1	16	8 s	31 ms before frame end	32 Hz
2	32	4 s	15 ms before frame end	64 Hz
3	64	2 s	8 ms before frame end	128 Hz
4	128	1 s	4 ms before frame end	256 Hz
5	256	500 ms	2 ms before frame end	512 Hz
6	512	250 ms	1 ms before frame end	1024 Hz
7	1024	125 ms	500 μ s before frame end	2048 Hz

Table 23-2. MTIF Initialization

Order	Required	Comment
1	Configure and enable the RTC	See RTC module chapter for details
2	Write to MTIFPGCNF (PGPW = 0x5A, PGFS = 5, PGCLR = 1, PGEN = 1)	Set grid frequency to 256 Hz, clear and enable the MTIF pulse generator
3	Write to MTIFPGCTL (PGPW = 0x5A, PGUR = 1)	Request update of grid frequency setting
4	Check for MTIFPGST.PGUA = 1	Check for acknowledge of PGFS setting
5	Write to MTIFPCCNF (PCPW = 0xA5, PCCLR = 1, PCEN = 1)	Clear and enable MTIF pulse counter
6	Write to MTIFTPCTL (TPPW = 0xC3, ACTIVATE = 1, TPOE = 1)	Enable MTIF output enable pin to control MTIF output

23.2.3 Setting the Pulse Rate

To set the pulse rate to n pulses per frame, write the required pulse count n into MTIFPGKVAL.KVAL = n with proper password and request an update by setting MTIFPGCTL.PKUR = 1 with password. The update occurs at the last time slot of the current frame. MTIFPGCTL.PKUA may be polled after for 1 to verify an update of MTIFPGKVAL.

[Table 23-3](#) shows a simple example to set the pulse rate.

Table 23-3. Setting the Pulse Rate

Order	LPM3 operation	LPM3.5 operation	Comment
1	Check if MTIFPGSR.PGUA = 0	Check if MTIFPGSR.PGUA = 0	Check if last request is complete
2	Write to MTIFPGKVAL (PGPW = 0x5A, KVAL = n)	Write to MTIFPGKVAL (PGPW = 0x5A, KVAL = n)	Set to n pulses
3	Write to MTIFPGCTL (PCPW = 0x5A, PGUR = 1)	Write to MTIFPGCTL (PCPW = 0x5A, PGUR = 1)	Request update of pulse generator
4	Enter LPM3	Enter LPM3.5	Optional, when required

23.2.4 Reading Pulse Counter

To read the pulse counter perform a read request to update the pulse counter read register by writing MTIFPCCTL.PCRR = 1 with password, then wait for the acknowledge on MTIFPCST.PCRA for confirmation.

Table 23-4. Reading the Pulse Rate

Order	LPM3 operation	Comment
1	Check if MTIFPCSR.PCRA = 0	Check if last request is complete
2	Write to MTIFPCCTL (PGPW = 0xA5, PCRR = 1)	Request update of pulse counter read register
3	Wait for MTIFPCSR.PCRA = 1	Wait until update request is acknowledged
4	Read MTIFPCR register	Treat the count value as signed 16 bit value
5	Subtract the old (previous) value from new value; this is done after reading the PCSR.OFL bit	Calculate the difference with wrap around correction when negative

MTIFPCSE.PCOFL indicates an wrap around of the pulse counter. MTIFPCSR.PCOFL can be considered as the 17th bit of the counter. XOR the previous read with the current read value to determine the overflow condition.

23.2.5 Synchronizing Pulse Generator Timing to Application

At the very first start up, perform an update request: set MTIFPGCTL.PGUR, poll MTIFPGSR.PGUA for changing, then read the RTC prescaler registers after the MTIFPGSR.PGUA change. The RTC prescaler value indicates the moment of the update slot for the selected frame rate. Alternatively, set the RTC prescaler registers after a MTIFPGSR.PGUA change for synchronization.

In most cases, the pulse generator frame rate and the measurement rate of the meter are kept the same. If the MTIFPGKVAL register is updated much faster than the pulse generator frame frequency, the value read during the update time slot is used for the next frame.

23.2.6 Various Resets During MTIF Operation

The MTIF module provides immunity against most system reset types. A power-up clear (PUC) resets the watchdog timer (WDT), and password violation resets do not cause any changes in setting, count, and behavior of the MTIF setting.

A power-on reset (POR) stops the RTC and low-frequency crystal and sets MTIFPGCNF.PGEN = 0 and MTIFPCCNF.PCEN = 0. After a POR, reactivate the module and settings and record the POR event as an error with a timestamp for later reference, if required by the application.

A POR (for example, after waking up from LPMx.5) resets the MTIF pin configuration, but the content of the pulse generator setting and the pulse counter are not affected. Reinitialize the MTIF pin configuration by software.

A BOR (resulting from a power supply drop) resets the MTIF module, including the pulse generating setting and pulse counter value.

A reset caused by the $\overline{\text{RST}}$ pin resets the MTIF module. To avoid this type of reset, select the NMI functionality on the $\overline{\text{RST}}$ /NMI pin. Restore SFRRPCR.SYSNMI on any other reset.

23.2.7 PUC Reset During Register Access

PUC resets can occur during register access of the pulse generator registers, pulse counter registers, or pin configuration registers. A PUC reset requires immediate action during the following reset start-up sequence, because the configuration and counter value may be corrupted. Alternatively, use software signatures in FRAM and Hoare_Monitors to keep track of what was happening and resolve the situation.

The application can keep FRAM structures (in separate storage clusters) that contain "old_value", "new_value", "transaction_number", and "transaction_state". Record and update each phase of the transaction. This allows recovery after any reset on the next start-up.

23.2.8 Enabling the Pulse Generator and the Pulse Counter

The configuration bits MTIFPGCNT.PGEN and MTIFPCCNF.PCEN are expected to be static during a regular metering operation. At start-up of the meter application, a reset caused by the $\overline{\text{RST}}$ pin after recovery from a power loss, the MTIF behaves as follows:

The prescaler of the pulse generator is cleared when the pulse generator is disabled. After the application enables MTIFPGCTL.PGEN, the pulse generation starts after 2 to 3 cycles (with the LF clock at 32 kHz). Even though the pulse counter may be cleared independent of MTIFPGCTL.PCEN, make sure that the counter is cleared either before or after 2 to 3 LFCLK cycles to avoid conflicts.

The pulse counter increments on high active pulses. Shorter pulses (artifacts) may be generated when enabling and disabling the MTIF output pin. A complete MTIFPCCNF.PCEN disable and enable cycle can cause up to two increments of the pulse counter when using the external pulse input pin.

23.3 MTIF Block Diagram

Figure 23-4 shows the interior of the MTIF module with an 3-pin MTIF I/O configuration. The dashed lines represent the power domain crossings to areas that are powered in LPM0 to LPM3.5.

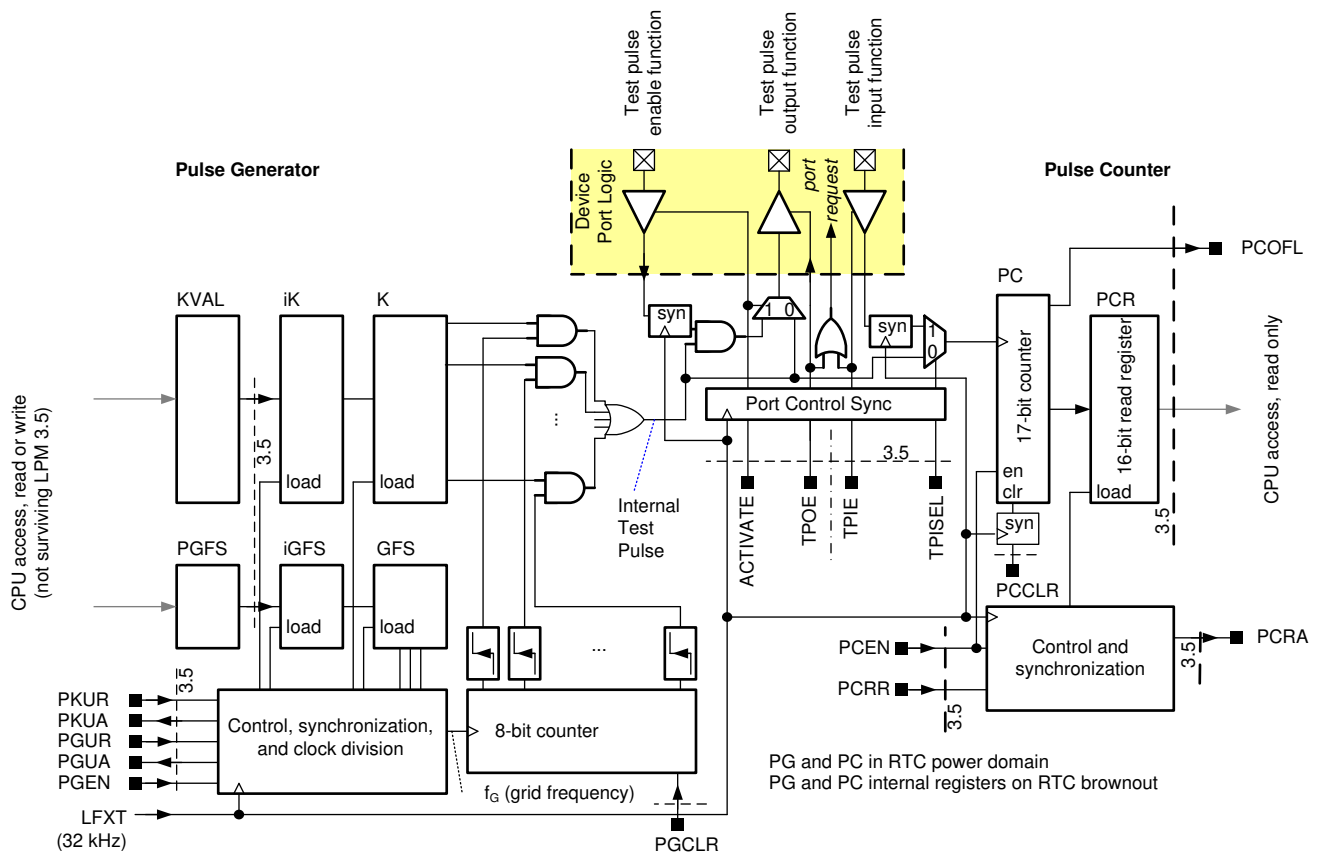


Figure 23-4. MTIF Block Diagram

23.3.1 Test Interface Input

The input does not feature any filter logic. To filter low-frequency contact bouncing, use an external RC filter. For higher ESD, EMI, and RFI tolerance, use appropriate shielding and filtering. The enable function of the test interface input is located in a separate register to fulfill the functional isolation requirement of various regulations.

The MTIF module supports one input and one output, which can be on separate pins or on a shared pin, depending on the specific device.

23.4 MTIF Registers

Table 23-5 lists the memory-mapped registers for the MTIF. All register offset addresses not listed in Table 23-5 should be considered as reserved locations and the register contents should not be modified.

Table 23-5. MTIF Registers

Offset	Acronym	Register Name	Type	Reset	Section
0h	MTIFPGCNF	Pulse Generator Configuration Register	read-write	6970h	Section 23.4.1
2h	MTIFPGKVAL	Pulse Generator Value Register	read-write	6900h	Section 23.4.2
4h	MTIFPGCTL	Pulse Generator Control Register	read-write	6900h	Section 23.4.3
6h	MTIFPGSR	Pulse Generator Status Register	read-write	0h	Section 23.4.4
8h	MTIFPCCNF	Pulse Counter Configuration Register	read-write	9600h	Section 23.4.5
Ah	MTIFPCR	Pulse Counter Value Register	read-write		Section 23.4.6
Ch	MTIFPCCTL	Pulse Counter Control Register	read-write	0h	Section 23.4.7
Eh	MTIFPCSR	Pulse Counter Status Register	read-write	0h	Section 23.4.8
10h	MTIFTPCTL	Measurement Test Port Control Register	read-write	F00h	Section 23.4.9

23.4.1 MTIFPGCNF Register (Offset = 0h) [reset = 6970h]

MTIFPGCNF is shown in [Figure 23-5](#) and described in [Table 23-6](#).

Return to [Summary Table](#).

Pulse Generator Configuration Register

Figure 23-5. MTIFPGCNF Register

15	14	13	12	11	10	9	8
PGPW							
RH/W-69h							
7	6	5	4	3	2	1	0
RESERVED	PGFS			RESERVED	PGCLR	RESERVED	PGEN
R/W-0h	R/W-7h			R/W-0h	RH/W1S-0h	R/W-0h	R/W-0h

Table 23-6. MTIFPGCNF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	PGPW	RH/W	69h	PG password. Always reads as 0x69. Must be written as 0x5A for register changes to be effective. This password differs from the pin configuration and pulse counter passwords. Reset type: PUC 5Ah (W) = PGPW : Must be written as 0x5A for register changes to be effective. 69h (R) = PGPW_R : Read value while locked
7	RESERVED	R/W	0h	
6-4	PGFS	R/W	7h	PG pulse grid frequency select. This value determines at which time grid pulses are generated. The pulse generator frame frequency is an 1/256th of this (MTIFPGCNF.PGEN has to be one to perform a change). Reset type: PUC 0h (R/W) = Pulse grid frequency is set to 8 Hz (nominal) 1h (R/W) = Pulse grid frequency is set to 16 Hz (nominal) 2h (R/W) = Pulse grid frequency is set to 32 Hz (nominal) 3h (R/W) = Pulse grid frequency is set to 64 Hz (nominal) 4h (R/W) = Pulse grid frequency is set to 128 Hz (nominal) 5h (R/W) = Pulse grid frequency is set to 256 Hz (nominal) 6h (R/W) = Pulse grid frequency is set to 512 Hz (nominal) 7h (R/W) = Pulse grid frequency is set to 1024 Hz (nominal) default
3	RESERVED	R/W	0h	
2	PGCLR	RH/W1S	0h	PG pulse counter clear. This bit allows to clear the pulse generator (MTIFPGCNF.PGEN has to be set to one to perform a clear). Note!: A clear request is being latched and released after the clear is executed. While MTIFPGCNF.PCEN =0 a time shift is generated. The clear occurs then after the clock is reenabled. This bit is for triggering only; it's state cannot be read back Reset type: PUC
1	RESERVED	R/W	0h	
0	PGEN	R/W	0h	PG sub module enable. This bit enables the PG sub module when set to one Reset type: POR

23.4.2 MTIFPGKVAL Register (Offset = 2h) [reset = 6900h]

MTIFPGKVAL is shown in [Figure 23-6](#) and described in [Table 23-7](#).

Return to [Summary Table](#).

Pulse Generator Value Register

Figure 23-6. MTIFPGKVAL Register

15	14	13	12	11	10	9	8
PGPW							
R/W-69h							
7	6	5	4	3	2	1	0
RESERVED				KVAL			
R/W-0h				R/W-0h			

Table 23-7. MTIFPGKVAL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	PGPW	R/W	69h	PG password. Always reads as 0x69. Must be written as 0x5A for register changes to be effective. This password differs from the pin configuration and pulse counter passwords. Reset type: PUC 5Ah (W) = PGPW : Must be written as 0x5A for register changes to be effective. 69h (R) = PGPW_R : Read value while locked
7	RESERVED	R/W	0h	
6-0	KVAL	R/W	0h	Pulse Count Number. This register value determines how many pulses are generated withing 256 periods of the pulse grid frequency(with password protection as in MTIFPGCNF). MTIFPGCNF.PGEN has to be one to perform a change. Reset type: PUC

23.4.3 MTIFPGCTL Register (Offset = 4h) [reset = 6900h]

MTIFPGCTL is shown in [Figure 23-7](#) and described in [Table 23-8](#).

Return to [Summary Table](#).

Pulse Generator Control Register

Figure 23-7. MTIFPGCTL Register

15	14	13	12	11	10	9	8
PGPW							
R/W-69h							
7	6	5	4	3	2	1	0
RESERVED						PGUR	PKUR
R/W-0h						RH/W1S-0h	RH/W1S-0h

Table 23-8. MTIFPGCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	PGPW	R/W	69h	PG password. Always reads as 0x69. Must be written as 0x5A for register changes to be effective. This password differs from the pin configuration and pulse counter passwords. Reset type: PUC 5Ah (W) = PGPW : Must be written as 0x5A for register changes to be effective. 69h (R) = PGPW_R : Read value while locked
7-2	RESERVED	R/W	0h	
1	PGUR	RH/W1S	0h	Pulse Grid Frequency Update Request (with password protection as in MTIFPGCNF). The update of PGFS occurs during the frequency grid slot 0xff (e.g. in the last 4ms of an second with an pulse grid frequency of 256Hz) Reset type: PUC
0	PKUR	RH/W1S	0h	Pulse K-Count Update Request (with password protection as in MTIFPGCNF). The update of KVAL occurs during the frequency grid slot 0xff (e.g. in the last 4ms of a second with a pulse grid frequency of 256Hz) Reset type: PUC

23.4.4 MTIFPGSR Register (Offset = 6h) [reset = 0h]

MTIFPGSR is shown in [Figure 23-8](#) and described in [Table 23-9](#).

Return to [Summary Table](#).

Pulse Generator Status Register

Figure 23-8. MTIFPGSR Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED						PGUA	PKUA
R/W-0h						RH/W-0h	RH/W-0h

Table 23-9. MTIFPGSR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	RESERVED	R/W	0h	
1	PGUA	RH/W	0h	Pulse Grid Frequency Update Acknowledge. This acknowledges a MTIFPGSR.PGUR 2-3 LFCLK cycles after the PGFS has been updated. Reset type: PUC
0	PKUA	RH/W	0h	Pulse K-Count Update Acknowledge. This acknowledges a MTIFPGSR.PKUR 2-3 LFCLK cycles after the K-values have been updated. Reset type: PUC

23.4.5 MTIFPCCNF Register (Offset = 8h) [reset = 9600h]

MTIFPCCNF is shown in [Figure 23-9](#) and described in [Table 23-10](#).

Return to [Summary Table](#).

Pulse Counter Configuration Register

Figure 23-9. MTIFPCCNF Register

15	14	13	12	11	10	9	8
PCPW							
RH/W-96h							
7	6	5	4	3	2	1	0
RESERVED					PCCLR	RESERVED	PCEN
R/W-0h					RH/W1S-0h	R/W-0h	R/W1S-0h

Table 23-10. MTIFPCCNF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	PCPW	RH/W	96h	Pulse counter password. Always reads as 0x96. Must be written as 0xA5 for register changes to be effective. This password differs from the pin configuration and pulse generator passwords Reset type: PUC 96h (R) = PCPW_R : Read value while locked A5h (W) = PCPW : 0xA5
7-3	RESERVED	R/W	0h	
2	PCCLR	RH/W1S	0h	Pulse counter clear. This bit allows to clear the pulse counter when set to one (MTIFPCCNF.PCEN has to be one to perform a clear). Note!: A clear request is being latched and released after the clear is executed. While MTIFPCCNF.PCEN=0 a time shift is generated. The clear occurs then after the clock is reenabled. This bit is for triggering only; it's state cannot be read back Reset type: PUC
1	RESERVED	R/W	0h	
0	PCEN	R/W1S	0h	PC sub module enable. This bit enables the PC sub module when set to one Reset type: POR

23.4.6 MTIFPCR Register (Offset = Ah) [reset = 0h]

MTIFPCR is shown in [Figure 23-10](#) and described in [Table 23-11](#).

Return to [Summary Table](#).

Pulse Counter Value Register

Figure 23-10. MTIFPCR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCR															
RH-															

Table 23-11. MTIFPCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	PCR	RH	0	Pulse Counter value register. This register returns the count value from the pulse counter.

23.4.7 MTIFPCCTL Register (Offset = Ch) [reset = 0h]

MTIFPCCTL is shown in [Figure 23-11](#) and described in [Table 23-12](#).

[Return to Summary Table.](#)

Pulse Counter Control Register

Figure 23-11. MTIFPCCTL Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							PCRR
R/W-0h							RH/W1S-0h

Table 23-12. MTIFPCCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	R/W	0h	
0	PCRR	RH/W1S	0h	Pulse Counter Read Request. Set this to request an update of MTIFPCR read register from the actual counter. Reset type: PUC

23.4.8 MTIFPCSR Register (Offset = Eh) [reset = 0h]

MTIFPCSR is shown in [Figure 23-12](#) and described in [Table 23-13](#).

Return to [Summary Table](#).

Pulse Counter Status Register

Figure 23-12. MTIFPCSR Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED						PCOFL	PCRA
R/W-0h						RH/W-0h	RH/W-0h

Table 23-13. MTIFPCSR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	RESERVED	R/W	0h	
1	PCOFL	RH/W	0h	Pulse counter overflow. This bit indicates an overflow of the pulse counter when its value changes since the last read request procedure. It is basically the 17th bit of the counter Reset type: PUC
0	PCRA	RH/W	0h	Pulse counter read acknowledge. This acknowledges the update of the PCR register as response to the MTIFPCCTL.PCRR read request. Note!: A read request is being latched. A temporary disable (MTIFPCCNF.PCEN=0) allows a time shift. The read will then be performed and acknowledged after the clock is reenabled. Reset type: PUC

23.4.9 MTIFTPCTL Register (Offset = 10h) [reset = F00h]

MTIFTPCTL is shown in [Figure 23-13](#) and described in [Table 23-14](#).

Return to [Summary Table](#).

Measurement Test Port Control Register

Figure 23-13. MTIFTPCTL Register

15	14	13	12	11	10	9	8
TPPW							
RH/W-Fh							
7	6	5	4	3	2	1	0
RESERVED				ACTIVATE	TPISEL	TPIE	TPOE
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 23-14. MTIFTPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	TPPW	RH/W	Fh	Test port password. Always reads as 0x0F. Must be written as 0xC3 for register changes to be effective. This password differs from the pulse generator and pulse counter passwords Reset type: PUC 0Fh (R) = TPPW_R : Read value while locked C3h (W) = TPPW : 0xC3
7-4	RESERVED	R/W	0h	
3	ACTIVATE	R/W	0h	Test port terminal enable activation. This value determines if the testport output is enabled solely by software or by software and hardware. Reset type: POR 0h (R/W) = The test port output is enabled solely by TPOE (enabled if MTIFTPCTL.TPOE=1) 1h (R/W) = The testport output requires both MTIFTPCTL.TPOE=1 and the MTPE_IN_ENABLE pin to be high to be enabled
2	TPISEL	R/W	0h	Test port input select for pulse counter. This value determines the source for the pulse counter. Reset type: POR 0h (R/W) = The pulse generator is used as input 1h (R/W) = The test port input terminal is selected as input
1	TPIE	R/W	0h	Test port input enable. This bit allows to enable the test input port Reset type: POR
0	TPOE	R/W	0h	Test port output enable. This bit allows to enable the test pulse output when set to one Reset type: POR

Watchdog Timer (WDT_A)

The watchdog timer is a 32-bit timer that can be used as a watchdog or as an interval timer. This chapter describes the watchdog timer. The enhanced watchdog timer, WDT_A, is implemented in all devices.

Topic	Page
24.1 WDT_A Introduction.....	636
24.2 WDT_A Operation	638
24.3 WDT_A Registers	640

24.1 WDT_A Introduction

The primary function of the watchdog timer (WDT_A) module is to perform a controlled system restart after a software problem occurs. If the selected time interval expires, a system reset is generated. If the watchdog function is not needed in an application, the module can be configured as an interval timer and can generate interrupts at selected time intervals.

Features of the watchdog timer module include:

- Eight software-selectable time intervals
- Watchdog mode
- Interval mode
- Password-protected access to Watchdog Timer Control (WDTCTL) register
- Selectable clock source
- Can be stopped to conserve power
- Clock fail-safe feature

The watchdog timer block diagram is shown in [Figure 24-1](#).

NOTE: Watchdog timer powers up active.

After a PUC, the WDT_A module is automatically configured in the watchdog mode with an initial approximately 32-ms reset interval using the SMCLK. The user must set up or halt the WDT_A before the initial reset interval expires.



24.2 WDT_A Operation

The watchdog timer module can be configured as either a watchdog or interval timer with the WDTCTL register. WDTCTL is a 16-bit password-protected read/write register. Any read or write access must use word instructions, and write accesses must include the write password 05Ah in the upper byte. A write to WDTCTL with any value other than 05Ah in the upper byte is a password violation and causes a PUC system reset, regardless of timer mode. Any read of WDTCTL reads 069h in the upper byte. Byte reads on WDTCTL high or low part result in the value of the low byte. Writing byte wide to upper or lower parts of WDTCTL results in a PUC.

24.2.1 Watchdog Timer Counter (WDTCNT)

The WDTCNT is a 32-bit up counter that is not directly accessible by software. The WDTCNT is controlled and its time intervals are selected through the Watchdog Timer Control (WDTCTL) register. The WDTCNT can be sourced from SMCLK, ACLK, VLOCLK, and X_CLK on some devices. The clock source is selected with the WDTSEL bits. The timer interval is selected with the WDTIS bits.

24.2.2 Watchdog Mode

After a PUC condition, the WDT module is configured in the watchdog mode with an initial 32-ms (approximate) reset interval using the SMCLK. The user must set up, halt, or clear the watchdog timer before this initial reset interval expires, or another PUC is generated. When the watchdog timer is configured to operate in watchdog mode, either writing to WDTCTL with an incorrect password or expiration of the selected time interval triggers a PUC. A PUC resets the watchdog timer to its default condition.

24.2.3 Interval Timer Mode

Setting the WDTTSEL bit to 1 selects the interval timer mode. This mode can be used to provide periodic interrupts. In interval timer mode, the WDTIFG flag is set at the expiration of the selected time interval. A PUC is not generated in interval timer mode at expiration of the selected timer interval, and the WDTIFG enable bit WDTIE remains unchanged.

When the WDTIE bit and the GIE bit are set, the WDTIFG flag requests an interrupt. The WDTIFG interrupt flag is automatically reset when its interrupt request is serviced, or may be reset by software. The interrupt vector address in interval timer mode is different from that in watchdog mode.

NOTE: Modifying the watchdog timer

The watchdog timer interval should be changed together with WDTCNTCL = 1 in a single instruction to avoid an unexpected immediate PUC or interrupt. The watchdog timer should be halted before changing the clock source to avoid a possible incorrect interval.

24.2.4 Watchdog Timer Interrupts

The watchdog timer uses two bits in the SFRs for interrupt control:

- WDT interrupt flag, WDTIFG, located in SFRIFG1.0
- WDT interrupt enable, WDTIE, located in SFRIFG1.0

When using the watchdog timer in the watchdog mode, the WDTIFG flag sources a reset vector interrupt. The WDTIFG can be used by the reset interrupt service routine to determine if the watchdog caused the device to reset. If the flag is set, the watchdog timer initiated the reset condition, either by timing out or by a password violation. If WDTIFG is cleared, the reset was caused by a different source.

When using the watchdog timer in interval timer mode, the WDTIFG flag is set after the selected time interval and requests a watchdog timer interval timer interrupt if the WDTIE and the GIE bits are set. The interval timer interrupt vector is different from the reset vector used in watchdog mode. In interval timer mode, the WDTIFG flag is reset automatically when the interrupt is serviced, or can be reset with software.

24.2.5 Fail-Safe Features

The WDT_A provides a fail-safe clocking feature, ensuring the clock to the WDT_A cannot be disabled while in watchdog mode. This means the low-power modes may be affected by the choice for the WDT_A clock.

In watchdog mode the WDT_A prevents LPMx.5 because in LPMx.5 the WDT_A cannot operate.

If SMCLK or ACLK fails as the WDT_A clock source, LFMODCLK clock is automatically selected as the WDT_A clock source.

When the WDT_A module is used in interval timer mode, there are no fail-safe features.

24.2.6 Operation in Low-Power Modes

The devices have several low-power modes. Different clock signals are available in different low-power modes. The requirements of the application and the type of clocking that is used determine how the WDT_A should be configured. For example, the WDT_A should not be configured in watchdog mode with a clock source that is originally sourced from DCO, XT1 in high-frequency mode, or XT2 sourcing SMCLK or ACLK if the user wants to use low-power mode 3. In this case, SMCLK or ACLK would remain enabled, increasing the current consumption of LPM3. When the watchdog timer is not required, the WDT_HOLD bit can be used to hold the WDT_CNT, reducing power consumption.

Any write operation to WDT_CTL must be a word operation with 05Ah (WDT_PW) in the upper byte (see [Example 24-1](#)).

Example 24-1. Writes to WDT_CTL

```
; Periodically clear an active watchdog
MOV #WDT_PW+WDTIS2+WDTIS1+WDT_CNTCL,&WDT_CTL
;
; Change watchdog timer interval
MOV #WDT_PW+WDT_CNTCL+SSEL,&WDT_CTL
;
; Stop the watchdog
MOV #WDT_PW+WDT_HOLD,&WDT_CTL
;
; Change WDT to interval timer mode, clock/8192 interval
MOV #WDT_PW+WDT_CNTCL+WDTT_MSEL+WDTIS2+WDTIS0,&WDT_CTL
```

24.3 WDT_A Registers

The watchdog timer module registers are listed in [Table 24-1](#). The base address for the watchdog timer module registers and special function registers (SFRs) can be found in the device-specific data sheets. The address offset is given in [Table 24-1](#).

Table 24-1. WDT_A Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	WDTCTL	Watchdog Timer Control	Read/write	Word	6904h	Section 24.3.1

24.3.1 WDTCTL Register

Watchdog Timer Control Register

Figure 24-2. WDTCTL Register

15	14	13	12	11	10	9	8
WDTPW							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
WDTHOLD	WDTSSSEL		WDTTMSEL	WDCNTCL	WDTIS		
rw-0	rw-0	rw-0	rw-0	r0(w)	rw-1	rw-0	rw-0

Table 24-2. WDTCTL Register Description

Bit	Field	Type	Reset	Description
15-8	WDTPW	RW	69h	Watchdog timer password. Always read as 069h. Must be written as 05Ah, or a PUC is generated.
7	WDTHOLD	RW	0h	Watchdog timer hold. This bit stops the watchdog timer. Setting WDTHOLD = 1 when the WDT is not in use conserves power. 0b = Watchdog timer is not stopped 1b = Watchdog timer is stopped
6-5	WDTSSSEL	RW	0h	Watchdog timer clock source select 00b = SMCLK 01b = ACLK 10b = VLOCLK 11b = X_CLK, same as VLOCLK if not defined differently in data sheet
4	WDTTMSEL	RW	0h	Watchdog timer mode select 0b = Watchdog mode 1b = Interval timer mode
3	WDCNTCL	RW	0h	Watchdog timer counter clear. Setting WDCNTCL = 1 clears the count value to 0000h. WDCNTCL is automatically reset. 0b = No action 1b = WDCNT = 0000h
2-0	WDTIS	RW	4h	Watchdog timer interval select. These bits select the watchdog timer interval to set the WDTIFG flag or generate a PUC. 000b = Watchdog clock source / 2^{31} (18:12:16 at 32.768 kHz) 001b = Watchdog clock source / 2^{27} (01:08:16 at 32.768 kHz) 010b = Watchdog clock source / 2^{23} (00:04:16 at 32.768 kHz) 011b = Watchdog clock source / 2^{19} (00:00:16 at 32.768 kHz) 100b = Watchdog clock source / 2^{15} (1 s at 32.768 kHz) 101b = Watchdog clock source / 2^{13} (250 ms at 32.768 kHz) 110b = Watchdog clock source / 2^9 (15.625 ms at 32.768 kHz) 111b = Watchdog clock source / 2^6 (1.95 ms at 32.768 kHz)

Timer_A

Timer_A is a 16-bit timer and counter with multiple capture/compare registers. There can be multiple Timer_A modules on a given device (see the device-specific data sheet). This chapter describes the operation and use of the Timer_A module.

Topic	Page
25.1 Timer_A Introduction	643
25.2 Timer_A Operation.....	645
25.3 Timer_A Registers	657

25.1 Timer_A Introduction

Timer_A is a 16-bit timer/counter with up to seven capture/compare registers. Timer_A can support multiple capture/compares, PWM outputs, and interval timing. Timer_A also has extensive interrupt capabilities. Interrupts may be generated from the counter on overflow conditions and from each of the capture/compare registers.

Timer_A features include:

- Asynchronous 16-bit timer/counter with four operating modes
- Selectable and configurable clock source
- Up to seven configurable capture/compare registers
- Configurable outputs with pulse width modulation (PWM) capability
- Asynchronous input and output latching
- Interrupt vector register for fast decoding of all Timer_A interrupts

The block diagram of Timer_A is shown in [Figure 25-1](#).

NOTE: Use of the word *count*

Count is used throughout this chapter. It means the counter must be in the process of counting for the action to take place. If a particular value is directly written to the counter, an associated action does not take place.

NOTE: Nomenclature

There may be multiple instantiations of Timer_A on a given device. The prefix TAx is used, where x is a greater than equal to zero indicating the Timer_A instantiation. For devices with one instantiation, x = 0. The suffix n, where n = 0 to 6, represents the specific capture/compare registers associated with the Timer_A instantiation.

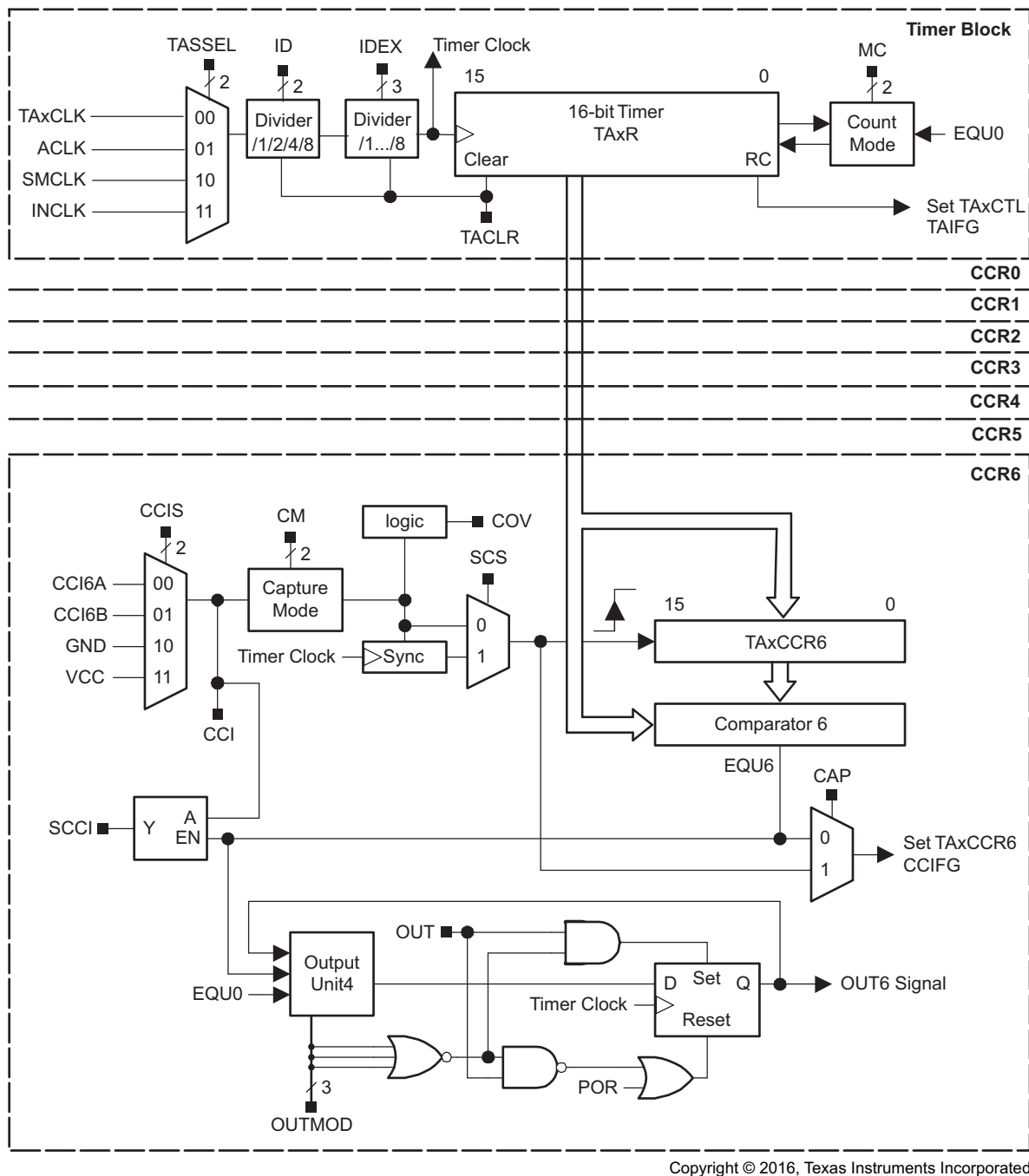


Figure 25-1. Timer_A Block Diagram

25.2 Timer_A Operation

The Timer_A module is configured with user software. The setup and operation of Timer_A are discussed in the following sections.

25.2.1 16-Bit Timer Counter

The 16-bit timer/counter register, [TAXR](#), increments or decrements (depending on mode of operation) with each rising edge of the clock signal. TAXR can be read or written with software. Additionally, the timer can generate an interrupt when it overflows.

TAR may be cleared by setting the TACLR bit. Setting TACLR also clears the clock divider counter logic (the divider setting remains unchanged) and count direction for up/down mode.

NOTE: Modifying Timer_A registers

TI recommends stopping the timer before modifying its operation (with exception of the interrupt enable, interrupt flag, and TACLR) to avoid errant operating conditions.

When the timer clock is asynchronous to the CPU clock, any read from [TAXR](#) should occur while the timer is not operating or the results may be unpredictable. Alternatively, the timer may be read multiple times while operating, and a majority vote taken in software to determine the correct reading. Any write to TAXR takes effect immediately.

25.2.1.1 Clock Source Select and Divider

The timer clock can be sourced from ACLK, SMCLK, or externally from TAXCLK or INCLK. The clock source is selected with the TASSEL bits. The selected clock source may be passed directly to the timer or divided by 2, 4, or 8, using the ID bits. The selected clock source can be further divided by 2, 3, 4, 5, 6, 7, or 8 using the TAIDEX bits. The timer clock divider logic is reset when TACLR is set.

NOTE: Timer_A dividers

After programming ID or TAIDEX bits, set the TACLR bit. This clears the contents of [TAXR](#) and resets the clock divider logic to a defined state. The clock dividers are implemented as down counters. Therefore, when the TACLR bit is cleared, the timer clock immediately begins clocking at the first rising edge of the Timer_A clock source selected with the TASSEL bits and continues clocking at the divider settings set by the ID and TAIDEX bits.

25.2.2 Starting the Timer

The timer may be started or restarted in the following ways:

- The timer counts when $MC > \{ 0 \}$ and the clock source is active.
- When the timer mode is either up or up/down, the timer may be stopped by writing 0 to [TAXCCR0](#). The timer may then be restarted by writing a nonzero value to TAXCCR0. In this scenario, the timer starts incrementing in the up direction from zero.

25.2.3 Timer Mode Control

The timer has four modes of operation: stop, up, continuous, and up/down (see Table 25-1). The operating mode is selected with the MC bits.

Table 25-1. Timer Modes

MC	Mode	Description
00	Stop	The timer is halted.
01	Up	The timer repeatedly counts from zero to the value of TAxCCR0
10	Continuous	The timer repeatedly counts from zero to 0FFFFh.
11	Up/down	The timer repeatedly counts from zero up to the value of TAxCCR0 and back down to zero.

25.2.3.1 Up Mode

The up mode is used if the timer period must be different from 0FFFFh counts. The timer repeatedly counts up to the value of compare register TAxCCR0, which defines the period (see Figure 25-2). The number of timer counts in the period is TAxCCR0 + 1. When the timer value equals TAxCCR0, the timer restarts counting from zero. If up mode is selected when the timer value is greater than TAxCCR0, the timer immediately restarts counting from zero.

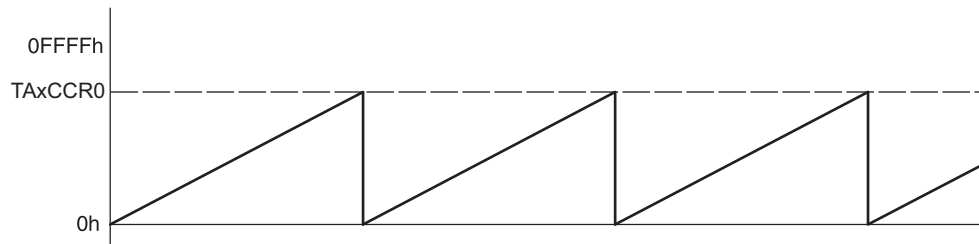


Figure 25-2. Up Mode

The TAxCCR0 CCIFG interrupt flag is set when the timer counts to the TAxCCR0 value. The TAIFG interrupt flag is set when the timer counts from TAxCCR0 to zero. Figure 25-3 shows the flag set cycle.

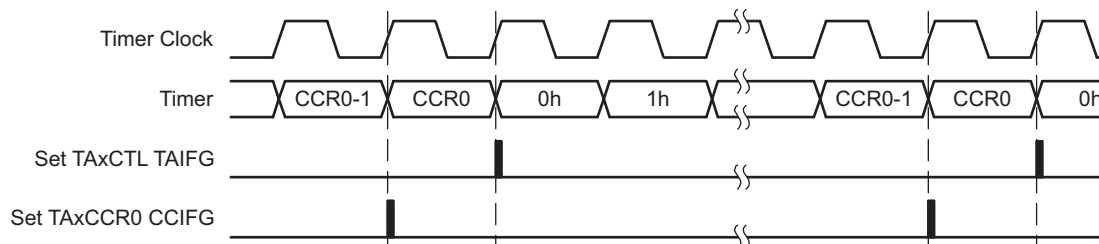


Figure 25-3. Up Mode Flag Setting

25.2.3.1.1 Changing Period Register TAxCCR0

When changing TAxCCR0 while the timer is running, if the new period is greater than or equal to the old period or greater than the current count value, the timer counts up to the new period. If the new period is less than the current count value, the timer rolls to zero. However, one additional count may occur before the counter rolls to zero.

25.2.3.2 Continuous Mode

In the continuous mode, the timer repeatedly counts up to 0FFFFh and restarts from zero as shown in Figure 25-4. The capture/compare register [TAxCCR0](#) works the same way as the other capture/compare registers.

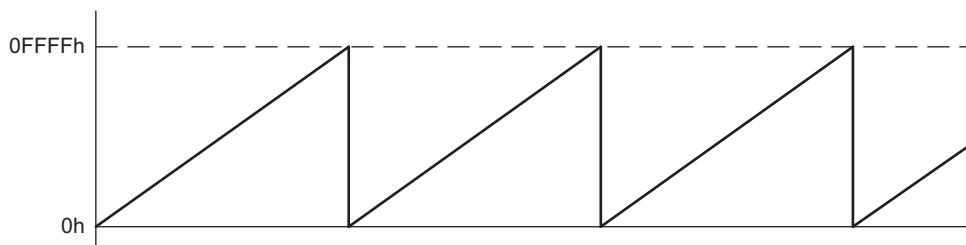


Figure 25-4. Continuous Mode

The TAIFG interrupt flag is set when the timer counts from 0FFFFh to zero. Figure 25-5 shows the flag set cycle.

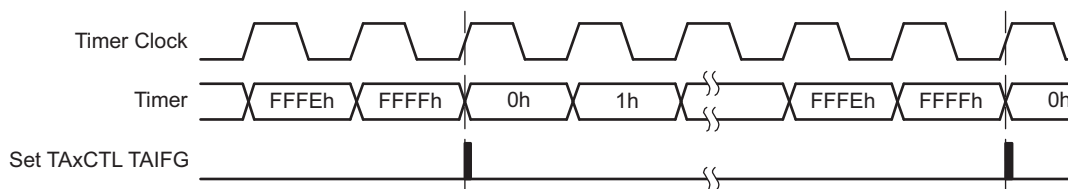


Figure 25-5. Continuous Mode Flag Setting

25.2.3.3 Use of Continuous Mode

The continuous mode can be used to generate independent time intervals and output frequencies. Each time an interval is completed, an interrupt is generated. The next time interval is added to the [TAxCCRn](#) register in the interrupt service routine. Figure 25-6 shows two separate time intervals, t_0 and t_1 , being added to the capture/compare registers. In this usage, the time interval is controlled by hardware, not software, without impact from interrupt latency. Up to n (where $n = 0$ to 6), independent time intervals or output frequencies can be generated using capture/compare registers.

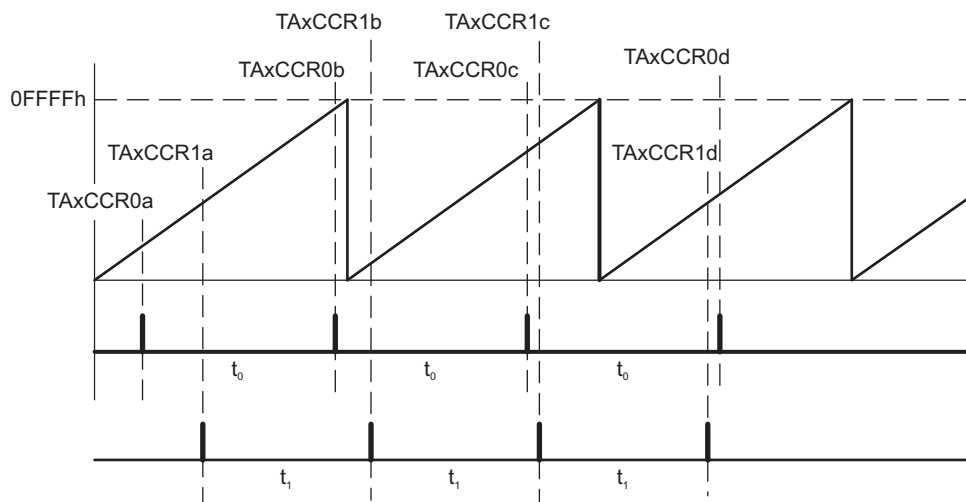


Figure 25-6. Continuous Mode Time Intervals

Time intervals can be produced with other modes as well, where [TAxCCR0](#) is used as the period register. Their handling is more complex since the sum of the old [TAxCCRn](#) data and the new period can be higher than the [TAxCCR0](#) value. When the previous [TAxCCRn](#) value plus t_x is greater than the [TAxCCR0](#) data, the [TAxCCR0](#) value must be subtracted to obtain the correct time interval.

25.2.3.4 Up/Down Mode

The up/down mode is used if the timer period must be different from 0FFFFh counts, and if symmetrical pulse generation is needed. The timer repeatedly counts up to the value of compare register [TAxCCR0](#) and back down to zero (see [Figure 25-7](#)). The period is twice the value in [TAxCCR0](#).

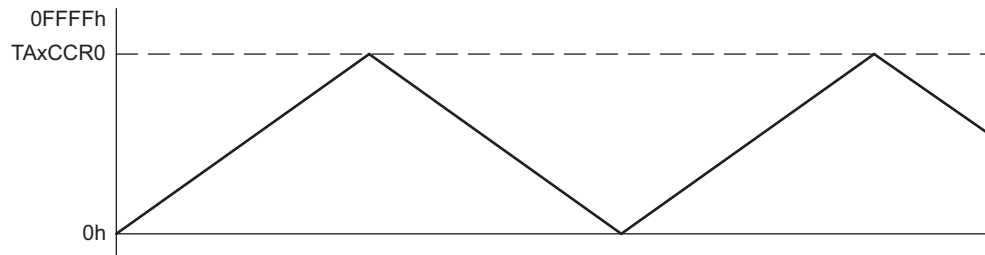


Figure 25-7. Up/Down Mode

The count direction is latched. This allows the timer to be stopped and then restarted in the same direction it was counting before it was stopped. If this is not desired, the [TACLR](#) bit must be set to clear the direction. Setting [TACLR](#) also clears the [TAR](#) value and the clock divider counter logic (the divider setting remains unchanged).

In up/down mode, the [TAxCCR0](#) CCIFG interrupt flag and the [TAIFG](#) interrupt flag are set only once during a period, separated by one-half the timer period. The [TAxCCR0](#) CCIFG interrupt flag is set when the timer *counts* from [TAxCCR0-1](#) to [TAxCCR0](#), and [TAIFG](#) is set when the timer completes *counting* down from 0001h to 0000h. [Figure 25-8](#) shows the flag set cycle.

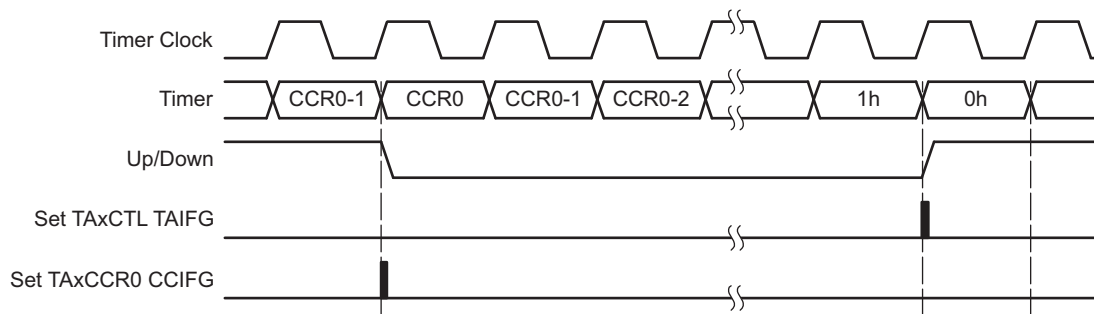


Figure 25-8. Up/Down Mode Flag Setting

25.2.3.4.1 Changing Period Register [TAxCCR0](#)

When changing [TAxCCR0](#) while the timer is running and counting in the down direction, the timer continues its descent until it reaches zero. The new period takes effect after the counter counts down to zero.

When the timer is counting in the up direction, and the new period is greater than or equal to the old period or greater than the current count value, the timer counts up to the new period before counting down.

When the timer is counting in the up direction and the new period is less than the current count value, the timer begins counting down. However, one additional count may occur before the counter begins counting down.

25.2.3.5 Use of Up/Down Mode

The up/down mode supports applications that require dead times between output signals (see [Section 25.2.5](#)). For example, to avoid overload conditions, two outputs driving an H-bridge must never be in a high state simultaneously. In the example shown in [Figure 25-9](#), the t_{dead} is:

$$t_{dead} = t_{timer} \times (TAxCCR1 - TAxCCR2)$$

Where:

t_{dead} = Time during which both outputs need to be inactive

t_{timer} = Cycle time of the timer clock

$TAxCCRn$ = Content of capture/compare register n

The $TAxCCRn$ registers are not buffered. They update immediately when written to. Therefore, any required dead time is not maintained automatically.

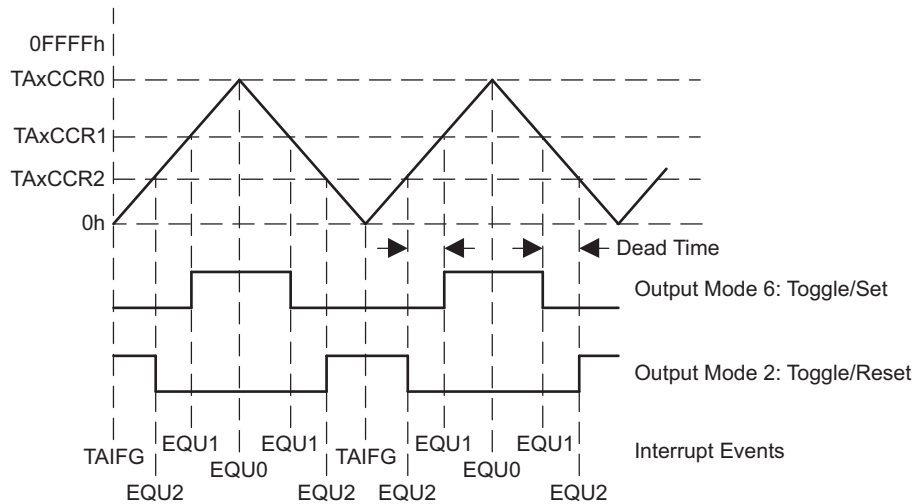


Figure 25-9. Output Unit in Up/Down Mode

25.2.4 Capture/Compare Blocks

Up to seven identical capture/compare blocks, $TAxCCRn$ (where $n = 0$ to 7), are present in Timer_A. Any of the blocks may be used to capture the timer data or to generate time intervals.

25.2.4.1 Capture Mode

The capture mode is selected when $CAP = 1$. Capture mode is used to record time events. It can be used for speed computations or time measurements. The capture inputs $CC1xA$ and $CC1xB$ are connected to external pins or internal signals and are selected with the $CCIS$ bits. The CM bits select the capture edge of the input signal as rising, falling, or both. A capture occurs on the selected edge of the input signal. If a capture occurs:

- The timer value is copied into the $TAxCCRn$ register.
- The interrupt flag $CCIFG$ is set.

The input signal level can be read at any time from the CCI bit. Devices may have different signals connected to $CC1xA$ and $CC1xB$. See the device-specific data sheet for the connections of these signals.

The capture signal can be asynchronous to the timer clock and cause a race condition. Setting the SCS bit synchronizes the capture with the next timer clock. Setting the SCS bit to synchronize the capture signal with the timer clock is recommended (see [Figure 25-10](#)).

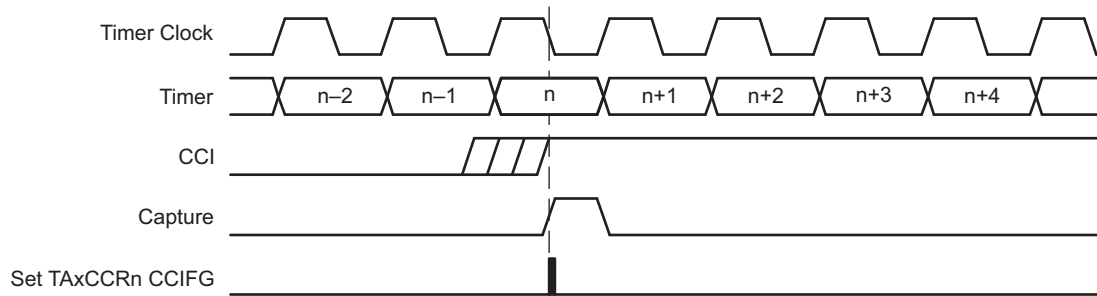


Figure 25-10. Capture Signal (SCS = 1)

NOTE: Changing Capture Inputs

Changing capture inputs while in capture mode may cause unintended capture events. To avoid this scenario, capture inputs should only be changed when capture mode is disabled (CM = {0} or CAP = 0).

Overflow logic is provided in each capture/compare register to indicate if a second capture was performed before the value from the first capture was read. Bit COV is set when this occurs as shown in [Figure 25-11](#). COV must be reset with software.

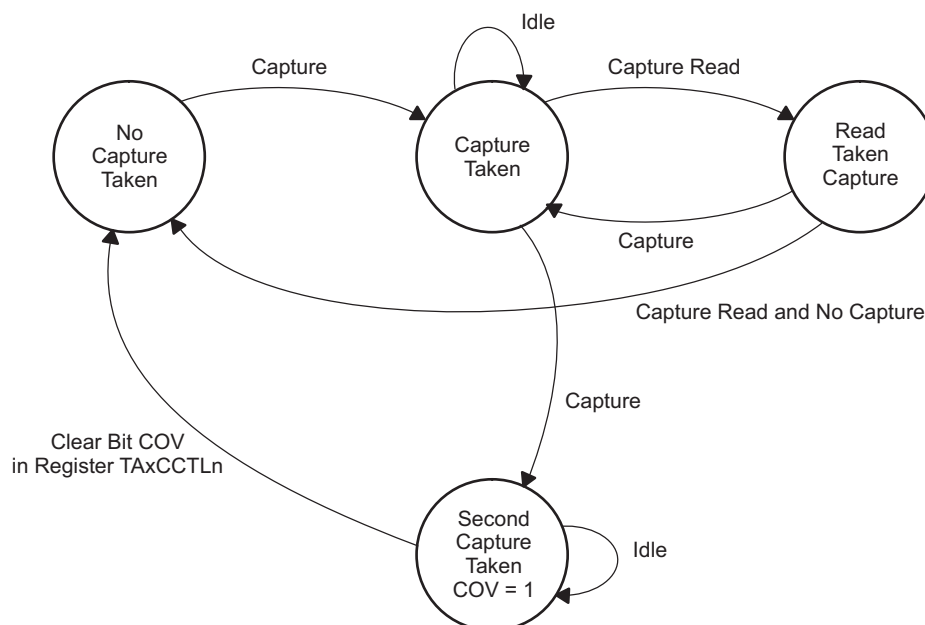


Figure 25-11. Capture Cycle

25.2.4.1.1 Capture Initiated by Software

Captures can be initiated by software. The CMx bits can be set for capture on both edges. Software then sets CCIS1 = 1 and toggles bit CCIS0 to switch the capture signal between V_{CC} and GND, initiating a capture each time CCIS0 changes state:

```
MOV  #CAP+SCS+CCIS1+CM_3,&TA0CCTL1 ; Setup TA0CCTL1, synch. capture mode
                                     ; Event trigger on both edges of capture input.
XOR  #CCIS0,&TA0CCTL1                ; TA0CCR1 = TA0R
```

NOTE: Capture Initiated by Software

In general, changing capture inputs while in capture mode may cause unintended capture events. For this scenario, switching the capture input between VCC and GND, disabling the capture mode is not required.

25.2.4.2 Compare Mode

The compare mode is selected when CAP = 0. The compare mode is used to generate PWM output signals or interrupts at specific time intervals. When *TaxR counts* to the value in a TAxCCRn, where n represents the specific capture/compare register.

- Interrupt flag CCIFG is set.
- Internal signal EQU_n = 1.
- EQU_n affects the output according to the output mode.
- The input signal CCI is latched into SCCI.

25.2.5 Output Unit

Each capture/compare block contains an output unit. The output unit is used to generate output signals, such as PWM signals. Each output unit has eight operating modes that generate signals based on the EQU0 and EQU_n signals.

25.2.5.1 Output Modes

The output modes are defined by the OUTMOD bits and are described in [Table 25-2](#). The OUT_n signal is changed with the rising edge of the timer clock for all modes except mode 0. Output modes 2, 3, 6, and 7 are not useful for output unit 0 because EQU_n = EQU0.

Table 25-2. Output Modes

OUTMODx	Mode	Description
000	Output	The output signal OUT _n is defined by the OUT bit. The OUT _n signal updates immediately when OUT is updated.
001	Set	The output is set when the timer <i>counts</i> to the TAxCCRn value. It remains set until a reset of the timer, or until another output mode is selected and affects the output.
010	Toggle/Reset	The output is toggled when the timer <i>counts</i> to the TAxCCRn value. It is reset when the timer <i>counts</i> to the TAxCCR0 value.
011	Set/Reset	The output is set when the timer <i>counts</i> to the TAxCCRn value. It is reset when the timer <i>counts</i> to the TAxCCR0 value.
100	Toggle	The output is toggled when the timer <i>counts</i> to the TAxCCRn value. The output period is double the timer period.
101	Reset	The output is reset when the timer <i>counts</i> to the TAxCCRn value. It remains reset until another output mode is selected and affects the output.
110	Toggle/Set	The output is toggled when the timer <i>counts</i> to the TAxCCRn value. It is set when the timer <i>counts</i> to the TAxCCR0 value.
111	Reset/Set	The output is reset when the timer <i>counts</i> to the TAxCCRn value. It is set when the timer <i>counts</i> to the TAxCCR0 value.

25.2.5.1.1 Output Example—Timer in Up Mode

The OUTn signal is changed when the timer *counts* up to the TAxCCRn value and rolls from TAxCCR0 to zero, depending on the output mode. An example is shown in Figure 25-12 using TAxCCR0 and TAxCCR1.

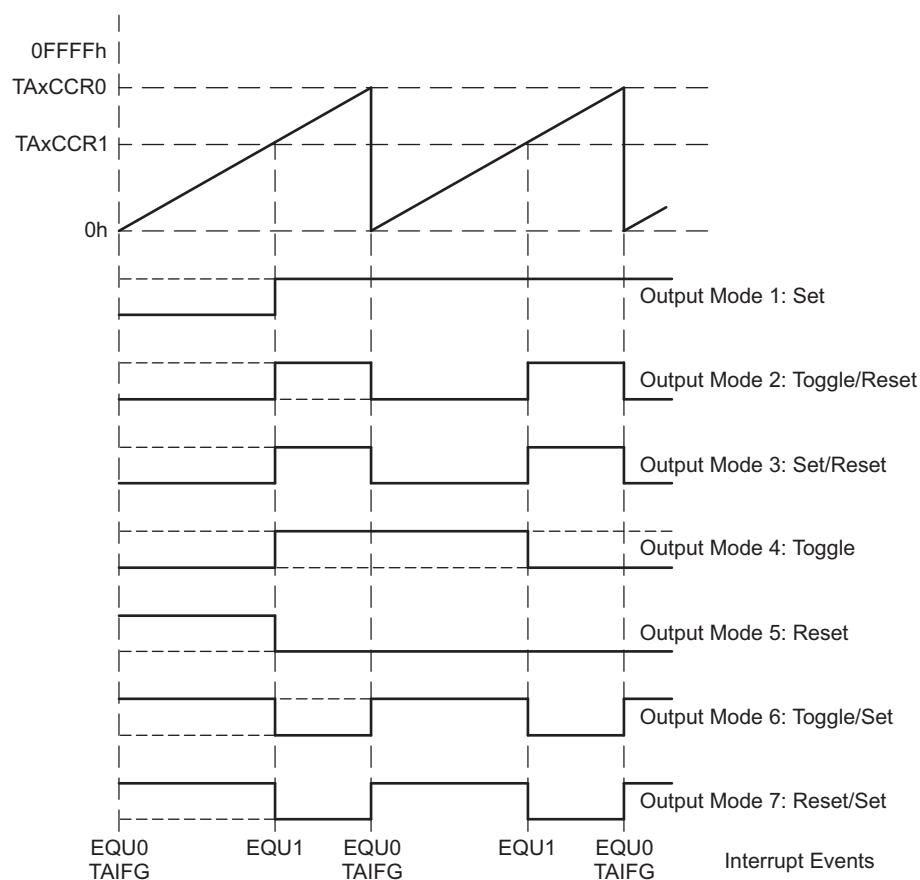


Figure 25-12. Output Example – Timer in Up Mode

25.2.5.1.2 Output Example – Timer in Continuous Mode

The OUTn signal is changed when the timer reaches the TAxCCRn and TAxCCR0 values, depending on the output mode. An example is shown in Figure 25-13 using TAxCCR0 and TAxCCR1.

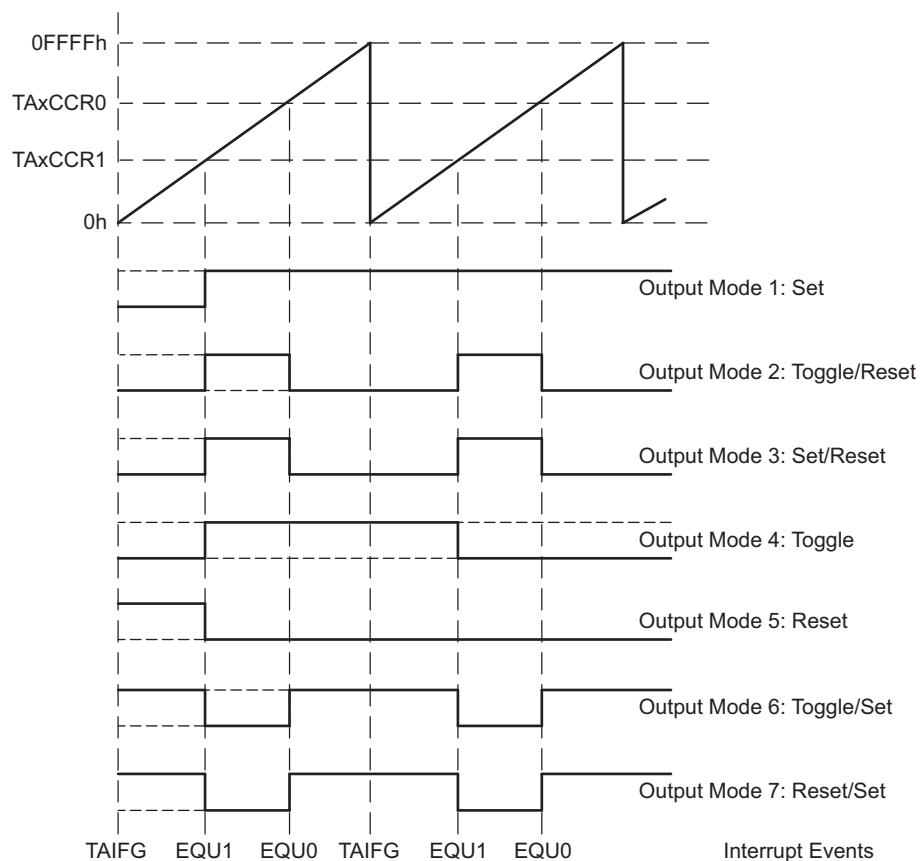


Figure 25-13. Output Example – Timer in Continuous Mode

25.2.5.1.3 Output Example – Timer in Up/Down Mode

The OUTn signal changes when the timer equals TAxCCRn in either count direction and when the timer equals TAxCCR0, depending on the output mode. An example is shown in Figure 25-14 using TAxCCR0 and TAxCCR2.

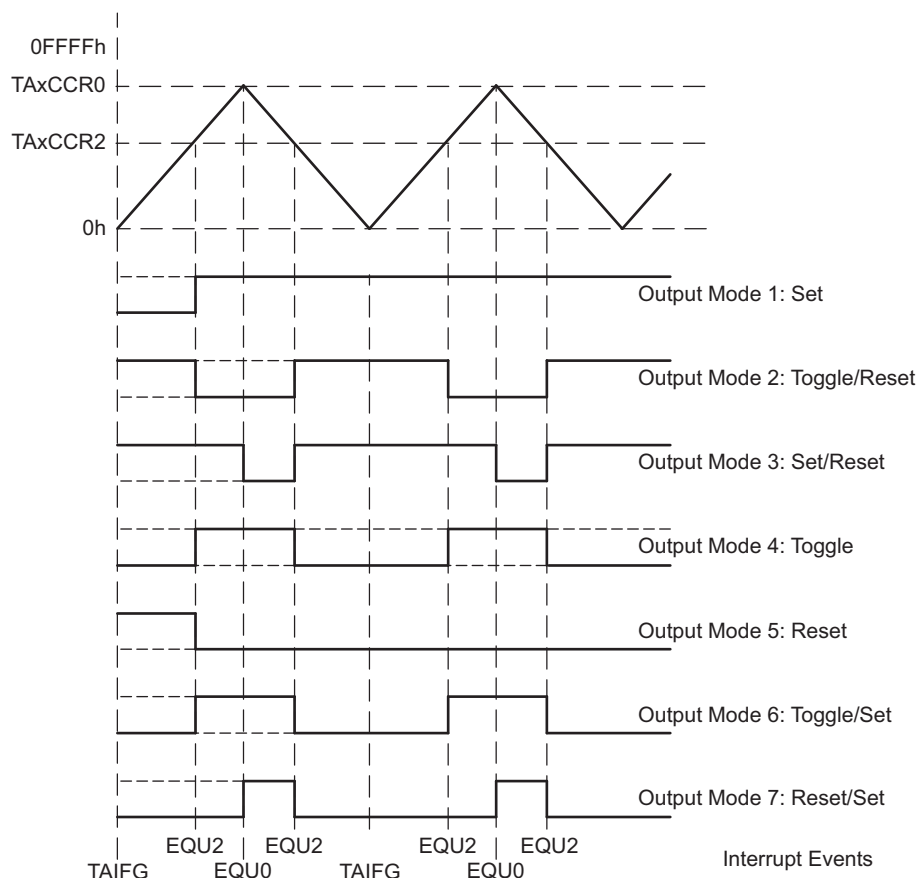


Figure 25-14. Output Example – Timer in Up/Down Mode

NOTE: Switching between output modes

When switching between output modes, one of the OUTMOD bits should remain set during the transition, unless switching to mode 0. Otherwise, output glitching can occur, because a NOR gate decodes output mode 0. A safe method for switching between output modes is to use output mode 7 as a transition state:

```
BIS    #OUTMOD_7,&TA0CCTL1    ; Set output mode=7
BIC    #OUTMOD,&TA0CCTL1      ; Clear unwanted bits
```


25.2.6 Timer_A Interrupts

Two interrupt vectors are associated with the 16-bit Timer_A module:

- **TAXCCR0** interrupt vector for TAXCCR0 CCIFG
- **TAXIV** interrupt vector for all other CCIFG flags and TAIFG

In capture mode, any CCIFG flag is set when a timer value is captured in the associated TAXCCRN register. In compare mode, any CCIFG flag is set if **TAXR** counts to the associated TAXCCRN value. Software may also set or clear any CCIFG flag. All CCIFG flags request an interrupt when their corresponding CCIE bit and the GIE bit are set.

25.2.6.1 TAXCCR0 Interrupt

The **TAXCCR0** CCIFG flag has the highest Timer_A interrupt priority and has a dedicated interrupt vector as shown in [Figure 25-15](#). The TAXCCR0 CCIFG flag is automatically reset when the TAXCCR0 interrupt request is serviced.

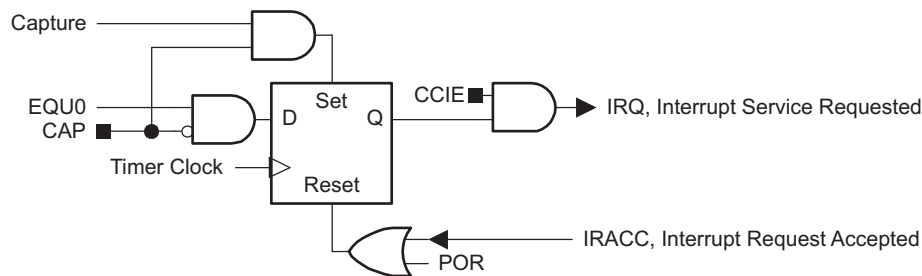


Figure 25-15. Capture/Compare TAXCCR0 Interrupt Flag

25.2.6.2 TAXIV, Interrupt Vector Generator

The TAXCCRY CCIFG flags and TAIFG flags are prioritized and combined to source a single interrupt vector. The interrupt vector register **TAXIV** is used to determine which flag requested an interrupt.

The highest-priority enabled interrupt generates a number in the **TAXIV** register (see register description). This number can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled Timer_A interrupts do not affect the TAXIV value.

Any access, read or write, of the **TAXIV** register automatically resets the highest-pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. For example, if the **TAXCCR1** and **TAXCCR2** CCIFG flags are set when the interrupt service routine accesses the TAXIV register, TAXCCR1 CCIFG is reset automatically. After the RETI instruction of the interrupt service routine is executed, the TAXCCR2 CCIFG flag generates another interrupt.

25.2.6.2.1 TAxIV Software Example

The following software example shows the recommended use of TAxIV and the handling overhead. The TAxIV value is added to the PC to automatically jump to the appropriate routine. The example assumes a single instantiation of the largest timer configuration available.

The numbers at the right margin show the necessary CPU cycles for each instruction. The software overhead for different interrupt sources includes interrupt latency and return-from-interrupt cycles, but not the task handling itself. The latencies are:

- Capture/compare block TA0CCR0: 11 cycles
- Capture/compare blocks TA0CCR1, TA0CCR2, TA0CCR3, TA0CCR4, TA0CCR5, TA0CCR6: 16 cycles
- Timer overflow TA0IFG: 14 cycles

```
; Interrupt handler for TA0CCR0 CCIFG.
CCIFG_0_HND
;      ...      ; Start of handler Interrupt latency    6
;      RETI      ;                                       5

; Interrupt handler for TA0IFG, TA0CCR1 through TA0CCR6 CCIFG.

TA0_HND      ...      ; Interrupt latency    6
      ADD      &TA0IV,PC      ; Add offset to Jump table    3
      RETI      ; Vector 0: No interrupt    5
      JMP      CCIFG_1_HND      ; Vector 2: TA0CCR1    2
      JMP      CCIFG_2_HND      ; Vector 4: TA0CCR2    2
      JMP      CCIFG_3_HND      ; Vector 6: TA0CCR3    2
      JMP      CCIFG_4_HND      ; Vector 8: TA0CCR4    2
      JMP      CCIFG_5_HND      ; Vector 10: TA0CCR5    2
      JMP      CCIFG_6_HND      ; Vector 12: TA0CCR6    2

TA0IFG_HND      ; Vector 14: TA0IFG Flag
      ...      ; Task starts here
      RETI      ;                                       5

CCIFG_6_HND      ; Vector 12: TA0CCR6
      ...      ; Task starts here
      RETI      ; Back to main program    5

CCIFG_5_HND      ; Vector 10: TA0CCR5
      ...      ; Task starts here
      RETI      ; Back to main program    5

CCIFG_4_HND      ; Vector 8: TA0CCR4
      ...      ; Task starts here
      RETI      ; Back to main program    5

CCIFG_3_HND      ; Vector 6: TA0CCR3
      ...      ; Task starts here
      RETI      ; Back to main program    5

CCIFG_2_HND      ; Vector 4: TA0CCR2
      ...      ; Task starts here
      RETI      ; Back to main program    5

CCIFG_1_HND      ; Vector 2: TA0CCR1
      ...      ; Task starts here
      RETI      ; Back to main program    5
```

25.3 Timer_A Registers

Timer_A registers are listed in [Table 25-3](#) for the largest configuration available. The base address can be found in the device-specific data sheet.

Table 25-3. Timer_A Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	TAxCTL	Timer_Ax Control	Read/write	Word	0000h	Section 25.3.1
02h	TAxCTL0	Timer_Ax Capture/Compare Control 0	Read/write	Word	0000h	Section 25.3.3
04h	TAxCTL1	Timer_Ax Capture/Compare Control 1	Read/write	Word	0000h	Section 25.3.3
06h	TAxCTL2	Timer_Ax Capture/Compare Control 2	Read/write	Word	0000h	Section 25.3.3
08h	TAxCTL3	Timer_Ax Capture/Compare Control 3	Read/write	Word	0000h	Section 25.3.3
0Ah	TAxCTL4	Timer_Ax Capture/Compare Control 4	Read/write	Word	0000h	Section 25.3.3
0Ch	TAxCTL5	Timer_Ax Capture/Compare Control 5	Read/write	Word	0000h	Section 25.3.3
0Eh	TAxCTL6	Timer_Ax Capture/Compare Control 6	Read/write	Word	0000h	Section 25.3.3
10h	TAxR	Timer_Ax Counter	Read/write	Word	0000h	Section 25.3.2
12h	TAxCCR0	Timer_Ax Capture/Compare 0	Read/write	Word	0000h	Section 25.3.4
14h	TAxCCR1	Timer_Ax Capture/Compare 1	Read/write	Word	0000h	Section 25.3.4
16h	TAxCCR2	Timer_Ax Capture/Compare 2	Read/write	Word	0000h	Section 25.3.4
18h	TAxCCR3	Timer_Ax Capture/Compare 3	Read/write	Word	0000h	Section 25.3.4
1Ah	TAxCCR4	Timer_Ax Capture/Compare 4	Read/write	Word	0000h	Section 25.3.4
1Ch	TAxCCR5	Timer_Ax Capture/Compare 5	Read/write	Word	0000h	Section 25.3.4
1Eh	TAxCCR6	Timer_Ax Capture/Compare 6	Read/write	Word	0000h	Section 25.3.4
2Eh	TAxIV	Timer_Ax Interrupt Vector	Read only	Word	0000h	Section 25.3.5
20h	TAxEX0	Timer_Ax Expansion 0	Read/write	Word	0000h	Section 25.3.6

25.3.1 TAxCTL Register

Timer_Ax Control Register

Figure 25-16. TAxCTL Register

15	14	13	12	11	10	9	8
Reserved						TASSEL	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ID		MC		Reserved	TACLR	TAIE	TAIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	w-(0)	rw-(0)	rw-(0)

Table 25-4. TAxCTL Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	RW	0h	Reserved
9-8	TASSEL	RW	0h	Timer_A clock source select 00b = TAxCLK 01b = ACLK 10b = SMCLK 11b = INCLK
7-6	ID	RW	0h	Input divider. These bits along with the TAIDEX bits select the divider for the input clock. 00b = /1 01b = /2 10b = /4 11b = /8
5-4	MC	RW	0h	Mode control. Setting MC = 00h when Timer_A is not in use conserves power. 00b = Stop mode: Timer is halted 01b = Up mode: Timer counts up to TAxCCR0 10b = Continuous mode: Timer counts up to 0FFFFh 11b = Up/down mode: Timer counts up to TAxCCR0 then down to 0000h
3	Reserved	RW	0h	Reserved
2	TACLR	RW	0h	Timer_A clear. Setting this bit clears TAR, the clock divider logic (the divider setting remains unchanged), and the count direction. The TACLR bit is automatically reset and is always read as zero.
1	TAIE	RW	0h	Timer_A interrupt enable. This bit enables the TAIFG interrupt request. 0b = Interrupt disabled 1b = Interrupt enabled
0	TAIFG	RW	0h	Timer_A interrupt flag 0b = No interrupt pending 1b = Interrupt pending

25.3.2 TAxR Register

Timer_Ax Counter Register

Figure 25-17. TAxR Register

15	14	13	12	11	10	9	8
TAxR							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TAxR							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 25-5. TAxR Register Description

Bit	Field	Type	Reset	Description
15-0	TAxR	RW	0h	Timer_A register. The TAxR register is the count of Timer_A.

25.3.3 TAxCTLn Register

Timer_Ax Capture/Compare Control n Register

Figure 25-18. TAxCTLn Register

15	14	13	12	11	10	9	8
CM		CCIS		SCS	SCCI	Reserved	CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)	r-(0)	rw-(0)
7	6	5	4	3	2	1	0
OUTMOD			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

Table 25-6. TAxCTLn Register Description

Bit	Field	Type	Reset	Description
15-14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture on rising edge 10b = Capture on falling edge 11b = Capture on both rising and falling edges
13-12	CCIS	RW	0h	Capture/compare input select. These bits select the TAxCCR0 input signal. See the device-specific data sheet for specific signal connections. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0b = Asynchronous capture 1b = Synchronous capture
10	SCCI	RW	0h	Synchronized capture/compare input. The selected CCI input signal is latched with the EQUx signal and can be read from this bit.
9	Reserved	R	0h	Reserved. Reads as 0.
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMOD	RW	0h	Output mode. Modes 2, 3, 6, and 7 are not useful for TAxCCR0 because EQUx = EQU0. 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.
2	OUT	RW	0h	Output. For output mode 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high

Table 25-6. TAxCTLn Register Description (continued)

Bit	Field	Type	Reset	Description
1	COV	RW	0h	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software. 0b = No capture overflow occurred 1b = Capture overflow occurred
0	CCIFG	RW	0h	Capture/compare interrupt flag 0b = No interrupt pending 1b = Interrupt pending

25.3.4 TAxCCRn Register

Timer_A Capture/Compare n Register

Figure 25-19. TAxCCRn Register

15	14	13	12	11	10	9	8
TAxCCRn							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TAxCCRn							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 25-7. TAxCCRn Register Description

Bit	Field	Type	Reset	Description
15-0	TAxCCR0	RW	0h	Compare mode: TAxCCRn holds the data for the comparison to the timer value in the Timer_A Register, TAR. Capture mode: The Timer_A Register, TAR, is copied into the TAxCCRn register when a capture is performed.

25.3.5 TAxIV Register

Timer_Ax Interrupt Vector Register

Figure 25-20. TAxIV Register

15	14	13	12	11	10	9	8
TAIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
TAIV							
r0	r0	r0	r0	r-(0)	r-(0)	r-(0)	r0

Table 25-8. TAxIV Register Description

Bit	Field	Type	Reset	Description
15-0	TAIV	R	0h	Timer_A interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Capture/compare 1; Interrupt Flag: TAxCCR1 CCIFG; Interrupt Priority: Highest 04h = Interrupt Source: Capture/compare 2; Interrupt Flag: TAxCCR2 CCIFG 06h = Interrupt Source: Capture/compare 3; Interrupt Flag: TAxCCR3 CCIFG 08h = Interrupt Source: Capture/compare 4; Interrupt Flag: TAxCCR4 CCIFG 0Ah = Interrupt Source: Capture/compare 5; Interrupt Flag: TAxCCR5 CCIFG 0Ch = Interrupt Source: Capture/compare 6; Interrupt Flag: TAxCCR6 CCIFG 0Eh = Interrupt Source: Timer overflow; Interrupt Flag: TAxCTL TAIFG; Interrupt Priority: Lowest

25.3.6 TAxEX0 Register

Timer_Ax Expansion 0 Register

Figure 25-21. TAxEX0 Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved					TAIDEX ⁽¹⁾		
r0	r0	r0	r0	r0	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ After programming TAIDEX bits and configuration of the timer, set TACLR bit to ensure proper reset of the timer divider logic.

Table 25-9. TAxEX0 Register Description

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0h	Reserved. Reads as 0.
2-0	TAIDEX	RW	0h	Input divider expansion. These bits along with the ID bits select the divider for the input clock. 000b = Divide by 1 001b = Divide by 2 010b = Divide by 3 011b = Divide by 4 100b = Divide by 5 101b = Divide by 6 110b = Divide by 7 111b = Divide by 8

Timer_B

Timer_B is a 16-bit timer/counter with multiple capture/compare registers. There can be multiple Timer_B modules on a given device (see the device-specific data sheet). This chapter describes the operation and use of the Timer_B module.

Topic	Page
26.1 Timer_B Introduction	665
26.2 Timer_B Operation.....	667
26.3 Timer_B Registers	680

26.1 Timer_B Introduction

Timer_B is a 16-bit timer/counter with up to seven capture/compare registers. Timer_B can support multiple capture/compares, PWM outputs, and interval timing. Timer_B also has extensive interrupt capabilities. Interrupts may be generated from the counter on overflow conditions and from each of the capture/compare registers.

Timer_B features include:

- Asynchronous 16-bit timer/counter with four operating modes and four selectable lengths
- Selectable and configurable clock source
- Up to seven configurable capture/compare registers
- Configurable outputs with PWM capability
- Double-buffered compare latches with synchronized loading
- Interrupt vector register for fast decoding of all Timer_B interrupts

The block diagram of Timer_B is shown in [Figure 26-1](#).

NOTE: Use of the word *count*

Count is used throughout this chapter. It means the counter must be in the process of counting for the action to take place. If a particular value is directly written to the counter, an associated action does not take place.

NOTE: Nomenclature

There may be multiple instantiations of Timer_B on a given device. The prefix TBx is used, where x is a greater than equal to zero indicating the Timer_B instantiation. For devices with one instantiation, x = 0. The suffix n, where n = 0 to 6, represents the specific capture/compare registers associated with the Timer_B instantiation.

26.1.1 Similarities and Differences From Timer_A

Timer_B is identical to Timer_A with the following exceptions:

- The length of Timer_B is programmable to be 8, 10, 12, or 16 bits.
- Timer_B TBxCCRn registers are double-buffered and can be grouped.
- All Timer_B outputs can be put into a high-impedance state.
- The SCCI bit function is not implemented in Timer_B.

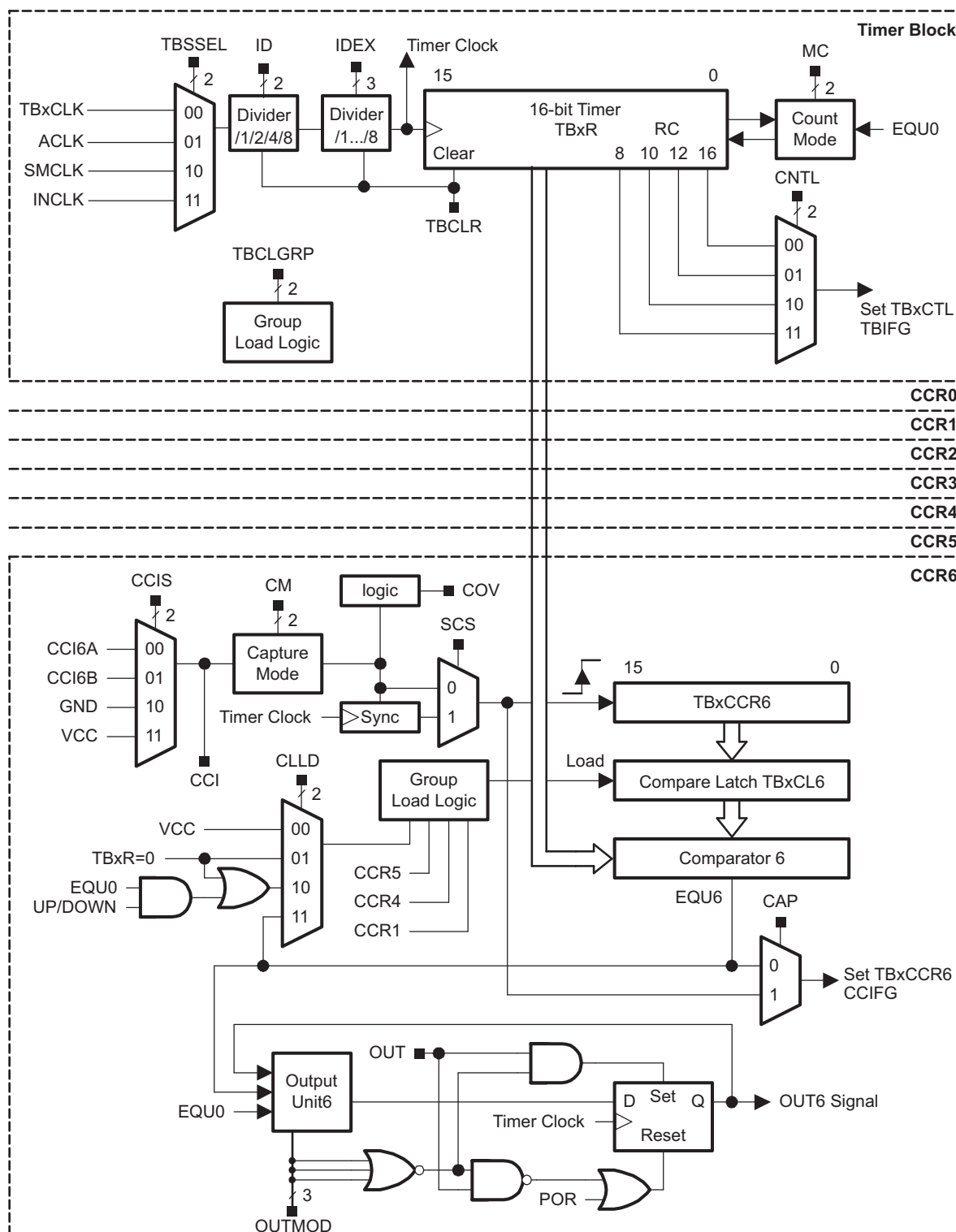


Figure 26-1. Timer_B Block Diagram

26.2 Timer_B Operation

The Timer_B module is configured with user software. The setup and operation of Timer_B is discussed in the following sections.

26.2.1 16-Bit Timer Counter

The 16-bit timer/counter register, TBxR, increments or decrements (depending on mode of operation) with each rising edge of the clock signal. TBxR can be read or written with software. Additionally, the timer can generate an interrupt when it overflows.

TBxR may be cleared by setting the TBCLR bit. Setting TBCLR also clears the clock divider counter logic (the divider setting remains unchanged) and count direction for up/down mode.

NOTE: Modifying Timer_B registers

TI recommends stopping the timer before modifying its operation (with exception of the interrupt enable, interrupt flag, and TBCLR) to avoid errant operating conditions.

When the timer clock is asynchronous to the CPU clock, any read from TBxR should occur while the timer is not operating or the results may be unpredictable. Alternatively, the timer may be read multiple times while operating, and a majority vote taken in software to determine the correct reading. Any write to TBxR takes effect immediately.

26.2.1.1 TBxR Length

Timer_B is configurable to operate as an 8-, 10-, 12-, or 16-bit timer with the CNTL bits. The maximum count value, TBxR_(max), for the selectable lengths is 0FFh, 03FFh, 0FFFh, and 0FFFFh, respectively. Data written to the TBxR register in 8-, 10-, and 12-bit mode is right justified with leading zeros.

26.2.1.2 Clock Source Select and Divider

The timer clock can be sourced from ACLK, SMCLK, or externally from TBxCCLK or INCLK. The clock source is selected with the TBSSEL bits. The selected clock source may be passed directly to the timer or divided by 2, 4, or 8, using the ID bits. The selected clock source can be further divided by 2, 3, 4, 5, 6, 7, or 8 using the TBIDEX bits. The timer clock divider logic is reset when TBCLR is set.

NOTE: Timer_B dividers

After programming ID or TBIDEX bits, set the TBCLR bit. This clears the contents of TBxR and resets the clock divider logic to a defined state. The clock dividers are implemented as down counters. Therefore, when the TBCLR bit is cleared, the timer clock immediately begins clocking at the first rising edge of the Timer_B clock source selected with the TBSSEL bits and continues clocking at the divider settings set by the ID and TBIDEX bits.

26.2.2 Starting the Timer

The timer may be started or restarted in the following ways:

- The timer counts when MC > { 0 } and the clock source is active.
- When the timer mode is either up or up/down, the timer may be stopped by loading 0 to TBxCLO. The timer may then be restarted by loading a nonzero value to TBxCLO. In this scenario, the timer starts incrementing in the up direction from zero.

26.2.3 Timer Mode Control

The timer has four modes of operation: stop, up, continuous, and up/down (see Table 26-1). The operating mode is selected with the MC bits.

Table 26-1. Timer Modes

MC	Mode	Description
00	Stop	The timer is halted.
01	Up	The timer repeatedly counts from zero to the value of compare register TBxCL0.
10	Continuous	The timer repeatedly counts from zero to the value selected by the CNTL bits.
11	Up/down	The timer repeatedly counts from zero up to the value of TBxCL0 and then back down to zero.

26.2.3.1 Up Mode

The up mode is used if the timer period must be different from $TBxR_{(max)}$ counts. The timer repeatedly counts up to the value of compare latch TBxCL0, which defines the period (see Figure 26-2). The number of timer counts in the period is TBxCL0 + 1. When the timer value equals TBxCL0, the timer restarts counting from zero. If up mode is selected when the timer value is greater than TBxCL0, the timer immediately restarts counting from zero.

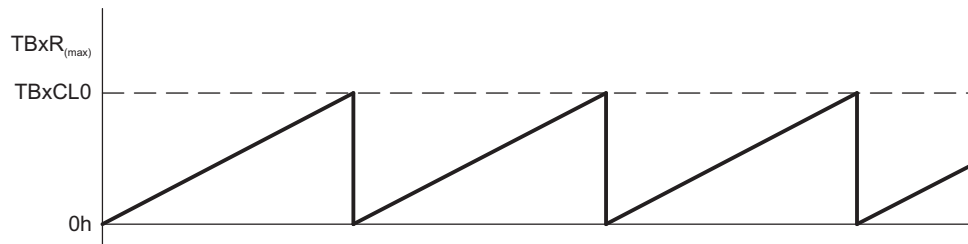


Figure 26-2. Up Mode

The TBxCCR0 CCIFG interrupt flag is set when the timer counts to the TBxCL0 value. The TBIFG interrupt flag is set when the timer counts from TBxCL0 to zero. Figure 26-3 shows the flag set cycle.

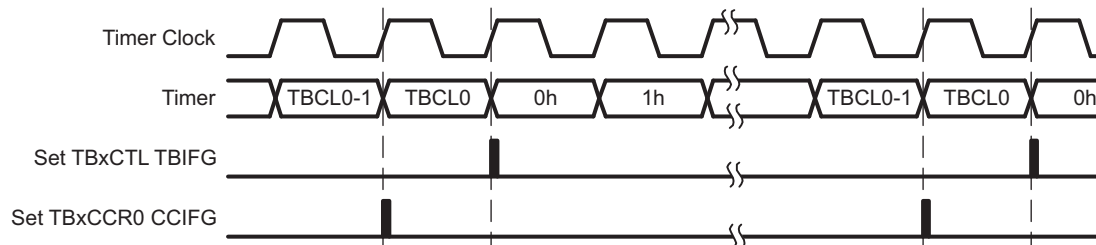


Figure 26-3. Up Mode Flag Setting

26.2.3.1.1 Changing Period Register TBxCL0

When changing TBxCL0 while the timer is running and when the TBxCL0 load mode is *immediate*, if the new period is greater than or equal to the old period or greater than the current count value, the timer counts up to the new period. If the new period is less than the current count value, the timer rolls to zero. However, one additional count may occur before the counter rolls to zero.

26.2.3.2 Continuous Mode

In continuous mode, the timer repeatedly counts up to $TBxR_{(max)}$ and restarts from zero (see Figure 26-4). The compare latch $TBxCL0$ works the same way as the other capture/compare registers.

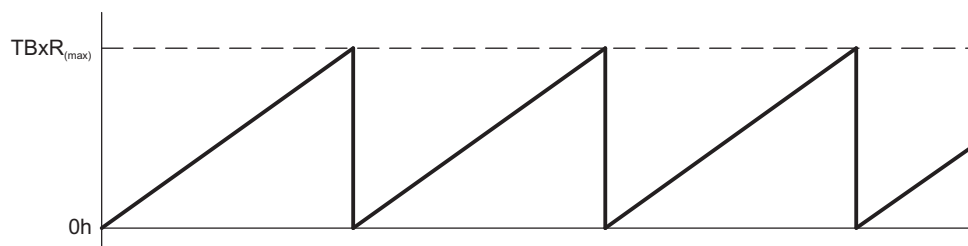


Figure 26-4. Continuous Mode

The TBIFG interrupt flag is set when the timer *counts* from $TBxR_{(max)}$ to zero. Figure 26-5 shows the flag set cycle.

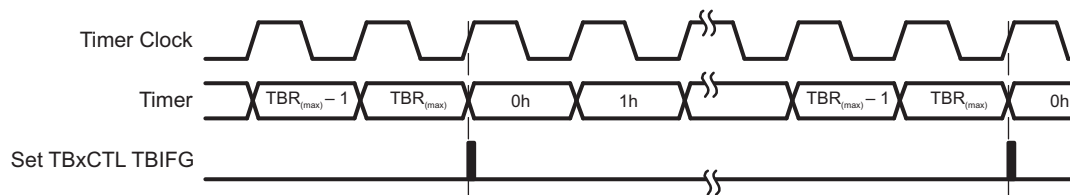


Figure 26-5. Continuous Mode Flag Setting

26.2.3.3 Use of Continuous Mode

The continuous mode can be used to generate independent time intervals and output frequencies. Each time an interval is completed, an interrupt is generated. The next time interval is added to the $TBxCLn$ latch in the interrupt service routine. Figure 26-6 shows two separate time intervals, t_0 and t_1 , being added to the capture/compare registers. The time interval is controlled by hardware, not software, without impact from interrupt latency. Up to n (where $n = 0$ to 7), independent time intervals or output frequencies can be generated using capture/compare registers.

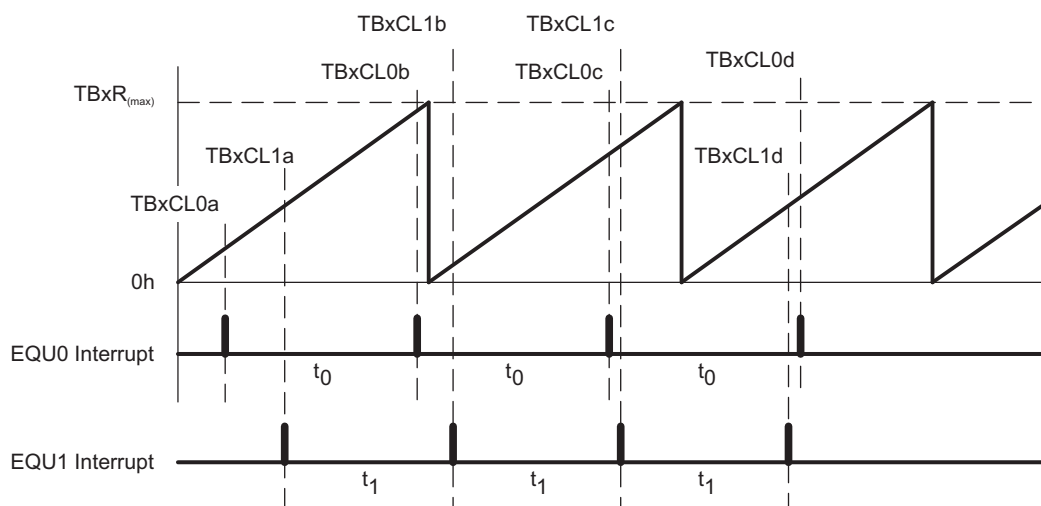


Figure 26-6. Continuous Mode Time Intervals

Time intervals can be produced with other modes as well, where TBxCL0 is used as the period register. Their handling is more complex, because the sum of the old TBxCLn data and the new period can be higher than the TBxCL0 value. When the sum of the previous TBxCLn value plus t_x is greater than the TBxCL0 data, the old TBxCL0 value must be subtracted to obtain the correct time interval.

26.2.3.4 Up/Down Mode

The up/down mode is used if the timer period must be different from TBxR_(max) counts and if symmetrical pulse generation is needed. The timer repeatedly counts up to the value of compare latch TBxCL0, and back down to zero (see Figure 26-7). The period is twice the value in TBxCL0.

NOTE: TBxCL0 > TBxR_(max)

If TBxCL0 > TBxR_(max), the counter operates as if it were configured for continuous mode. It does not count down from TBxR_(max) to zero.

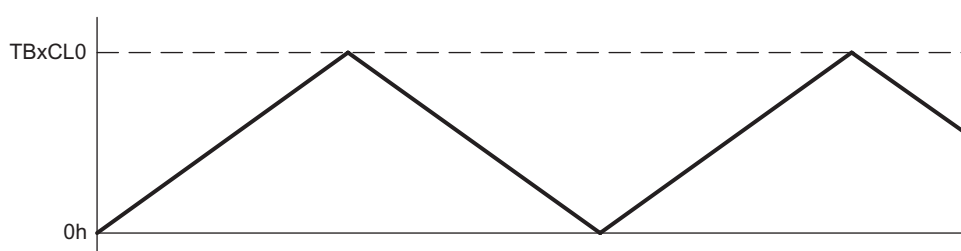


Figure 26-7. Up/Down Mode

The count direction is latched. This allows the timer to be stopped and then restarted in the same direction it was counting before it was stopped. If this is not desired, the TBCLR bit must be used to clear the direction. Setting TBCLR also clears the TBxR value and the clock divider counter logic (the divider setting remains unchanged).

In up/down mode, the TBxCCR0 CCIFG interrupt flag and the TBIFG interrupt flag are set only once during the period, separated by one-half the timer period. The TBxCCR0 CCIFG interrupt flag is set when the timer counts from TBxCL0-1 to TBxCL0, and TBIFG is set when the timer completes counting down from 0001h to 0000h. Figure 26-8 shows the flag set cycle.

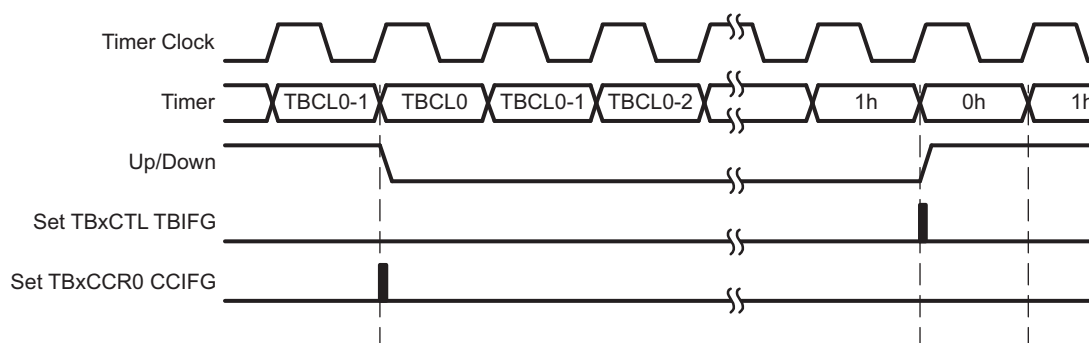


Figure 26-8. Up/Down Mode Flag Setting

26.2.3.4.1 Changing the Value of Period Register TBxCL0

When changing TBxCL0 while the timer is running and counting in the down direction, and when the TBxCL0 load mode is *immediate*, the timer continues its descent until it reaches zero. The new period takes effect after the counter counts down to zero.

If the timer is counting in the up direction when the new period is latched into TBxCL0, and the new period is greater than or equal to the old period or greater than the current count value, the timer counts up to the new period before counting down. When the timer is counting in the up direction, and the new period is less than the current count value when TBxCL0 is loaded, the timer begins counting down. However, one additional count may occur before the counter begins counting down.

26.2.3.5 Use of Up/Down Mode

The up/down mode supports applications that require dead times between output signals (see [Section 26.2.5](#)). For example, to avoid overload conditions, two outputs driving an H-bridge must never be in a high state simultaneously. In the example shown in [Figure 26-9](#), the t_{dead} is:

$$t_{dead} = t_{timer} \times (TBxCL1 - TBxCL3)$$

Where:

t_{dead} = Time during which both outputs need to be inactive

t_{timer} = Cycle time of the timer clock

TBxCLn = Content of compare latch n

The ability to simultaneously load grouped compare latches ensures the dead times.

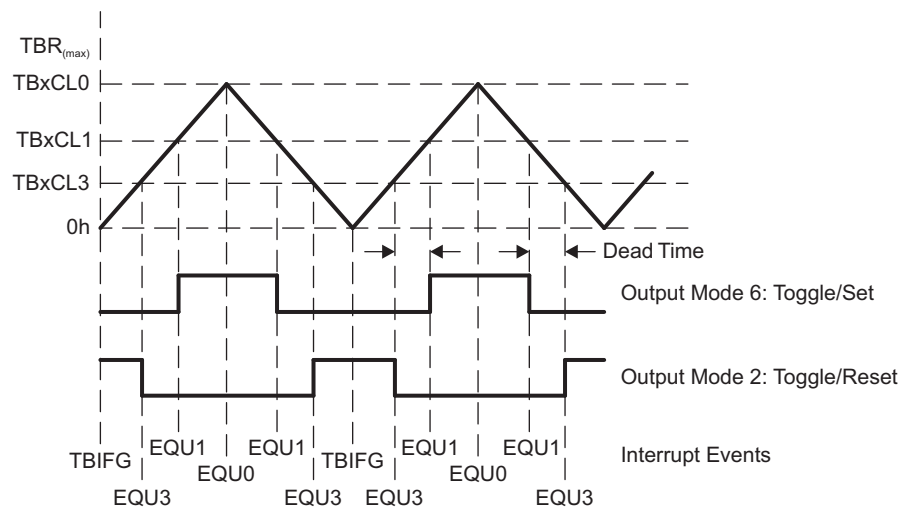


Figure 26-9. Output Unit in Up/Down Mode

26.2.4 Capture/Compare Blocks

Up to seven identical capture/compare blocks, TBxCCRn (where n = 0 to 6), are present in Timer_B. Any of the blocks may be used to capture the timer data or to generate time intervals.

26.2.4.1 Capture Mode

The capture mode is selected when CAP = 1. Capture mode is used to record time events. It can be used for speed computations or time measurements. The capture inputs CC1xA and CC1xB are connected to external pins or internal signals and are selected with the CCIS bits. The CM bits select the capture edge of the input signal as rising, falling, or both. A capture occurs on the selected edge of the input signal. If a capture is performed:

- The timer value is copied into the TBxCCRn register.
- The interrupt flag CCIFG is set.

The input signal level can be read at any time from the CCI bit. Devices may have different signals connected to CC1xA and CC1xB. See the device-specific data sheet for the connections of these signals.

The capture signal can be asynchronous to the timer clock and cause a race condition. Setting the SCS bit synchronizes the capture with the next timer clock. TI recommends setting the SCS bit to synchronize the capture signal with the timer clock (see [Figure 26-10](#)).

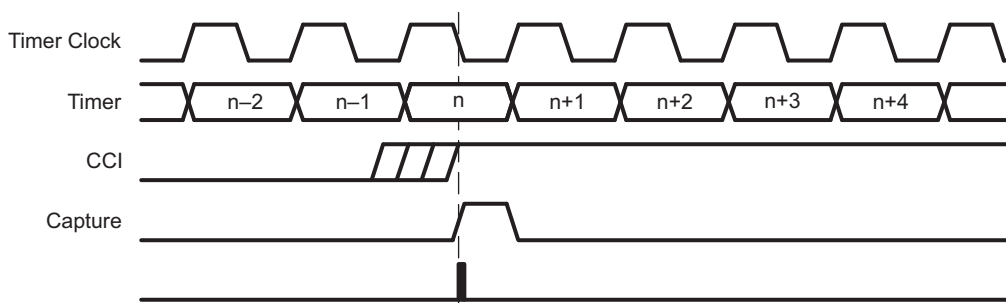


Figure 26-10. Capture Signal (SCS = 1)

NOTE: Changing Capture Inputs

Changing capture inputs while in capture mode may cause unintended capture events. To avoid this scenario, capture inputs should only be changed when capture mode is disabled (CM = {0} or CAP = 0).

Overflow logic is provided in each capture/compare register to indicate if a second capture was performed before the value from the first capture was read. Bit COV is set when this occurs (see [Figure 26-11](#)). COV must be reset with software.

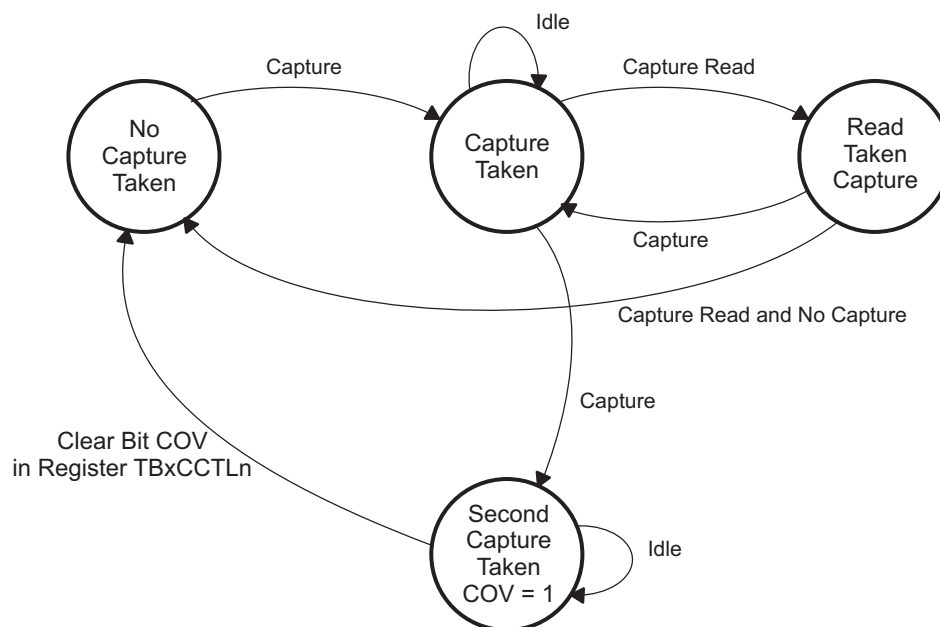


Figure 26-11. Capture Cycle

26.2.4.1.1 Capture Initiated by Software

Captures can be initiated by software. The CM bits can be set for capture on both edges. Software then sets bit CCIS1 = 1 and toggles bit CCIS0 to switch the capture signal between V_{CC} and GND, initiating a capture each time CCIS0 changes state:

```
MOV      #CAP+SCS+CCIS1+CM_3, &TB0CCTL1    ; Setup TB0CCTL1
XOR      #CCIS0, &TB0CCTL1                  ; TB0CCR1 = TB0R
```

NOTE: Capture Initiated by Software

In general, changing capture inputs while in capture mode may cause unintended capture events. For this scenario, switching the capture input between VCC and GND, disabling the capture mode is not required.

26.2.4.2 Compare Mode

The compare mode is selected when CAP = 0. Compare mode is used to generate PWM output signals or interrupts at specific time intervals. When TBxR *counts* to the value in a TBxCLn, where n represents the specific capture/compare latch:

- Interrupt flag CCIFG is set.
- Internal signal EQUn = 1.
- EQUn affects the output according to the output mode.

26.2.4.2.1 Compare Latch TBxCLn

The TBxCCRn compare latch, TBxCLn, holds the data for the comparison to the timer value in compare mode. TBxCLn is buffered by TBxCCRn. The buffered compare latch gives the user control over when a compare period updates. The user cannot directly access TBxCLn. Compare data is written to each TBxCCRn and automatically transferred to TBxCLn. The timing of the transfer from TBxCCRn to TBxCLn is user selectable, with the CLLD bits as described in [Table 26-2](#).

Table 26-2. TBxCLn Load Events

CLLD	Description
00	New data is transferred from TBxCCRn to TBxCLn immediately when TBxCCRn is written to.
01	New data is transferred from TBxCCRn to TBxCLn when TBxR <i>counts</i> to 0.
10	New data is transferred from TBxCCRn to TBxCLn when TBxR <i>counts</i> to 0 for up and continuous modes. New data is transferred to from TBxCCRn to TBxCLn when TBxR <i>counts</i> to the old TBxCL0 value or to 0 for up/down mode.
11	New data is transferred from TBxCCRn to TBxCLn when TBxR <i>counts</i> to the old TBxCLn value.

26.2.4.2.2 Grouping Compare Latches

Multiple compare latches may be grouped together for simultaneous updates with the TBCLGRP_x bits. When using groups, the CLLD bits of the lowest numbered TBxCCR_n in the group determine the load event for each compare latch of the group, except when TBCLGRP = 3 (see [Table 26-3](#)). The CLLD bits of the controlling TBxCCR_n must not be set to zero. When the CLLD bits of the controlling TBxCCR_n are set to zero, all compare latches update immediately when their corresponding TBxCCR_n is written; no compare latches are grouped.

Two conditions must exist for the compare latches to be loaded when grouped. First, all TBxCCR_n registers of the group must be updated, even when new TBxCCR_n data = old TBxCCR_n data. Second, the load event must occur.

Table 26-3. Compare Latch Operating Modes

TBCLGRP _x	Grouping	Update Control
00	None	Individual
01	TBxCL1+TBxCL2 TBxCL3+TBxCL4 TBxCL5+TBxCL6	TBxCCR1 TBxCCR3 TBxCCR5
10	TBxCL1+TBxCL2+TBxCL3 TBxCL4+TBxCL5+TBxCL6	TBxCCR1 TBxCCR4
11	TBxCL0+TBxCL1+TBxCL2+TBxCL3+TBxCL4+TBxCL5+TBxCL6	TBxCCR1

26.2.5 Output Unit

Each capture/compare block contains an output unit. The output unit is used to generate output signals, such as PWM signals. Each output unit has eight operating modes that generate signals based on the EQU0 and EQU_n signals. The TBOUTH pin function can be used to put all Timer_B outputs into a high-impedance state. When the TBOUTH pin function is selected for the pin (corresponding PSEL bit is set, and port configured as input) and when the pin is pulled high, all Timer_B outputs are in a high-impedance state.

26.2.5.1 Output Modes

The output modes are defined by the OUTMOD bits and are described in [Table 26-4](#). The OUT_n signal is changed with the rising edge of the timer clock for all modes except mode 0. Output modes 2, 3, 6, and 7 are not useful for output unit 0 because EQU_n = EQU0.

Table 26-4. Output Modes

OUTMOD	Mode	Description
000	Output	The output signal OUT _n is defined by the OUT bit. The OUT _n signal updates immediately when OUT is updated.
001	Set	The output is set when the timer <i>counts</i> to the TBxCL _n value. It remains set until a reset of the timer, or until another output mode is selected and affects the output.
010	Toggle/Reset	The output is toggled when the timer <i>counts</i> to the TBxCL _n value. It is reset when the timer <i>counts</i> to the TBxCL0 value.
011	Set/Reset	The output is set when the timer <i>counts</i> to the TBxCL _n value. It is reset when the timer <i>counts</i> to the TBxCL0 value.
100	Toggle	The output is toggled when the timer <i>counts</i> to the TBxCL _n value. The output period is double the timer period.
101	Reset	The output is reset when the timer <i>counts</i> to the TBxCL _n value. It remains reset until another output mode is selected and affects the output.
110	Toggle/Set	The output is toggled when the timer <i>counts</i> to the TBxCL _n value. It is set when the timer <i>counts</i> to the TBxCL0 value.
111	Reset/Set	The output is reset when the timer <i>counts</i> to the TBxCL _n value. It is set when the timer <i>counts</i> to the TBxCL0 value.

26.2.5.1.1 Output Example – Timer in Up Mode

The OUTn signal is changed when the timer *counts* up to the TBxCLn value, and rolls from TBxCL0 to zero, depending on the output mode. An example is shown in [Figure 26-12](#) using TBxCL0 and TBxCL1.

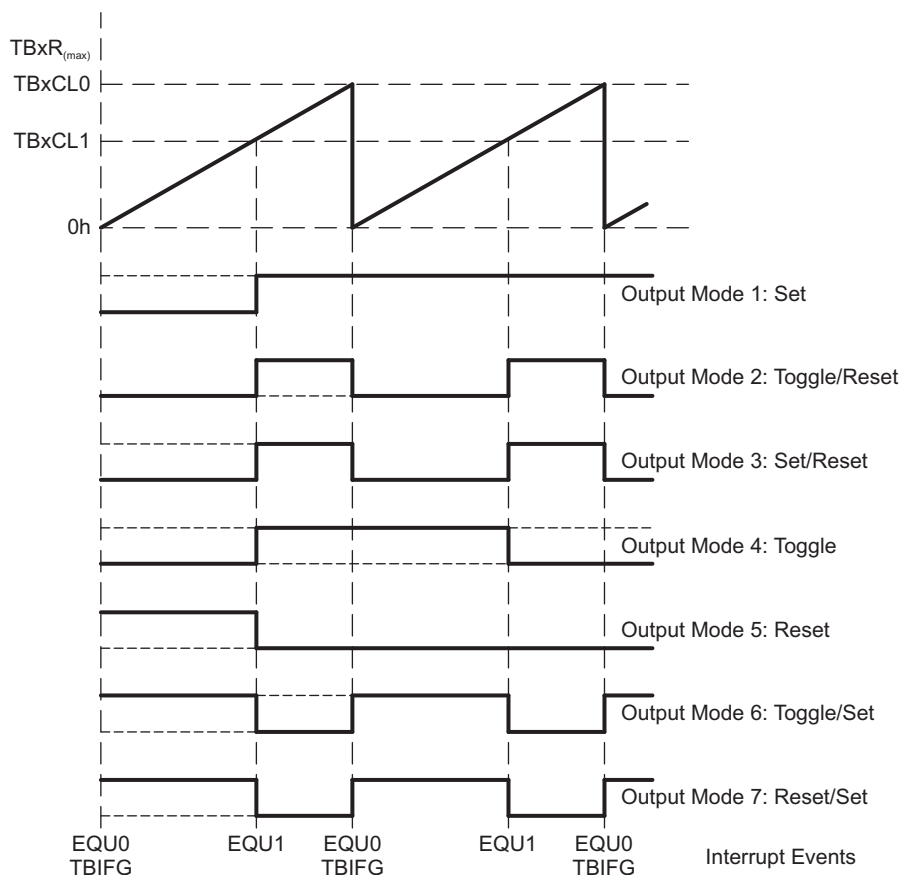


Figure 26-12. Output Example – Timer in Up Mode

26.2.5.1.2 Output Example – Timer in Continuous Mode

The OUTn signal is changed when the timer reaches the TBxCLn and TBxCL0 values, depending on the output mode. An example is shown in [Figure 26-13](#) using TBxCL0 and TBxCL1.

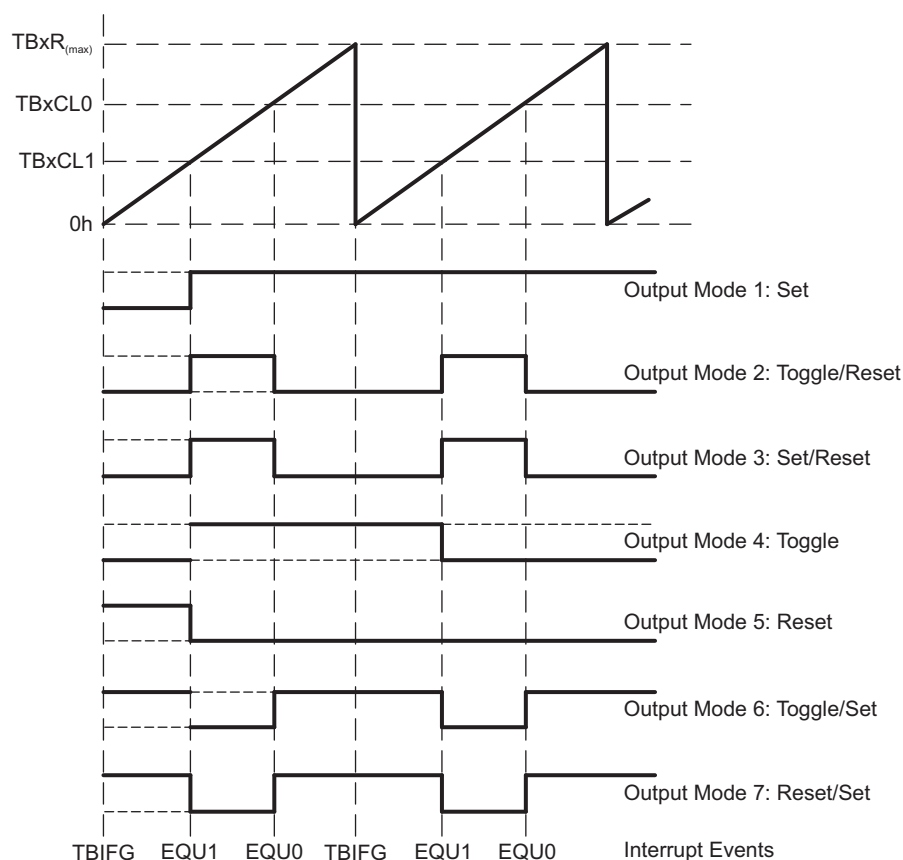


Figure 26-13. Output Example – Timer in Continuous Mode

26.2.5.1.3 Output Example – Timer in Up/Down Mode

The OUTn signal changes when the timer equals TBxCLn in either count direction and when the timer equals TBxCL0, depending on the output mode. An example is shown in [Figure 26-14](#) using TBxCL0 and TBxCL3.

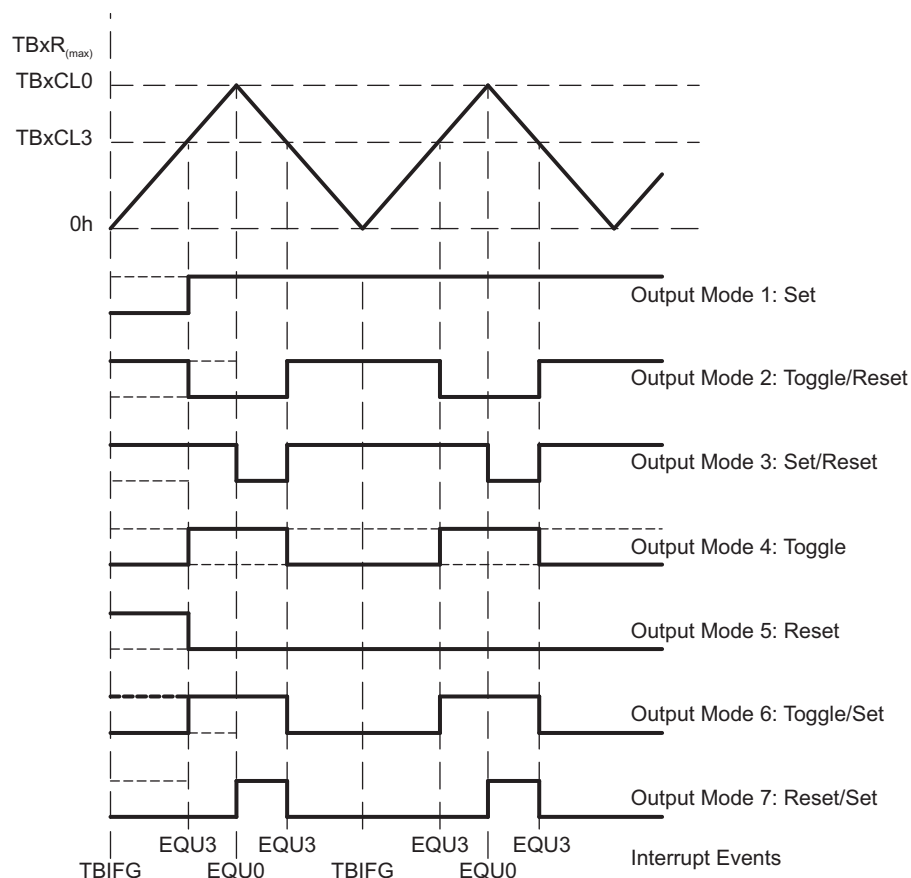


Figure 26-14. Output Example – Timer in Up/Down Mode

NOTE: Switching between output modes

When switching between output modes, one of the OUTMOD bits should remain set during the transition, unless switching to mode 0. Otherwise, output glitching can occur because a NOR gate decodes output mode 0. A safe method for switching between output modes is to use output mode 7 as a transition state:

```
BIS #OUTMOD_7,&TBCCTLx ; Set output mode=7
BIC #OUTMOD,&TBCCTLx ; Clear unwanted bits
```

26.2.6 Timer_B Interrupts

Two interrupt vectors are associated with the 16-bit Timer_B module:

- TBxCCR0 interrupt vector for TBxCCR0 CCIFG
- TBIV interrupt vector for all other CCIFG flags and TBIFG

In capture mode, any CCIFG flag is set when a timer value is captured in the associated TBxCCRn register. In compare mode, any CCIFG flag is set when TBxR counts to the associated TBxCLn value. Software may also set or clear any CCIFG flag. All CCIFG flags request an interrupt when their corresponding CCIE bit and the GIE bit are set.

26.2.6.1 TBxCCR0 Interrupt Vector

The TBxCCR0 CCIFG flag has the highest Timer_B interrupt priority and has a dedicated interrupt vector (see Figure 26-15). The TBxCCR0 CCIFG flag is automatically reset when the TBxCCR0 interrupt request is serviced.

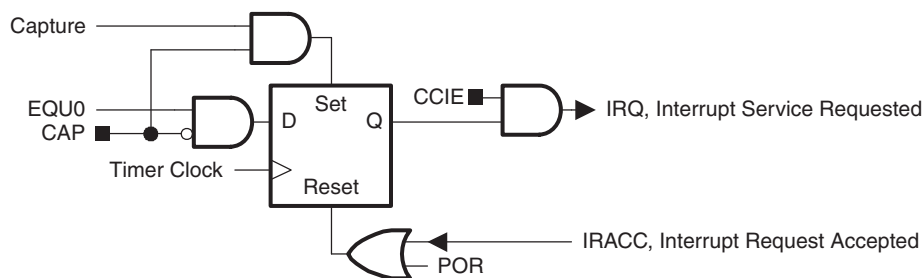


Figure 26-15. Capture/Compare TBxCCR0 Interrupt Flag

26.2.6.2 TBxIV, Interrupt Vector Generator

The TBIFG flag and TBxCCRn CCIFG flags (excluding TBxCCR0 CCIFG) are prioritized and combined to source a single interrupt vector. The interrupt vector register TBxIV is used to determine which flag requested an interrupt.

The highest-priority enabled interrupt (excluding TBxCCR0 CCIFG) generates a number in the TBxIV register (see register description). This number can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled Timer_B interrupts do not affect the TBxIV value.

Any access, read or write, of the TBxIV register automatically resets the highest-pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. For example, if the TBxCCR1 and TBxCCR2 CCIFG flags are set when the interrupt service routine accesses the TBxIV register, TBxCCR1 CCIFG is reset automatically. After the RETI instruction of the interrupt service routine is executed, the TBxCCR2 CCIFG flag generates another interrupt.

26.2.6.3 TBxIV, Interrupt Handler Examples

The following software example shows the recommended use of TBxIV and the handling overhead. The TBxIV value is added to the PC to automatically jump to the appropriate routine. The example assumes a single instantiation of the largest timer configuration available.

The numbers at the right margin show the necessary CPU clock cycles for each instruction. The software overhead for different interrupt sources includes interrupt latency and return-from-interrupt cycles, but not the task handling itself. The latencies are:

- Capture/compare block CCR0: 11 cycles
- Capture/compare blocks CCR1 to CCR6: 16 cycles
- Timer overflow TBIFG: 14 cycles

The following software example shows the recommended use of TBxIV for Timer_B3.

```

; Interrupt handler for TB0CCR0 CCIFG.                                Cycles
CCIFG_0_HND
;      ...      ; Start of handler Interrupt latency      6
;      RETI      5

; Interrupt handler for TB0IFG, TB0CCR1 through TB0CCR6 CCIFG.

TB0_HND      ...      ; Interrupt latency      6
      ADD      &TB0IV,PC      ; Add offset to Jump table      3
      RETI      ; Vector 0: No interrupt      5
      JMP      CCIFG_1_HND      ; Vector 2: TB0CCR1      2
      JMP      CCIFG_2_HND      ; Vector 4: TB0CCR2      2
      JMP      CCIFG_3_HND      ; Vector 6: TB0CCR3      2
      JMP      CCIFG_4_HND      ; Vector 8: TB0CCR4      2
      JMP      CCIFG_5_HND      ; Vector 10: TB0CCR5      2
      JMP      CCIFG_6_HND      ; Vector 12: TB0CCR6      2

TB0IFG_HND      ; Vector 14: TB0IFG Flag
      ...      ; Task starts here
      RETI      5

CCIFG_6_HND      ; Vector 12: TB0CCR6
      ...      ; Task starts here
      RETI      ; Back to main program      5

CCIFG_5_HND      ; Vector 10: TB0CCR5
      ...      ; Task starts here
      RETI      ; Back to main program      5

CCIFG_4_HND      ; Vector 8: TB0CCR4
      ...      ; Task starts here
      RETI      ; Back to main program      5

CCIFG_3_HND      ; Vector 6: TB0CCR3
      ...      ; Task starts here
      RETI      ; Back to main program      5

CCIFG_2_HND      ; Vector 4: TB0CCR2
      ...      ; Task starts here
      RETI      ; Back to main program      5

CCIFG_1_HND      ; Vector 2: TB0CCR1
      ...      ; Task starts here
      RETI      ; Back to main program      5

```

26.3 Timer_B Registers

The Timer_B registers are listed in [Table 26-5](#). The base address can be found in the device-specific data sheet. The address offset is listed in [Table 26-5](#).

Table 26-5. Timer_B Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	TBxCTL	Timer_B Control	Read/write	Word	0000h	Section 26.3.1
02h	TBxCCTL0	Timer_B Capture/Compare Control 0	Read/write	Word	0000h	Section 26.3.3
04h	TBxCCTL1	Timer_B Capture/Compare Control 1	Read/write	Word	0000h	Section 26.3.3
06h	TBxCCTL2	Timer_B Capture/Compare Control 2	Read/write	Word	0000h	Section 26.3.3
08h	TBxCCTL3	Timer_B Capture/Compare Control 3	Read/write	Word	0000h	Section 26.3.3
0Ah	TBxCCTL4	Timer_B Capture/Compare Control 4	Read/write	Word	0000h	Section 26.3.3
0Ch	TBxCCTL5	Timer_B Capture/Compare Control 5	Read/write	Word	0000h	Section 26.3.3
0Eh	TBxCCTL6	Timer_B Capture/Compare Control 6	Read/write	Word	0000h	Section 26.3.3
10h	TBxR	Timer_B Counter	Read/write	Word	0000h	Section 26.3.2
12h	TBxCCR0	Timer_B Capture/Compare 0	Read/write	Word	0000h	Section 26.3.4
14h	TBxCCR1	Timer_B Capture/Compare 1	Read/write	Word	0000h	Section 26.3.4
16h	TBxCCR2	Timer_B Capture/Compare 2	Read/write	Word	0000h	Section 26.3.4
18h	TBxCCR3	Timer_B Capture/Compare 3	Read/write	Word	0000h	Section 26.3.4
1Ah	TBxCCR4	Timer_B Capture/Compare 4	Read/write	Word	0000h	Section 26.3.4
1Ch	TBxCCR5	Timer_B Capture/Compare 5	Read/write	Word	0000h	Section 26.3.4
1Eh	TBxCCR6	Timer_B Capture/Compare 6	Read/write	Word	0000h	Section 26.3.4
2Eh	TBxIV	Timer_B Interrupt Vector	Read only	Word	0000h	Section 26.3.5
20h	TBxEX0	Timer_B Expansion 0	Read/write	Word	0000h	Section 26.3.6

26.3.1 TBxCTL Register

Timer_B x Control Register

Figure 26-16. TBxCTL Register

15	14	13	12	11	10	9	8
Reserved	TBCLGRP _x		CNTL		Reserved	TBSSEL	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ID	MC		Reserved	TBCLR	TBIE	TBIFG	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	w-(0)	rw-(0)	rw-(0)

Table 26-6. TBxCTL Register Description

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved. Always reads as 0.
14-13	TBCLGRP	RW	0h	TBxCL _n group 00b = Each TBxCL _n latch loads independently. 01b = TBxCL1+TBxCL2 (TBxCCR1 CLLD bits control the update); TBxCL3+TBxCL4 (TBxCCR3 CLLD bits control the update); TBxCL5+TBxCL6 (TBxCCR5 CLLD bits control the update); TBxCL0 independent 10b = TBxCL1+TBxCL2+TBxCL3 (TBxCCR1 CLLD bits control the update); TBxCL4+TBxCL5+TBxCL6 (TBxCCR4 CLLD bits control the update); TBxCL0 independent 11b = TBxCL0+TBxCL1+TBxCL2+TBxCL3+TBxCL4+TBxCL5+TBxCL6 (TBxCCR1 CLLD bits control the update)
12-11	CNTL	RW	0h	Counter length 00b = 16-bit, TBxR(max) = 0FFFFh 01b = 12-bit, TBxR(max) = 0FFFh 10b = 10-bit, TBxR(max) = 03FFh 11b = 8-bit, TBxR(max) = 0FFh
10	Reserved	R	0h	Reserved. Always reads as 0.
9-8	TBSSEL	RW	0h	Timer_B clock source select 00b = TBxCLK 01b = ACLK 10b = SMCLK 11b = INCLK
7-6	ID	RW	0h	Input divider. These bits, along with the TBIDEX bits, select the divider for the input clock. 00b = /1 01b = /2 10b = /4 11b = /8
5-4	MC	RW	0h	Mode control. Setting MC = 00h when Timer_B is not in use conserves power. 00b = Stop mode: Timer is halted 01b = Up mode: Timer counts up to TBxCL0 10b = Continuous mode: Timer counts up to the value set by CNTL 11b = Up/down mode: Timer counts up to TBxCL0 and down to 0000h
3	Reserved	R	0h	Reserved. Always reads as 0.
2	TBCLR	RW	0h	Timer_B clear. Setting this bit clears TBR, the clock divider logic (the divider setting remains unchanged), and the count direction. The TBCLR bit is automatically reset and is always read as zero.
1	TBIE	RW	0h	Timer_B interrupt enable. This bit enables the TBIFG interrupt request. 0b = Interrupt disabled 1b = Interrupt enabled

Table 26-6. TBxCTL Register Description (continued)

Bit	Field	Type	Reset	Description
0	TBIFG	RW	0h	Timer_B interrupt flag 0b = No interrupt pending 1b = Interrupt pending

26.3.2 TBxR Register

Timer_B x Counter Register

Figure 26-17. TBxR Register

15	14	13	12	11	10	9	8
TBxR							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TBxR							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 26-7. TBxR Register Description

Bit	Field	Type	Reset	Description
15-0	TBxR	RW	0h	Timer_B register. The TBxR register is the count of Timer_B.

26.3.3 TBxCCTLn Register

Timer_B x Capture/Compare Control Register n

Figure 26-18. TBxCCTLn Register

15	14	13	12	11	10	9	8
CM		CCIS		SCS	CLLD		CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
OUTMOD			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

Table 26-8. TBxCCTLn Register Description

Bit	Field	Type	Reset	Description
15-14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture on rising edge 10b = Capture on falling edge 11b = Capture on both rising and falling edges
13-12	CCIS	RW	0h	Capture/compare input select. These bits select the TBxCCRn input signal. See the device-specific data sheet for specific signal connections. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0b = Asynchronous capture 1b = Synchronous capture
10-9	CLLD	RW	0h	Compare latch load. These bits select the compare latch load event. 00b = TBxCLn loads on write to TBxCCRn 01b = TBxCLn loads when TBxR counts to 0 10b = TBxCLn loads when TBxR counts to 0 (up or continuous mode). TBxCLn loads when TBxR counts to TBxCL0 or to 0 (up/down mode). 11b = TBxCLn loads when TBxR counts to TBxCLn
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMOD	RW	0h	Output mode. Modes 2, 3, 6, and 7 are not useful for TBxCL0 because EQU0 = EQU0. 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	Undef	Capture/compare input. The selected input signal can be read by this bit.

Table 26-8. TBxCCTLn Register Description (continued)

Bit	Field	Type	Reset	Description
2	OUT	RW	0h	Output. For output mode 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high
1	COV	RW	0h	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software. 0b = No capture overflow occurred 1b = Capture overflow occurred
0	CCIFG	RW	0h	Capture/compare interrupt flag 0b = No interrupt pending 1b = Interrupt pending

26.3.4 TBxCCRn Register

Timer_B x Capture/Compare Register n

Figure 26-19. TBxCCRn Register

15	14	13	12	11	10	9	8
TBxCCRn							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TBxCCRn							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 26-9. TBxCCRn Register Description

Bit	Field	Type	Reset	Description
15-0	TBxCCRn	RW	0h	Timer_B capture/compare register. Compare mode: TBxCCRn holds the data for the comparison to the timer value in the Timer_B Register, TBR. Capture mode: The Timer_B Register, TBR, is copied into the TBxCCRn register when a capture is performed.

26.3.5 TBxIV Register

Timer_B x Interrupt Vector Register

Figure 26-20. TBxIV Register

15	14	13	12	11	10	9	8
TBIV							
r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)
7	6	5	4	3	2	1	0
TBIV							
r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)

Table 26-10. TBxIV Register Description

Bit	Field	Type	Reset	Description
15-0	TBIV	R	0h	Timer_B interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Capture/compare 1; Interrupt Flag: TBxCCR1 CCIFG; Interrupt Priority: Highest 04h = Interrupt Source: Capture/compare 2; Interrupt Flag: TBxCCR2 CCIFG 06h = Interrupt Source: Capture/compare 3; Interrupt Flag: TBxCCR3 CCIFG 08h = Interrupt Source: Capture/compare 4; Interrupt Flag: TBxCCR4 CCIFG 0Ah = Interrupt Source: Capture/compare 5; Interrupt Flag: TBxCCR5 CCIFG 0Ch = Interrupt Source: Capture/compare 6; Interrupt Flag: TBxCCR6 CCIFG 0Eh = Interrupt Source: Timer overflow; Interrupt Flag: TBxCTL TBIFG; Interrupt Priority: Lowest

26.3.6 TBxEX0 Register

Timer_B x Expansion Register 0

Figure 26-21. TBxEX0 Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved					TBIDEX ⁽¹⁾		
r0	r0	r0	r0	r0	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ After programming TBIDEX bits and configuration of the timer, set TBCLR bit to ensure proper reset of the timer divider logic.

Table 26-11. TBxEX0 Register Description

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0h	Reserved. Always reads as 0.
2-0	TBIDEX	RW	0h	Input divider expansion. These bits along with the ID bits select the divider for the input clock. 000b = Divide by 1 001b = Divide by 2 010b = Divide by 3 011b = Divide by 4 100b = Divide by 5 101b = Divide by 6 110b = Divide by 7 111b = Divide by 8

Real-Time Clock (RTC) Overview

27.1 RTC Overview

Table 27-1. RTC Overview

Feature	RTC_B LPM3.5, Calendar Mode Only	RTC_C Protection + Improved Calibration and Compensation
Calendar Mode	Yes	Yes
Counter Mode	No	Optional (device-dependent) ⁽¹⁾
Programmable Alarms	Yes	Yes
Password Protected Calendar Registers	No	Yes
Input Clocks	32-kHz crystal oscillator	32-kHz crystal oscillator
LPM3.5 Support	Yes	Yes
Offset Calibration Register	Yes	Yes
Temperature Compensation Register	No	Yes
Frequency Adjustment Range	-2.17 ppm × 59 ≈ -128 ppm +4.34 ppm × 59 ≈ +256 ppm	-240 ppm, +240 ppm ⁽²⁾
Frequency Adjustment Steps	-2.17 ppm, +4.34 ppm	-1 ppm, +1 ppm
Temperature Compensation	With software, manipulating offset calibration value	With software using separate temperature compensation register
Calibration and Compensation Period	60 min	1 min
BCD to Binary Conversion	Integrated for Calendar Mode plus separate conversion registers	Integrated for Calendar Mode plus separate conversion registers
Event/Tamper Detect with Time Stamp	No	Optional (device-dependent) ⁽¹⁾

⁽¹⁾ See the device-specific data sheet.

⁽²⁾ Total adjustment range of offset calibration plus temperature compensation. See the RTC_C chapter for details.

Real-Time Clock B (RTC_B)

The real-time clock RTC_B module provides clock counters with calendar mode, a flexible programmable alarm, and calibration. Note that the RTC_B supports only calendar mode and not counter mode. The RTC_B also support operation in LPM3.5. See the device-specific data sheet for the supported features. This chapter describes the RTC_B module.

Topic	Page
28.1 Real-Time Clock RTC_B Introduction	691
28.2 RTC_B Operation.....	693
28.3 RTC_B Registers	698

28.1 Real-Time Clock RTC_B Introduction

The RTC_B module provides configurable clock counters.

RTC_B features include:

- Real-time clock and calendar mode providing seconds, minutes, hours, day of week, day of month, month, and year (including leap year correction)

Note that only the calendar mode is supported by RTC_B; the counter mode that is available in some other RTC modules is not supported.

- Interrupt capability
- Selectable BCD or binary format
- Programmable alarms
- Calibration logic for time offset correction
- Operation in LPM3.5

The RTC_B block diagram for devices supporting LPM3.5 is shown in [Figure 28-1](#).

NOTE: Real-time clock initialization

Most RTC_B module registers have no initial condition. These registers must be configured by user software before use.

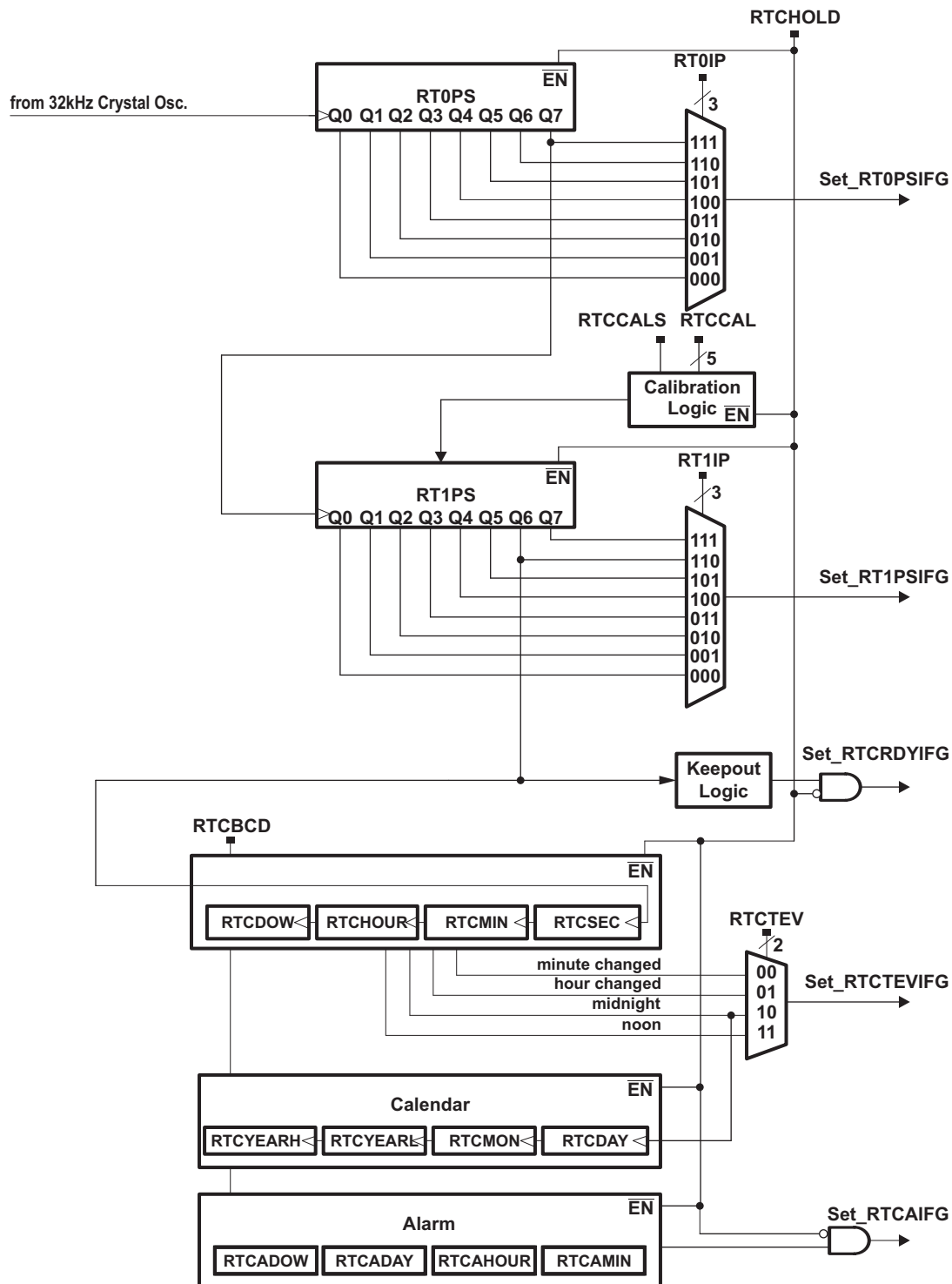


Figure 28-1. RTC_B Block Diagram

28.2 RTC_B Operation

The RTC_B module provides seconds, minutes, hours, day of week, day of month, month, and year in selectable BCD or hexadecimal format. The calendar includes a leap-year algorithm that considers all years evenly divisible by four as leap years. This algorithm is accurate from the year 1901 through 2099.

28.2.1 Real-Time Clock and Prescale Dividers

The prescale dividers, RT0PS and RT1PS, are automatically configured to provide a 1-s clock interval for the RTC_B. The low-frequency oscillator must be operated at 32768 Hz (nominal) for proper RTC_B operation. RT0PS is sourced from the low-frequency oscillator XT1. The output of RT0PS / 256 (Q7) is used to source RT1PS. RT1PS is further divider and the /128 output sources the real-time clock counter registers providing the required 1-second time interval.

When RTCBCD = 1, BCD format is selected for the calendar registers. It is possible to switch between BCD and hexadecimal format while the RTC is counting.

Setting RTCHOLD halts the real-time counters and prescale counters, RT0PS, and RT1PS.

28.2.2 Real-Time Clock Alarm Function

The RTC_B module provides for a flexible alarm system. There is a single user-programmable alarm that can be programmed based on the settings contained in the alarm registers for minutes, hours, day of week, and day of month.

Each alarm register contains an alarm enable (AE) bit that can be used to enable the respective alarm register. By setting AE bits of the various alarm registers, a variety of alarm events can be generated.

- Example 1: A user wishes to set an alarm every hour at 15 minutes past the hour (that is, at 00:15:00, 01:15:00, 02:15:00, etc). This is possible by setting RTCAMIN to 15. By setting the AE bit of the RTCAMIN and clearing all other AE bits of the alarm registers, the alarm is enabled. When enabled, the RTCAIFG is set when the count transitions from 00:14:59 to 00:15:00, 01:14:59 to 01:15:00, 02:14:59 to 02:15:00, and so on.
- Example 2: A user wishes to set an alarm every day at 04:00:00. This is possible by setting RTCAHOUR to 4. By setting the AE bit of the RTCAHOUR and clearing all other AE bits of the alarm registers, the alarm is enabled. When enabled, the RTCAIFG is set when the count transitions from 03:59:59 to 04:00:00.
- Example 3: A user wishes to set an alarm for 06:30:00. RTCAHOUR would be set to 6 and RTCAMIN would be set to 30. By setting the AE bits of RTCAHOUR and RTCAMIN, the alarm is enabled. Once enabled, the RTCAIFG is set when the time count transitions from 06:29:59 to 06:30:00. In this case, the alarm event occurs every day at 06:30:00.
- Example 4: A user wishes to set an alarm every Tuesday at 06:30:00. RTCADOW would be set to 2, RTCAHOUR would be set to 6, and RTCAMIN would be set to 30. By setting the AE bits of RTCADOW, RTCAHOUR, and RTCAMIN, the alarm is enabled. Once enabled, the RTCAIFG is set when the time count transitions from 06:29:59 to 06:30:00 and the RTCDOW transitions from 1 to 2.
- Example 5: A user wishes to set an alarm the fifth day of each month at 06:30:00. RTCADAY would be set to 5, RTCAHOUR would be set to 6, and RTCAMIN would be set to 30. By setting the AE bits of RTCADAY, RTCAHOUR, and RTCAMIN, the alarm is enabled. Once enabled, the RTCAIFG is set when the time count transitions from 06:29:59 to 06:30:00 and the RTCDAY equals 5.

NOTE: Setting the alarm

Before setting an initial alarm, all alarm registers including the AE bits should be cleared.

To prevent potential erroneous alarm conditions from occurring, the alarms should be disabled by clearing the RTCAIE, RTCAIFG, and AE bits before writing initial or new time values to the RTC time registers.

NOTE: Invalid alarm settings

Invalid alarm settings are not checked by hardware. It is the user's responsibility that valid alarm settings are entered.

NOTE: Invalid time and date values

Writing of invalid date or time information or data values outside the legal ranges specified in the RTCSEC, RTCMIN, RTCHOUR, RTCDAY, RTCDOW, RTCYEAR, RTCAMIN, RTCAHOUR, RTCADAY, and RTCADOW registers can result in unpredictable behavior.

28.2.3 Reading or Writing Real-Time Clock Registers

Because the system clock may in fact be asynchronous to the RTC_B clock source, special care must be used when accessing the real-time clock registers.

The real-time clock registers are updated once per second. To prevent reading any real-time clock register at the time of an update, which could result in an invalid time being read, a keep-out window is provided. The keep-out window is centered approximately 128/32768 seconds around the update transition. The read-only RTCRDY bit is reset during the keep-out window period and set outside the keep-out the window period. Any read of the clock registers while RTCRDY is reset is considered to be potentially invalid, and the time read should be ignored.

An easy way to safely read the real-time clock registers is to utilize the RTCRDYIFG interrupt flag. Setting RTCRDYIE enables the RTCRDYIFG interrupt. Once enabled, an interrupt is generated based on the rising edge of the RTCRDY bit, causing the RTCRDYIFG to be set. At this point, the application has nearly a complete second to safely read any or all of the real-time clock registers. This synchronization process prevents reading the time value during transition. The RTCRDYIFG flag is reset automatically when the interrupt is serviced, or it can be reset with software.

NOTE: Reading or writing real-time clock registers

When the counter clock is asynchronous to the CPU clock, any read from any RTCSEC, RTCMIN, RTCHOUR, RTCDOW, RTCDAY, RTCMON, or RTCYEAR register while the RTCRDY is reset may result in invalid data being read. To safely read the counting registers, either polling of the RTCRDY bit or the synchronization procedure previously described can be used. Alternatively, the counter register can be read multiple times while operating, and a majority vote taken in software to determine the correct reading. Reading the RT0PS and RT1PS can only be handled by reading the registers multiple times and a majority vote taken in software to determine the correct reading.

Any write to any counting register takes effect immediately. However, the clock is stopped during the write. In addition, RT0PS and RT1PS registers are reset. This could result in losing up to 1 second during a write. Writing of data outside the legal ranges or invalid time stamp combinations results in unpredictable behavior.

28.2.4 Real-Time Clock Interrupts

Six sources for interrupts are available, namely RT0PSIFG, RT1PSIFG, RTCRDYIFG, RTCTEVIFG, RTCAIFG, and RTCOFIFG. These flags are prioritized and combined to source a single interrupt vector. The interrupt vector register (RTCIV) is used to determine which flag requested an interrupt.

The highest-priority enabled interrupt generates a number in the RTCIV register (see register description). This number can be evaluated or added to the program counter (PC) to automatically enter the appropriate software routine. Disabled RTC interrupts do not affect the RTCIV value.

Any access, read or write, of the RTCIV register automatically resets the highest-pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. In addition, all flags can be cleared by software.

The user-programmable alarm event sources the real-time clock interrupt, RTCAIFG. Setting RTCAIE enables the interrupt. In addition to the user-programmable alarm, the RTC_B module provides for an interval alarm that sources real-time clock interrupt, RTCTEVIFG. The interval alarm can be selected to cause an alarm event when RTCMIN changed or RTCHOUR changed, every day at midnight (00:00:00) or every day at noon (12:00:00). The event is selectable with the RTCTEV bits. Setting the RTCTEVIE bit enables the interrupt.

The RTCRDY bit sources the real-time clock interrupt, RTCRDYIFG, and is useful in synchronizing the read of time registers with the system clock. Setting the RTCRDYIE bit enables the interrupt.

RT0PSIFG can be used to generate interrupt intervals selectable by the RT0IP bits. RT0PS is sourced with low-frequency oscillator clock at 32768 Hz, so intervals of 16384 Hz, 8192 Hz, 4096 Hz, 2048 Hz, 1024 Hz, 512 Hz, 256 Hz, or 128 Hz are possible. Setting the RT0PSIE bit enables the interrupt.

RT1PSIFG can be used to generate interrupt intervals selectable by the RT1IP bits. RT1PS is sourced with the output of RT0PS, which is 128 Hz (32768/256 Hz). Therefore, intervals of 64 Hz, 32 Hz, 16 Hz, 8 Hz, 4 Hz, 2 Hz, 1 Hz, or 0.5 Hz are possible. Setting the RT1PSIE bit enables the interrupt.

NOTE: Changing RT0IP or RT1IP

Changing the settings of the interrupt interval bits RT0IP or RT1IP while the corresponding prescaler is running or is stopped in a non-zero state can result in setting the corresponding interrupt flags.

The RTCOFIFG bit flags a failure of the 32-kHz crystal oscillator. Its main purpose is to wake up the CPU from LPM3.5 if an oscillator failure occurs.

28.2.4.1 RTCIV Software Example

The following software example shows the recommended use of RTCIV and the handling overhead. The RTCIV value is added to the PC to automatically jump to the appropriate routine.

The numbers at the right margin show the necessary CPU cycles for each instruction. The software overhead for different interrupt sources includes interrupt latency and return-from-interrupt cycles, but not the task handling itself.

; Interrupt handler for RTC interrupt flags.

```

RTC_HND                ; Interrupt latency        6
    ADD &RTCIV,PC       ; Add offset to Jump table 3
    RETI                ; Vector 0: No interrupt   5
    JMP RTCRDYIFG_HND   ; Vector 2: RTCRDYIFG      2
    JMP RTCTEVIFG_HND   ; Vector 4: RTCTEVIFG      2
    JMP RTCAIFG_HND     ; Vector 6: RTCAIFG        5
    JMP RT0PSIFG_HND    ; Vector 8: RT0PSIFG      5
    JMP RT1PSIFG_HND    ; Vector A: RT1PSIFG      5
    JMP RTCOFIFG_HND    ; Vector C: RTCOFIFG      5
    RETI                ; Vector E: Reserved      5

RTCRDYIFG_HND          ; Vector 2: RTCRDYIFG Flag
    ...                ; Task starts here
    RETI                ; Back to main program    5

RTCTEVIFG_HND          ; Vector 4: RTCTEVIFG Flag
    ...                ; Task starts here
    RETI                ; Back to main program    5

RTCAIFG_HND            ; Vector 6: RTCAIFG Flag
    ...                ; Task starts here
    RETI                ; Back to main program    5

RT0PSIFG_HND           ; Vector 8: RT0PSIFG Flag

```

```

...                ; Task starts here
RETI               ; Back to main program          5

RT1PSIFG_HND       ; Vector A: RT1PSIFG Flag
...                ; Task starts here
RETI               ; Back to main program          5

RTC0FIFG_HND       ; Vector C: RTC0FIFG Flag
...                ; Task starts here
RETI               ; Back to main program          5

```

28.2.5 Real-Time Clock Calibration

The RTC_B module has calibration logic that allows for adjusting the crystal frequency in approximately +4-ppm or -2-ppm steps, allowing for higher time keeping accuracy from standard crystals. The RTCCALx bits are used to adjust the frequency. When RTCCALS is set, each RTCCALx LSB causes an approximately +4-ppm adjustment. When RTCCALS is cleared, each RTCCALx LSB causes an approximately -2-ppm adjustment.

Calibration is accomplished by periodically adjusting the RT1PS counter based on the RTCCALS and RTCCALx settings. The RT0PS divides the nominal 32768-Hz low-frequency (LF) crystal clock input by 256. A 60-minute period has 32768 cycles/sec × 60 sec/min × 60 min = 117964800 cycles. Therefore, a -2-ppm reduction in frequency (down calibration) approximately equates to adding an additional 256 cycles every 117964800 cycles (256/117964800 = 2.17 ppm). This is accomplished by holding the RT1PS counter for one additional clock of the RT0PS output within a 60-minute period. Similarly, a +4-ppm increase in frequency (up calibration) approximately equates to removing 512 cycles every 117964800 cycle (512/117964800 = 4.34 ppm). This is accomplished by incrementing the RT1PS counter for two additional clocks of the RT0PS output within a 60-minute period. Each RTCCALx calibration bit causes either 256 LF crystal clock cycles to be added every 60 minutes or 512 LF crystal clock cycles to be subtracted every 60 minutes, giving a frequency adjustment of approximately -2 ppm or +4 ppm, respectively.

To calibrate the frequency, the RTCCLK output signal is available at a pin. RTCCALF bits can be used to select the frequency rate of the output signal, either no signal, 512 Hz, 256 Hz, or 1 Hz.

The basic flow to calibrate the frequency is as follows:

1. Configure the RTCCLK pin.
2. Measure the RTCCLK output signal with an appropriate resolution frequency counter; that is, within the resolution required.
3. Compute the absolute error in ppm: Absolute Error (ppm) = $10^6 \times (f_{\text{MEASURED}} - f_{\text{RTCCLK}}) / f_{\text{RTCCLK}}$, where f_{RTCCLK} is the expected frequency of 512 Hz, 256 Hz, or 1 Hz.
4. Adjust the frequency:
 - a. If the frequency is too low, set RTCCALS = 1 and apply the appropriate RTCCALx bits, where $\text{RTCCALx} = ((\text{Absolute Error}) / 4.069) - 1$, rounded to the nearest integer.
 - b. If the frequency is too high, clear RTCCALS = 0 and apply the appropriate RTCCALx bits, where $\text{RTCCALx} = ((\text{Absolute Error}) / 2.035) - 1$, rounded to the nearest integer.

For example, assume that RTCCLK is output at a frequency of 512 Hz. The measured RTCCLK is 511.9658 Hz. The frequency error is approximately 66.8 ppm low. To increase the frequency by 66.8 ppm, RTCCALS would be set, and RTCCAL would be set to 15 $((66.8 / 4.069) - 1)$. Similarly, assume that the measured RTCCLK is 512.0125 Hz. The frequency error is approximately 24.4 ppm high. To decrease the frequency by 24.4 ppm, RTCCALS would be cleared, and RTCCAL would be set to 11 $((24.4 / 2.035) - 1)$.

The calibration corrects only initial offsets and does not adjust for temperature and aging effects. These effects can be managed by periodically measuring the temperature and using the crystal's characteristic curve to adjust the ppm based on temperature as required. In counter mode (RTCMODE = 0), the calibration logic is disabled.

NOTE: Minimum possible calibration

For RTC calibration, RTCCAL+1 is used. The minimal calibration possible is -4 ppm or +8 ppm (when RTCCAL = 1h). For example, setting RTCCALS = 0 and RTCCAL = 0h results in a -4 ppm decrease in frequency. Similarly, setting RTCCALS = 1 and RTCCAL = 0h results in a +8 ppm increase in frequency.

NOTE: Calibration output frequency

The 512-Hz and 256-Hz output frequencies observed at the RTCCLK pin are not affected by changes in the calibration settings, because these output frequencies are generated before the calibration logic. The 1-Hz output frequency is affected by changes in the calibration settings. Because the frequency change is small and infrequent over a very long time interval, it can be difficult to observe.

28.2.6 Real-Time Clock Operation in LPM3.5 Low-Power Mode

The regulator of the Power Management Module (PMM) is disabled upon entering LPM3.5, which causes most of the RTC_B configuration registers to be lost; only the counters are retained. [Table 28-1](#) lists the retained registers in LPM3.5. Also the configuration of the interrupts is stored so that the configured interrupts can cause a wakeup upon exit from LPM3.5. Interrupt flags that are set before entering LPM3.5 are cleared upon entering LPM3.5 (Note: this can only happen if the corresponding interrupt is not enabled). The interrupt flags RTCTEVIFG, RTCAIFG, RT1PSIFG, and RTCOFIFG can be used as RTC_B wake-up interrupt sources. After restoring the configuration registers (and clearing LOCKLPM5) the interrupts can be serviced as usual. The detailed flow is as follows:

1. Set all I/Os to general purpose I/Os and configure as needed. Optionally configure input interrupt pins for wake-up. Configure RTC_B interrupts for wake-up (set RTCTEVIE, RTCAIE, RT1PSIE, or RTCOFIE. If the alarm interrupt is also used as wake-up event, the alarm registers must be configured as needed).
2. Enter LPMx.5 with LPMx.5 entry sequence.


```
MOV #PMMKEY + PMMREGOFF, &PMMCTL0 ; Open PMM registers for write and set PMMREGOFF
;
BIS #LPM4, SR ; Enter LPMx.5 when PMMREGOFF is set
```
3. LOCKLPM5 is automatically set by hardware upon entering LPMx.5, the core voltage regulator is disabled, and all clocks are disabled except for the 32-kHz crystal oscillator clock if the RTC is enabled with RTCHOLD = 0.
4. An LPMx.5 wake-up event, such as an edge on a wake-up input pin, or an RTC_B interrupt event will start the BOR entry sequence together with the core voltage regulator. All peripheral registers are set to their default conditions. The I/O pin state remains locked as well as the interrupt configuration for the RTC_B.
5. The device can be configured. The I/O configuration and the RTC_B interrupt configuration that was not retained during LPM3.5 should be restored to the values before entering LPM3.5. Then the LOCKLPM5 bit can be cleared, this releases the I/O pin conditions as well as the RTC_B interrupt configuration.
6. After enabling I/O and RTC_B interrupts, the interrupt that caused the wake-up can be serviced.
7. To re-enter LPMx.5, the LOCKLPM5 bit must be cleared before re-entry, otherwise LPMx.5 is not entered.

If the RTC is enabled (RTCHOLD = 0), the 32-kHz oscillator remains active during LPM3.5. The fault detection also remains functional. If a fault occurs during LPM3.5 and the RTCOFIE was set before entering LPM3.5, a wake-up event is issued.

28.3 RTC_B Registers

The RTC_B module registers are listed in [Table 28-1](#). This table also lists the retention during LPMx.5. Registers that are not retained during LPMx.5 must be restored after exit from LPMx.5. The base address for the RTC_B module registers can be found in the device-specific data sheet. The address offsets are given in [Table 28-1](#).

NOTE: Most registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 28-1. RTC_B Registers

Offset	Acronym	Register Name	Type	Access	Reset	LPMx.5 and Backup Operation
00h	RTCCTL01	Real-Time Clock Control 0, 1	Read/write	Word	7000h	not retained
00h	RTCCTL0 or RTCCTL01_L	Real-Time Clock Control 0	Read/write	Byte	00h	not retained
01h	RTCCTL1 or RTCCTL01_H	Real-Time Clock Control 1	Read/write	Byte	70h	not retained
02h	RTCCTL23	Real-Time Clock Control 2, 3	Read/write	Word	0000h	retained
02h	RTCCTL2 or RTCCTL23_L	Real-Time Clock Control 2	Read/write	Byte	00h	retained
03h	RTCCTL3 or RTCCTL23_H	Real-Time Clock Control 3	Read/write	Byte	00h	retained
08h	RTCPS0CTL	Real-Time Prescale Timer 0 Control	Read/write	Word	0000h	not retained
08h	RTCPS0CTLH or RTCPS0CTL_L		Read/write	Byte	00h	not retained
09h	RTCPS0CTLH or RTCPS0CTL_H		Read/write	Byte	00h	not retained
0Ah	RTCPS1CTL	Real-Time Prescale Timer 1 Control	Read/write	Word	0000h	not retained
0Ah	RTCPS1CTLH or RTCPS1CTL_L		Read/write	Byte	00h	not retained
0Bh	RTCPS0CTLH or RTCPS0CTL_H		Read/write	Byte	00h	not retained
0Ch	RTCPS	Real-Time Prescale Timer 0, 1 Counter	Read/write	Word	none	retained
0Ch	RT0PS or RTCPS_L	Real-Time Prescale Timer 0 Counter	Read/write	Byte	none	retained
0Dh	RT1PS or RTCPS_H	Real-Time Prescale Timer 1 Counter	Read/write	Byte	none	retained
0Eh	RTCIV	Real Time Clock Interrupt Vector	Read	Word	0000h	not retained
10h	RTCTIM0	Real-Time Clock Seconds, Minutes	Read/write	Word	undefined	retained
10h	RTCSEC or RTCTIM0_L	Real-Time Clock Seconds	Read/write	Byte	undefined	retained
11h	RTCMIN or RTCTIM0_H	Real-Time Clock Minutes	Read/write	Byte	undefined	retained
12h	RTCTIM1	Real-Time Clock Hour, Day of Week	Read/write	Word	undefined	retained
12h	RTCHOUR or RTCTIM1_L	Real-Time Clock Hour	Read/write	Byte	undefined	retained
13h	RTCDOW or RTCTIM1_H	Real-Time Clock Day of Week	Read/write	Byte	undefined	retained

Table 28-1. RTC_B Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	LPMx.5 and Backup Operation
14h	RTCDATE	Real-Time Clock Date	Read/write	Word	undefined	retained
14h	RTCDAY or RTCDATE_L	Real-Time Clock Day of Month	Read/write	Byte	undefined	retained
15h	RTCMON or RTCDATE_H	Real-Time Clock Month	Read/write	Byte	undefined	retained
16h	RTCYEAR	Real-Time Clock Year ⁽¹⁾	Read/write	Word	undefined	retained
18h	RTCAMINHR	Real-Time Clock Minutes, Hour Alarm	Read/write	Word	undefined	retained
18h	RTCAMIN or RTCAMINHR_L	Real-Time Clock Minutes Alarm	Read/write	Byte	undefined	retained
19h	RTCAHOUR or RTCAMINHR_H	Real-Time Clock Hours Alarm	Read/write	Byte	undefined	retained
1Ah	RTCADOWDAY	Real-Time Clock Day of Week, Day of Month Alarm	Read/write	Word	undefined	retained
1Ah	RTCADOW or RTCADOWDAY_L	Real-Time Clock Day of Week Alarm	Read/write	Byte	undefined	retained
1Bh	RTCADAY or RTCADOWDAY_H	Real-Time Clock Day of Month Alarm	Read/write	Byte	undefined	retained
1Ch	BIN2BCD	Binary-to-BCD Conversion Register	Read/write	Word	00h	not retained
1Eh	BCD2BIN	BCD-to-Binary Conversion Register	Read/write	Word	00h	not retained

⁽¹⁾ Do not access the RTCYEAR register in byte mode.

28.3.1 RTCCTL0 Register

Real-Time Clock Control 0 Register

Figure 28-2. RTCCTL0 Register

7	6	5	4	3	2	1	0
RTCOFIE ⁽¹⁾	RTCDEVIE ⁽¹⁾	RTCAIE ⁽¹⁾	RTCRDYIE	RTCOFIG	RTCDEVIFG	RTCAIFG	RTCRDYIFG
rw-0	rw-0	rw-0	rw-0	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ The configuration of these bits is retained during LPMx.5 until LOCKLPM5 is cleared, but not the register bits itself; therefore, reconfiguration after wake-up from LPMx.5 before clearing LOCKLPM5 is required.

Table 28-2. RTCCTL0 Register Description

Bit	Field	Type	Reset	Description
7	RTCOFIE	RW	0h	32-kHz crystal oscillator fault interrupt enable. This interrupt can be used as LPMx.5 wake-up event. 0b = Interrupt not enabled 1b = Interrupt enabled (LPMx.5 wake-up enabled)
6	RTCDEVIE	RW	0h	Real-time clock time event interrupt enable. In modules supporting LPMx.5 this interrupt can be used as LPMx.5 wake-up event. 0b = Interrupt not enabled 1b = Interrupt enabled (LPMx.5 wake-up enabled)
5	RTCAIE	RW	0h	Real-time clock alarm interrupt enable. In modules supporting LPMx.5 this interrupt can be used as LPMx.5 wake-up event. 0b = Interrupt not enabled 1b = Interrupt enabled (LPMx.5 wake-up enabled)
4	RTCRDYIE	RW	0h	Real-time clock ready interrupt enable. 0b = Interrupt not enabled 1b = Interrupt enabled
3	RTCOFIG	RW	0h	32-kHz crystal oscillator fault interrupt flag. This interrupt can be used as LPMx.5 wake-up event. It also indicates a clock failure during backup operation. 0b = No interrupt pending 1b = Interrupt pending. A 32-kHz crystal oscillator fault occurred after last reset.
2	RTCDEVIFG	RW	0h	Real-time clock time event interrupt flag. In modules supporting LPMx.5 this interrupt can be used as LPMx.5 wake-up event. 0b = No time event occurred 1b = Time event occurred
1	RTCAIFG	RW	0h	Real-time clock alarm interrupt flag. In modules supporting LPMx.5 this interrupt can be used as LPMx.5 wake-up event. 0b = No time event occurred 1b = Time event occurred
0	RTCRDYIFG	RW	0h	Real-time clock ready interrupt flag 0b = RTC cannot be read safely 1b = RTC can be read safely

28.3.2 RTCCTL1 Register

Real-Time Clock Control Register 1

Figure 28-3. RTCCTL1 Register

7	6	5	4	3	2	1	0
RTCB CD	RTCHOLD ⁽¹⁾	Reserved	RTCRDY	Reserved	Reserved	RTCTEVx ⁽¹⁾	
rw-(0)	rw-(1)	r1	r-(1)	r0	r0	rw-(0)	rw-(0)

⁽¹⁾ The configuration of these bits is retained during LPMx.5 until LOCKLPM5 is cleared, but not the register bits itself; therefore, reconfiguration after wake-up from LPMx.5 before clearing LOCKLPM5 is required.

Table 28-3. RTCCTL1 Register Description

Bit	Field	Type	Reset	Description
7	RTCB CD	RW	0h	Real-time clock BCD select. Selects BCD counting for real-time clock. 0b = Binary-hexadecimal code selected 1b = BCD Binary coded decimal (BCD) code selected
6	RTCHOLD	RW	1h	Real-time clock hold 0b = Real-time clock is operational. 1b = The calendar is stopped as well as the prescale counters, RT0PS, and RT1PS.
5	Reserved	R	1h	Reserved. Always read as 1.
4	RTCRDY	RW	1h	Real-time clock ready 0b = RTC time values in transition 1b = RTC time values safe for reading. This bit indicates when the real-time clock time values are safe for reading.
3-2	Reserved	R	0h	Reserved. Always read as 0.
1-0	RTCTEVx	RW	0h	Real-time clock time interrupt event 00b = Minute changed 01b = Hour changed 10b = Every day at midnight (00:00) 11b = Every day at noon (12:00)

28.3.3 RTCCTL2 Register

Real-Time Clock Control 2 Register

Figure 28-4. RTCCTL2 Register

7	6	5	4	3	2	1	0
RTCCALS	Reserved	RTCCALx					
rw-(0)	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 28-4. RTCCTL2 Register Description

Bit	Field	Type	Reset	Description
7	RTCCALS	RW	0h	Real-time clock calibration sign 0b = Frequency adjusted down 1b = Frequency adjusted up
6	Reserved	R	0h	Reserved. Always read as 0.
5-0	RTCCALx	RW	0h	Real-time clock calibration. Each LSB represents approximately +4-ppm (RTCCALS = 1) or a -2-ppm (RTCCALS = 0) adjustment in frequency.

28.3.4 RTCCTL3 Register

Real-Time Clock Control 3 Register

Figure 28-5. RTCCTL3 Register

7	6	5	4	3	2	1	0
Reserved						RTCCALFx	
r0	r0	r0	r0	r0	r0	rw-(0)	rw-(0)

Table 28-5. RTCCTL3 Register Description

Bit	Field	Type	Reset	Description
7-2	Reserved	R	0h	Reserved. Always read as 0.
1-0	RTCCALFx	RW	0h	Real-time clock calibration frequency. Selects frequency output to RTCCLK pin for calibration measurement. The corresponding port must be configured for the peripheral module function. 00b = No frequency output to RTCCLK pin 01b = 512 Hz 10b = 256 Hz 11b = 1 Hz

28.3.5 RTCSEC Register – Hexadecimal Format

Real-Time Clock Seconds Register – Hexadecimal Format

Figure 28-6. RTCSEC Register

7	6	5	4	3	2	1	0
0	0	Seconds					
r-0	r-0	rw	rw	rw	rw	rw	rw

Table 28-6. RTCSEC Register Description

Bit	Field	Type	Reset	Description
7-6	0	R	0h	Always reads as 0.
5-0	Seconds	RW	undefined	Seconds. Valid values are 0 to 59.

28.3.6 RTCSEC Register – BCD Format

Real-Time Clock Seconds Register – BCD Format

Figure 28-7. RTCSEC Register

7	6	5	4	3	2	1	0
0	Seconds – high digit			Seconds – low digit			
r-0	rw	rw	rw	rw	rw	rw	rw

Table 28-7. RTCSEC Register Description

Bit	Field	Type	Reset	Description
7	0	R	0h	Always reads as 0.
6-4	Seconds – high digit	RW	undefined	Seconds – high digit. Valid values are 0 to 5.
3-0	Seconds – low digit	RW	undefined	Seconds – low digit. Valid values are 0 to 9.

28.3.7 RTCMIN Register – Hexadecimal Format

Real-Time Clock Minutes Register – Hexadecimal Format

Figure 28-8. RTCMIN Register

7	6	5	4	3	2	1	0
0	0	Minutes					
r-0	r-0	rw	rw	rw	rw	rw	rw

Table 28-8. RTCMIN Register Description

Bit	Field	Type	Reset	Description
7-6	0	R	0h	Always reads as 0.
5-0	Minutes	RW	undefined	Minutes. Valid values are 0 to 59.

28.3.8 RTCMIN Register – BCD Format

Real-Time Clock Minutes Register – BCD Format

Figure 28-9. RTCMIN Register

7	6	5	4	3	2	1	0
0	Minutes – high digit			Minutes – low digit			
r-0	rw	rw	rw	rw	rw	rw	rw

Table 28-9. RTCMIN Register Description

Bit	Field	Type	Reset	Description
7	0	R	0h	Always reads as 0.
6-4	Minutes – high digit	RW	undefined	Minutes – high digit. Valid values are 0 to 5.
3-0	Minutes – low digit	RW	undefined	Minutes – low digit. Valid values are 0 to 9.

28.3.9 RTCHOUR Register – Hexadecimal Format

Real-Time Clock Hours Register – Hexadecimal Format

Figure 28-10. RTCHOUR Register

7	6	5	4	3	2	1	0
0	0	0	Hours				
r-0	r-0	r-0	rw	rw	rw	rw	rw

Table 28-10. RTCHOUR Register Description

Bit	Field	Type	Reset	Description
7-5	0	R	0h	Always reads as 0.
4-0	Hours	RW	undefined	Hours. Valid values are 0 to 23.

28.3.10 RTCHOUR Register – BCD Format

Real-Time Clock Hours Register – BCD Format

Figure 28-11. RTCHOUR Register

7	6	5	4	3	2	1	0
0	0	Hours – high digit		Hours – low digit			
r-0	r-0	rw	rw	rw	rw	rw	rw

Table 28-11. RTCHOUR Register Description

Bit	Field	Type	Reset	Description
7-6	0	R	0h	Always reads as 0.
5-4	Hours – high digit	RW	undefined	Hours – high digit. Valid values are 0 to 2.
3-0	Hours – low digit	RW	undefined	Hours – low digit. Valid values are 0 to 9.

28.3.11 RTCDOW Register

Real-Time Clock Day of Week Register

Figure 28-12. RTCDOW Register

7	6	5	4	3	2	1	0
0	0	0	0	0	Day of week		
r-0	r-0	r-0	r-0	r-0	rw	rw	rw

Table 28-12. RTCDOW Register Description

Bit	Field	Type	Reset	Description
7-3	0	R	0h	Always reads as 0.
2-0	Day of week	RW	undefined	Day of week. Valid values are 0 to 6.

28.3.12 RTCDAY Register – Hexadecimal Format

Real-Time Clock Day of Month Register – Hexadecimal Format

Figure 28-13. RTCDAY Register

7	6	5	4	3	2	1	0
0	0	0	Day of month				
r-0	r-0	r-0	rw	rw	rw	rw	rw

Table 28-13. RTCDAY Register Description

Bit	Field	Type	Reset	Description
7-5	0	R	0h	Always reads as 0.
4-0	Day of month	RW	undefined	Day of month. Valid values are 1 to 31.

28.3.13 RTCDAY Register – BCD Format

Real-Time Clock Day of Month Register – BCD Format

Figure 28-14. RTCDAY Register

7	6	5	4	3	2	1	0
0	0	Day of month – high digit		Day of month – low digit			
r-0	r-0	rw	rw	rw	rw	rw	rw

Table 28-14. RTCDAY Register Description

Bit	Field	Type	Reset	Description
7-6	0	R	0h	Always reads as 0.
5-4	Day of month – high digit	RW	undefined	Day of month – high digit. Valid values are 0 to 3.
3-0	Day of month – low digit	RW	undefined	Day of month – low digit. Valid values are 0 to 9.

28.3.14 RTCMON Register – Hexadecimal Format

Real-Time Clock Month Register – Hexadecimal Format

Figure 28-15. RTCMON Register

7	6	5	4	3	2	1	0
0	0	0	0	Month			
r-0	r-0	r-0	r-0	rw	rw	rw	rw

Table 28-15. RTCMON Register Description

Bit	Field	Type	Reset	Description
7-4	0	R	0h	Always reads as 0.
3-0	Month	RW	undefined	Month. Valid values are 1 to 12.

28.3.15 RTCMON Register – BCD Format

Real-Time Clock Month Register

Figure 28-16. RTCMON Register

7	6	5	4	3	2	1	0
0	0	0	Month – high digit	Month – low digit			
r-0	r-0	r-0	rw	rw	rw	rw	rw

Table 28-16. RTCMON Register Description

Bit	Field	Type	Reset	Description
7-5	0	R	0h	Always reads as 0.
4	Month – high digit	RW	undefined	Month – high digit. Valid values are 0 or 1.
3-0	Month – low digit	RW	undefined	Month – low digit. Valid values are 0 to 9.

28.3.16 RTCYEAR Register – Hexadecimal Format

Real-Time Clock Year Register – Hexadecimal Format

Figure 28-17. RTCYEAR Register

15	14	13	12	11	10	9	8
0	0	0	0	Year – high byte			
r-0	r-0	r-0	r-0	rw	rw	rw	rw
7	6	5	4	3	2	1	0
rw	rw	rw	rw	rw	rw	rw	rw

Table 28-17. RTCYEAR Register Description

Bit	Field	Type	Reset	Description
15-12	0	R	0h	Always reads as 0.
11-8	Year – high byte	RW	undefined	Year – high byte. Valid values of Year are 0 to 4095.
7-0	Year – low byte	RW	undefined	Year – low byte. Valid values of Year are 0 to 4095.

28.3.17 RTCYEAR Register – BCD Format

Real-Time Clock Year Register – BCD Format

Figure 28-18. RTCYEAR Register

15	14	13	12	11	10	9	8
0	Century – high digit				Century – low digit		
r-0	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
Decade				Year – lowest digit			
rw	rw	rw	rw	rw	rw	rw	rw

Table 28-18. RTCYEAR Register Description

Bit	Field	Type	Reset	Description
15	0	R	0h	Always reads as 0.
14-12	Century – high digit	RW	undefined	Century – high digit . Valid values are 0 to 4.
11-8	Century – low digit	RW	undefined	Century – low digit. Valid values are 0 to 9.
7-4	Decade	RW	undefined	Decade. Valid values are 0 to 9.
3-0	Year – lowest digit	RW	undefined	Year – lowest digit. Valid values are 0 to 9.

28.3.18 RTCAMIN Register – Hexadecimal Format

Real-Time Clock Minutes Alarm Register – Hexadecimal Format

Figure 28-19. RTCAMIN Register

7	6	5	4	3	2	1	0
AE	0	Minutes					
rw	r-0	rw	rw	rw	rw	rw	rw

Table 28-19. RTCAMIN Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6	0	R	0h	Always reads as 0.
5-0	Minutes	RW	undefined	Minutes. Valid values are 0 to 59.

28.3.19 RTCAMIN Register – BCD Format

Real-Time Clock Minutes Alarm Register – BCD Format

Figure 28-20. RTCAMIN Register

7	6	5	4	3	2	1	0
AE	Minutes – high digit			Minutes – low digit			
rw	rw	rw	rw	rw	rw	rw	rw

Table 28-20. RTCAMIN Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6-4	Minutes – high digit	RW	undefined	Minutes – high digit. Valid values are 0 to 5.
3-0	Minutes – low digit	RW	undefined	Minutes – low digit. Valid values are 0 to 9.

28.3.20 RTCAHOUR Register – Hexadecimal Format

Real-Time Clock Hours Alarm Register – Hexadecimal Format

Figure 28-21. RTCAHOUR Register

7	6	5	4	3	2	1	0
AE	0	0	Hours				
rw	r-0	r-0	rw	rw	rw	rw	rw

Table 28-21. RTCAHOUR Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6-5	0	R	0h	Always reads as 0.
4-0	Hours	RW	undefined	Hours. Valid values are 0 to 23.

28.3.21 RTCAHOUR Register – BCD Format

Real-Time Clock Hours Alarm Register – BCD Format

Figure 28-22. RTCAHOUR Register

7	6	5	4	3	2	1	0
AE	0	Hours – high digit		Hours – low digit			
rw	r-0	rw	rw	rw	rw	rw	rw

Table 28-22. RTCAHOUR Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6	0	R	0h	Always reads as 0.
5-4	Hours – high digit	RW	undefined	Hours – high digit. Valid values are 0 to 2.
3-0	Hours – low digit	RW	undefined	Hours – low digit. Valid values are 0 to 9.

28.3.22 RTCADOW Register

Real-Time Clock Day of Week Alarm Register

Figure 28-23. RTCADOW Register

7	6	5	4	3	2	1	0
AE	0	0	0	0	Day of week		
rw	r-0	r-0	r-0	r-0	rw	rw	rw

Table 28-23. RTCADOW Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6-3	0	R	0h	Always reads as 0.
2-0	Day of week	RW	undefined	Day of week. Valid values are 0 to 6.

28.3.23 RTCADAY Register – Hexadecimal Format

Real-Time Clock Day of Month Alarm Register – Hexadecimal Format

Figure 28-24. RTCADAY Register

7	6	5	4	3	2	1	0
AE	0	0	Day of month				
rw	r-0	r-0	rw	rw	rw	rw	rw

Table 28-24. RTCADAY Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6-5	0	R	0h	Always reads as 0.
4-0	Day of month	RW	undefined	Day of month. Valid values are 1 to 31.

28.3.24 RTCADAY Register – BCD Format

Real-Time Clock Day of Month Alarm Register – BCD Format

Figure 28-25. RTCADAY Register

7	6	5	4	3	2	1	0
AE	0	Day of month – high digit		Day of month – low digit			
rw	r-0	rw	rw	rw	rw	rw	rw

Table 28-25. RTCADAY Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6	0	R	0h	Always reads as 0.
5-4	Day of month – high digit	RW	undefined	Day of month – high digit. Valid values are 0 to 3.
3-0	Day of month – low digit	RW	undefined	Day of month – low digit. Valid values are 0 to 9.

28.3.25 RTCPS0CTL Register

Real-Time Clock Prescale Timer 0 Control Register

Figure 28-26. RTCPS0CTL Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved			RT0IPx ⁽¹⁾			RT0PSIE	RT0PSIFG
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-(0)

⁽¹⁾ The configuration of these bits is retained during LPMx.5 until LOCKLPM5 is cleared, but not the register bits itself; therefore, reconfiguration after wake-up from LPMx.5 before clearing LOCKLPM5 is required.

Table 28-26. RTCPS0CTL Register Description

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Always reads as 0.
4-2	RT0IPx	RW	0h	Prescale timer 0 interrupt interval 000b = Divide by 2 001b = Divide by 4 010b = Divide by 8 011b = Divide by 16 100b = Divide by 32 101b = Divide by 64 110b = Divide by 128 111b = Divide by 256
1	RT0PSIE	RW	0h	Prescale timer 0 interrupt enable 0b = Interrupt not enabled 1b = Interrupt enabled
0	RT0PSIFG	RW	0h	Prescale timer 0 interrupt flag 0b = No time event occurred 1b = Time event occurred

28.3.26 RTCPS1CTL Register

Real-Time Clock Prescale Timer 1 Control Register

Figure 28-27. RTCPS1CTL Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved			RT1IPx ⁽¹⁾			RT1PSIE ⁽¹⁾	RT1PSIFG
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-(0)

⁽¹⁾ The configuration of these bits is retained during LPMx.5 until LOCKLPM5 is cleared, but not the register bits themselves; therefore, reconfiguration after wake-up from LPMx.5 before clearing LOCKLPM5 is required.

Table 28-27. RTCPS1CTL Register Description

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Always reads as 0.
4-2	RT1IPx	RW	0h	Prescale timer 1 interrupt interval 000b = Divide by 2 001b = Divide by 4 010b = Divide by 8 011b = Divide by 16 100b = Divide by 32 101b = Divide by 64 110b = Divide by 128 111b = Divide by 256
1	RT1PSIE	RW	0h	Prescale timer 1 interrupt enable 0b = Interrupt not enabled 1b = Interrupt enabled (LPMx.5 wake-up enabled)
0	RT1PSIFG	RW	0h	Prescale timer 1 interrupt flag. In modules supporting LPMx.5 this interrupt can be used as LPMx.5 wake-up event. 0b = No time event occurred 1b = Time event occurred

28.3.27 RTCPS0 Register

Real-Time Clock Prescale Timer 0 Counter Register

Figure 28-28. RTCPS0 Register

7	6	5	4	3	2	1	0
RT0PS							
rw	rw	rw	rw	rw	rw	rw	rw

Table 28-28. RTCPS0 Register Description

Bit	Field	Type	Reset	Description
7-0	RT0PS	RW	undefined	Prescale timer 0 counter value

28.3.28 RTCPS1 Register

Real-Time Clock Prescale Timer 1 Counter Register

Figure 28-29. RTCPS1 Register

7	6	5	4	3	2	1	0
RT1PS							
rw	rw	rw	rw	rw	rw	rw	rw

Table 28-29. RTCPS1 Register Description

Bit	Field	Type	Reset	Description
7-0	RT1PS	RW	undefined	Prescale timer 1 counter value

28.3.29 RTCIV Register

Real-Time Clock Interrupt Vector Register

Figure 28-30. RTCIV Register

15	14	13	12	11	10	9	8
RTCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
RTCIVx							
r0	r0	r0	r0	r-(0)	r-(0)	r-(0)	r0

Table 28-30. RTCIV Register Description

Bit	Field	Type	Reset	Description
15-0	RTCIVx	R	0h	Real-time clock interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: RTC ready; Interrupt Flag: RTCRDYIFG; Interrupt Priority: Highest 04h = Interrupt Source: RTC interval timer; Interrupt Flag: RTCTEVIFG 06h = Interrupt Source: RTC user alarm; Interrupt Flag: RTCAIFG 08h = Interrupt Source: RTC prescaler 0; Interrupt Flag: RT0PSIFG 0Ah = Interrupt Source: RTC prescaler 1; Interrupt Flag: RT1PSIFG 0Ch = Interrupt Source: RTC oscillator failure; Interrupt Flag: RTCOFIFG 0Eh = Reserved; Interrupt Priority: Lowest

28.3.30 BIN2BCD Register

Binary-to-BCD Conversion Register

Figure 28-31. BIN2BCD Register

15	14	13	12	11	10	9	8
BIN2BCDx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
BIN2BCDx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 28-31. BIN2BCD Register Description

Bit	Field	Type	Reset	Description
15-0	BIN2BCDx	RW	0h	Read: 16-bit BCD conversion of previously written 12-bit binary number Write: 12-bit binary number to be converted

28.3.31 BCD2BIN Register

BCD-to-Binary Conversion Register

Figure 28-32. BCD2BIN Register

15	14	13	12	11	10	9	8
BCD2BINx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
BCD2BINx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 28-32. BCD2BIN Register Description

Bit	Field	Type	Reset	Description
15-0	BCD2BINx	RW	0h	Read: 12-bit binary conversion of previously written 16-bit BCD number Write: 16-bit BCD number to be converted

Real-Time Clock C (RTC_C)

The Real-Time Clock C (RTC_C) module provides clock counters with calendar mode, a flexible programmable alarm, offset calibration, and a provision for temperature compensation. The RTC_C also supports operation in LPM3.5. This chapter describes the RTC_C module.

Topic	Page
29.1 Real-Time Clock (RTC_C) Introduction	719
29.2 RTC_C Operation.....	721
29.3 RTC_C Operation - Device-Dependent Features	729
29.4 RTC_C Registers	734

29.1 Real-Time Clock (RTC_C) Introduction

The RTC_C module provides configurable clock counters.

RTC_C features include:

- Real-time clock and calendar mode that provides seconds, minutes, hours, day of week, day of month, month, and year (including leap year correction)
- Protection for real-time clock registers
- Interrupt capability
- Selectable BCD or binary format
- Programmable alarms
- Real-time clock calibration for crystal offset error
- Real-time clock compensation for crystal temperature drift
- Operation in LPM3.5

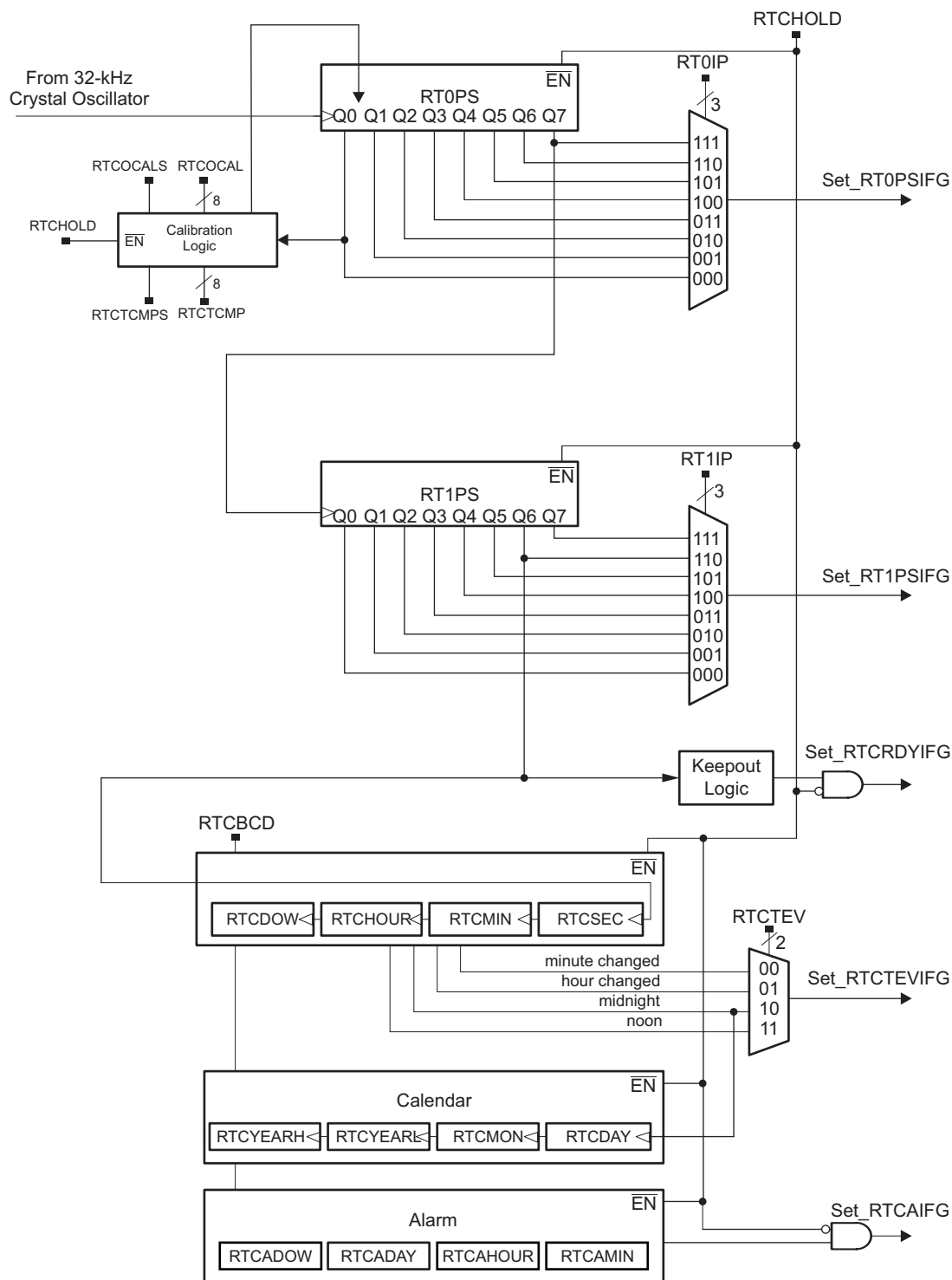
The RTC_C module can provide the following device-dependent features. Refer to the device-specific data sheet to determine if these features are available in a particular device.

- General-purpose counter mode (see [Section 29.3.1](#))
- Event and tamper detection with time stamp (see [Section 29.3.2](#))
- Operation from a separate voltage supply

NOTE: Real-time clock initialization

Most RTC_C module registers have no initial condition. These registers must be configured by user software before use.

[Figure 29-1](#) shows the RTC_C block diagram.



Copyright © 2016, Texas Instruments Incorporated

Figure 29-1. RTC_C Block Diagram (RTCMODE = 1)

29.2 RTC_C Operation

29.2.1 Calendar Mode

Calendar mode is selected when RTCMODE is set. In calendar mode, the RTC_C module provides seconds, minutes, hours, day of week, day of month, month, and year in selectable BCD or hexadecimal format. The calendar includes a leap-year algorithm that considers all years evenly divisible by four as leap years. This algorithm is accurate from the year 1901 through 2099. Switching from counter mode (if available) to calendar mode **does not** reset the calendar registers (RTCSEC, RTCMIN, RTCHOUR, RTCDAY, RTCDOW, and RTCYEAR) and the prescale counters (RT0PS, RT1PS). These registers must be configured by user software before use.

29.2.2 Real-Time Clock and Prescale Dividers

The prescale dividers, RT0PS and RT1PS, are automatically configured to provide a 1-second clock interval for the RTC_C. The low-frequency oscillator must be operated at 32768 Hz (nominal) for proper RTC_C operation. RT0PS is sourced from the low-frequency oscillator XT1. The output of RT0PS divided by 256 (Q7) sources RT1PS. RT1PS is further divided by 128 to source the real-time clock counter registers that provide the required 1-second time interval.

When RTCBCD = 1, BCD format is selected for the calendar registers. It is possible to switch between BCD and hexadecimal format while the RTC is counting.

Setting RTCHOLD halts the real-time counters and prescale counters, RT0PS and RT1PS.

NOTE: For reliable update to all Calendar Mode registers

Set RTCHOLD = 1 before writing into any of the calendar or prescaler registers (RTCPS0, RTCPS1, RTCSEC, RTCMIN, RTCHOUR, RTCDAY, RTCDOW, RTCMON, and RTCYEAR).

29.2.3 Real-Time Clock Alarm Function

The RTC_C module provides for a flexible alarm system. There is a single user-programmable alarm that can be programmed based on the settings contained in the alarm registers for minutes, hours, day of week, and day of month.

Each alarm register contains an alarm enable (AE) bit that can be used to enable the respective alarm register. By setting AE bits of the various alarm registers, a variety of alarm events can be generated.

- Example 1: A user wishes to set an alarm every hour at 15 minutes past the hour (that is, 00:15:00, 01:15:00, 02:15:00, and so on). This is possible by setting RTCAMIN to 15. By setting the AE bit of the RTCAMIN and clearing all other AE bits of the alarm registers, the alarm is enabled. When enabled, the RTCAIFG is set when the count transitions from 00:14:59 to 00:15:00, 01:14:59 to 01:15:00, 02:14:59 to 02:15:00, and so on.
- Example 2: A user wishes to set an alarm every day at 04:00:00. This is possible by setting RTCAHOUR to 4. By setting the AE bit of the RTCHOUR and clearing all other AE bits of the alarm registers, the alarm is enabled. When enabled, the RTCAIFG is set when the count transitions from 03:59:59 to 04:00:00.
- Example 3: A user wishes to set an alarm for 06:30:00. RTCAHOUR would be set to 6, and RTCAMIN would be set to 30. By setting the AE bits of RTCAHOUR and RTCAMIN, the alarm is enabled. When enabled, the RTCAIFG is set when the time count transitions from 06:29:59 to 06:30:00. In this case, the alarm event occurs every day at 06:30:00.
- Example 4: A user wishes to set an alarm every Tuesday at 06:30:00. RTCADOW would be set to 2, RTCAHOUR would be set to 6, and RTCAMIN would be set to 30. By setting the AE bits of RTCADOW, RTCAHOUR, and RTCAMIN, the alarm is enabled. When enabled, the RTCAIFG is set when the time count transitions from 06:29:59 to 06:30:00 and the RTCDOW transitions from 1 to 2.
- Example 5: A user wishes to set an alarm the fifth day of each month at 06:30:00. RTCADAY would be set to 5, RTCAHOUR would be set to 6, and RTCAMIN would be set to 30. By setting the AE bits of RTCADAY, RTCAHOUR, and RTCAMIN, the alarm is enabled. When enabled, the RTCAIFG is set when the time count transitions from 06:29:59 to 06:30:00 and the RTCDAY equals 5.

NOTE: Invalid alarm settings

Invalid alarm settings are not checked by hardware. It is the user's responsibility that valid alarm settings are entered.

NOTE: Invalid time and date values

Writing of invalid date or time information or data values outside the legal ranges specified in the RTCSEC, RTCMIN, RTCHOUR, RTCDAY, RTCDOW, RTCYEARH, RTCYEARL, RTCAMIN, RTCAHOUR, RTCADAY, and RTCADOW registers can result in unpredictable behavior.

NOTE: Setting the alarm

Before setting an initial alarm, all alarm registers including the AE bits should be cleared.

To prevent potential erroneous alarm conditions from occurring, the alarms should be disabled by clearing the RTCAIE, RTCAIFG, and AE bits before writing initial or new time values to the RTC time registers.

29.2.4 Real-Time Clock Protection

RTC_C registers are key protected to ensure clock integrity and module configuration against software crash or from runaway code. Key protection does not apply for reads from the RTC_C registers. That is, any RTC_C register can be read at any time without having to unlock the module. Some predefined registers of RTC_C are key protected for write access. The control registers, clock registers, calendar register, prescale timer registers, and offset error calibration registers are protected. RTC_C alarm function registers, prescale timer control registers, interrupt vector register, and temperature compensation registers are not protected. RTC_C registers that are not protected can be written at any time without unlocking the module. [Table 29-2](#) shows which registers are affected by the protection scheme.

The RTCCTL0_H register implements key protection and controls the lock or unlock state of the module. When this register is written with correct key, 0A5h, the module is unlocked and unlimited write access possible to RTC_C registers. After the module is unlocked, it remains unlocked until the user writes any incorrect key or until the module is reset. A read from RTCCTL0_H register returns value 96h. Write access to any protected registers of RTC_C is ignored when the module is locked.

RTC_C Key Protection Software Example

```
; Unlock/lock sequence for RTC_C
MOV.B  #RTCKEY, &RTCCTL0_H      ; Write correct key to unlock RTC_C
MOV.B  #8bit_value, &RTCSEC      ; Write 8 bit value into RTCSEC
MOV.B  #8bit_value, &RTCMIN      ; Write 8 bit value into RTCMIN
MOV.W  #16bit_value, &RTCTIM1    ; Write 16bit value into RTCTIM1
MOV.W  #RTCKEY+8bit_value, &RTCCTL0 ; Write into RTCCTL0 with correct key in word mode
...

MOV.B  #00h, &RTCCTL0_H          ; Write incorrect key to lock RTC_C
```

29.2.5 Reading or Writing Real-Time Clock Registers

Because the system clock may be asynchronous to the RTC_C clock source, special care must be used when accessing the real-time clock registers.

The real-time clock registers are updated once per second. To prevent reading any real-time clock register at the time of an update that could result in an invalid time being read, a keep-out window is provided. The keep-out window is centered approximately 128/32768 seconds around the update transition. The read-only RTCRDY bit is reset during the keep-out window period and set outside the keep-out the window period. Any read of the clock registers while RTCRDY is reset is considered to be potentially invalid, and the time read should be ignored.

An easy way to safely read the real-time clock registers is to use the RTCRDYIFG interrupt flag. Setting RTCRDYIE enables the RTCRDYIFG interrupt. When enabled, an interrupt is generated based on the rising edge of the RTCRDY bit, causing the RTCRDYIFG to be set. At this point, the application has nearly a complete second to safely read any or all of the real-time clock registers. This synchronization process prevents reading the time value during transition. The RTCRDYIFG flag is reset automatically when the interrupt is serviced or can be reset with software.

NOTE: Reading or writing real-time clock registers

When the counter clock is asynchronous to the CPU clock, any read from any RTCSEC, RTCMIN, RTCHOUR, RTCDOW, RTCDAY, RTCMON, RTCYEARL, or RTCYEARH register while the RTCRDY is reset may result in invalid data being read. To safely read the counting registers, either polling of the RTCRDY bit or the synchronization procedure previously described can be used. Alternatively, the counter register can be read multiple times while operating, and a majority vote taken in software to determine the correct reading. Reading the RT0PS and RT1PS can only be handled by reading the registers multiple times and a majority vote taken in software to determine the correct reading.

Any write to any counting register takes effect immediately. However, the clock is stopped during the write. In addition, RT0PS and RT1PS registers are reset. This could result in losing up to 1 second during a write. Writing of data outside the legal ranges or invalid time stamp combinations results in unpredictable behavior.

29.2.6 Real-Time Clock Interrupts

At least six sources for interrupts are available, namely RT0PSIFG, RT1PSIFG, RTCRDYIFG, RTCTEVIFG, RTCAIFG, and RTCOFIFG. These flags are prioritized and combined to source a single interrupt vector. The interrupt vector register (RTCIV) is used to determine which flag requested an interrupt.

The highest-priority enabled interrupt generates a number in the RTCIV register (see register description). This number can be evaluated or added to the program counter (PC) to automatically enter the appropriate software routine. Disabled RTC interrupts do not affect the RTCIV value.

Writes into RTCIV register clear all pending interrupt conditions. Reads from RTCIV register clear the highest priority pending interrupt condition. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. In addition, all flags can be cleared by software.

The user-programmable alarm event sources the real-time clock interrupt, RTCAIFG. Setting RTCAIE enables the interrupt. In addition to the user-programmable alarm, the RTC_C module provides for an interval alarm that sources real-time clock interrupt, RTCTEVIFG. The interval alarm can be selected to cause an alarm event when RTCMIN changed or RTCHOUR changed, every day at midnight (00:00:00) or every day at noon (12:00:00). The event is selectable with the RTCTEV bits. Setting the RTCTEVIE bit enables the interrupt.

The RTCRDY bit sources the real-time clock interrupt, RTCRDYIFG, and is useful in synchronizing the read of time registers with the system clock. Setting the RTCRDYIE bit enables the interrupt.

RT0PSIFG can be used to generate interrupt intervals selectable by the RT0IP bits. RT0PS is sourced with low-frequency oscillator clock at 32768 Hz, so intervals of 16384 Hz, 8192 Hz, 4096 Hz, 2048 Hz, 1024 Hz, 512 Hz, 256 Hz, or 128 Hz are possible. Setting the RT0PSIE bit enables the interrupt.

RT1PSIFG can be used to generate interrupt intervals selectable by the RT1IP bits. RT1PS is sourced with the output of RT0PS, which is 128 Hz (32768/256 Hz). Therefore, intervals of 64 Hz, 32 Hz, 16 Hz, 8 Hz, 4 Hz, 2 Hz, 1 Hz, or 0.5 Hz are possible. Setting the RT1PSIE bit enables the interrupt.

NOTE: Changing RT0IP or RT1IP

Changing the settings of the interrupt interval bits RT0IP or RT1IP while the corresponding prescaler is running or is stopped in a non-zero state can result in setting the corresponding interrupt flags.

The RTCOFIFG bit flags the failure of the 32-kHz crystal oscillator. Its main purpose is to wake up the CPU from LPM3.5 if an oscillator failure occurs. On devices with separate supply for RTC, this flag also stores a failure event that occurs when the core supply is not available.

NOTE: Disable the RTC interrupts when changing the V_{CORE} level. The process of changing the V_{CORE} level disables SYS_H , and RTC_C is accessible only when SVS_H is enabled.

29.2.6.1 RTCIV Software Example

The following software example shows the recommended use of RTCIV and the handling overhead. The RTCIV value is added to the PC to automatically jump to the appropriate routine.

The numbers at the right margin show the necessary CPU cycles for each instruction. The software overhead for different interrupt sources includes interrupt latency and return-from-interrupt cycles, but not the task handling itself.

; Interrupt handler for RTC interrupt flags.

```

RTC_HND                                ; Interrupt latency                6
    ADD    &RTCIV,PC                    ; Add offset to Jump table        3
    RETI                                     ; Vector 0: No interrupt            5
    JMP     RTCOFIFG_HND                 ; Vector 2: RTCOFIFG              2
    JMP     RTCRDYIFG_HND                ; Vector 4: RTCRDYIFG            2
    JMP     RTCTEVIFG_HND                ; Vector 6: RTCTEVIFG           2
    JMP     RTCAIFG                      ; Vector 8: RTCAIFG              5
    JMP     RT0PSIFG                     ; Vector A: RT0PSIFG             5
    JMP     RT1PSIFG                     ; Vector C: RT1PSIFG            5
    RETI                                     ; Vector E: Reserved            5

RTCOFIFG_HND                           ; Vector 2: RTCOFIFG Flag
...                                     ; Task starts here
    RETI                                     ; Back to main program          5

RTCRDYIFG_HND                           ; Vector 4: RTCRDYIFG Flag
...                                     ; Task starts here
    RETI                                     ; Back to main program          5

RTCTEVIFG_HND                           ; Vector 6: RTCTEVIFG
...                                     ; Task starts here
    RETI                                     ; Back to main program          5

RTCAIFG_HND                             ; Vector 8: RTCAIFG
...                                     ; Task starts here
    RETI                                     ; Back to main program          5

RT0PSIFG_HND                             ; Vector A: RT0PSIFG
...                                     ; Task starts here
    RETI                                     ; Back to main program          5

RT1PSIFG_HND                             ; Vector C: RT1PSIFG

```

```

...           ; Task starts here
RETI          ; Back to main program      5

```

29.2.7 Real-Time Clock Calibration for Crystal Offset Error

The RTC_C module can be calibrated for crystal manufacturing tolerance or offset error to enable better accuracy of time keeping accuracy. The crystal frequency error of up to ± 240 ppm can be calibrated smoothly over a period of 60 seconds. RTCOCAL_L register is used to adjust the frequency. The calibration value is written into RTCOCAL_L register, and each LSB in this register represent approximately ± 1 -ppm correction based on RTCOCALS bit in RTCOCAL_H register. When RTCOCALS bit is set (up calibration), each LSB in RTCOCAL_L represent +1-ppm adjustment. When RTCOCALS is cleared (down calibration), each LSB in RTCOCAL_L represent -1-ppm adjustment to frequency. Both RTCOCAL_L and RTCOCAL_H registers are protected and require RTC_C to be unlocked before writing into these registers.

29.2.7.1 Calibration Frequency

To calibrate the frequency, the RTCCLK output signal is available at a pin. RTCCALFx bits in RTCCTL3 register can be used to select the frequency rate of the output signal. When RTCCALFx = 00, no signal is output on RTCCLK pin. The other settings of RTCCALFx select one the three frequencies: 512 Hz, 256 Hz, or 1 Hz. RTCCLK can be measured, and the result of this measurement can be applied to the RTCOCALS and RTCOCALx bits to effectively reduce the initial offset of the clock.

29.2.7.1.1 Calibration Mechanism

RTCOCAL_L is an 8-bit register. Software can write a value of up to 256 ppm into this register, but the maximum frequency error that can be corrected is only 240 ppm. Software must make sure to write legal values into this register. A read from RTCOCAL always returns the value that was written by software. Real-time clock offset error calibration is inactive when RTC_C is not enabled (RTCHOLD = 0) or when RTCOCALx bits are zero. RTCOCAL should only be written when RTCHOLD = 1. Writing RTCOCAL resets temperature compensation to zero.

In RTC_C, the offset error calibration takes place over a period of 60 seconds. To achieve approximately ± 1 -ppm correction, the 16-kHz clock (Q0 output of RT0PS) is adjusted to add or subtract one clock pulse. For +1-ppm correction, one clock pulse is added to the 16-kHz clock, and for -1-ppm correction, one clock pulse is subtracted from the 16-kHz clock. This correction happens once every quarter second until the programmed ppm error is compensated.

$$f_{\text{ACLK,meas}} < 32768 \text{ Hz} \rightarrow \text{RTCOCALS} = 1, \text{RTCOCALx} = \text{Round} (60 \times 16384 \times (1 - f_{\text{ACLK,meas}}/32768))$$

$$f_{\text{ACLK,meas}} \geq 32768 \text{ Hz} \rightarrow \text{RTCOCALS} = 0, \text{RTCOCALx} = \text{Round} (60 \times 16384 \times (1 - f_{\text{ACLK,meas}}/32768))$$

As an example for up calibration, when the measured frequency is 511.9658 Hz against the reference frequency of 512 Hz, the frequency error is approximately 67 ppm low. To increase the frequency by 67 ppm, RTCOCALS should be set, and RTCOCALx should be set to $\text{Round} (60 \times 16384 \times (1 - 511.9658 \times 64 / 32768)) = 66$.

As an example for down calibration, when the measured frequency is 512.0241 Hz against the reference frequency of 512 Hz, the frequency error is approximately 47 ppm high. To decrease the frequency by 47 ppm, RTCOCALS should be cleared, and RTCOCALx should be set to $\text{Round} (60 \times 16384 \times (1 - 512.0241 \times 64 / 32768)) = 46$.

All three possible output frequencies (512 Hz, 256 Hz, and 1 Hz) at RTCCLK pin are affected by calibration settings. RT0PS interrupt triggered by RT0PS - Q0 (RT0IPx = 000) is based on the uncalibrated clock, while RT0PS interrupt triggered by RT0PS - Q1 to Q7 (RT0IPx $\neq$ 000) is based on the calibrated clock. RT1PS interrupt (RT1PSIFG) and RTC counter interrupt (RTCTEVIFG) are also based on the calibrated clock.

29.2.8 Real-Time Clock Compensation for Crystal Temperature Drift

The frequency output of the crystal varies considerably due to drift in temperature. It would be necessary to compensate the real-time clock for this temperature drift for higher time keeping accuracy from standard crystals. A hybrid software and hardware approach can be followed to achieve temperature compensation for RTC_C.

The software can make use of an (on-chip) temperature sensor to measure the temperature at desired intervals (for example, once every few seconds or minutes). The temperature sensor parameters are calibrated at production and stored in the nonvolatile memory. Using the temperature sensor parameters and the measured temperature, software can do parabolic calculations to find out the corresponding frequency error in ppm.

This frequency error can be written into RTCTCMP_L register for temperature compensation. RTCTCMP_L is an 8-bit register that allows correction for a frequency error up to ± 240 ppm. Each LSB in this register represent ± 1 ppm based on the RTCTCMPS bit in the RTCTCMP_H register. When RTCTCMPS bit is set, each LSB in RTCTCMP represents +1-ppm adjustment (up calibration). When RTCTCMPS is cleared, each LSB in RTCTCMP represents -1-ppm adjustment (down calibration). RTCTCMP register is not protected and can be written any time without unlocking RTC_C.

29.2.8.1 Temperature Compensation Scheme

RTCTCMP_L is an 8-bit register. Software can write up to value of 256 ppm into this register, but the maximum frequency error that can be corrected including the crystal offset error is 240 ppm. Real-time clock temperature compensation is inactive when RTC_C is not enabled (RTCHOLD = 0) or when RTCTCMPx bits are zero.

When the temperature compensation value is written into RTCTCMP_L, it is added to the offset error calibration value, and the resulting value is taken into account from next calibration cycle onwards. The ongoing calibration cycle is not affected by writes into the RTCTCMP register. The maximum frequency error that can be corrected to account for both offset error and temperature variation is ± 240 ppm. This means the sign addition of offset error value and temperature compensation value should not exceed maximum of ± 240 ppm; otherwise, the excess value above ± 240 ppm is ignored by hardware. Reading from the RTCTCMP register at any time returns the cumulative value which is the signed addition of RTCOCALx and RTCTCMPx values. (Note that writing RTCOCAL resets the temperature compensation value to zero.)

For example, when RTCOCAL value is +150 ppm, and the value written into RTCTCMP is +200 ppm, the effective value taken in for next calibration cycle is +240 ppm. Software is expected to do temperature measurement at certain regularity, calculate the frequency error, and write into RTCTCMP register to not exceed the maximum limit of ± 240 ppm.

Changing the sign bit by writing to RTCTCMP_H becomes effective only after also writing RTCTCMP_L. Thus TI recommends writing the sign bit together with compensation value as a 16-bit value into RTCTCMP.

29.2.8.2 Writing to RTCTCMP Register

Because the system clock can be asynchronous to the RTC_C clock source, the RTCTCRDY bit in the RTCTCMP_H register should be considered for reliable writing into RTCTCMP register. RTCTCRDY is a read-only bit that is set when the hardware is ready to take in the new temperature compensation value. A write to RTCTCMP should be avoided when RTCTCRDY bit is reset. Writes into RTCTCMP register when RTCTCRDY is reset are ignored.

RTCTCOK is a status bit that indicates if the write to RTCTCMP register is successful or not. RTCTCOK is set if the write to RTCTCMP is successful and reset if the write is unsuccessful. The status remains the same until the next write to the RTCTCMP register. If the write to RTCTCMP is unsuccessful, then the user needs to attempt writing into RTCTCMP again when RTCTCRDY is set.

Figure 29-2 shows the scheme for real-time clock offset error calibration and temperature compensation.

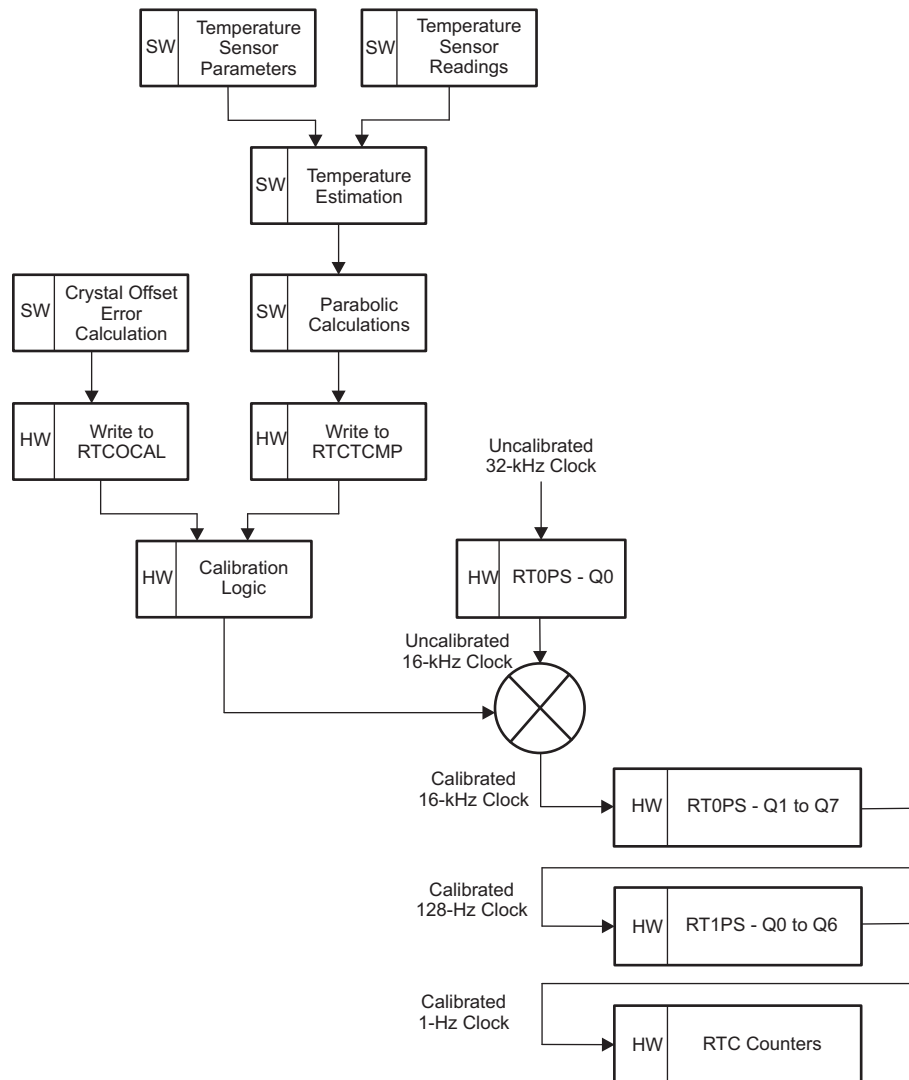


Figure 29-2. RTC_C Offset Error Calibration and Temperature Compensation Scheme

29.2.8.3 Temperature Measurement and Updates to RTC_C

The user may wish to perform temperature measurement once every few seconds or once every minute or once in several minutes. Writing to RTCTCMP register for temperature compensation is effective always once in one minute. This means that if the user performs temperature measurement every minute and updates RTCTCMP register with the frequency error, compensation would immediately work fine. But if software performs temperature measurement more frequently than once per minute (for example once every 5 seconds) then it needs to average the error over one minute and update RTCTCMP register once per minute. If the software performs temperature measurement less frequently than once per minute (for example, once every 5 minutes) then it needs to calculate the frequency error for the measured temperature and write into RTCTCMP register. The value written into RTCTCMP in this case would be effective until it is updated again by software.

29.2.9 Real-Time Clock Operation in LPM3.5 Low-Power Mode

The regulator of the Power Management Module (PMM) is disabled when the device enters LPM3.5, which causes most of the RTC_C configuration registers to be lost; only the counters and calibration registers are retained. [Table 29-2](#) shows which registers are retained in LPM3.5. Also the configuration of the interrupt enables is stored so that the configured interrupts can cause a wakeup upon exit from LPM3.5. Interrupt flags that are set before entering LPM3.5 are cleared upon entering LPM3.5 (Note: this can only happen if the corresponding interrupt is not enabled). The interrupt flags RTCTEVIFG, RTCAIFG, RT1PSIFG, and RTCOFIFG can be used as RTC_C wakeup interrupt sources. Any interrupt event that occurs during LPM3.5 is stored in the corresponding flags, but only enabled interrupts can wake up the device. After restoring the configuration registers (and clearing LOCKLPM5), the interrupts can be serviced as usual.

The detailed flow is as follows:

1. Set all I/Os to general-purpose I/Os and configure as needed. Optionally, configure input interrupt pins for wakeup. Configure RTC_C interrupts for wake-up (set RTCTEVIE, RTCAIE, RT1PSIE, or RTCOFIE. If the alarm interrupt is also used as wake-up event, the alarm registers must be configured as needed).
2. Enter LPM3.5 with LPM3.5 entry sequence:


```
bic #RTCHOLD, &RTCCTL13
bis #PMMKEY + REGOFF, &PMMCTL0
bis #LPM4, SR
```
3. LOCKLPM5 is automatically set by hardware upon entering LPM3.5, the core voltage regulator is disabled, and all clocks are disabled except for the 32-kHz crystal oscillator clock as the RTC_C is enabled with RTCHOLD = 0.
4. An LPM3.5 wake-up event like an edge on a wake-up input pin or an RTC_C interrupt event starts the BOR entry sequence and the core voltage regulator. All peripheral registers are set to their default conditions. The I/O pin state and the interrupt configuration for the RTC_C remain locked.
5. The device can be configured. The I/O configuration and the RTC_C interrupt configuration that was not retained during LPM3.5 should be restored to the values that they had before entering LPM3.5. Then the LOCKLPM5 bit can be cleared, which releases the I/O pin conditions and the RTC_C interrupt configuration. Registers that are retained during LPM3.5 should not be altered before LOCKLPM5 is cleared.
6. After enabling I/O and RTC_C interrupts, the interrupt that caused the wake-up can be serviced.

If the RTC_C is enabled (RTCHOLD = 0), the 32-kHz oscillator remains active during LPM3.5. The fault detection also remains functional. If a fault occurs during LPM3.5 and the RTCOFIE was set before entering LPM3.5, a wake-up event is issued.

29.3 RTC_C Operation - Device-Dependent Features

29.3.1 Counter Mode

NOTE: This feature is available only on selected devices. See the device-specific data sheet to determine if this feature is available.

The RTC_C module can be configured as a real-time clock with calendar function (calendar mode) or as a 32-bit general-purpose counter (counter mode) with the RTCMODE bit.

Counter mode is selected when RTCMODE is reset. In this mode, a 32-bit counter is provided that is directly accessible by software. Figure 29-3 shows the functional block diagram of a RTC_C module in counter mode (RTCMODE = 0).

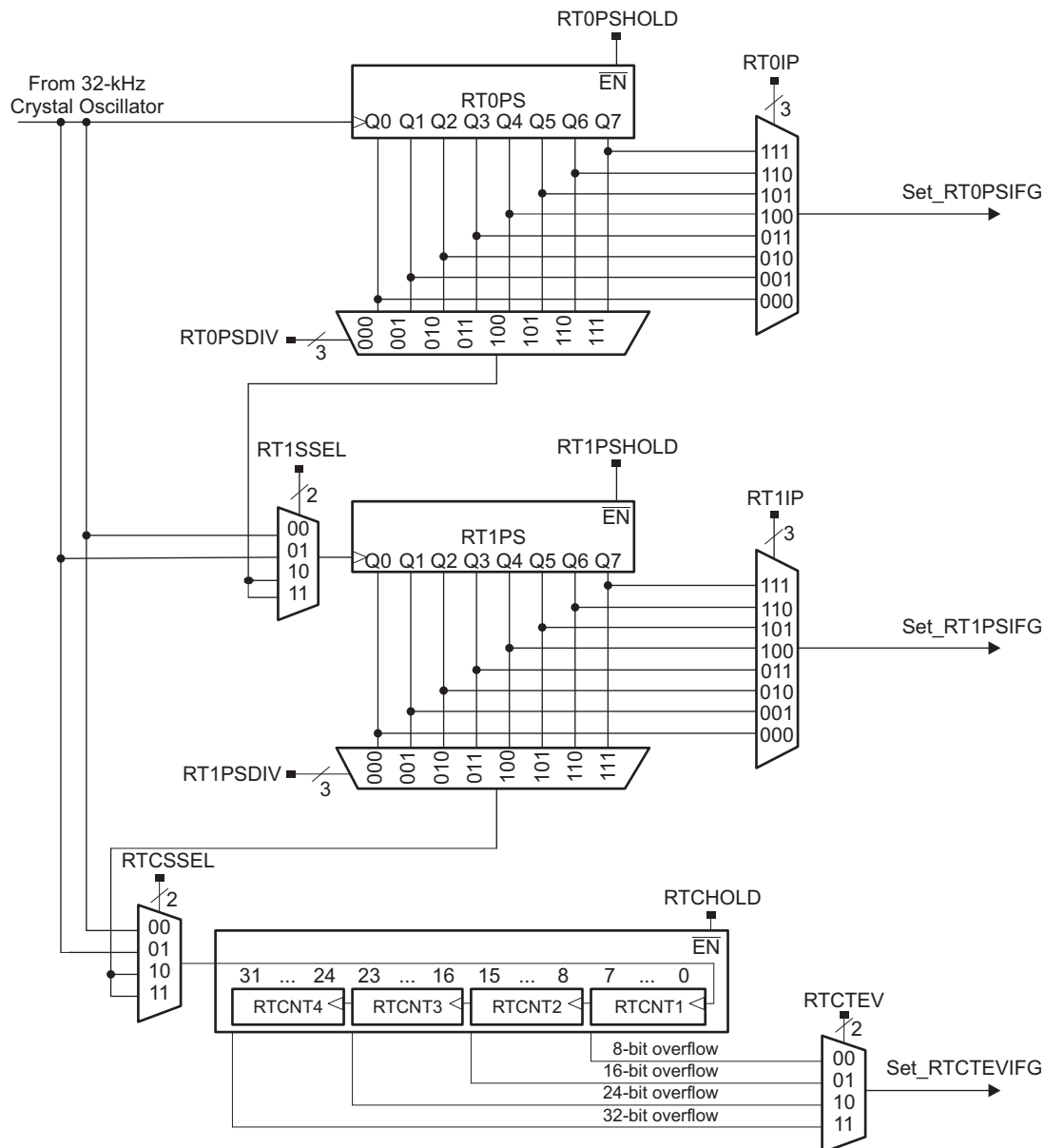


Figure 29-3. RTC_C Functional Block Diagram in Counter Mode (RTCMODE = 0)

29.3.1.1 Switching between Calendar and Counter Mode

To switch from calendar mode to counter mode do the following steps:

1. Stop the RTC with RTCHOLD=1.
2. Clear the RTCMODE bit to enable the counter mode.
3. Initialize the count registers (RTCNT1, RTCNT2, RTCNT3, RTCNT4) and the prescale counters (RT0PS, RT1PS) as needed. (The switch from calendar mode to counter mode **does not** reset the count value (RTCNT1, RTCNT2, RTCNT3, RTCNT4) or the prescale counters (RT0PS, RT1PS). These registers must be configured by user software before use.)
4. Clear the RTCHOLD bit to start the counters.

To switch back from counter mode to calendar mode do the following steps:

1. Stop the counters with RTCHOLD=1.
2. Set RTCMODE=1 to enable the calendar mode.
3. Initialize the clock and calendar registers and the prescale counters (RT0PS, RT1PS) as needed. (The switch from counter mode to calendar mode **does not** reset the clock and calendar registers nor the prescale counters (RT0PS, RT1PS). These registers must be configured by user software before use.)
4. Clear the RTCHOLD bit to start the RTC.

29.3.1.2 Counter Mode Operation

The clock that increments the counter can be sourced from the 32-kHz crystal oscillator or from prescaled versions of the 32-kHz crystal oscillator clock. Prescaled versions are sourced from the prescale dividers (RT0PS and RT1PS). RT0PS and RT1PS can output /2, /4, /8, 16, /32, /64, /128, and /256 versions of the 32-kHz clock. The output of RT0PS can be cascaded with RT1PS. The cascaded output can also be used as a clock source input to the 32-bit counter.

Four individual 8-bit counters are cascaded to provide the 32-bit counter. This provides 8-bit, 16-bit, 24-bit, or 32-bit overflow intervals of the counter clock. The RTCTEV bits select the respective trigger event. An RTCTEV event can trigger an interrupt by setting the RTCTEVIE bit. Each counter, RTCNT1 through RTCNT4, is individually accessible and may be written.

RT0PS and RT1PS can be configured as two 8-bit counters or cascaded into a single 16-bit counter. RT0PS and RT1PS can be halted on an individual basis by setting their respective RT0PSHOLD and RT1PSHOLD bits. When RT0PS is cascaded with RT1PS, setting RT0PSHOLD causes both RT0PS and RT1PS to be halted. The 32-bit counter can be halted several ways, depending on the configuration. If the 32-bit counter is sourced directly by the 32-kHz crystal clock, it can be halted by setting RTCHOLD. If it is sourced from the output of RT1PS, it can be halted by setting RT1PSHOLD or RTCHOLD. Finally, if it is sourced from the cascaded outputs of RT0PS and RT1PS, it can be halted by setting RT0PSHOLD, RT1PSHOLD, or RTCHOLD.

NOTE: Accessing the RTCNT1, RTCNT2, RTCNT3, RTCNT4, RT0PS, RT1PS registers

When the counter clock is asynchronous to the CPU clock, any read from any RTCNT1, RTCNT2, RTCNT3, RTCNT4, RT0PS, or RT1PS register should occur while the counter is not operating. Otherwise, the results may be unpredictable. Alternatively, the counter may be read multiple times while operating, and a majority vote taken in software to determine the correct reading. Any write to these registers takes effect immediately.

NOTE: For reliable update to all Counter Mode registers

Depending on the cascading of counters, when a write occurs, hold all subsequent counters. For example, if RTPS0 is being updated, set RTCPS1HOLD = 1, and if RTPS1 is being updated, set RTCHOLD = 1.

29.3.1.3 Real-Time Clock Interrupts in Counter Mode

In counter mode, four interrupt sources are available: RT0PSIFG, RT1PSIFG, RTCTEVIFG, and RTCOFIFG. RTCAIFG and RTCRDYIFG are cleared. RTCRDYIE and RTCAIE are don't care.

RT0PSIFG can be used to generate interrupt intervals selectable by the RT0IP bits. In counter mode, divide ratios of /2, /4, /8, /16, /32, /64, /128, and /256 of the clock source are possible. Setting the RT0PSIE bit enables the interrupt.

RT1PSIFG can be used to generate interrupt intervals selectable by the RT1IP bits. In counter mode, RT1PS is sourced with low-frequency oscillator clock, or the output of RT0PS, so divide ratios of /2, /4, /8, /16, /32, /64, /128, and /256 of the respective clock source are possible. Setting the RT1PSIE bit enables the interrupt.

In Counter Mode, the RTC_C module provides for an interval timer that sources real-time clock interrupt, RTCTEVIFG. The interval timer can be selected to cause an interrupt event when an 8-bit, 16-bit, 24-bit, or 32-bit overflow occurs within the 32-bit counter. The event is selectable with the RTCTEV bits. Setting the RTCTEVIE bit enables the interrupt.

The RTCOFIFG bit flags a failure of the 32-kHz crystal oscillator. It's main purpose is to wake-up the CPU from LPM3.5 in case an oscillator failure occurred.

29.3.2 Real-Time Clock Event/Tamper Detection With Time Stamp

NOTE: This feature is available only on selected devices. See the device-specific data sheet to determine if this feature is available.

The RTC_C module provides an external event or tamper detection and time stamp for up to two external events. The pins RTCCAP0 and RTCCAP1⁽²⁾ can be used as an event or tamper detection input of an external switch (mechanical or electronic). After device power-up, this feature can be enabled by setting the TCEN bit in the RTCTCCTL0 register. Event and tamper detection with time stamp is supported in all MSP430 operating modes, as long as there is a valid RTC power supply.

- When there is an event on RTCCAPx pin and the time capture feature is enabled (TCEN = 1), the corresponding CAPEV bit in RTCCAPxCTL register is set and the corresponding time stamp information (seconds, minutes, hours, day of month, month and year) is stored in the respective backup registers (RTCSECBAX, RTCMINBAKx, RTCHOURBAKx, RTCDAYBAKx, RTCMONBAKx and RTCYEARBAKx).
- In case of multiple events, **ONLY** the time stamp of the event that occurred first is stored in the respective backup registers. After CAPEV is set by the first event on RTCCAPx, all subsequent events on RTCCAPx are ignored until the CAPEV bit is cleared by the user.
- The CAPES bit in the RTCCAPxCTL register sets the event edge for the corresponding RTCCAPx pin.
 - Bit = 0: CAPEV flag is set with a low-to-high transition.
 - Bit = 1: CAPEV flag is set with a high-to-low transition.

NOTE: Writing to CAPESx

Writing to CAPES can result in setting the corresponding interrupt flags.

CAPESx	RTCCAPx	RTCCAPIFG
0 → 1	0	May be set
0 → 1	1	Unchanged
1 → 0	0	Unchanged
1 → 0	1	May be set

- The interrupt flag RTCCAPIFG is set when any of the individual CAPEV bits are set. If the RTCIV is read, RTCCAPIFG is cleared but not the status flags (CAPEV bits). They are then read by the CPU and must be cleared by software only.
- By setting the RTCCAPIE bit, an event on RTCCAPx generates an interrupt. This interrupt can be used as LPM3.5 or LPM4.5 wake-up event in modules that support LPM3.5 or LPM4.5.
- When the time capture feature is enabled (TCEN = 1), all of the backup registers (RTCSECBAX, RTCMINBAKx, RTCHOURBAKx, RTCDAYBAKx, RTCMONBAKx, and RTCYEARBAKx) are read-only to user and can be written only by the RTC hardware. When RTCBCD = 1 and TCEN = 1, BCD format is selected for the backup registers. If the backup registers were written to by the hardware before TCEN was set, then the previous values are retained until there is a time capture event that overrides the values with the time stamp.
- When the time capture feature is disabled (TCEN = 0), all of the backup registers (RTCSECBAX, RTCMINBAKx, RTCHOURBAKx, RTCDAYBAKx, RTCMONBAKx, and RTCYEARBAKx) can be written only by the CPU. When TCEN = 0, the RTCBCD bit setting is ignored for the backup registers. The data in the backup registers when TCEN = 1 is retained until the user writes new values after TCEN is cleared.
- When TCEN is cleared, all CAPEV bits and RTCCAPIFG are cleared.
- [Table 29-1](#) shows how to use the DIR, REN, and OUT bits in RTCCAPxCTL for proper configuration of RTCCAPx pins.

⁽²⁾ These pins are present only on devices that support this feature of RTC_C. Refer to the device-specific data sheet to determine the availability of this feature.

Table 29-1. RTCCAPx Pin Configuration

DIR	REN	OUT	RTCCAPx Configuration
0	0	x	Input
0	1	0	Input with pulldown resistor
0	1	1	Input with pullup resistor
1	x	x	Output

29.3.2.1 Real-Time Clock Event/Tamper Detection Interrupts

With the event or tamper detection feature, one additional interrupt sources is available, RTCCAPIFG. This flag is prioritized and combined with the other interrupt flags to source a single interrupt vector. The interrupt vector register (RTCIV) is used to determine which flag requested an interrupt.

The RTCCAPIFG bit flags the occurrence of a tamper event. The exact source of the interrupt among multiple tamper events can be found out by reading the CAPEV bit in the respective RTCCAPxCTL registers (one per tamper source). When RTCIV is read, the RTCCAPIFG is cleared but not the status flags (CAPEV bits).

29.4 RTC_C Registers

The RTC_C module registers are shown in [Table 29-2](#). This table also shows which registers are key protected and which are retained during LPM3.5. The registers that are retained during LPM3.5 and given with a reset value are not reset on POR. Registers that are not retained during LPM3.5 must be restored after exit from LPM3.5.

The high-side SVS must not be disabled by software if the real-time clock feature is needed. When the high-side SVS is disabled, the RTC_C registers with LPM3.5 retention are not accessible by the CPU.

The base address for the RTC_C module registers can be found in the device-specific data sheet. The address offsets are shown in [Table 29-2](#).

The additional registers that are available if Event/Tamper Detection is implemented are shown in [Table 29-3](#) together with the corresponding address offsets.

If the counter mode is supported, the register aliases shown in [Table 29-4](#) can be used to access the counter registers.

NOTE: Most registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 29-2. RTC_C Registers

Offset	Acronym	Register Name	Type	Access	Reset	Key Protected	LPM3.5 Retention
00h	RTCCTL0	Real-Time Clock Control 0	Read/write	Word	9600h	yes	not retained ⁽¹⁾
00h	RTCCTL0_L	Real-Time Clock Control 0 Low	Read/write	Byte	00h	yes	not retained ⁽¹⁾
01h	RTCCTL0_H	Real-Time Clock Control 0 High	Read/write	Byte	96h	n/a	not retained ⁽¹⁾
02h	RTCCTL13	Real-Time Clock Control 1, 3	Read/write	Word	0070h	yes	high byte retained
02h	RTCCTL1	Real-Time Clock Control 1	Read/write	Byte	70h	yes	not retained
	or RTCCTL13_L						
03h	RTCCTL3	Real-Time Clock Control 3	Read/write	Byte	00h	yes	retained
	or RTCCTL13_H						
04h	RTCOCAL	Real-Time Clock Offset Calibration	Read/write	Word	0000h	yes	retained
04h	RTCOCAL_L		Read/write	Byte	00h	yes	retained
05h	RTCOCAL_H		Read/write	Byte	00h	yes	retained
06h	RTCTCMP	Real-Time Clock Temperature Compensation	Read/write	Word	4000h	no	partially retained
06h	RTCTCMP_L		Read/write	Byte	00h	no	partially retained
07h	RTCTCMP_H		Read/write	Byte	40h	no	partially retained
08h	RTCPS0CTL	Real-Time Prescale Timer 0 Control	Read/write	Word	0100h	no	not retained
08h	RTCPS0CTL_L		Read/write	Byte	00h	no	not retained
09h	RTCPS0CTL_H		Read/write	Byte	01h	no	not retained
0Ah	RTCPS1CTL	Real-Time Prescale Timer 1 Control	Read/write	Word	0100h	no	not retained
0Ah	RTCPS1CTL_L		Read/write	Byte	00h	no	not retained
0Bh	RTCPS1CTL_H		Read/write	Byte	01h	no	not retained

⁽¹⁾ Some bits in this register are retained. See the register description to determine which bits are retained.

Table 29-2. RTC_C Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Key Protected	LPM3.5 Retention
0Ch	RTCPS	Real-Time Prescale Timer 0, 1 Counter	Read/write	Word	none	yes	retained
0Ch	RT0PS	Real-Time Prescale Timer 0 Counter	Read/write	Byte	none	yes	retained
	or RTCPS_L						
0Dh	RT1PS	Real-Time Prescale Timer 1 Counter	Read/write	Byte	none	yes	retained
	or RTCPS_H						
0Eh	RTCIV	Real Time Clock Interrupt Vector	Read	Word	0000h	no	not retained ⁽¹⁾
10h	RTCTIM0	Real-Time Clock Seconds, Minutes	Read/write	Word	undefined	yes	retained
10h	RTCSEC	Real-Time Clock Seconds	Read/write	Byte	undefined	yes	retained
	or RTCTIM0_L						
11h	RTCMIN	Real-Time Clock Minutes	Read/write	Byte	undefined	yes	retained
	or RTCTIM0_H						
12h	RTCTIM1	Real-Time Clock Hour, Day of Week	Read/write	Word	undefined	yes	retained
12h	RTCHOUR	Real-Time Clock Hour	Read/write	Byte	undefined	yes	retained
	or RTCTIM1_L						
13h	RTCDOW	Real-Time Clock Day of Week	Read/write	Byte	undefined	yes	retained
	or RTCTIM1_H						
14h	RTCDATE	Real-Time Clock Date	Read/write	Word	undefined	yes	retained
14h	RTCDAY	Real-Time Clock Day of Month	Read/write	Byte	undefined	yes	retained
	or RTCDATE_L						
15h	RTCMON	Real-Time Clock Month	Read/write	Byte	undefined	yes	retained
	or RTCDATE_H						
16h	RTCYEAR	Real-Time Clock Year ⁽²⁾	Read/write	Word	undefined	yes	retained
18h	RTCAMINHR	Real-Time Clock Minutes, Hour Alarm	Read/write	Word	undefined	no	retained
18h	RTCAMIN	Real-Time Clock Minutes Alarm	Read/write	Byte	undefined	no	retained
	or RTCAMINHR_L						
19h	RTCAHOUR	Real-Time Clock Hours Alarm	Read/write	Byte	undefined	no	retained
	or RTCAMINHR_H						
1Ah	RTCADOWDAY	Real-Time Clock Day of Week, Day of Month Alarm	Read/write	Word	undefined	no	retained
1Ah	RTCADOW	Real-Time Clock Day of Week Alarm	Read/write	Byte	undefined	no	retained
	or RTCADOWDAY_L						
1Bh	RTCADAY	Real-Time Clock Day of Month Alarm	Read/write	Byte	undefined	no	retained
	or RTCADOWDAY_H						
1Ch	BIN2BCD	Binary-to-BCD conversion register	Read/write	Word	0000h	no	not retained
1Eh	BCD2BIN	BCD-to-binary conversion register	Read/write	Word	0000h	no	not retained

⁽²⁾ Do not access the year register RTCYEAR in byte mode.

Table 29-3. RTC_C Event and Tamper Detection Registers

Offset	Acronym	Register Name	Type	Access	Reset	Key Protected	LPM3.5 Retention
20h	RTCTCCTL0	Real-Time Clock Time Capture Control Register 0	Read/write	Byte	02h	yes	retained
21h	RTCTCCTL1	Real-Time Clock Time Capture Control Register 1	Read/write	Byte	00h	yes	not retained
22h	RTCCAP0CTL	Tamper Detect Pin 0 Control Register	Read/write	Byte	00h	yes	not retained
23h	RTCCAP1CTL	Tamper Detect Pin 1 Control Register	Read/write	Byte	00h	yes	not retained
30h	RTCSECBK0	Real-Time Clock Seconds Backup Register 0	Read/write	Byte	00h	yes	retained
31h	RTCMINBAK0	Real-Time Clock Minutes Backup Register 0	Read/write	Byte	00h	yes	retained
32h	RTCHOURBAK0	Real-Time Clock Hours Backup Register 0	Read/write	Byte	00h	yes	retained
33h	RTCDAYBAK0	Real-Time Clock Days Backup Register 0	Read/write	Byte	00h	yes	retained
34h	RTCMONBAK0	Real-Time Clock Months Backup Register 0	Read/write	Byte	00h	yes	retained
36h	RTCYEARBAK0	Real-Time Clock year Backup Register 0	Read/write	Word	00h	yes	retained
38h	RTCSECBK1	Real-Time Clock Seconds Backup Register 1	Read/write	Byte	00h	yes	retained
39h	RTCMINBAK1	Real-Time Clock Minutes Backup Register 1	Read/write	Byte	00h	yes	retained
3Ah	RTCHOURBAK1	Real-Time Clock Hours Backup Register 1	Read/write	Byte	00h	yes	retained
3Bh	RTCDAYBAK1	Real-Time Clock Days Backup Register 1	Read/write	Byte	00h	yes	retained
3Ch	RTCMONBAK1	Real-Time Clock Months Backup Register 1	Read/write	Byte	00h	yes	retained
3Eh	RTCYEARBAK1	Real-Time Clock Year Backup Register 1	Read/write	Word	00h	yes	retained

Table 29-4. RTC_C Real-Time Clock Counter Mode Aliases

Offset	Acronym	Register Name	Type	Access	Reset	Key Protected	LPM3.5 Retention
10h	RTCCNT12	Real-Time Counter 1, 2	Read/write	Word	undefined	yes	retained
10h	RTCCNT1	Real-Time Counter 1	Read/write	Byte	undefined	yes	retained
11h	RTCCNT2	Real-Time Counter 2	Read/write	Byte	undefined	yes	retained
12h	RTCCNT34	Real-Time Counter 3, 4	Read/write	Word	undefined	yes	retained
12h	RTCCNT3	Real-Time Counter 3	Read/write	Byte	undefined	yes	retained
13h	RTCCNT4	Real-Time Counter 4	Read/write	Byte	undefined	yes	retained

29.4.1 RTCCTL0_L Register

Real-Time Clock Control 0 Low Register

Figure 29-4. RTCCTL0_L Register

7	6	5	4	3	2	1	0
RTCOFIE ⁽¹⁾	RTCDEVIE ⁽¹⁾	RTCAIE ⁽¹⁾	RTCRDYIE	RTCOFIG	RTCDEVIFG	RTCAIFG	RTCRDYIFG
rw-0	rw-0	rw-0	rw-0	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ The configuration of these bits is retained during LPMx.5 until LOCKLPM5 is cleared, but not the register bits themselves; therefore, reconfiguration is required after wakeup from LPMx.5 before clearing LOCKLPM5.

Table 29-5. RTCCTL0_L Register Description

Bit	Field	Type	Reset	Description
7	RTCOFIE	RW	0h	32-kHz crystal oscillator fault interrupt enable. This interrupt can be used as LPM3.5 wake-up event. 0b = Interrupt not enabled 1b = Interrupt enabled (LPM3.5 wake-up enabled)
6	RTCDEVIE	RW	0h	Real-time clock time event interrupt enable. In modules supporting LPM3.5 this interrupt can be used as LPM3.5 wake-up event. 0b = Interrupt not enabled 1b = Interrupt enabled (LPM3.5 wake-up enabled)
5	RTCAIE	RW	0h	Real-time clock alarm interrupt enable. In modules supporting LPM3.5 this interrupt can be used as LPM3.5 wake-up event. 0b = Interrupt not enabled 1b = Interrupt enabled (LPM3.5 wake-up enabled)
4	RTCRDYIE	RW	0h	Real-time clock ready interrupt enable 0b = Interrupt not enabled 1b = Interrupt enabled
3	RTCOFIG	RW	0h	32-kHz crystal oscillator fault interrupt flag. This interrupt can be used as LPM3.5 wake-up event. It also indicates a clock failure during backup operation. 0b = No interrupt pending 1b = Interrupt pending. A 32-kHz crystal oscillator fault occurred after last reset.
2	RTCDEVIFG	RW	0h	Real-time clock time event interrupt flag. In modules supporting LPM3.5 this interrupt can be used as LPM3.5 wake-up event. 0b = No time event occurred 1b = Time event occurred
1	RTCAIFG	RW	0h	Real-time clock alarm interrupt flag. In modules supporting LPM3.5 this interrupt can be used as LPM3.5 wake-up event. 0b = No time event occurred 1b = Time event occurred
0	RTCRDYIFG	RW	0h	Real-time clock ready interrupt flag 0b = RTC cannot be read safely 1b = RTC can be read safely

29.4.2 RTCCTL0_H Register

Real-Time Clock Control 0 High Register

Figure 29-5. RTCCTL0_H Register

7	6	5	4	3	2	1	0
RTCKEY							
rw-1	rw-0	rw-0	rw-1	rw-0	rw-1	rw-1	rw-0

Table 29-6. RTCCTL0_H Register Description

Bit	Field	Type	Reset	Description
7-0	RTCKEY	RW	96h	Real-time clock key. This register should be written with A5h to unlock RTC_C. A write with a value other than A5h locks the module. A read from this register always returns 96h.

29.4.3 RTCCTL1 Register

Real-Time Clock Control Register 1

Figure 29-6. RTCCTL1 Register

7	6	5	4	3	2	1	0
RTCBCD	RTCHOLD ⁽¹⁾	RTCMODE ⁽¹⁾	RTCRDY	RTCSSELx ⁽¹⁾		RTCDEVx ⁽¹⁾	
rw-(0)	rw-(1)	rw-(1)	r-(1)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ The configuration of these bits is retained during LPMx.5 until LOCKLPM5 is cleared, but not the register bits themselves; therefore, reconfiguration is required after wakeup from LPMx.5 before clearing LOCKLPM5.

Table 29-7. RTCCTL1 Register Description

Bit	Field	Type	Reset	Description
7	RTCBCD	RW	0h	Real-time clock BCD select. Selects BCD counting for real-time clock. Applies to calendar mode (RTCMODE = 1) only; setting is ignored in counter mode. 0b = Binary (hexadecimal) code selected 1b = Binary coded decimal (BCD) code selected
6	RTCHOLD	RW	1h	Real-time clock hold 0b = Real-time clock (32-bit counter or calendar mode) is operational. 1b = In counter mode (RTCMODE = 0), only the 32-bit counter is stopped. In calendar mode (RTCMODE = 1), the calendar is stopped as well as the prescale counters, RT0PS and RT1PS. RT0PSHOLD and RT1PSHOLD are don't care.
5	RTCMODE	RW	1h	Real-time clock mode. In RTC_C modules without counter mode support this bit is read-only and always reads 1. 0b = 32-bit counter mode 1b = Calendar mode. Switching between counter and calendar mode does not reset the real-time clock counter registers. These registers must be configured by user software before use.
4	RTCRDY	R	1h	Real-time clock ready 0b = RTC time values in transition (calendar mode only) 1b = RTC time values safe for reading (calendar mode only). This bit indicates when the real-time clock time values are safe for reading (calendar mode only). In counter mode, RTCRDY remains cleared.
3-2	RTCSSELx	RW	0h	Real-time clock source select. In counter mode, selects clock input source to the 32-bit counter. In calendar mode, these bits are don't care. The clock input is automatically set to the output of RT1PS. 00b = 32-kHz crystal oscillator clock 01b = 32-kHz crystal oscillator clock 10b = Output from RT1PS 11b = Output from RT1PS
1-0	RTCDEVx	RW	0h	Real-time clock time event Calendar Mode (RTCMODE = 1) 00b = Minute changed 01b = Hour changed 10b = Every day at midnight (00:00) 11b = Every day at noon (12:00) Counter Mode (RTCMODE = 0) 00b = 8-bit overflow 01b = 16-bit overflow 10b = 24-bit overflow 11b = 32-bit overflow

29.4.4 RTCCTL3 Register

Real-Time Clock Control 3 Register

Figure 29-7. RTCCTL3 Register

7	6	5	4	3	2	1	0
r0	r0	r0	Reserved	r0	r0	RTCCALFx ⁽¹⁾	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-8. RTCCTL3 Register Description

Bit	Field	Type	Reset	Description
7-2	Reserved	R	0h	Reserved. Always reads as 0.
1-0	RTCCALFx	RW	0h	Real-time clock calibration frequency. Selects frequency output to RTCCLK pin for calibration measurement. The corresponding port must be configured for the peripheral module function. The RTCCLK is not available in counter mode and remains low, and the RTCCALF bits are don't care. 00b = No frequency output to RTCCLK pin 01b = 512 Hz 10b = 256 Hz 11b = 1 Hz

29.4.5 RTCOCAL Register

Real-Time Clock Offset Calibration Register

Figure 29-8. RTCOCAL Register

15	14	13	12	11	10	9	8
RTCOALS ⁽¹⁾	Reserved						
rw-(0)	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
RTCOCALx ⁽¹⁾							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-9. RTCOCAL Register Description

Bit	Field	Type	Reset	Description
15	RTCOALS	RW	0h	Real-time clock offset error calibration sign. This bit decides the sign of offset error calibration. 0b = Down calibration. Frequency adjusted down. 1b = Up calibration. Frequency adjusted up.
14-8	Reserved	R	0h	Reserved. Always reads as 0.
7-0	RTCOCALx	RW	0h	Real-time clock offset error calibration. Each LSB represents approximately +1 ppm (RTCOALS = 1) or –1 ppm (RTCOALS = 0) adjustment in frequency. Maximum effective calibration value is ±240 ppm. Excess values written above ±240 ppm are ignored by hardware.

29.4.6 RTCTCMP Register

Real-Time Clock Temperature Compensation Register

Figure 29-9. RTCTCMP Register

15	14	13	12	11	10	9	8
RTCTCMPS ⁽¹⁾	RTCTCRDY ⁽¹⁾	RTCTCOK	Reserved				
rw-(0)	r-(1)	r-(0)	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
RTCTCMPx ⁽¹⁾							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-10. RTCTCMP Register Description

Bit	Field	Type	Reset	Description
15	RTCTCMPS	RW	0h	Real-time clock temperature compensation sign. This bit decides the sign of temperature compensation. ⁽¹⁾ 0b = Down calibration. Frequency adjusted down. 1b = Up calibration. Frequency adjusted up.
14	RTCTCRDY	R	1h	Real-time clock temperature compensation ready. This is a read only bit that indicates when the RTCTCMPx can be written. Write to RTCTCMPx should be avoided when RTCTCRDY is reset.
13	RTCTCOK	R	0h	Real-time clock temperature compensation write OK. This is a read-only bit that indicates if the write to RTCTCMP is successful or not. 0b = Write to RTCTCMPx is unsuccessful 1b = Write to RTCTCMPx is successful
12-8	Reserved	R	0h	Reserved. Always reads as 0.
7-0	RTCTCMPx	RW	0h	Real-time clock temperature compensation. Value written into this register is used for temperature compensation of RTC_C. Each LSB represents approximately +1 ppm (RTCTCMPS = 1) or –1 ppm (RTCTCMPS = 0) adjustment in frequency. Maximum effective calibration value is ±240 ppm. Excess values written above ±240 ppm are ignored by hardware.

⁽¹⁾ Changing the sign-bit by writing to RTCTCMP_H becomes effective only after also writing RTCTCMP_L.

29.4.7 RTCNT1 Register

Real-Time Clock Counter 1 Register – Counter Mode

Figure 29-10. RTCNT1 Register

7	6	5	4	3	2	1	0
RTCNT1							
rw	rw	rw	rw	rw	rw	rw	rw

Table 29-11. RTCNT1 Register Description

Bit	Field	Type	Reset	Description
7-0	RTCNT1	RW	undefined	The RTCNT1 register is the count of RTCNT1

29.4.8 RTCNT2 Register

Real-Time Clock Counter 2 Register – Counter Mode

Figure 29-11. RTCNT2 Register

7	6	5	4	3	2	1	0
RTCNT2							
rw	rw	rw	rw	rw	rw	rw	rw

Table 29-12. RTCNT2 Register Description

Bit	Field	Type	Reset	Description
7-0	RTCNT2	RW	undefined	The RTCNT2 register is the count of RTCNT2

29.4.9 RTCNT3 Register

Real-Time Clock Counter 3 Register – Counter Mode

Figure 29-12. RTCNT3 Register

7	6	5	4	3	2	1	0
RTCNT3							
rw	rw	rw	rw	rw	rw	rw	rw

Table 29-13. RTCNT3 Register Description

Bit	Field	Type	Reset	Description
7-0	RTCNT3	RW	undefined	The RTCNT3 register is the count of RTCNT3

29.4.10 RTCNT4 Register

Real-Time Clock Counter 4 Register – Counter Mode

Figure 29-13. RTCNT4 Register

7	6	5	4	3	2	1	0
RTCNT4							
rw	rw	rw	rw	rw	rw	rw	rw

Table 29-14. RTCNT4 Register Description

Bit	Field	Type	Reset	Description
7-0	RTCNT4	RW	undefined	The RTCNT4 register is the count of RTCNT4.

29.4.11 RTCSEC Register – Calendar Mode With Hexadecimal Format

Real-Time Clock Seconds Register – Calendar Mode With Hexadecimal Format

Figure 29-14. RTCSEC Register

7	6	5	4	3	2	1	0
0		Seconds					
r-0	r-0	rw	rw	rw	rw	rw	rw

Table 29-15. RTCSEC Register Description

Bit	Field	Type	Reset	Description
7-6	0	R	0h	Always 0
5-0	Seconds	RW	undefined	Seconds (0 to 59)

29.4.12 RTCSEC Register – Calendar Mode With BCD Format

Real-Time Clock Seconds Register – Calendar Mode With BCD Format

Figure 29-15. RTCSEC Register

7	6	5	4	3	2	1	0
0	Seconds – high digit			Seconds – low digit			
r-0	rw	rw	rw	rw	rw	rw	rw

Table 29-16. RTCSEC Register Description

Bit	Field	Type	Reset	Description
7	0	R	0h	Always 0
6-4	Seconds – high digit	RW	undefined	Seconds – high digit (0 to 5)
3-0	Seconds – low digit	RW	undefined	Seconds – low digit (0 to 9)

29.4.13 RTCMIN Register – Calendar Mode With Hexadecimal Format

Real-Time Clock Minutes Register – Calendar Mode With Hexadecimal Format

Figure 29-16. RTCMIN Register

7	6	5	4	3	2	1	0
0		Minutes					
r-0	r-0	rw	rw	rw	rw	rw	rw

Table 29-17. RTCMIN Register Description

Bit	Field	Type	Reset	Description
7-6	0	R	0h	Always 0
5-0	Minutes	RW	undefined	Minutes (0 to 59)

29.4.14 RTCMIN Register – Calendar Mode With BCD Format

Real-Time Clock Minutes Register – Calendar Mode With BCD Format

Figure 29-17. RTCMIN Register

7	6	5	4	3	2	1	0
0	Minutes – high digit			Minutes – low digit			
r-0	rw	rw	rw	rw	rw	rw	rw

Table 29-18. RTCMIN Register Description

Bit	Field	Type	Reset	Description
7	0	R	0h	Always 0
6-4	Minutes – high digit	RW	undefined	Minutes – high digit (0 to 5)
3-0	Minutes – low digit	RW	undefined	Minutes – low digit (0 to 9)

29.4.15 RTCHOUR Register – Calendar Mode With Hexadecimal Format

Real-Time Clock Hours Register – Calendar Mode With Hexadecimal Format

Figure 29-18. RTCHOUR Register

7	6	5	4	3	2	1	0
0			Hours				
r-0	r-0	r-0	rw	rw	rw	rw	rw

Table 29-19. RTCHOUR Register Description

Bit	Field	Type	Reset	Description
7-5	0	R	0h	Always 0
4-0	Hours	RW	undefined	Hours (0 to 23)

29.4.16 RTCHOUR Register – Calendar Mode With BCD Format

Real-Time Clock Hours Register – Calendar Mode With BCD Format

Figure 29-19. RTCHOUR Register

7	6	5	4	3	2	1	0
0		Hours – high digit		Hours – low digit			
r-0	r-0	rw	rw	rw	rw	rw	rw

Table 29-20. RTCHOUR Register Description

Bit	Field	Type	Reset	Description
7-6	0	R	0h	Always 0
5-4	Hours – high digit	RW	undefined	Hours – high digit (0 to 2)
3-0	Hours – low digit	RW	undefined	Hours – low digit (0 to 9)

29.4.17 RTCDOW Register – Calendar Mode

Real-Time Clock Day of Week Register – Calendar Mode

Figure 29-20. RTCDOW Register

7	6	5	4	3	2	1	0
0					Day of week		
r-0	r-0	r-0	r-0	r-0	rw	rw	rw

Table 29-21. RTCDOW Register Description

Bit	Field	Type	Reset	Description
7-3	0	R	0h	Always 0
2-0	Day of week	RW	undefined	Day of week (0 to 6)

29.4.18 RTCDAY Register – Calendar Mode With Hexadecimal Format

Real-Time Clock Day of Month Register – Calendar Mode With Hexadecimal Format

Figure 29-21. RTCDAY Register

7	6	5	4	3	2	1	0
0			Day of month				
r-0	r-0	r-0	rw	rw	rw	rw	rw

Table 29-22. RTCDAY Register Description

Bit	Field	Type	Reset	Description
7-5	0	R	0h	Always 0
4-0	Day of month	RW	undefined	Day of month (1 to 28, 29, 30, 31)

29.4.19 RTCDAY Register – Calendar Mode With BCD Format

Real-Time Clock Day of Month Register – Calendar Mode With BCD Format

Figure 29-22. RTCDAY Register

7	6	5	4	3	2	1	0
0		Day of month – high digit		Day of month – low digit			
r-0	r-0	rw	rw	rw	rw	rw	rw

Table 29-23. RTCDAY Register Description

Bit	Field	Type	Reset	Description
7-6	0	R	0h	
5-4	Day of month – high digit	RW	undefined	Day of month – high digit (0 to 3)
3-0	Day of month – low digit	RW	undefined	Day of month – low digit (0 to 9)

29.4.20 RTCMON Register – Calendar Mode With Hexadecimal Format

Real-Time Clock Month Register – Calendar Mode With Hexadecimal Format

Figure 29-23. RTCMON Register

7	6	5	4	3	2	1	0
0				Month			
r-0	r-0	r-0	r-0	rw	rw	rw	rw

Table 29-24. RTCMON Register Description

Bit	Field	Type	Reset	Description
7-4	0	R	0h	Always 0
3-0	Month	RW	undefined	Month (1 to 12)

29.4.21 RTCMON Register – Calendar Mode With BCD Format

Real-Time Clock Month Register – Calendar Mode With BCD Format

Figure 29-24. RTCMON Register

7	6	5	4	3	2	1	0
0			Month – high digit	Month – low digit			
r-0	r-0	r-0	rw	rw	rw	rw	rw

Table 29-25. RTCMON Register Description

Bit	Field	Type	Reset	Description
7-5	0	R	0h	Always 0
4	Month – high digit	RW	undefined	Month – high digit (0 or 1)
3-0	Month – low digit	RW	undefined	Month – low digit (0 to 9)

29.4.22 RTCYEAR Register – Calendar Mode With Hexadecimal Format

Real-Time Clock Year Low-Byte Register – Calendar Mode With Hexadecimal Format

Figure 29-25. RTCYEAR Register

15	14	13	12	11	10	9	8
0				Year – high byte			
r-0	r-0	r-0	r-0	rw	rw	rw	rw
7	6	5	4	3	2	1	0
rw	rw	rw	rw	rw	rw	rw	rw

Table 29-26. RTCYEAR Register Description

Bit	Field	Type	Reset	Description
15-12	0	R	0h	Always 0
11-8	Year – high byte	RW	undefined	Year – high byte. Valid values for Year are 0 to 4095.
7-0	Year – low byte	RW	undefined	Year – low byte. Valid values for Year are 0 to 4095.

29.4.23 RTCYEAR Register – Calendar Mode With BCD Format

Real-Time Clock Year Low-Byte Register – Calendar Mode With BCD Format

Figure 29-26. RTCYEAR Register

15	14	13	12	11	10	9	8
0	Century – high digit				Century – low digit		
r-0	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
Decade				Year – lowest digit			
rw	rw	rw	rw	rw	rw	rw	rw

Table 29-27. RTCYEAR Register Description

Bit	Field	Type	Reset	Description
15	0	R	0h	Always 0
14-10	Century – high digit	RW	undefined	Century – high digit (0 to 4)
11-8	Century – low digit	RW	undefined	Century – low digit (0 to 9)
7-4	Decade	RW	undefined	Decade (0 to 9)
3-0	Year – lowest digit	RW	undefined	Year – lowest digit (0 to 9)

29.4.24 RTCAMIN Register – Calendar Mode With Hexadecimal Format

Real-Time Clock Minutes Alarm Register – Calendar Mode With Hexadecimal Format

Figure 29-27. RTCAMIN Register

7	6	5	4	3	2	1	0
AE	0	Minutes					
rw	r-0	rw	rw	rw	rw	rw	rw

Table 29-28. RTCAMIN Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6	0	R	0h	Always 0.
5-0	Minutes	RW	undefined	Minutes (0 to 59)

29.4.25 RTCAMIN Register – Calendar Mode With BCD Format

Real-Time Clock Minutes Alarm Register – Calendar Mode With BCD Format

Figure 29-28. RTCAMIN Register

7	6	5	4	3	2	1	0
AE	Minutes – high digit			Minutes – low digit			
rw	rw	rw	rw	rw	rw	rw	rw

Table 29-29. RTCAMIN Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6-4	Minutes – high digit	RW	undefined	Minutes – high digit (0 to 5)
3-0	Minutes – low digit	RW	undefined	Minutes – low digit (0 to 9)

29.4.26 RTCAHOUR Register

Real-Time Clock Hours Alarm Register – Calendar Mode With Hexadecimal Format

Figure 29-29. RTCAHOUR Register

7	6	5	4	3	2	1	0
AE	0				Hours		
rw	r-0	r-0	rw	rw	rw	rw	rw

Table 29-30. RTCAHOUR Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6-5	0	R	0h	Always 0
4-0	Hours	RW	undefined	Hours (0 to 23)

29.4.27 RTCAHOUR Register – Calendar Mode With BCD Format

Real-Time Clock Hours Alarm Register – Calendar Mode With BCD Format

Figure 29-30. RTCAHOUR Register

7	6	5	4	3	2	1	0
AE	0	Hours – high digit			Hours – low digit		
rw	r-0	rw	rw	rw	rw	rw	rw

Table 29-31. RTCAHOUR Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6	0	R	0h	Always 0
5-4	Hours – high digit	RW	undefined	Hours – high digit (0 to 2)
3-0	Hours – low digit	RW	undefined	Hours – low digit (0 to 9)

29.4.28 RTCADOW Register – Calendar Mode

Real-Time Clock Day of Week Alarm Register – Calendar Mode

Figure 29-31. RTCADOW Register

7	6	5	4	3	2	1	0
AE	0				Day of week		
rw	r-0	r-0	r-0	r-0	rw	rw	rw

Table 29-32. RTCADOW Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6-3	0	R	0h	Always 0
2-0	Day of week	RW	undefined	Day of week (0 to 5)

29.4.29 RTCADAY Register – Calendar Mode With Hexadecimal Format

Real-Time Clock Day of Month Alarm Register – Calendar Mode With Hexadecimal Format

Figure 29-32. RTCADAY Register

7	6	5	4	3	2	1	0
AE	0		Day of month				
rw	r-0	r-0	rw	rw	rw	rw	rw

Table 29-33. RTCADAY Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6-5	0	R	0h	Always 0
4-0	Day of month	RW	undefined	Day of month (1 to 28, 29, 30, 31)

29.4.30 RTCADAY Register – Calendar Mode With BCD Format

Real-Time Clock Day of Month Alarm Register – Calendar Mode With BCD Format

Figure 29-33. RTCADAY Register

7	6	5	4	3	2	1	0
AE	0	Day of month – high digit		Day of month – low digit			
rw	r-0	rw	rw	rw	rw	rw	rw

Table 29-34. RTCADAY Register Description

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable 0b = This alarm register is disabled 1b = This alarm register is enabled
6	0	R	0h	Always 0
5-4	Day of month – high digit	RW	undefined	Day of month – high digit (0 to 3)
3-0	Day of month – low digit	RW	undefined	Day of month – low digit (0 to 9)

29.4.31 RTCPS0CTL Register

Real-Time Clock Prescale Timer 0 Control Register

Figure 29-34. RTCPS0CTL Register

15	14	13	12	11	10	9	8
Reserved		RT0PSDIV ⁽¹⁾			Reserved		RT0PSHOLD ⁽¹⁾
r0	r0	rw-(0)	rw-(0)	rw-(0)	r0	r0	rw-(1)
7	6	5	4	3	2	1	0
Reserved			RT0IP ⁽¹⁾			RT0PSIE	RT0PSIFG
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-(0)

⁽¹⁾ The configuration of these bits is retained during LPMx.5 until LOCKLPM5 is cleared, but not the register bits themselves; therefore, reconfiguration is required after wake-up from LPMx.5 before clearing LOCKLPM5.

Table 29-35. RTCPS0CTL Register Description

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0h	Reserved. Always reads as 0.
13-11	RT0PSDIV	RW	0h	Prescale timer 0 clock divide. These bits control the divide ratio of the RT0PS counter. In real-time clock calendar mode, these bits are don't care for RT0PS and RT1PS. RT0PS clock output is automatically set to /256. RT1PS clock output is automatically set to /128. 000b = Divide by 2 001b = Divide by 4 010b = Divide by 8 011b = Divide by 16 100b = Divide by 32 101b = Divide by 64 110b = Divide by 128 111b = Divide by 256
10-9	Reserved	R	0h	Reserved. Always reads as 0.
8	RT0PSHOLD	RW	1h	Prescale timer 0 hold. In real-time clock calendar mode, this bit is don't care. RT0PS is stopped by the RTCHOLD bit. 0b = RT0PS is operational 1b = RT0PS is held
7-5	Reserved	R	0h	Reserved. Always reads as 0.
4-2	RT0IP	RW	0h	Prescale timer 0 interrupt interval 000b = Divide by 2 001b = Divide by 4 010b = Divide by 8 011b = Divide by 16 100b = Divide by 32 101b = Divide by 64 110b = Divide by 128 111b = Divide by 256
1	RT0PSIE	RW	0h	Prescale timer 0 interrupt enable 0b = Interrupt not enabled 1b = Interrupt enabled
0	RT0PSIFG	RW	0h	Prescale timer 0 interrupt flag 0b = No time event occurred 1b = Time event occurred

29.4.32 RTCPS1CTL Register

Real-Time Clock Prescale Timer 1 Control Register

Figure 29-35. RTCPS1CTL Register

15	14	13	12	11	10	9	8
RT1SSELx ⁽¹⁾		RT1PSDIVx ⁽¹⁾			Reserved		RT1PSHOLD ⁽¹⁾
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0	r0	rw-(1)
7	6	5	4	3	2	1	0
Reserved			RT1IPx ⁽¹⁾			RT1PSIE	RT1PSIFG
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-(0)

⁽¹⁾ The configuration of these bits is retained during LPMx.5 until LOCKLPM5 is cleared, but not the register bits themselves; therefore, reconfiguration is required after wake-up from LPMx.5 before clearing LOCKLPM5.

Table 29-36. RTCPS1CTL Register Description

Bit	Field	Type	Reset	Description
15-14	RT1SSELx	RW	0h	Prescale timer 1 clock source select. Selects clock input source to the RT1PS counter. In real-time clock calendar mode, these bits are do not care. RT1PS clock input is automatically set to the output of RT0PS. 00b = 32-kHz crystal oscillator clock 01b = 32-kHz crystal oscillator clock 10b = Output from RT0PS 11b = Output from RT0PS
13-11	RT1PSDIVx	RW	0h	Prescale timer 1 clock divide. These bits control the divide ratio of the RT0PS counter. In real-time clock calendar mode, these bits are don't care for RT0PS and RT1PS. RT0PS clock output is automatically set to /256. RT1PS clock output is automatically set to /128. 000b = Divide by 2 001b = Divide by 4 010b = Divide by 8 011b = Divide by 16 100b = Divide by 32 101b = Divide by 64 110b = Divide by 128 111b = Divide by 256
10-9	Reserved	R	0h	Reserved. Always reads as 0.
8	RT1PSHOLD	RW	1h	Prescale timer 1 hold. In real-time clock calendar mode, this bit is don't care. RT1PS is stopped by the RTCHOLD bit. 0b = RT1PS is operational 1b = RT1PS is held
7-5	Reserved	R	0h	Reserved. Always reads as 0.
4-2	RT1IPx	RW	0h	Prescale timer 1 interrupt interval 000b = Divide by 2 001b = Divide by 4 010b = Divide by 8 011b = Divide by 16 100b = Divide by 32 101b = Divide by 64 110b = Divide by 128 111b = Divide by 256
1	RT1PSIE	RW	0h	Prescale timer 1 interrupt enable 0b = Interrupt not enabled 1b = Interrupt enabled (LPMx.5 wake-up enabled)

Table 29-36. RTCPS1CTL Register Description (continued)

Bit	Field	Type	Reset	Description
0	RT1PSIFG	RW	0h	Prescale timer 1 interrupt flag. This interrupt can be used as LPMx.5 wake-up event. 0b = No time event occurred 1b = Time event occurred

29.4.33 RTCPS0 Register

Real-Time Clock Prescale Timer 0 Counter Register

Figure 29-36. RTCPS0 Register

7	6	5	4	3	2	1	0
RT0PS							
rw	rw	rw	rw	rw	rw	rw	rw

Table 29-37. RTCPS0 Register Description

Bit	Field	Type	Reset	Description
7-0	RT0PS	RW	undefined	Prescale timer 0 counter value

29.4.34 RTCPS1 Register

Real-Time Clock Prescale Timer 1 Counter Register

Figure 29-37. RTCPS1 Register

7	6	5	4	3	2	1	0
RT1PS							
rw	rw	rw	rw	rw	rw	rw	rw

Table 29-38. RTCPS1 Register Description

Bit	Field	Type	Reset	Description
7-0	RT1PS	RW	undefined	Prescale timer 1 counter value

29.4.35 RTCIV Register

Real-Time Clock Interrupt Vector Register

Figure 29-38. RTCIV Register

15	14	13	12	11	10	9	8
RTCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
RTCIVx							
r0	r0	r0	r-(0)	r-(0)	r-(0)	r-(0)	r0

Table 29-39. RTCIV Register Description

Bit	Field	Type	Reset	Description
15-0	RTCIVx	R	0h	Real-time clock interrupt vector value Without Event/Tamper Detection implemented: 00h = No interrupt pending 02h = Interrupt Source: RTC oscillator failure; Interrupt Flag: RTCOFIFG; Interrupt Priority: Highest 04h = Interrupt Source: RTC ready; Interrupt Flag: RTCRDYIFG 06h = Interrupt Source: RTC interval timer; Interrupt Flag: RTCTEVIFG 08h = Interrupt Source: RTC user alarm; Interrupt Flag: RTCAIFG 0Ah = Interrupt Source: RTC prescaler 0; Interrupt Flag: RT0PSIFG 0Ch = Interrupt Source: RTC prescaler 1; Interrupt Flag: RT1PSIFG 0Eh = Reserved 10h = Reserved ; Interrupt Priority: Lowest With Event/Tamper Detection implemented: 00h = No interrupt pending 02h = Interrupt Source: RTC oscillator failure; Interrupt Flag: RTCOFIFG; Interrupt Priority: Highest 04h = Interrupt Source: RTC Tamper Event; Interrupt Flag: RTCCAPIFG 06h = Interrupt Source: RTC ready; Interrupt Flag: RTCRDYIFG 08h = Interrupt Source: RTC interval timer; Interrupt Flag: RTCTEVIFG 0Ah = Interrupt Source: RTC user alarm; Interrupt Flag: RTCAIFG 0Ch = Interrupt Source: RTC prescaler 0; Interrupt Flag: RT0PSIFG 0Eh = Interrupt Source: RTC prescaler 1; Interrupt Flag: RT1PSIFG 10h = Reserved ; Interrupt Priority: Lowest

29.4.36 BIN2BCD Register

Binary-to-BCD Conversion Register

Figure 29-39. BIN2BCD Register

15	14	13	12	11	10	9	8
BIN2BCDx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
BIN2BCDx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 29-40. BIN2BCD Register Description

Bit	Field	Type	Reset	Description
15-0	BIN2BCDx	RW	0h	Read: 16-bit BCD conversion of previously written 12-bit binary number. Write: 12-bit binary number to be converted.

29.4.37 BCD2BIN Register

BCD-to-Binary Conversion Register

Figure 29-40. BCD2BIN Register

15	14	13	12	11	10	9	8
BCD2BINx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
BCD2BINx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 29-41. BCD2BIN Register Description

Bit	Field	Type	Reset	Description
15-0	BCD2BINx	RW	0h	Read: 12-bit binary conversion of previously written 16-bit BCD number. Write: 16-bit BCD number to be converted.

29.4.38 RTCSECBAX Register – Hexadecimal Format

Real-Time Clock Seconds Backup Register – Hexadecimal Format

Figure 29-41. RTCSECBAX Register

7	6	5	4	3	2	1	0
0	0	Seconds ⁽¹⁾					
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-42. RTCSECBAX Register Description

Bit	Field	Type	Reset	Description
7-6	0	RW	0h	Always 0.
5-0	Seconds	RW	0h	Seconds. Valid values are 0 to 59.

29.4.39 RTCSECBAX Register – BCD Format

Real-Time Clock Seconds Backup Register – BCD Format

Figure 29-42. RTCSECBAX Register

7	6	5	4	3	2	1	0
0	Seconds – high digit ⁽¹⁾			Seconds – low digit ⁽¹⁾			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-43. RTCSECBAX Register Description

Bit	Field	Type	Reset	Description
7	0	RW	0h	Always 0.
6-4	Seconds – high digit	RW	0h	Seconds – high digit. Valid values are 0 to 5.
3-0	Seconds – low digit	RW	0h	Seconds – low digit. Valid values are 0 to 9.

29.4.40 RTCMINBAKx Register – Hexadecimal Format

Real-Time Clock Minutes Backup Register – Hexadecimal Format

Figure 29-43. RTCMINBAKx Register

7	6	5	4	3	2	1	0
0	0	Minutes ⁽¹⁾					
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-44. RTCMINBAKx Register Description

Bit	Field	Type	Reset	Description
7-6	0	RW	0h	Always 0.
5-0	Minutes	RW	0h	Minutes. Valid values are 0 to 59.

29.4.41 RTCMINBAKx Register – BCD Format

Real-Time Clock Minutes Backup Register – BCD Format

Figure 29-44. RTCMINBAKx Register

7	6	5	4	3	2	1	0
0	Minutes – high digit ⁽¹⁾			Minutes – low digit ⁽¹⁾			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-45. RTCMINBAKx Register Description

Bit	Field	Type	Reset	Description
7	0	RW	0h	Always 0.
6-4	Minutes – high digit	RW	0h	Minutes – high digit. Valid values are 0 to 5.
3-0	Minutes – low digit	RW	0h	Minutes – low digit. Valid values are 0 to 9.

29.4.42 RTCHOURBAKx Register – Hexadecimal Format

Real-Time Clock Hours Backup Register – Hexadecimal Format

Figure 29-45. RTCHOURBAKx Register

7	6	5	4	3	2	1	0
0	0	0	Hours ⁽¹⁾				
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-46. RTCHOURBAKx Register Description

Bit	Field	Type	Reset	Description
7-5	0	RW	0h	Always 0.
4-0	Hours	RW	0h	Hours. Valid values are 0 to 23.

29.4.43 RTCHOURBAKx Register – BCD Format

Real-Time Clock Hours Backup Register – BCD Format

Figure 29-46. RTCHOURBAKx Register

7	6	5	4	3	2	1	0
0	0	Hours – high digit ⁽¹⁾		Hours – low digit ⁽¹⁾			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-47. RTCHOURBAKx Register Description

Bit	Field	Type	Reset	Description
7-6	0	RW	0h	Always 0.
5-4	Hours – high digit	RW	0h	Hours – high digit. Valid values are 0 to 2.
3-0	Hours – low digit	RW	0h	Hours – low digit. Valid values are 0 to 9.

29.4.44 RTCDAYBAKx Register – Hexadecimal Format

Real-Time Clock Day of Month Backup Register – Hexadecimal Format

Figure 29-47. RTCDAYBAKx Register

7	6	5	4	3	2	1	0
0	0	0	Day of month ⁽¹⁾				
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-48. RTCDAYBAKx Register Description

Bit	Field	Type	Reset	Description
7-5	0	RW	0h	Always 0.
4-0	Day of month	RW	0h	Day of month. Valid values are 1 to 31.

29.4.45 RTCDAYBAKx Register – BCD Format

Real-Time Clock Day of Month Backup Register – BCD Format

Figure 29-48. RTCDAYBAKx Register

7	6	5	4	3	2	1	0
0	0	Day of month – high digit ⁽¹⁾		Day of month – low digit ⁽¹⁾			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-49. RTCDAYBAKx Register Description

Bit	Field	Type	Reset	Description
7-6	0	RW	0h	Always 0.
5-4	Day of month – high digit	RW	0h	Day of month – high digit. Valid values are 0 to 3.
3-0	Day of month – low digit	RW	0h	Day of month – low digit. Valid values are 0 to 9.

29.4.46 RTCMONBAKx Register – Hexadecimal Format

Real-Time Clock Month Backup Register – Hexadecimal Format

Figure 29-49. RTCMONBAKx Register

7	6	5	4	3	2	1	0
0	0	0	0	Month ⁽¹⁾			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-50. RTCMONBAKx Register Description

Bit	Field	Type	Reset	Description
7-4	0	RW	0h	Always 0.
3-0	Month	RW	0h	Month. Valid values are 1 to 12.

29.4.47 RTCMONBAKx Register – BCD Format

Real-Time Clock Month Backup Register – BCD Format

Figure 29-50. RTCMONBAKx Register

7	6	5	4	3	2	1	0
0	0	Month – high digit ⁽¹⁾		Month – low digit ⁽¹⁾			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-51. RTCMONBAKx Register Description

Bit	Field	Type	Reset	Description
7-6	0	RW	0h	Always 0.
5-4	Month – high digit	RW	0h	Month – high digit. Valid values are 0 to 3.
3-0	Month – low digit	RW	0h	Month – low digit. Valid values are 0 to 9.

29.4.48 RTCYEARBAKx Register – Hexadecimal Format

Real-Time Clock Year Low-Byte Backup Register – Hexadecimal Format

Figure 29-51. RTCYEARBAKx Register

15	14	13	12	11	10	9	8
0	0	0	0	Year – high byte ⁽¹⁾			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-52. RTCYEARBAKx Register Description

Bit	Field	Type	Reset	Description
15-12	0	RW	0h	Always 0.
11-8	Year – high byte	RW	0h	Year – high byte. Valid values of Year are 0 to 4095.
7-0	Year – low byte	RW	0h	Year – low byte. Valid values of Year are 0 to 4095.

29.4.49 RTCYEARBAKx Register – BCD Format

Real-Time Clock Year Low-Byte Backup Register – BCD Format

Figure 29-52. RTCYEARBAKx Register

15	14	13	12	11	10	9	8
0	Century – high digit ⁽¹⁾				Century – low digit ⁽¹⁾		
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
Decade ⁽¹⁾				Year – lowest digit ⁽¹⁾			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-53. RTCYEARBAKx Register Description

Bit	Field	Type	Reset	Description
15	0	RW	0h	Always 0.
14-12	Century – high digit	RW	0h	Century – high digit. Valid values are 0 to 4.
11-8	Century – low digit	RW	0h	Century – low digit. Valid values are 0 to 9.
7-4	Decade	RW	0h	Decade. Valid values are 0 to 9.
3-0	Year – lowest digit	RW	0h	Year – lowest digit. Valid values are 0 to 9.

29.4.50 RTCTCTL0 Register

Real-Time Clock Time Capture Control Register 0

Figure 29-53. RTCTCTL0 Register

7	6	5	4	3	2	1	0
Reserved						AUX3RST ⁽¹⁾	TCEN ⁽¹⁾
r-0	r-0	r-0	r-0	r-0	r-0	rw-(1)	rw-(0)

⁽¹⁾ These bits are not reset on POR.

Table 29-54. RTCTCTL0 Register Description

Bit	Field	Type	Reset	Description
7-2	Reserved	R	0h	Reserved. Always reads as 0.
1	AUX3RST	RW	1h	Indication of power cycle on AUXVCC3 0b = No power cycle on AUXVCC3 since the last clear by the User 1b = Indication of AUXVCC3 power cycle. Needs to be cleared by User to observe the next power cycle on AUXVCC3
0	TCEN	RW	0h	Enable for RTC tamper detection with time stamp 0b = Tamper detection with time stamp disabled 1b = Tamper detection with time stamp enabled

29.4.51 RTCTCTL1 Register

Real-Time Clock Time Capture Control Register 1

Figure 29-54. RTCTCTL1 Register

7	6	5	4	3	2	1	0
Reserved						RTCCAPIE	RTCCAPIFG
r-0	r-0	r-0	r-0	r-0	r-0	rw-(0)	rw-(0)

Table 29-55. RTCTCTL1 Register Description

Bit	Field	Type	Reset	Description
7-2	Reserved	R	0h	Reserved. Always reads as 0.
1	RTCCAPIE	RW	0h	Tamper event interrupt enable. In modules that support LPM3.5 or LPM4.5, this interrupt can be used as LPM3.5 or LPM4.5 wake-up event. 0b = Interrupt not enabled 1b = Interrupt enabled (LPM3.5 and LPM4.5 wake-up enabled)
0	RTCCAPIFG	RW	0h	Common interrupt flag for all tamper events. In modules that support LPM3.5 or LPM4.5, this interrupt can be used as LPM3.5 or LPM4.5 wake-up event. 0b = Tamper event did not occur 1b = At least one tamper event occurred. Status of individual tamper events can be found from the CAPEV bit in RTCCAPxCTL.

29.4.52 RTCCAPxCTL Register

Tamper Detect Pin Control Register

Figure 29-55. RTCCAPxCTL Register

7	6	5	4	3	2	1	0
Reserved	OUT ⁽¹⁾	DIR ⁽¹⁾	IN ⁽¹⁾	REN ⁽¹⁾	CAPES ⁽¹⁾	Reserved	CAPEV ⁽¹⁾
r-0	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	r-0	r/w0

⁽¹⁾ The configuration of these bits is retained during LPMx.5 until LOCKLPM5 is cleared, but not the bits themselves; therefore, reconfiguration is required after wakeup from LPMx.5 before clearing LOCKLPM5.

Table 29-56. RTCCAPxCTL Register Description

Bit	Field	Type	Reset	Description
7	Reserved	R	0h	Reserved. Always reads as 0.
6	OUT	RW	0h	RTCCAPx output 0b = Output low 1b = Output high
5	DIR	RW	0h	RTCCAPx pin direction 0b = RTCCAPx pin configured as input 1b = RTCCAPx pin configured as output
4	IN	R	0h	RTCCAPx input. The external input on RTCCAPx pin can be read by this bit. 0b = Input is low 1b = Input is high
3	REN	RW	0h	RTCCAPx pin pullup or pulldown resistor enable. When respective pin is configured as input, setting this bit enables the pullup or pulldown (see Table 29-1). 0b = Pullup or pulldown disabled 1b = Pullup or pulldown enabled
2	CAPES	RW	0h	Event edge selection 0b = Event on a low-to-high transition 1b = Event on a high-to-low transition
1	Reserved	R	0h	Reserved. Always reads as 0.
0	CAPEV	RW	0h	Tamper event status flag. All subsequent events on RTCCAPx after CAPEV is set are ignored until CAPEV is cleared by the user. Can only be written as 0. 0b = Tamper event did not occur 1b = Tamper event occurred

Enhanced Universal Serial Communication Interface (eUSCI) – UART Mode

The enhanced universal serial communication interface A (eUSCI_A) supports multiple serial communication modes with one hardware module. This chapter discusses the operation of the asynchronous UART mode.

Topic	Page
30.1 Enhanced Universal Serial Communication Interface A (eUSCI_A) Overview	768
30.2 eUSCI_A Introduction – UART Mode	768
30.3 eUSCI_A Operation – UART Mode	770
30.4 eUSCI_A UART Registers	786

30.1 Enhanced Universal Serial Communication Interface A (eUSCI_A) Overview

The eUSCI_A module supports two serial communication modes:

- UART mode
- SPI mode

30.2 eUSCI_A Introduction – UART Mode

In asynchronous mode, the eUSCI_Ax modules connect the device to an external system through two external pins, UCxRXD and UCxTXD. UART mode is selected when the UCSYNC bit is cleared.

UART mode features include:

- 7-bit or 8-bit data with odd, even, or no parity
- Independent transmit and receive shift registers
- Separate transmit and receive buffer registers
- LSB-first or MSB-first data transmit and receive
- Built-in idle-line and address-bit communication protocols for multiprocessor systems
- Receiver start edge detection for automatic wake from LPMx modes (wake from LPMx.5 is not supported)
- Programmable baud rate with modulation for fractional baud-rate support
- Status flags for error detection and suppression
- Status flags for address detection
- Independent interrupt capability for receive, transmit, start bit received, and transmit complete

[Figure 30-1](#) shows the eUSCI_Ax when configured for UART mode.

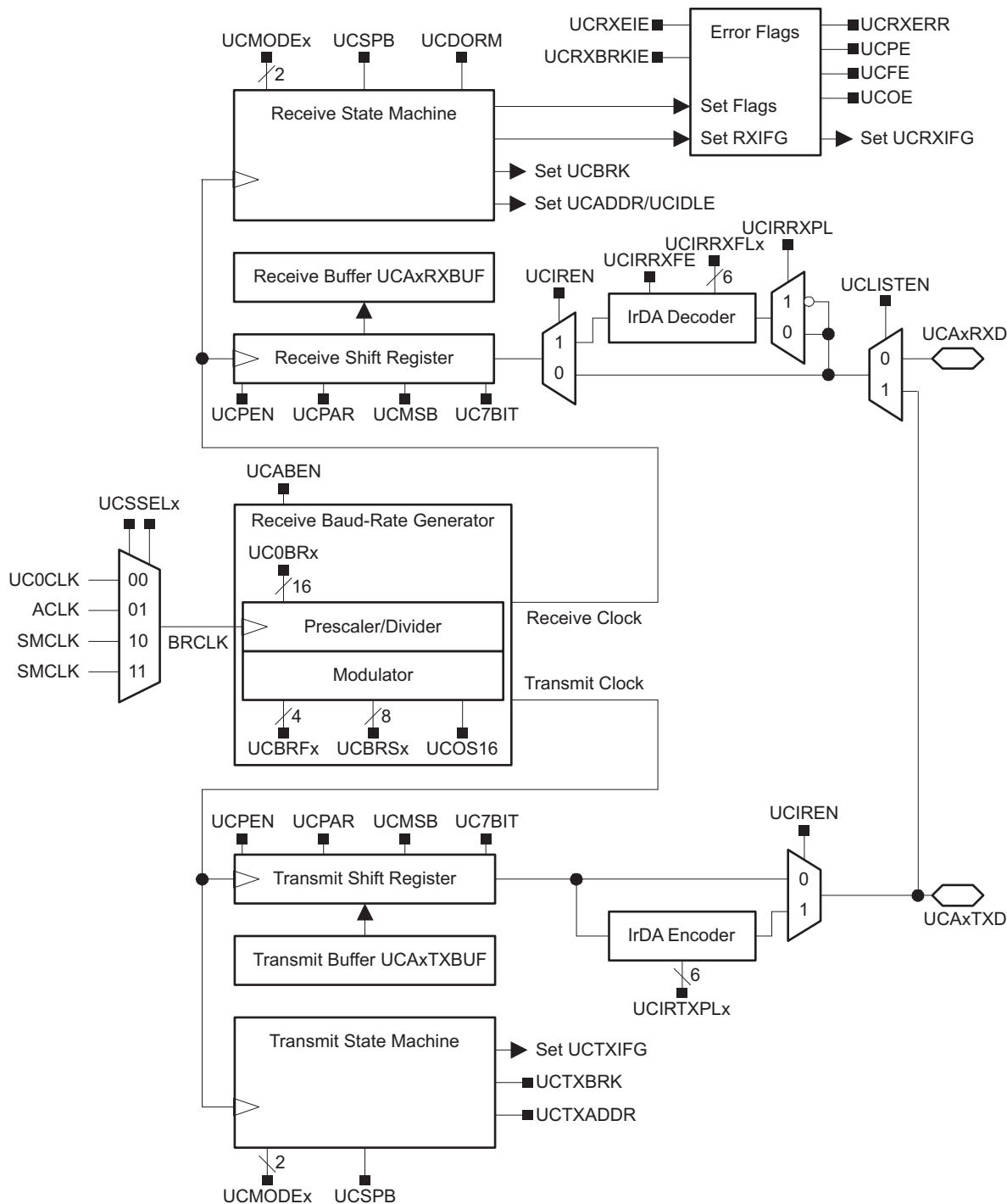


Figure 30-1. eUSCI_Ax Block Diagram – UART Mode (UCSYNC = 0)

30.3 eUSCI_A Operation – UART Mode

In UART mode, the eUSCI_A transmits and receives characters at a bit rate asynchronous to another device. Timing for each character is based on the selected baud rate of the eUSCI_A. The transmit and receive functions use the same baud-rate frequency.

30.3.1 eUSCI_A Initialization and Reset

The eUSCI_A is reset by a PUC or by setting the UCSWRST bit. After a PUC, the UCSWRST bit is automatically set, keeping the eUSCI_A in a reset condition. When set, the UCSWRST bit sets the UCTXIFG bit and resets the UCRXIE, UCTXIE, UCRXIFG, UCRXERR, UCBRK, UCPE, UCOE, UCFE, UCSTOE, and UCBTOE bits. Clearing UCSWRST releases the eUSCI_A for operation.

Configuring and reconfiguring the eUSCI_A module should be done when UCSWRST is set to avoid unpredictable behavior.

NOTE: Initializing or reconfiguring the eUSCI_A module

The recommended eUSCI_A initialization/reconfiguration process is:

1. Set UCSWRST (BIS.B
#UCSWRST, &UCAxCTL1).
 2. Initialize all eUSCI_A registers with UCSWRST = 1 (including UCAxCTL1).
 3. Configure ports.
 4. Clear UCSWRST through software (BIC.B
#UCSWRST, &UCAxCTL1).
 5. Enable interrupts (optional) through UCRXIE or UCTXIE.
-

30.3.2 Character Format

The UART character format (see [Figure 30-2](#)) consists of a start bit, seven or eight data bits, an even/odd/no parity bit, an address bit (address-bit mode), and one or two stop bits. The UCMSB bit controls the direction of the transfer and selects LSB or MSB first. LSB first is typically required for UART communication.

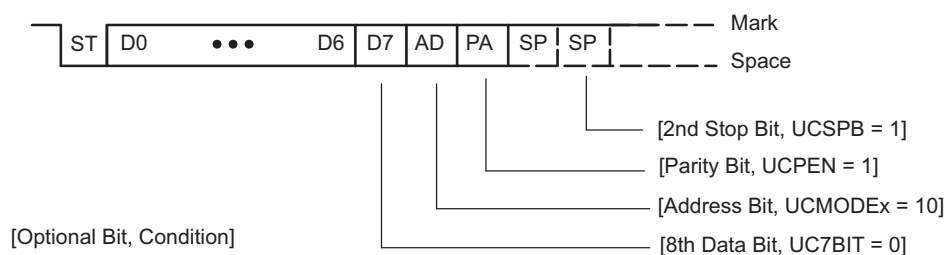


Figure 30-2. Character Format

30.3.3 Asynchronous Communication Format

When two devices communicate asynchronously, no multiprocessor format is required for the protocol. When three or more devices communicate, the eUSCI_A supports the idle-line and address-bit multiprocessor communication formats.

30.3.3.1 Idle-Line Multiprocessor Format

When UCMODEx = 01, the idle-line multiprocessor format is selected. Blocks of data are separated by an idle time on the transmit or receive lines (see [Figure 30-3](#)). An idle receive line is detected when ten or more continuous ones (marks) are received after the one or two stop bits of a character. The baud-rate generator is switched off after reception of an idle line until the next start edge is detected. When an idle line is detected, the UCIDLE bit is set.

The first character received after an idle period is an address character. The UCIDLE bit is used as an address tag for each block of characters. In idle-line multiprocessor format, this bit is set when a received character is an address.

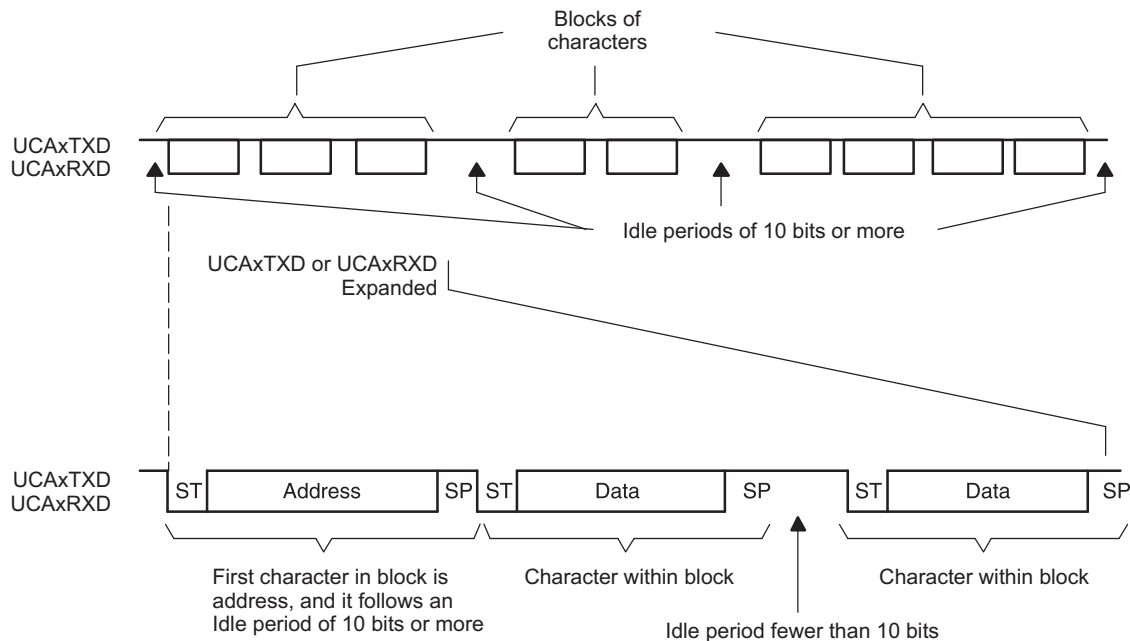


Figure 30-3. Idle-Line Format

The UCDORM bit is used to control data reception in the idle-line multiprocessor format. When UCDORM = 1, all non-address characters are assembled but not transferred into the UCxRXBUF, and interrupts are not generated. When an address character is received, the character is transferred into UCxRXBUF, UCRXIFG is set, and any applicable error flag is set when UCRXEIE = 1. When UCRXEIE = 0 and an address character is received but has a framing error or parity error, the character is not transferred into UCxRXBUF and UCRXIFG is not set.

If an address is received, user software can validate the address and must reset UCDORM to continue receiving data. If UCDORM remains set, only address characters are received. When UCDORM is cleared during the reception of a character, the receive interrupt flag is set after the reception completed. The UCDORM bit is not modified automatically by the eUSCI_A hardware.

For address transmission in idle-line multiprocessor format, a precise idle period can be generated by the eUSCI_A to generate address character identifiers on UCxTXD. The double-buffered UCTXADDR flag indicates if the next character loaded into UCxTXBUF is preceded by an idle line of 11 bits. UCTXADDR is automatically cleared when the start bit is generated.

30.3.3.1.1 Transmitting an Idle Frame

The following procedure sends out an idle frame to indicate an address character followed by associated data:

1. Set UCTXADDR, then write the address character to UCxTXBUF. UCxTXBUF must be ready for new data (UCTXIFG = 1).

This generates an idle period of exactly 11 bits followed by the address character. UCTXADDR is reset automatically when the address character is transferred from UCxTXBUF into the shift register.

2. Write desired data characters to UCxTXBUF. UCxTXBUF must be ready for new data (UCTXIFG = 1).

The data written to UCxTXBUF is transferred to the shift register and transmitted as soon as the shift register is ready for new data.

The idle-line time must not be exceeded between address and data transmission or between data transmissions. Otherwise, the transmitted data is misinterpreted as an address.

30.3.3.2 Address-Bit Multiprocessor Format

When UCMODEx = 10, the address-bit multiprocessor format is selected. Each processed character contains an extra bit used as an address indicator (see Figure 30-4). The first character in a block of characters carries a set address bit that indicates that the character is an address. The eUSCI_A UCADDR bit is set when a received character has its address bit set and is transferred to UCxRXBUF.

The UCDORM bit is used to control data reception in the address-bit multiprocessor format. When UCDORM is set, data characters with address bit = 0 are assembled by the receiver but are not transferred to UCxRXBUF and no interrupts are generated. When a character containing a set address bit is received, the character is transferred into UCxRXBUF, UCRXIFG is set, and any applicable error flag is set when UCRXEIE = 1. When UCRXEIE = 0 and a character containing a set address bit is received but has a framing error or parity error, the character is not transferred into UCxRXBUF and UCRXIFG is not set.

If an address is received, user software can validate the address and must reset UCDORM to continue receiving data. If UCDORM remains set, only address characters with address bit = 1 are received. The UCDORM bit is not modified by the eUSCI_A hardware automatically.

When UCDORM = 0, all received characters set the receive interrupt flag UCRXIFG. If UCDORM is cleared during the reception of a character, the receive interrupt flag is set after the reception is completed.

For address transmission in address-bit multiprocessor mode, the address bit of a character is controlled by the UCTXADDR bit. The value of the UCTXADDR bit is loaded into the address bit of the character transferred from UCxTXBUF to the transmit shift register. UCTXADDR is automatically cleared when the start bit is generated.

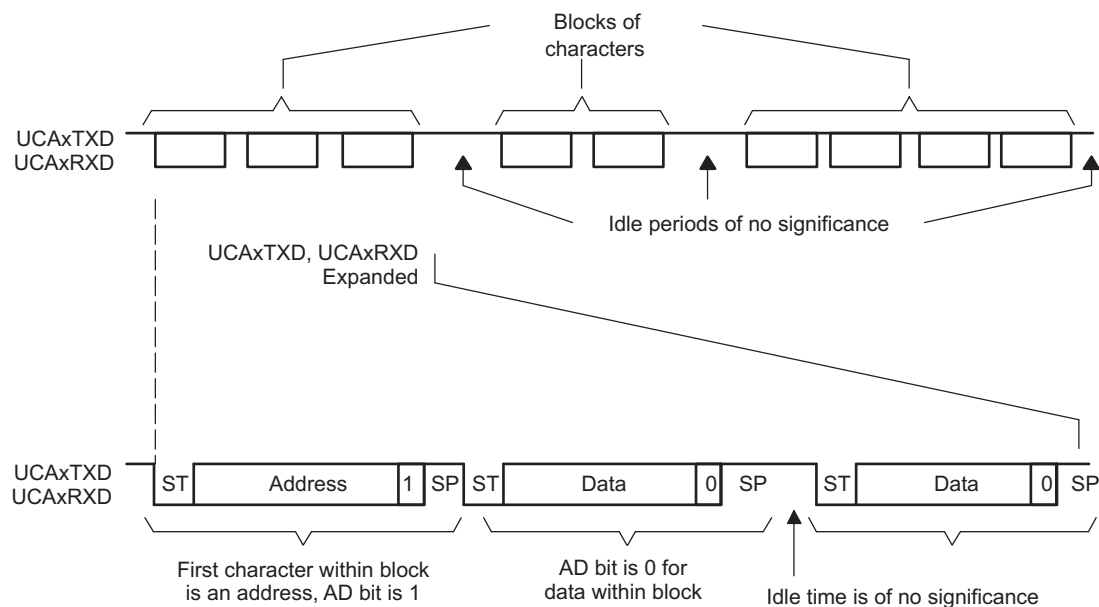


Figure 30-4. Address-Bit Multiprocessor Format

30.3.3.2.1 Break Reception and Generation

When UCMODEx = 00, 01, or 10, the receiver detects a break when all data, parity, and stop bits are low, regardless of the parity, address mode, or other character settings. When a break is detected, the UCBRK bit is set. If the break interrupt enable bit (UCBRKIE) is set, the receive interrupt flag UCRXIFG is also set. In this case, the value in UCxRXBUF is 0h, because all data bits were zero.

To transmit a break, set the UCTXBRK bit, then write 0h to UCxTXBUF. UCxTXBUF must be ready for new data (UCTXIFG = 1). This generates a break with all bits low. UCTXBRK is automatically cleared when the start bit is generated.

30.3.4 Automatic Baud-Rate Detection

When UCMODEx = 11, UART mode with automatic baud-rate detection is selected. For automatic baud-rate detection, a data frame is preceded by a synchronization sequence that consists of a break and a synch field. A break is detected when 11 or more continuous zeros (spaces) are received. If the length of the break exceeds 21 bit times, the break timeout error flag UCBTOE is set. The eUSCI_A cannot transmit data while receiving the break/synch field. The synch field follows the break as shown in Figure 30-5.

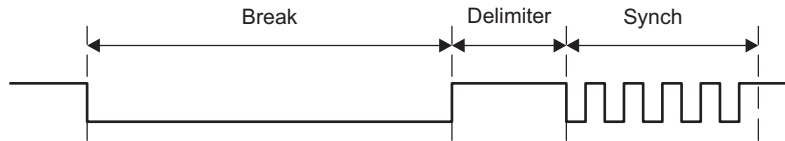


Figure 30-5. Auto Baud-Rate Detection – Break/Synch Sequence

For LIN conformance, the character format should be set to eight data bits, LSB first, no parity, and one stop bit. No address bit is available.

The synch field consists of the data 055h inside a byte field (see Figure 30-6). The synchronization is based on the time measurement between the first falling edge and the last falling edge of the pattern. The transmit baud-rate generator is used for the measurement if automatic baud-rate detection is enabled by setting UCABDEN. Otherwise, the pattern is received but not measured. The result of the measurement is transferred into the baud-rate control registers (UCAxBRW and UCAxMCTLW). If the length of the synch field exceeds the measurable time, the synch timeout error flag UCSTOE is set. The result can be read after the receive interrupt flag UCRXIFG is set.

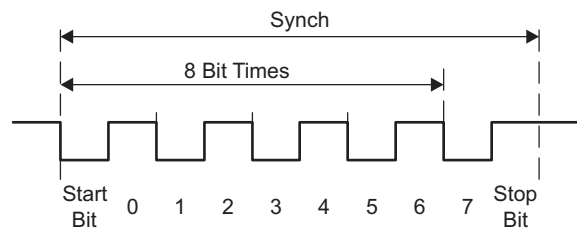


Figure 30-6. Auto Baud-Rate Detection – Synch Field

The UCDORM bit is used to control data reception in this mode. When UCDORM is set, all characters are received but not transferred into the UCAxRXBUF, and interrupts are not generated. When a break/synch field is detected, the UCBRK flag is set. The character following the break/synch field is transferred into UCAxRXBUF and the UCRXIFG interrupt flag is set. Any applicable error flag is also set. If the UCBRKIE bit is set, reception of the break/synch sets the UCRXIFG. The UCBRK bit is reset by user software or by reading the receive buffer UCAxRXBUF.

When a break/synch field is received, user software must reset UCDORM to continue receiving data. If UCDORM remains set, only the character after the next reception of a break/synch field is received. The UCDORM bit is not modified by the eUSCI_A hardware automatically.

When UCDORM = 0, all received characters set the receive interrupt flag UCRXIFG. If UCDORM is cleared during the reception of a character, the receive interrupt flag is set after the reception is complete.

The counter used to detect the baud rate is limited to 0FFFFh (2^{16}) counts. This means the minimum baud rate detectable is 244 baud in oversampling mode and 15 baud in low-frequency mode. The highest detectable baud rate is 1 Mbaud.

The automatic baud-rate detection mode can be used in a full-duplex communication system with some restrictions. The eUSCI_A cannot transmit data while receiving the break/synch field and, if a 0h byte with framing error is received, any data transmitted during this time is corrupted. The latter case can be discovered by checking the received data and the UCFE bit.

30.3.4.1 Transmitting a Break/Synch Field

The following procedure transmits a break/synch field:

1. Set UCTXBRK with UMODEx = 11.
2. Write 055h to UCAxTXBUF. UCAxTXBUF must be ready for new data (UCTXIFG = 1).
This generates a break field of 13 bits followed by a break delimiter and the synch character. The length of the break delimiter is controlled with the UCDELIMx bits. UCTXBRK is reset automatically when the synch character is transferred from UCAxTXBUF into the shift register.
3. Write desired data characters to UCAxTXBUF. UCAxTXBUF must be ready for new data (UCTXIFG = 1).
The data written to UCAxTXBUF is transferred to the shift register and transmitted as soon as the shift register is ready for new data.

30.3.5 IrDA Encoding and Decoding

When UCIREN is set, the IrDA encoder and decoder are enabled and provide hardware bit shaping for IrDA communication.

30.3.5.1 IrDA Encoding

The encoder sends a pulse for every zero bit in the transmit bitstream coming from the UART (see Figure 30-7). The pulse duration is defined by UCIRTXPLx bits specifying the number of one-half clock periods of the clock selected by UCIRTXCLK.

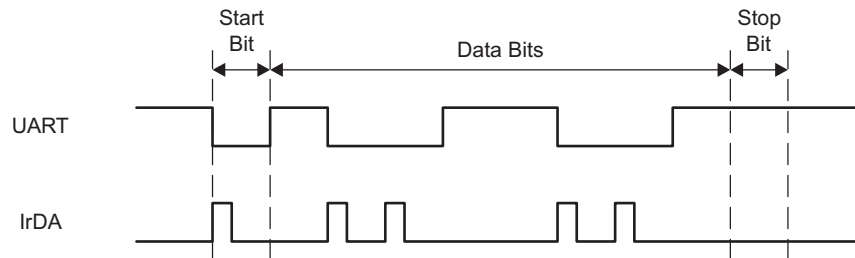


Figure 30-7. UART vs IrDA Data Format

To set the pulse time of 3/16 bit period required by the IrDA standard, the BITCLK16 clock is selected with UCIRTXCLK = 1, and the pulse length is set to six one-half clock cycles with UCIRTXPLx = 6 – 1 = 5.

When UCIRTXCLK = 0, the pulse length t_{PULSE} is based on BRCLK and is calculated as:

$$UCIRTXPLx = t_{PULSE} \times 2 \times f_{BRCLK} - 1$$

When UCIRTXCLK = 0, the prescaler UCBRx must be set to a value greater or equal to 5.

30.3.5.2 IrDA Decoding

The decoder detects high pulses when UCIRRXPL = 0. Otherwise, it detects low pulses. In addition to the analog deglitch filter, an additional programmable digital filter stage can be enabled by setting UCIRRXFE. When UCIRRXFE is set, only pulses longer than the programmed filter length are passed. Shorter pulses are discarded. The equation to program the filter length UCIRRXFLx is:

$$UCIRRXFLx = (t_{PULSE} - t_{WAKE}) \times 2 \times f_{BRCLK} - 4$$

Where:

t_{PULSE} = Minimum receive pulse width

t_{WAKE} = Wake time from any low-power mode. Zero when the device is in active mode.

30.3.6 Automatic Error Detection

Glitch suppression prevents the eUSCI_A from being accidentally started. Any pulse on UCAXRXD shorter than the deglitch time t_d (selected by UCGLITx) is ignored (see the device-specific data sheet for parameters).

When a low period on UCAXRXD exceeds t_d , a majority vote is taken for the start bit. If the majority vote fails to detect a valid start bit, the eUSCI_A halts character reception and waits for the next low period on UCAXRXD. The majority vote is also used for each bit in a character to prevent bit errors.

The eUSCI_A module automatically detects framing errors, parity errors, overrun errors, and break conditions when receiving characters. The bits UCFE, UCPE, UCOE, and UCBRK are set when their respective condition is detected. When the error flags UCFE, UCPE, or UCOE are set, UCRXERR is also set. The error conditions are described in [Table 30-1](#).

Table 30-1. Receive Error Conditions

Error Condition	Error Flag	Description
Framing error	UCFE	A framing error occurs when a low stop bit is detected. When two stop bits are used, both stop bits are checked for framing error. When a framing error is detected, the UCFE bit is set.
Parity error	UCPE	A parity error is a mismatch between the number of 1s in a character and the value of the parity bit. When an address bit is included in the character, it is included in the parity calculation. When a parity error is detected, the UCPE bit is set.
Receive overrun	UCOE	An overrun error occurs when a character is loaded into UCAXRXBUF before the prior character has been read. When an overrun occurs, the UCOE bit is set.
Break condition	UCBRK	When not using automatic baud-rate detection, a break is detected when all data, parity, and stop bits are low. When a break condition is detected, the UCBRK bit is set. A break condition can also set the interrupt flag UCRXIFG if the break interrupt enable UCBRKIE bit is set.

When UCRXEIE = 0 and a framing error or parity error is detected, no character is received into UCAXRXBUF. When UCRXEIE = 1, characters are received into UCAXRXBUF and any applicable error bit is set.

When any of the UCFE, UCPE, UCOE, UCBRK, or UCRXERR bit is set, the bit remains set until user software resets it or UCAXRXBUF is read. UCOE must be reset by reading UCAXRXBUF. Otherwise, it does not function properly. To detect overflows reliably, TI recommends the following flow. After a character is received and UCRXIFG is set, first read UCAXSTATW to check the error flags including the overflow flag UCOE. Read UCAXRXBUF next. This clears all error flags except UCOE, if UCAXRXBUF was overwritten between the read access to UCAXSTATW and to UCAXRXBUF. Therefore, the UCOE flag should be checked after reading UCAXRXBUF to detect this condition. Note that, in this case, the UCRXERR flag is not set.

30.3.7 eUSCI_A Receive Enable

The eUSCI_A module is enabled by clearing the UCSWRST bit and the receiver is ready and in an idle state. The receive baud rate generator is in a ready state but is not clocked nor producing any clocks.

The falling edge of the start bit enables the baud rate generator and the UART state machine checks for a valid start bit. If no valid start bit is detected the UART state machine returns to its idle state and the baud rate generator is turned off again. If a valid start bit is detected, a character is received.

When the idle-line multiprocessor mode is selected with UCMODEx = 01, the UART state machine checks for an idle line after receiving a character. If a start bit is detected, another character is received. Otherwise, the UCIDLE flag is set after 10 ones are received, the UART state machine returns to its idle state, and the baud rate generator is turned off.

30.3.7.1 Receive Data Glitch Suppression

Glitch suppression prevents the eUSCI_A from being accidentally started. Any glitch on UCAXRXD shorter than the deglitch time t_i is ignored by the eUSCI_A, and further action is initiated as shown in [Figure 30-8](#) (see the device-specific data sheet for parameters). The deglitch time t_i can be set to four different values using the UCGLITx bits.

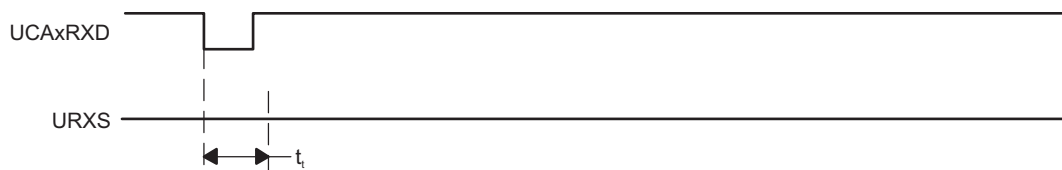


Figure 30-8. Glitch Suppression, eUSCI_A Receive Not Started

When a glitch is longer than t_i or a valid start bit occurs on UCAXRXD, the eUSCI_A receive operation is started and a majority vote is taken (see [Figure 30-9](#)). If the majority vote fails to detect a start bit, the eUSCI_A halts character reception.

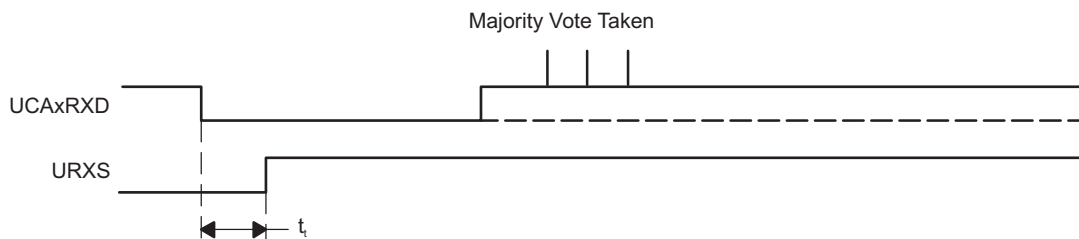


Figure 30-9. Glitch Suppression, eUSCI_A Activated

30.3.8 eUSCI_A Transmit Enable

The eUSCI_A module is enabled by clearing the UCSWRST bit and the transmitter is ready and in an idle state. The transmit baud-rate generator is ready but is not clocked nor producing any clocks.

A transmission is initiated by writing data to UCAXTXBUF. When this occurs, the baud-rate generator is enabled, and the data in UCAXTXBUF is moved to the transmit shift register on the next BITCLK after the transmit shift register is empty. UCTXIFG is set when new data can be written into UCAXTXBUF.

Transmission continues as long as new data is available in UCAXTXBUF at the end of the previous byte transmission. If new data is not in UCAXTXBUF when the previous byte has transmitted, the transmitter returns to its idle state and the baud-rate generator is turned off.

30.3.9 UART Baud-Rate Generation

The eUSCI_A baud-rate generator is capable of producing standard baud rates from nonstandard source frequencies. It provides two modes of operation selected by the UCOS16 bit.

A quick setup for finding the correct baud rate settings for the eUSCI_A can be found in [Section 30.3.10](#).

30.3.9.1 Low-Frequency Baud-Rate Generation

The low-frequency mode is selected when UCOS16 = 0. This mode allows generation of baud rates from low-frequency clock sources (for example, 9600 baud from a 32768-Hz crystal). By using a lower input frequency, the power consumption of the module is reduced. Using this mode with higher frequencies and higher prescaler settings causes the majority votes to be taken in an increasingly smaller window and, thus, decrease the benefit of the majority vote.

In low-frequency mode, the baud-rate generator uses one prescaler and one modulator to generate bit clock timing. This combination supports fractional divisors for baud-rate generation. In this mode, the maximum eUSCI_A baud rate is one-third the UART source clock frequency BRCLK.

Timing for each bit is shown in [Figure 30-10](#). For each bit received, a majority vote is taken to determine the bit value. These samples occur at the $N/2 - 1/2$, $N/2$, and $N/2 + 1/2$ BRCLK periods, where N is the number of BRCLKs per BITCLK.

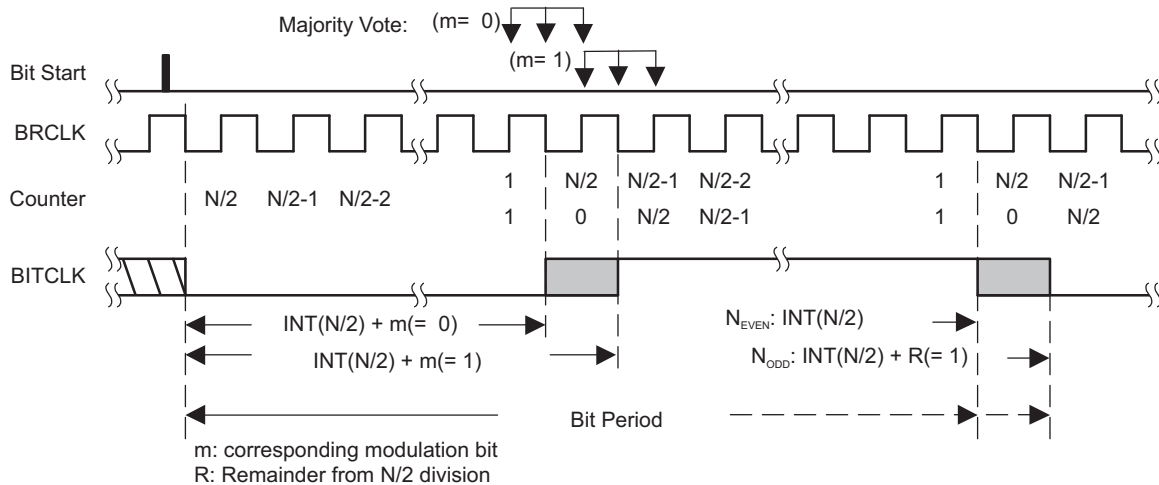


Figure 30-10. BITCLK Baud-Rate Timing With UCOS16 = 0

Modulation is based on the UCBRSx setting as shown in [Table 30-2](#). A 1 in the table indicates that m = 1 and the corresponding BITCLK period is one BRCLK period longer than a BITCLK period with m = 0. The modulation wraps around after 8 bits but restarts with each new start bit.

Table 30-2. Modulation Pattern Examples

UCBRSx	Bit 0 (Start Bit)	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
0x00	0	0	0	0	0	0	0	0
0x01	0	0	0	0	0	0	0	1
⋮								
0x35	0	0	1	1	0	1	0	1
0x36	0	0	1	1	0	1	1	0
0x37	0	0	1	1	0	1	1	1
⋮								
0xff	1	1	1	1	1	1	1	1

The correct setting of UCBRSx can be found as described in [Section 30.3.10](#).

30.3.9.2 Oversampling Baud-Rate Generation

The oversampling mode is selected when UCOS16 = 1. This mode supports sampling a UART bitstream with higher input clock frequencies. This results in majority votes that are always 1/16 of a bit clock period apart. This mode also easily supports IrDA pulses with a 3/16 bit time when the IrDA encoder and decoder are enabled.

This mode uses one prescaler and one modulator to generate the BITCLK16 clock that is 16 times faster than the BITCLK. An additional divider by 16 and modulator stage generates BITCLK from BITCLK16. This combination supports fractional divisions of both BITCLK16 and BITCLK for baud-rate generation. In this mode, the maximum eUSCI_A baud rate is 1/16 the UART source clock frequency BRCLK.

Modulation for BITCLK16 is based on the UCBRFx setting (see [Table 30-3](#)). A 1 in the table indicates that the corresponding BITCLK16 period is one BRCLK period longer than the periods m = 0. The modulation restarts with each new bit timing.

Modulation for BITCLK is based on the UCBRSx setting as previously described.

Table 30-3. BITCLK16 Modulation Pattern

UCBRFx	Number of BITCLK16 Clocks After Last Falling BITCLK Edge															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01h	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02h	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
03h	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
04h	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1
05h	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1
06h	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1
07h	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1
08h	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1
09h	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1
0Ah	0	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1
0Bh	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1
0Ch	0	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1
0Dh	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
0Eh	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0Fh	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

30.3.10 Setting a Baud Rate

For a given BRCLK clock source, the baud rate used determines the required division factor N:

$$N = f_{BRCLK} / \text{baud rate}$$

The division factor N is often a noninteger value, thus, at least one divider and one modulator stage is used to meet the factor as closely as possible.

If N is equal or greater than 16, TI recommends using the oversampling baud-rate generation mode by setting UCOS16.

NOTE: Baud-rate settings quick set up

To calculate the correct the correct settings for the baud-rate generation, perform these steps:

1. Calculate $N = f_{BRCLK} / \text{baud rate}$ [if $N > 16$ continue with step 3, otherwise with step 2]
2. $OS16 = 0$, $UCBRx = \text{INT}(N)$ [continue with step 4]
3. $OS16 = 1$, $UCBRx = \text{INT}(N/16)$, $UCBRFx = \text{INT}([(N/16) - \text{INT}(N/16)] \times 16)$
4. UCBRSx can be found by looking up the fractional part of $N (= N - \text{INT}(N))$ in table [Table 30-4](#)
5. If $OS16 = 0$ was chosen, TI recommends performing a detailed error calculation.

[Table 30-4](#) can be used as a lookup table for finding the correct UCBRSx modulation pattern for the corresponding fractional part of N. The values there are optimized for transmitting.

Table 30-4. UCBRSx Settings for Fractional Portion of $N = f_{BRCLK} / \text{Baud Rate}$

Fractional Portion of N	UCBRSx ⁽¹⁾	Fractional Portion of N	UCBRSx ⁽¹⁾
0.0000	0x00	0.5002	0xAA
0.0529	0x01	0.5715	0x6B
0.0715	0x02	0.6003	0xAD
0.0835	0x04	0.6254	0xB5
0.1001	0x08	0.6432	0xB6
0.1252	0x10	0.6667	0xD6
0.1430	0x20	0.7001	0xB7
0.1670	0x11	0.7147	0xBB
0.2147	0x21	0.7503	0xDD
0.2224	0x22	0.7861	0xED
0.2503	0x44	0.8004	0xEE
0.3000	0x25	0.8333	0xBF
0.3335	0x49	0.8464	0xDF
0.3575	0x4A	0.8572	0xEF
0.3753	0x52	0.8751	0xF7
0.4003	0x92	0.9004	0xFB
0.4286	0x53	0.9170	0xFD
0.4378	0x55	0.9288	0xFE

⁽¹⁾ The UCBRSx setting in one row is valid from the fractional portion given in that row until the one in the next row

30.3.10.1 Low-Frequency Baud-Rate Mode Setting

In low-frequency mode, the integer portion of the divisor is realized by the prescaler:

$$UCBRx = \text{INT}(N)$$

The fractional portion is realized by the modulator with its UCBRSx setting. The recommended way of determining the correct UCBRSx is performing a detailed error calculation as explained in the following sections. However it is also possible to look up the correct settings in table with typical crystals (see [Table 30-5](#)).

30.3.10.2 Oversampling Baud-Rate Mode Setting

In the oversampling mode, the prescaler is set to:

$$\text{UCBRx} = \text{INT}(N/16)$$

and the first stage modulator is set to:

$$\text{UCBRFx} = \text{INT}([(N/16) - \text{INT}(N/16)] \times 16)$$

The second modulation stage setting (UCBRSx) can be found by performing a detailed error calculation or by using [Table 30-4](#) and the fractional part of $N = f_{\text{BRCLK}}/\text{baud rate}$.

30.3.11 Transmit Bit Timing - Error calculation

The timing for each character is the sum of the individual bit timings. Using the modulation features of the baud-rate generator reduces the cumulative bit error. The individual bit error can be calculated using the following steps.

30.3.11.1 Low-Frequency Baud-Rate Mode Bit Timing

In low-frequency mode, calculation of the length of bit i $T_{\text{bit,TX}}[i]$ is based on the UCBRx and UCBRSx settings:

$$T_{\text{bit,TX}}[i] = (1/f_{\text{BRCLK}})(\text{UCBRx} + m_{\text{UCBRSx}}[i])$$

Where:

$$m_{\text{UCBRSx}}[i] = \text{Modulation of bit } i \text{ of UCBRSx}$$

30.3.11.2 Oversampling Baud-Rate Mode Bit Timing

In oversampling baud-rate mode, calculation of the length of bit i $T_{\text{bit,TX}}[i]$ is based on the baud-rate generator UCBRx, UCBRFx and UCBRSx settings:

$$t_{\text{bit,TX}}[i] = \frac{1}{f_{\text{BRCLK}}} \left((16 \times \text{UCBRx}) + \sum_{j=0}^{15} m_{\text{UCBRFx}}[j] + m_{\text{UCBRSx}}[i] \right)$$

Where:

$$\sum_{j=0}^{15} m_{\text{UCBRFx}}[j] = \text{Sum of ones from the corresponding row in } \text{Table 30-3}$$

$$m_{\text{UCBRSx}}[i] = \text{Modulation of bit } i \text{ of UCBRSx}$$

This results in an end-of-bit time $t_{\text{bit,TX}}[i]$ equal to the sum of all previous and the current bit times:

$$t_{\text{bit,TX}}[i] = \sum_{j=0}^i t_{\text{bit,TX}}[j]$$

To calculate bit error, this time is compared to the ideal bit time $t_{\text{bit,ideal,TX}}[i]$:

$$t_{\text{bit,ideal,TX}}[i] = (1/\text{baud rate})(i + 1)$$

This results in an error normalized to one ideal bit time (1/baud rate):

$$\text{Error}_{\text{TX}}[i] = (t_{\text{bit,TX}}[i] - t_{\text{bit,ideal,TX}}[i]) \times \text{baud rate} \times 100\%$$

30.3.12 Receive Bit Timing – Error Calculation

Receive timing error consists of two error sources. The first is the bit-to-bit timing error similar to the transmit bit timing error. The second is the error between a start edge occurring and the start edge being accepted by the eUSCI_A module. [Figure 30-11](#) shows the asynchronous timing errors between data on the UCAxRXD pin and the internal baud-rate clock. This results in an additional synchronization error. The synchronization error t_{SYNC} is between -0.5 BRCLKs and $+0.5 \text{ RCLKs}$, independent of the selected baud-rate generation mode.

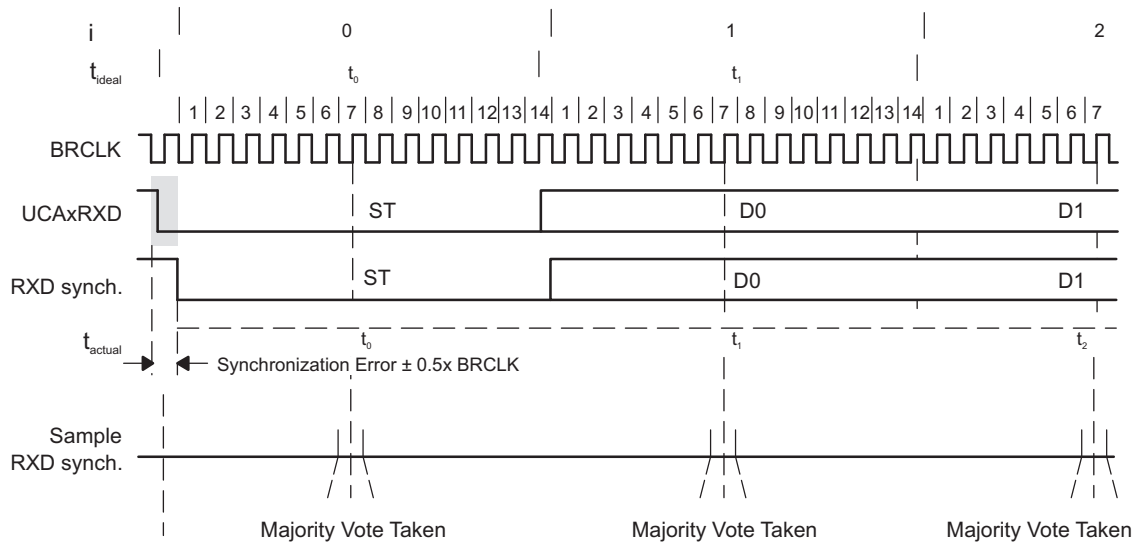


Figure 30-11. Receive Error

The ideal sampling time $t_{\text{bit,ideal,RX}}[i]$ is in the middle of a bit period:

$$t_{\text{bit,ideal,RX}}[i] = (1/\text{baud rate})(i + 0.5)$$

The real sampling time, $t_{\text{bit,RX}}[i]$, is equal to the sum of all previous bits according to the formulas shown in the transmit timing section, plus one-half BITCLK for the current bit i , plus the synchronization error t_{SYNC} .

This results in the following $t_{\text{bit,RX}}[i]$ for the low-frequency baud-rate mode:

$$t_{\text{bit,RX}}[i] = t_{\text{SYNC}} + \sum_{j=0}^{i-1} T_{\text{bit,RX}}[j] + \frac{1}{f_{\text{BRCLK}}} \left(\text{INT}(\frac{1}{2} \text{UCBRx}) + m_{\text{UCBRsx}}[i] \right)$$

Where:

$$T_{\text{bit,RX}}[i] = (1/f_{\text{BRCLK}})(\text{UCBRx} + m_{\text{UCBRsx}}[i])$$

$$m_{\text{UCBRsx}}[i] = \text{Modulation of bit } i \text{ of UCBRSx}$$

For the oversampling baud-rate mode, the sampling time $t_{\text{bit,RX}}[i]$ of bit i is calculated by:

$$t_{\text{bit,RX}}[i] = t_{\text{SYNC}} + \sum_{j=0}^{i-1} T_{\text{bit,RX}}[j] + \frac{1}{f_{\text{BRCLK}}} \left((8 * \text{UCBRx}) + \sum_{j=0}^7 m_{\text{UCBRfx}}[j] + m_{\text{UCBRsx}}[i] \right)$$

Where:

$$t_{\text{bit,RX}}[i] = \frac{1}{f_{\text{BRCLK}}} \left((16 * \text{UCBRx}) + \sum_{j=0}^{15} m_{\text{UCBRfx}}[j] + m_{\text{UCBRsx}}[i] \right)$$

$$\sum_{j=0}^{7+m_{\text{UCBRsx}}[i]} m_{\text{UCBRfx}}[j]$$

= Sum of ones from columns 0 to $(7 + m_{\text{UCBRsx}}[i])$ from the corresponding row in

[Table 30-3](#).

$$m_{\text{UCBRsx}}[i] = \text{Modulation of bit } i \text{ of UCBRSx}$$

This results in an error normalized to one ideal bit time ($1/\text{baud rate}$) according to the following formula:

$$\text{Error}_{\text{RX}}[i] = (t_{\text{bit,RX}}[i] - t_{\text{bit,ideal,RX}}[i]) \times \text{baud rate} \times 100\%$$

30.3.13 Typical Baud Rates and Errors

Standard baud-rate data for UCBRx, UCBRSx, and UCBRFx are listed in [Table 30-5](#) for a 32768-Hz crystal sourcing ACLK and typical SMCLK frequencies. Make sure that the selected BRCLK frequency does not exceed the device specific maximum eUSCI_A input frequency (see the device-specific data sheet).

The receive error is the accumulated time versus the ideal scanning time in the middle of each bit. The worst-case error is given for the reception of an 8-bit character with parity and one stop bit including synchronization error.

The transmit error is the accumulated timing error versus the ideal time of the bit period. The worst-case error is given for the transmission of an 8-bit character with parity and stop bit.

Table 30-5. Recommended Settings for Typical Crystals and Baud Rates⁽¹⁾

BRCLK	Baud Rate	UCOS16	UCBRx	UCBRFx	UCBRSx ⁽²⁾	TX Error ⁽²⁾ (%)		RX Error ⁽²⁾ (%)	
						neg	pos	neg	pos
32768	1200	1	1	11	0x25	-2.29	2.25	-2.56	5.35
32768	2400	0	13	-	0xB6	-3.12	3.91	-5.52	8.84
32768	4800	0	6	-	0xEE	-7.62	8.98	-21	10.25
32768	9600	0	3	-	0x92	-17.19	16.02	-23.24	37.3
1000000	9600	1	6	8	0x20	-0.48	0.64	-1.04	1.04
1000000	19200	1	3	4	0x2	-0.8	0.96	-1.84	1.84
1000000	38400	1	1	10	0x0	0	1.76	0	3.44
1000000	57600	0	17	-	0x4A	-2.72	2.56	-3.76	7.28
1000000	115200	0	8	-	0xD6	-7.36	5.6	-17.04	6.96
1048576	9600	1	6	13	0x22	-0.46	0.42	-0.48	1.23
1048576	19200	1	3	6	0xAD	-0.88	0.83	-2.36	1.18
1048576	38400	1	1	11	0x25	-2.29	2.25	-2.56	5.35
1048576	57600	0	18	-	0x11	-2	3.37	-5.31	5.55
1048576	115200	0	9	-	0x08	-5.37	4.49	-5.93	14.92
4000000	9600	1	26	0	0xB6	-0.08	0.16	-0.28	0.2
4000000	19200	1	13	0	0x84	-0.32	0.32	-0.64	0.48
4000000	38400	1	6	8	0x20	-0.48	0.64	-1.04	1.04
4000000	57600	1	4	5	0x55	-0.8	0.64	-1.12	1.76
4000000	115200	1	2	2	0xBB	-1.44	1.28	-3.92	1.68
4000000	230400	0	17	-	0x4A	-2.72	2.56	-3.76	7.28
4194304	9600	1	27	4	0xFB	-0.11	0.1	-0.33	0
4194304	19200	1	13	10	0x55	-0.21	0.21	-0.55	0.33
4194304	38400	1	6	13	0x22	-0.46	0.42	-0.48	1.23
4194304	57600	1	4	8	0xEE	-0.75	0.74	-2	0.87
4194304	115200	1	2	4	0x92	-1.62	1.37	-3.56	2.06
4194304	230400	0	18	-	0x11	-2	3.37	-5.31	5.55
8000000	9600	1	52	1	0x49	-0.08	0.04	-0.1	0.14
8000000	19200	1	26	0	0xB6	-0.08	0.16	-0.28	0.2
8000000	38400	1	13	0	0x84	-0.32	0.32	-0.64	0.48
8000000	57600	1	8	10	0xF7	-0.32	0.32	-1	0.36
8000000	115200	1	4	5	0x55	-0.8	0.64	-1.12	1.76
8000000	230400	1	2	2	0xBB	-1.44	1.28	-3.92	1.68
8000000	460800	0	17	-	0x4A	-2.72	2.56	-3.76	7.28
8388608	9600	1	54	9	0xEE	-0.06	0.06	-0.11	0.13
8388608	19200	1	27	4	0xFB	-0.11	0.1	-0.33	0
8388608	38400	1	13	10	0x55	-0.21	0.21	-0.55	0.33
8388608	57600	1	9	1	0xB5	-0.31	0.31	-0.53	0.78
8388608	115200	1	4	8	0xEE	-0.75	0.74	-2	0.87

⁽¹⁾ The listed UCBRSx settings are determined by a search algorithm for the lowest error. Other settings for UCBRSx might result in similar or same errors.

⁽²⁾ Assumes a stable clock source for BRCLK with negligible jitter (for example, from a crystal oscillator). Any frequency variation or jitter of the clock source will make the errors worse.

Table 30-5. Recommended Settings for Typical Crystals and Baud Rates⁽¹⁾ (continued)

BRCLK	Baud Rate	UCOS16	UCBRx	UCBRFx	UCBR5x ⁽²⁾	TX Error ⁽²⁾ (%)		RX Error ⁽²⁾ (%)	
						neg	pos	neg	pos
8388608	230400	1	2	4	0x92	-1.62	1.37	-3.56	2.06
8388608	460800	0	18	-	0x11	-2	3.37	-5.31	5.55
12000000	9600	1	78	2	0x0	0	0	0	0.04
12000000	19200	1	39	1	0x0	0	0	0	0.16
12000000	38400	1	19	8	0x65	-0.16	0.16	-0.4	0.24
12000000	57600	1	13	0	0x25	-0.16	0.32	-0.48	0.48
12000000	115200	1	6	8	0x20	-0.48	0.64	-1.04	1.04
12000000	230400	1	3	4	0x2	-0.8	0.96	-1.84	1.84
12000000	460800	1	1	10	0x0	0	1.76	0	3.44
16000000	9600	1	104	2	0xD6	-0.04	0.02	-0.09	0.03
16000000	19200	1	52	1	0x49	-0.08	0.04	-0.1	0.14
16000000	38400	1	26	0	0xB6	-0.08	0.16	-0.28	0.2
16000000	57600	1	17	5	0xDD	-0.16	0.2	-0.3	0.38
16000000	115200	1	8	10	0xF7	-0.32	0.32	-1	0.36
16000000	230400	1	4	5	0x55	-0.8	0.64	-1.12	1.76
16000000	460800	1	2	2	0xBB	-1.44	1.28	-3.92	1.68
16777216	9600	1	109	3	0xB5	-0.03	0.02	-0.05	0.06
16777216	19200	1	54	9	0xEE	-0.06	0.06	-0.11	0.13
16777216	38400	1	27	4	0xFB	-0.11	0.1	-0.33	0
16777216	57600	1	18	3	0x44	-0.16	0.15	-0.2	0.45
16777216	115200	1	9	1	0xB5	-0.31	0.31	-0.53	0.78
16777216	230400	1	4	8	0xEE	-0.75	0.74	-2	0.87
16777216	460800	1	2	4	0x92	-1.62	1.37	-3.56	2.06
20000000	9600	1	130	3	0x25	-0.02	0.03	0	0.07
20000000	19200	1	65	1	0xD6	-0.06	0.03	-0.1	0.1
20000000	38400	1	32	8	0xEE	-0.1	0.13	-0.27	0.14
20000000	57600	1	21	11	0x22	-0.16	0.13	-0.16	0.38
20000000	115200	1	10	13	0xAD	-0.29	0.26	-0.46	0.66
20000000	230400	1	5	6	0xEE	-0.67	0.51	-1.71	0.62
20000000	460800	1	2	11	0x92	-1.38	0.99	-1.84	2.8

30.3.14 Using the eUSCI_A Module in UART Mode With Low-Power Modes

The eUSCI_A module provides automatic clock activation for use with low-power modes. When the eUSCI_A clock source is inactive because the device is in a low-power mode, the eUSCI_A module automatically activates it when needed, regardless of the control-bit settings for the clock source. The clock remains active until the eUSCI_A module returns to its idle condition. After the eUSCI_A module returns to the idle condition, control of the clock source reverts to the settings of its control bits.

NOTE: Clock Activation Time

If the clock source is not already active when the eUSCI_A module requests it then the clock must be activated. This takes time. This clock activation time depending on the selected clock source and the selected low power mode. If the DCO is used as clock source the activation time is approximately the wake-up time as specified in the device-specific data sheet.

30.3.15 eUSCI_A Interrupts in UART Mode

The eUSCI_A has only one interrupt vector that is shared for transmission and for reception.

30.3.15.1 UART Transmit Interrupt Operation

The UCTXIFG interrupt flag is set by the transmitter to indicate that UCAXTXBUF is ready to accept another character. An interrupt request is generated if UCTXIE and GIE are also set. UCTXIFG is automatically reset if a character is written to UCAXTXBUF.

UCTXIFG is set after a PUC or when UCSWRST = 1. UCTXIE is reset after a PUC or when UCSWRST = 1.

30.3.15.2 UART Receive Interrupt Operation

The UCRXIFG interrupt flag is set each time a character is received and loaded into UCAXRXBUF. An interrupt request is generated if UCRXIE and GIE are also set. UCRXIFG and UCRXIE are reset by a system reset PUC signal or when UCSWRST = 1. UCRXIFG is automatically reset when UCAXRXBUF is read.

Additional interrupt control features include:

- When UCAXRXEIE = 0, erroneous characters do not set UCRXIFG.
- When UCDORM = 1, nonaddress characters do not set UCRXIFG in multiprocessor modes. In plain UART mode, no characters are set UCRXIFG.
- When UCBRKIE = 1, a break condition sets the UCBRK bit and the UCRXIFG flag.

30.3.15.3 UART State Change Interrupt Operation

[Table 30-6](#) describes the UART state change interrupt flags.

Table 30-6. UART State Change Interrupt Flags

Interrupt Flag	Interrupt Condition
UCSTTIFG	START byte received interrupt. This flag is set when the UART module receives a START byte. This flag can be cleared by writing 0 to it.
UCTXCPTIFG	Transmit complete interrupt. This flag is set after the complete UART byte in the internal shift register including STOP bit is shifted out. This flag can be cleared by writing 0 to it.

30.3.15.4 UCAXIV, Interrupt Vector Generator

The eUSCI_A interrupt flags are prioritized and combined to source a single interrupt vector. The interrupt vector register UCAXIV is used to determine which flag requested an interrupt. The highest-priority enabled interrupt generates a number in the UCAXIV register that can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled interrupts do not affect the UCAXIV value.

Read access of the UCAXIV register automatically resets the highest-pending Interrupt condition and flag. Write access of the UCAXIV register clears all pending Interrupt conditions and flags. If another interrupt flag is set, another interrupt is generated immediately after servicing the initial interrupt.

[Example 30-1](#) shows the recommended use of UCAXIV. The UCAXIV value is added to the PC to automatically jump to the appropriate routine. The following example is given for eUSCI_A0.

Example 30-1. UCAXIV Software Example

```
#pragma vector = USCI_A0_VECTOR __interrupt void USCI_A0_ISR(void) {
    switch(__even_in_range(UCA0IV,18)) {
        case 0x00:      // Vector 0: No interrupts
            break;
        case 0x02: ...  // Vector 2: UCRXIFG
            break;
        case 0x04: ...  // Vector 4: UCTXIFG
            break;
        case 0x06: ...  // Vector 6: UCSTTIFG
            break;
        case 0x08: ...  // Vector 8: UCTXCPITIFG
            break;
        default: break;
    }
}
```

30.3.16 DMA Operation

In devices with a DMA controller, the eUSCI module can trigger DMA transfers when the transmit buffer UCAXTXBUF is empty or when data was received in the UCAXRXBUF buffer. The DMA trigger signals correspond to the UCTXIFG transmit interrupt flag and the UCRXIFG receive interrupt flag, respectively. The interrupt functionality must be disabled for the selected DMA triggers with UCTXIE = 0 and UCRXIE = 0.

A DMA read access to UCAXRXBUF has the same effects as a CPU (software) read: all error flags (UCRXERR, UCFE, UCPE, UCOE, and UCBRK) are cleared after the read. Thus these errors might go unnoticed.

30.4 eUSCI_A UART Registers

The eUSCI_A registers applicable in UART mode and their address offsets are listed in [Table 30-7](#). The base address can be found in the device-specific data sheet.

Table 30-7. eUSCI_A UART Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	UCAxCTLW0	eUSCI_Ax Control Word 0	Read/write	Word	0001h	Section 30.4.1
01h	UCAxCTL0 ⁽¹⁾	eUSCI_Ax Control 0	Read/write	Byte	00h	
00h	UCAxCTL1	eUSCI_Ax Control 1	Read/write	Byte	01h	
02h	UCAxCTLW1	eUSCI_Ax Control Word 1	Read/write	Word	0003h	Section 30.4.2
06h	UCAxBRW	eUSCI_Ax Baud Rate Control Word	Read/write	Word	0000h	Section 30.4.3
06h	UCAxBR0 ⁽¹⁾	eUSCI_Ax Baud Rate Control 0	Read/write	Byte	00h	
07h	UCAxBR1	eUSCI_Ax Baud Rate Control 1	Read/write	Byte	00h	
08h	UCAxMCTLW	eUSCI_Ax Modulation Control Word	Read/write	Word	00h	Section 30.4.4
0Ah	UCAxSTATW	eUSCI_Ax Status	Read/write	Word	00h	Section 30.4.5
0Ch	UCAxRXBUF	eUSCI_Ax Receive Buffer	Read/write	Word	00h	Section 30.4.6
0Eh	UCAxTXBUF	eUSCI_Ax Transmit Buffer	Read/write	Word	00h	Section 30.4.7
10h	UCAxABCTL	eUSCI_Ax Auto Baud Rate Control	Read/write	Word	00h	Section 30.4.8
12h	UCAxIRCTL	eUSCI_Ax IrDA Control	Read/write	Word	0000h	Section 30.4.9
12h	UCAxIRTCTL	eUSCI_Ax IrDA Transmit Control	Read/write	Byte	00h	
13h	UCAxIRRCTL	eUSCI_Ax IrDA Receive Control	Read/write	Byte	00h	
1Ah	UCAxIE	eUSCI_Ax Interrupt Enable	Read/write	Word	00h	Section 30.4.10
1Ch	UCAxIFG	eUSCI_Ax Interrupt Flag	Read/write	Word	02h	Section 30.4.11
1Eh	UCAxIV	eUSCI_Ax Interrupt Vector	Read	Word	0000h	Section 30.4.12

⁽¹⁾ It is recommended to access these registers using 16-bit access. If 8-bit access is used, the corresponding bit names must be followed by "_H".

30.4.1 UCxCTLW0 Register

eUSCI_Ax Control Word Register 0

Figure 30-12. UCxCTLW0 Register

15	14	13	12	11	10	9	8
UCPEN	UCPAR	UCMSB	UC7BIT	UCSPB	UCMODEx		UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCSSELx		UCRXEIE	UCBRKIE	UCDORM	UCTXADDR	UCTXBRK	UCSWRST
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

Can be modified only when UCSWRST = 1.

Table 30-8. UCxCTLW0 Register Description

Bit	Field	Type	Reset	Description
15	UCPEN	RW	0h	Parity enable 0b = Parity disabled 1b = Parity enabled. Parity bit is generated (UCxTXD) and expected (UCxRXD). In address-bit multiprocessor mode, the address bit is included in the parity calculation.
14	UCPAR	RW	0h	Parity select. UCPAR is not used when parity is disabled. 0b = Odd parity 1b = Even parity
13	UCMSB	RW	0h	MSB first select. Controls the direction of the receive and transmit shift register. 0b = LSB first 1b = MSB first
12	UC7BIT	RW	0h	Character length. Selects 7-bit or 8-bit character length. 0b = 8-bit data 1b = 7-bit data
11	UCSPB	RW	0h	Stop bit select. Number of stop bits. 0b = One stop bit 1b = Two stop bits
10-9	UCMODEx	RW	0h	eUSCI_A mode. The UCMODEx bits select the asynchronous mode when UCSYNC = 0. 00b = UART mode 01b = Idle-line multiprocessor mode 10b = Address-bit multiprocessor mode 11b = UART mode with automatic baud-rate detection
8	UCSYNC	RW	0h	Synchronous mode enable 0b = Asynchronous mode 1b = Synchronous mode
7-6	UCSSELx	RW	0h	eUSCI_A clock source select. These bits select the BRCLK source clock. 00b = UCLK 01b = ACLK 10b = SMCLK 11b = SMCLK
5	UCRXEIE	RW	0h	Receive erroneous-character interrupt enable 0b = Erroneous characters rejected and UCRXIFG is not set. 1b = Erroneous characters received set UCRXIFG.
4	UCBRKIE	RW	0h	Receive break character interrupt enable 0b = Received break characters do not set UCRXIFG. 1b = Received break characters set UCRXIFG.

Table 30-8. UCAXCTLW0 Register Description (continued)

Bit	Field	Type	Reset	Description
3	UCDORM	RW	0h	Dormant. Puts eUSCI_A into sleep mode. 0b = Not dormant. All received characters set UCRXIFG. 1b = Dormant. Only characters that are preceded by an idle-line or with address bit set UCRXIFG. In UART mode with automatic baud-rate detection, only the combination of a break and synch field sets UCRXIFG.
2	UCTXADDR	RW	0h	Transmit address. Next frame to be transmitted is marked as address, depending on the selected multiprocessor mode. 0b = Next frame transmitted is data. 1b = Next frame transmitted is an address.
1	UCTXBRK	RW	0h	Transmit break. Transmits a break with the next write to the transmit buffer. In UART mode with automatic baud-rate detection, 055h must be written into UCAXTXBUF to generate the required break/synch fields. Otherwise, 0h must be written into the transmit buffer. 0b = Next frame transmitted is not a break. 1b = Next frame transmitted is a break or a break/synch.
0	UCSWRST	RW	1h	Software reset enable 0b = Disabled. eUSCI_A reset released for operation. 1b = Enabled. eUSCI_A logic held in reset state.

30.4.2 UCAXCTLW1 Register

eUSCI_Ax Control Word Register 1

Figure 30-13. UCAXCTLW1 Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved						UCGLITx	
r-0	r-0	r-0	r-0	r-0	r-0	rw-1	rw-1

Table 30-9. UCAXCTLW1 Register Description

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1-0	UCGLITx	RW	3h	Deglintch time 00b = Approximately 2 ns 01b = Approximately 50 ns 10b = Approximately 100 ns 11b = Approximately 200 ns

30.4.3 UCABRW Register

eUSCI_Ax Baud Rate Control Word Register

Figure 30-14. UCABRW Register

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when UCSWRST = 1.

Table 30-10. UCABRW Register Description

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Clock prescaler setting of the Baud rate generator

30.4.4 UCAMCTLW Register

eUSCI_Ax Modulation Control Word Register

Figure 30-15. UCAMCTLW Register

15	14	13	12	11	10	9	8
UCBRSx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCBRFx				Reserved			UCOS16
rw-0	rw-0	rw-0	rw-0	r0	r0	r0	rw-0

Can be modified only when UCSWRST = 1.

Table 30-11. UCAMCTLW Register Description

Bit	Field	Type	Reset	Description
15-8	UCBRSx	RW	0h	Second modulation stage select. These bits hold a free modulation pattern for BITCLK.
7-4	UCBRFx	RW	0h	First modulation stage select. These bits determine the modulation pattern for BITCLK16 when UCOS16 = 1. Ignored with UCOS16 = 0. The "Oversampling Baud-Rate Generation" section shows the modulation pattern.
3-1	Reserved	R	0h	Reserved
0	UCOS16	RW	0h	Oversampling mode enabled 0b = Disabled 1b = Enabled

30.4.5 UCxSTATW Register

eUSCI_Ax Status Register

Figure 30-16. UCxSTATW Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCLISTEN	UCFE	UCOE	UCPE	UCBRK	UCRXERR	UCADDR UCIDLE	UCBUSY
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r-0

Can be modified only when UCSWRST = 1.

Table 30-12. UCxSTATW Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7	UCLISTEN	RW	0h	Listen enable. The UCLISTEN bit selects loopback mode. 0b = Disabled 1b = Enabled. UCxTXD is internally fed back to the receiver.
6	UCFE	RW	0h	Framing error flag. UCFE is cleared when UCxRXBUF is read. 0b = No error 1b = Character received with low stop bit
5	UCOE	RW	0h	Overrun error flag. This bit is set when a character is transferred into UCxRXBUF before the previous character was read. UCOE is cleared automatically when UCxRXBUF is read, and must not be cleared by software. Otherwise, it does not function correctly. 0b = No error 1b = Overrun error occurred.
4	UCPE	RW	0h	Parity error flag. When UCPE = 0, UCPE is read as 0. UCPE is cleared when UCxRXBUF is read. 0b = No error 1b = Character received with parity error
3	UCBRK	RW	0h	Break detect flag. UCBRK is cleared when UCxRXBUF is read. 0b = No break condition 1b = Break condition occurred.
2	UCRXERR	RW	0h	Receive error flag. This bit indicates a character was received with one or more errors. When UCRXERR = 1, on or more error flags, UCFE, UCPE, or UCOE is also set. UCRXERR is cleared when UCxRXBUF is read. 0b = No receive errors detected 1b = Receive error detected
1	UCADDR UCIDLE	RW	0h	UCADDR: Address received in address-bit multiprocessor mode. UCADDR is cleared when UCxRXBUF is read. UCIDLE: Idle line detected in idle-line multiprocessor mode. UCIDLE is cleared when UCxRXBUF is read. 0b = UCADDR: Received character is data. UCIDLE: No idle line detected 1b = UCADDR: Received character is an address. UCIDLE: Idle line detected
0	UCBUSY	R	0h	eUSCI_A busy. This bit indicates if a transmit or receive operation is in progress. 0b = eUSCI_A inactive 1b = eUSCI_A transmitting or receiving

30.4.6 UCxRXBUF Register

eUSCI_Ax Receive Buffer Register

Figure 30-17. UCxRXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
UCRXBUFx							
r	r	r	r	r	r	r	r

Table 30-13. UCxRXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCRXBUFx	R	0h	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCxRXBUF resets the receive-error bits, the UCADDR or UCIDLE bit, and UCRXIFG. In 7-bit data mode, UCxRXBUF is LSB justified and the MSB is always reset.

30.4.7 UCxTXBUF Register

eUSCI_Ax Transmit Buffer Register

Figure 30-18. UCxTXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 30-14. UCxTXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTXBUFx	RW	0h	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted on UCxTXD. Writing to the transmit data buffer clears UCTXIFG. The MSB of UCxTXBUF is not used for 7-bit data and is reset.

30.4.8 UCxABCTL Register

eUSCI_Ax Auto Baud Rate Control Register

Figure 30-19. UCxABCTL Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved		UCDELIMx		UCSTOE	UCBTOE	Reserved	UCABDEN
r-0	r-0	rw-0	rw-0	rw-0	rw-0	r-0	rw-0

Can be modified only when UCSWRST = 1.

Table 30-15. UCxABCTL Register Description

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved
5-4	UCDELIMx	RW	0h	Break/synch delimiter length 00b = 1 bit time 01b = 2 bit times 10b = 3 bit times 11b = 4 bit times
3	UCSTOE	RW	0h	Synch field time out error 0b = No error 1b = Length of synch field exceeded measurable time.
2	UCBTOE	RW	0h	Break time out error 0b = No error 1b = Length of break field exceeded 22 bit times.
1	Reserved	R	0h	Reserved
0	UCABDEN	RW	0h	Automatic baud-rate detect enable 0b = Baud-rate detection disabled. Length of break and synch field is not measured. 1b = Baud-rate detection enabled. Length of break and synch field is measured and baud-rate settings are changed accordingly.

30.4.9 UCAXIRCTL Register

eUSCI_Ax IrDA Control Word Register

Figure 30-20. UCAXIRCTL Register

15	14	13	12	11	10	9	8
UCIRRXFLx						UCIRRXPL	UCIRRXFE
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCIRTXPLx						UCIRTXCLK	UCIREN
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when UCSWRST = 1.

Table 30-16. UCAXIRCTL Register Description

Bit	Field	Type	Reset	Description
15-10	UCIRRXFLx	RW	0h	Receive filter length. The minimum pulse length for receive is given by: $t_{MIN} = (UCIRRXFLx + 4) / [2 \times f_{IRTXCLK}]$
9	UCIRRXPL	RW	0h	IrDA receive input UCAXRXD polarity 0b = IrDA transceiver delivers a high pulse when a light pulse is seen. 1b = IrDA transceiver delivers a low pulse when a light pulse is seen.
8	UCIRRXFE	RW	0h	IrDA receive filter enabled 0b = Receive filter disabled 1b = Receive filter enabled
7-2	UCIRTXPLx	RW	0h	Transmit pulse length. Pulse length $t_{PULSE} = (UCIRTXPLx + 1) / [2 \times f_{IRTXCLK}]$
1	UCIRTXCLK	RW	0h	IrDA transmit pulse clock select 0b = BRCLK 1b = BITCLK16 when UCOS16 = 1. Otherwise, BRCLK.
0	UCIREN	RW	0h	IrDA encoder/decoder enable 0b = IrDA encoder/decoder disabled 1b = IrDA encoder/decoder enabled

30.4.10 UCxIE Register

eUSCI_Ax Interrupt Enable Register

Figure 30-21. UCxIE Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved				UCTXCPTIE	UCSTTIE	UCTXIE	UCRXIE
r-0	r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0

Table 30-17. UCxIE Register Description

Bit	Field	Type	Reset	Description
15-4	Reserved	R	0h	Reserved
3	UCTXCPTIE	RW	0h	Transmit complete interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
2	UCSTTIE	RW	0h	Start bit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
1	UCTXIE	RW	0h	Transmit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE	RW	0h	Receive interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

30.4.11 UCAXIFG Register

eUSCI_Ax Interrupt Flag Register

Figure 30-22. UCAXIFG Register

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved				UCTXCPTIFG	UCSTTIFG	UCTXIFG	UCRXIFG
r-0	r-0	r-0	r-0	rw-0	rw-0	rw-1	rw-0

Table 30-18. UCAXIFG Register Description

Bit	Field	Type	Reset	Description
15-4	Reserved	R	0h	Reserved
3	UCTXCPTIFG	RW	0h	Transmit complete interrupt flag. UCTXCPTIFG is set when the entire byte in the internal shift register got shifted out and UCAXTXBUF is empty. 0b = No interrupt pending 1b = Interrupt pending
2	UCSTTIFG	RW	0h	Start bit interrupt flag. UCSTTIFG is set after a Start bit was received 0b = No interrupt pending 1b = Interrupt pending
1	UCTXIFG	RW	1h	Transmit interrupt flag. UCTXIFG is set when UCAXTXBUF empty. 0b = No interrupt pending 1b = Interrupt pending
0	UCRXIFG	RW	0h	Receive interrupt flag. UCRXIFG is set when UCAXRXBUF has received a complete character. 0b = No interrupt pending 1b = Interrupt pending

30.4.12 UCxIV Register

eUSCI_Ax Interrupt Vector Register

Figure 30-23. UCxIV Register

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCIVx							
r0	r0	r0	r0	r-(0)	r-(0)	r-(0)	r0

Table 30-19. UCxIV Register Description

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI_A interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Receive buffer full; Interrupt Flag: UCRXIFG; Interrupt Priority: Highest 04h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG 06h = Interrupt Source: Start bit received; Interrupt Flag: UCSTTIFG 08h = Interrupt Source: Transmit complete; Interrupt Flag: UCTXCPRTIFG; Interrupt Priority: Lowest

Enhanced Universal Serial Communication Interface (eUSCI) – SPI Mode

The enhanced universal serial communication interfaces, eUSCI_A and eUSCI_B, support multiple serial communication modes with one hardware module. This chapter discusses the operation of the synchronous peripheral interface (SPI) mode.

Topic	Page
31.1 Enhanced Universal Serial Communication Interfaces (eUSCI_A, eUSCI_B) Overview	798
31.2 eUSCI Introduction – SPI Mode	798
31.3 eUSCI Operation – SPI Mode	800
31.4 eUSCI_A SPI Registers.....	806
31.5 eUSCI_B SPI Registers.....	815

31.1 Enhanced Universal Serial Communication Interfaces (eUSCI_A, eUSCI_B) Overview

Both the eUSCI_A and the eUSCI_B support serial communication in SPI mode.

31.2 eUSCI Introduction – SPI Mode

In synchronous mode, the eUSCI connects the device to an external system through three or four pins: UCxSIMO, UCxSOMI, UCxCLK, and UCxSTE. SPI mode is selected when the UCSYNC bit is set, and SPI mode (3-pin or 4-pin) is selected with the UCMODEx bits.

SPI mode features include:

- 7-bit or 8-bit data length
- LSB-first or MSB-first data transmit and receive
- 3-pin and 4-pin SPI operation
- Master or slave modes
- Independent transmit and receive shift registers
- Separate transmit and receive buffer registers
- Continuous transmit and receive operation
- Selectable clock polarity and phase control
- Programmable clock frequency in master mode
- Independent interrupt capability for receive and transmit
- Slave operation in LPM4

[Figure 31-1](#) shows the eUSCI when configured for SPI mode.

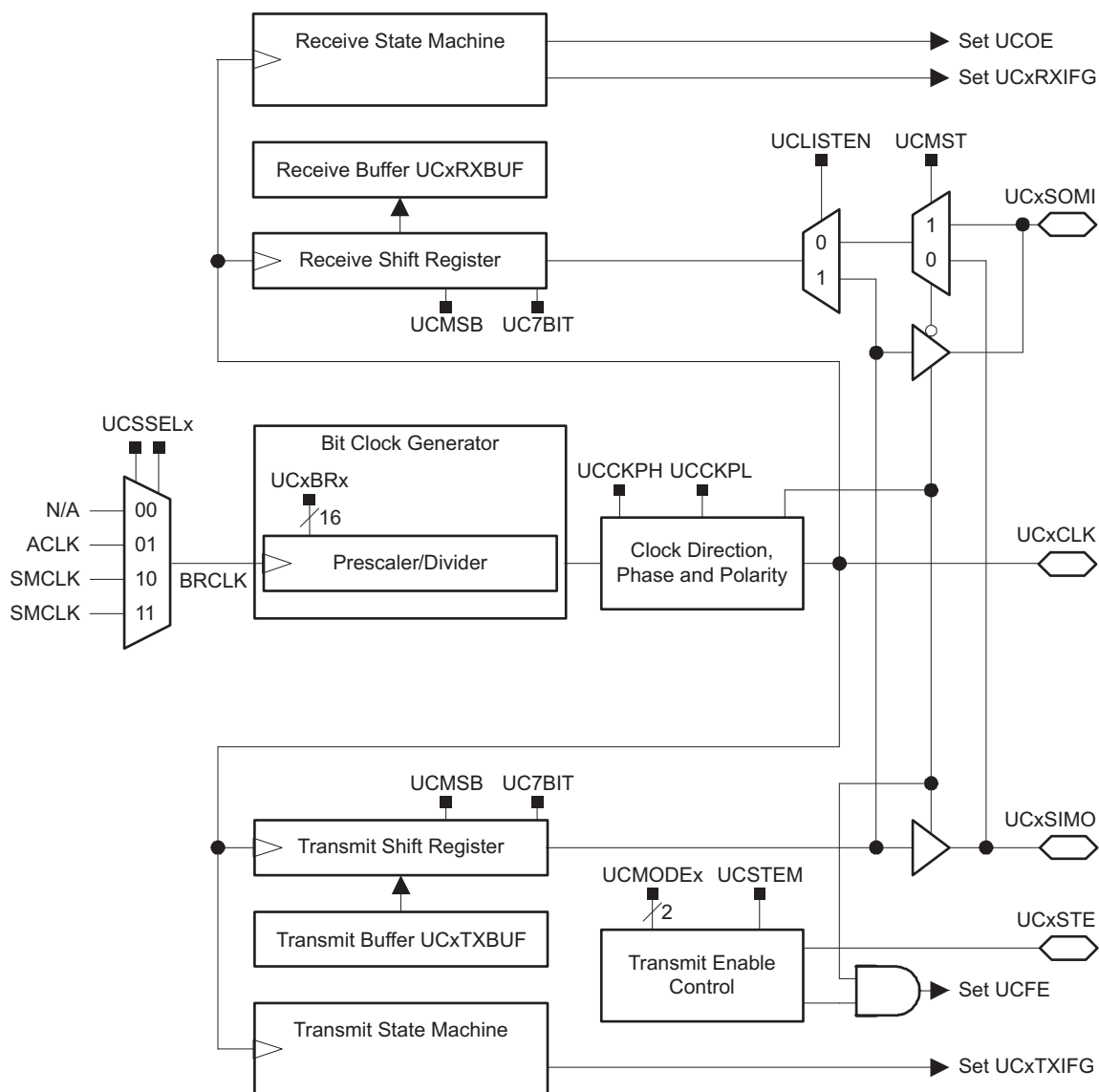


Figure 31-1. eUSCI Block Diagram – SPI Mode

31.3 eUSCI Operation – SPI Mode

In SPI mode, serial data is transmitted and received by multiple devices using a shared clock provided by the master. An additional pin controlled by the master, UCxSTE, is provided to enable a device to receive and transmit data.

Three or four signals are used for SPI data exchange:

- UCxSIMO – slave in, master out
Master mode: UCxSIMO is the data output line.
Slave mode: UCxSIMO is the data input line.
- UCxSOMI – slave out, master in
Master mode: UCxSOMI is the data input line.
Slave mode: UCxSOMI is the data output line.
- UCxCLK – eUSCI SPI clock
Master mode: UCxCLK is an output.
Slave mode: UCxCLK is an input.
- UCxSTE – slave transmit enable.

Used in 4-pin mode to allow multiple masters on a single bus. Not used in 3-pin mode. [Table 31-1](#) describes the UCxSTE operation.

Table 31-1. UCxSTE Operation

UCMODEx	UCxSTE Active State	UCxSTE	Slave	Master
01	High	0	Inactive	Active
		1	Active	Inactive
10	Low	0	Active	Inactive
		1	Inactive	Active

31.3.1 eUSCI Initialization and Reset

The eUSCI is reset by a PUC or by the UCSWRST bit. After a PUC, the UCSWRST bit is automatically set, keeping the eUSCI in a reset condition. When set, the UCSWRST bit resets the UCRXIE, UCTXIE, UCRXIFG, UCOE, and UCFE bits, and sets the UCTXIFG flag. Clearing UCSWRST releases the eUSCI for operation.

Configuring and reconfiguring the eUSCI module should be done when UCSWRST is set to avoid unpredictable behavior.

NOTE: Initializing or reconfiguring the eUSCI module

The recommended eUSCI initialization or reconfiguration process is:

1. Set UCSWRST.
`BIS.B #UCSWRST, &UCxCTL1`
2. Initialize all eUSCI registers with UCSWRST = 1 (including UCxCTL1).
3. Configure ports.
4. Ensure that any input signals into the SPI module such as UCxSOMI (in master mode) or UCxSIMO and UCxCLK (in slave mode) have settled to their final voltage levels before clearing UCSWRST and avoid any unwanted transitions during operation.
5. Clear UCSWRST.
`BIC.B #UCSWRST, &UCxCTL1`
6. Enable interrupts (optional) with UCRXIE or UCTXIE.

31.3.2 Character Format

The eUSCI module in SPI mode supports 7-bit and 8-bit character lengths selected by the UC7BIT bit. In 7-bit data mode, UCxRXBUF is LSB justified and the MSB is always reset. The UCMSB bit controls the direction of the transfer and selects LSB or MSB first.

NOTE: Default character format

The default SPI character transmission is LSB first. For communication with other SPI interfaces, MSB-first mode may be required.

NOTE: Character format for figures

Figures throughout this chapter use MSB-first format.

31.3.3 Master Mode

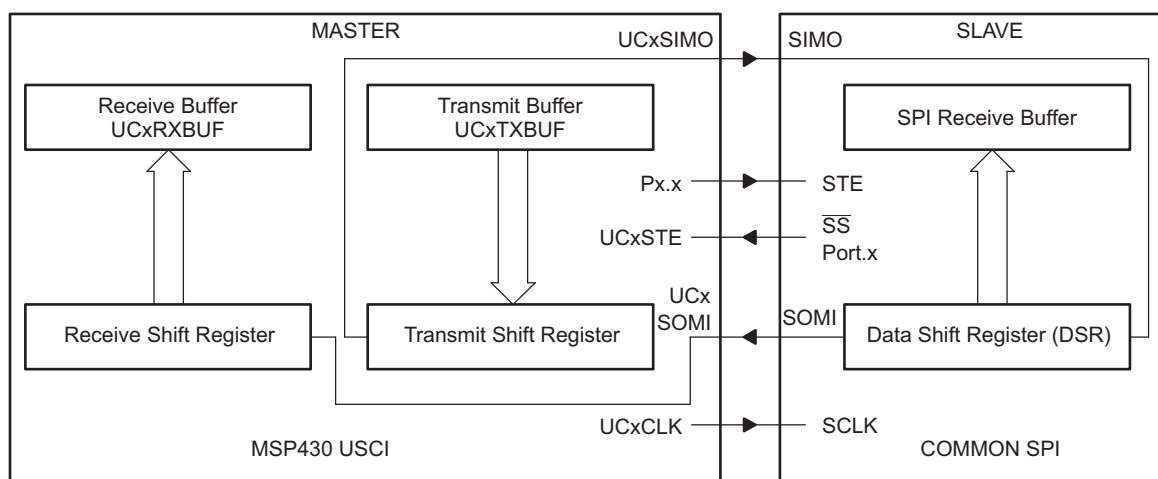


Figure 31-2. eUSCI Master and External Slave (UCSTEM = 0)

Figure 31-2 shows the eUSCI as a master in both 3-pin and 4-pin configurations. The eUSCI initiates data transfer when data is moved to the transmit data buffer UCxTXBUF. The UCxTXBUF data is moved to the transmit (TX) shift register when the TX shift register is empty, initiating data transfer on UCxSIMO starting with either the MSB or LSB, depending on the UCMSB setting. Data on UCxSOMI is shifted into the receive shift register on the opposite clock edge. When the character is received, the receive data is moved from the receive (RX) shift register to the received data buffer UCxRXBUF and the receive interrupt flag UCRXIFG is set, indicating the RX/TX operation is complete.

A set transmit interrupt flag, UCTXIFG, indicates that data has moved from UCxTXBUF to the TX shift register and UCxTXBUF is ready for new data. It does not indicate RX/TX completion.

To receive data into the eUSCI in master mode, data must be written to UCxTXBUF, because receive and transmit operations operate concurrently.

There are two different options for configuring the eUSCI as a 4-pin master, which are described in the next sections:

- The fourth pin is used as input to prevent conflicts with other masters (UCSTEM = 0).
- The fourth pin is used as output to generate a slave enable signal (UCSTEM = 1).

The bit UCSTEM is used to select the corresponding mode.

31.3.3.1 4-Pin SPI Master Mode (UCSTEM = 0)

In 4-pin master mode with UCSTEM = 0, UCxSTE is a digital input that can be used to prevent conflicts with another master and controls the master as described in [Table 31-1](#). When UCxSTE is in the master-inactive state and UCSTEM = 0:

- UCxSIMO and UCxCLK are set to inputs and no longer drive the bus.
- The error bit UCFE is set, indicating a communication integrity violation to be handled by the user.
- The internal state machines are reset and the shift operation is aborted.

If data is written into UCxTXBUF while the master is held inactive by UCxSTE, it is transmit as soon as UCxSTE transitions to the master-active state. If an active transfer is aborted by UCxSTE transitioning to the master-inactive state, the data must be rewritten into UCxTXBUF to be transferred when UCxSTE transitions back to the master-active state. The UCxSTE input signal is not used in 3-pin master mode.

31.3.3.2 4-Pin SPI Master Mode (UCSTEM = 1)

If UCSTEM = 1 in 4-pin master mode, UCxSTE is a digital output. In this mode the slave enable signal for a single slave is automatically generated on UCxSTE. The corresponding behavior can be seen in [Figure 31-4](#).

If multiple slaves are desired, this feature is not applicable and the software needs to use general purpose I/O pins instead to generate STE signals for each slave individually.

31.3.4 Slave Mode

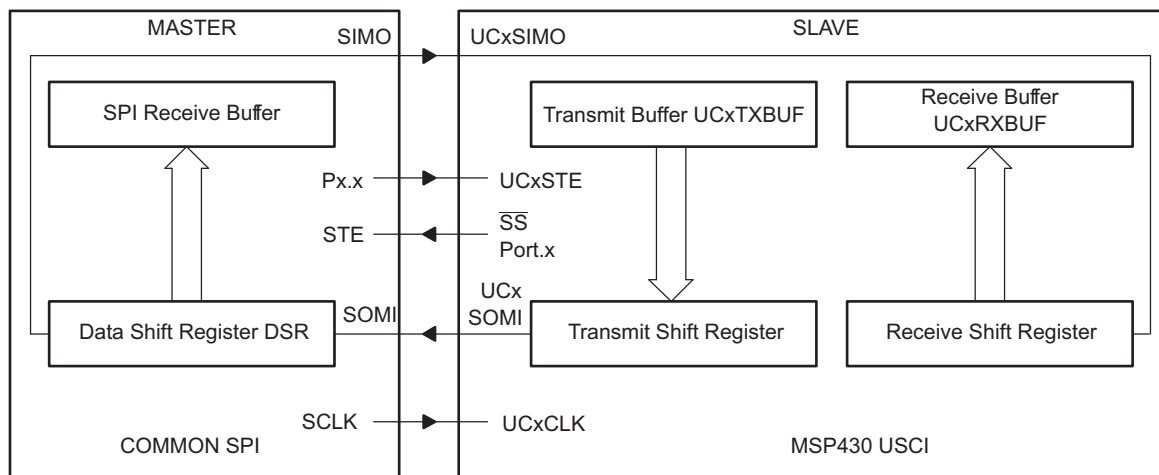


Figure 31-3. eUSCI Slave and External Master

[Figure 31-3](#) shows the eUSCI as a slave in both 3-pin and 4-pin configurations. UCxCLK is used as the input for the SPI clock and must be supplied by the external master. The data-transfer rate is determined by this clock and not by the internal bit clock generator. Data written to UCxTXBUF and moved to the TX shift register before the start of UCxCLK is transmitted on UCxSOMI. Data on UCxSIMO is shifted into the receive shift register on the opposite edge of UCxCLK and moved to UCxRXBUF when the set number of bits are received. When data is moved from the RX shift register to UCxRXBUF, the UCRXIFG interrupt flag is set, indicating that data has been received. The overrun error bit UCOE is set when the previously received data is not read from UCxRXBUF before new data is moved to UCxRXBUF.

31.3.4.1 4-Pin SPI Slave Mode

In 4-pin slave mode, UCxSTE is a digital input used by the slave to enable the transmit and receive operations and is driven by the SPI master. When UCxSTE is in the slave-active state, the slave operates normally. When UCxSTE is in the slave-inactive state:

- Any receive operation in progress on UCxSIMO is halted.
- UCxSOMI is set to the input direction.
- The shift operation is halted until the UCxSTE line transitions into the slave transmit active state.

The UCxSTE input signal is not used in 3-pin slave mode.

31.3.5 SPI Enable

When the eUSCI module is enabled by clearing the UCSWRST bit, it is ready to receive and transmit. In master mode, the bit clock generator is ready, but is not clocked nor producing any clocks. In slave mode, the bit clock generator is disabled and the clock is provided by the master.

A transmit or receive operation is indicated by UCBUSY = 1.

A PUC or set UCSWRST bit disables the eUSCI immediately and any active transfer is terminated.

31.3.5.1 Transmit Enable

In master mode, writing to UCxTXBUF activates the bit clock generator, and the data begins to transmit.

In slave mode, transmission begins when a master provides a clock and, in 4-pin mode, when the UCxSTE is in the slave-active state.

31.3.5.2 Receive Enable

The SPI receives data when a transmission is active. Receive and transmit operations operate concurrently.

31.3.6 Serial Clock Control

UCxCLK is provided by the master on the SPI bus. When UCMST = 1, the bit clock is provided by the eUSCI bit clock generator on the UCxCLK pin. The clock used to generate the bit clock is selected with the UCSSELx bits. When UCMST = 0, the eUSCI clock is provided on the UCxCLK pin by the master, the bit clock generator is not used, but the UCSSELx bits must be set to 0. The SPI receiver and transmitter operate in parallel and use the same clock source for data transfer.

The 16-bit value of UCBRx in the bit rate control registers UCxxBRW is the division factor of the eUSCI clock source, BRCLK. With UCBRx = 0 the maximum bit clock that can be generated in master mode is BRCLK. Modulation is not used in SPI mode, and UCAxMCTL should be cleared when using SPI mode for eUSCI_A.

The UCAxCLK or UCBxCLK frequency is given by:

$$f_{\text{BitClock}} = f_{\text{BRCLK}} / \text{UCBRx}$$

If UCBRx = 0, $f_{\text{BitClock}} = f_{\text{BRCLK}}$

Even UCBRx settings result in even divisions and, thus, generate a bit clock with a 50/50 duty cycle.

Odd UCBRx settings result in odd divisions. In this case, the high phase of the bit clock is one BRCLK cycle longer than the low phase.

When UCBRx = 0, no division is applied to BRCLK, and the bit clock equals BRCLK.

31.3.6.1 Serial Clock Polarity and Phase

The polarity and phase of UCxCLK are independently configured through the UCCKPL and UCCKPH control bits of the eUSCI. Timing for each case is shown in [Figure 31-4](#).

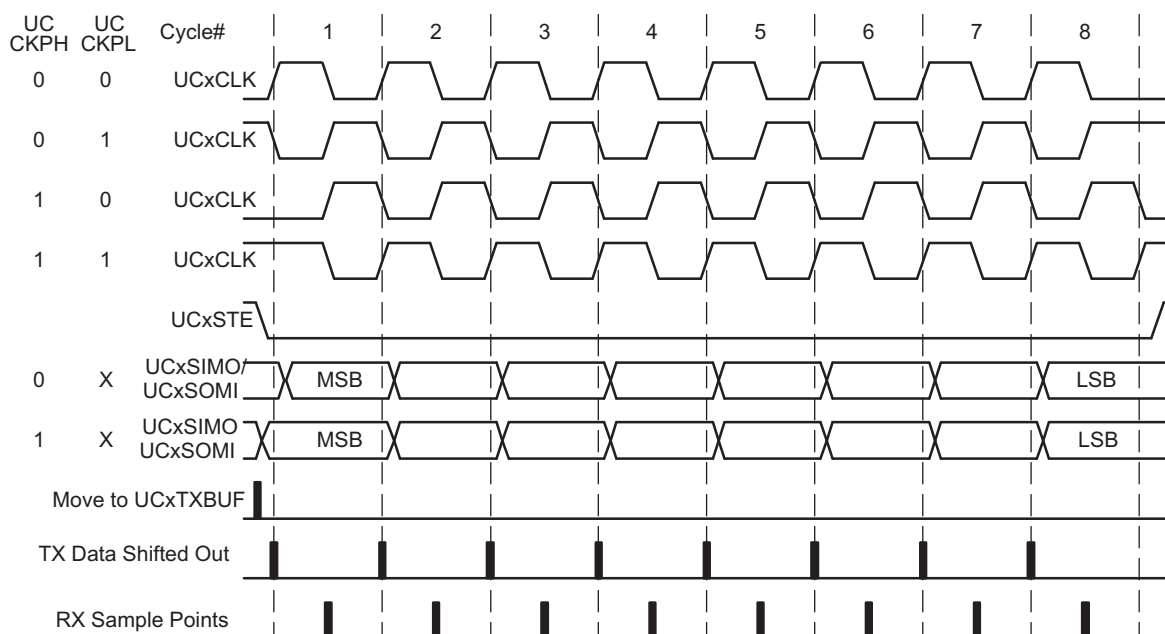


Figure 31-4. eUSCI SPI Timing With UCMSB = 1

31.3.7 Using the SPI Mode With Low-Power Modes

The eUSCI module provides automatic clock activation for use with low-power modes. When the eUSCI clock source is inactive because the device is in a low-power mode, the eUSCI module automatically activates it when needed, regardless of the control-bit settings for the clock source. The clock remains active until the eUSCI module returns to its idle condition. After the eUSCI module returns to the idle condition, control of the clock source reverts to the settings of its control bits.

In SPI slave mode, no internal clock source is required because the clock is provided by the external master. It is possible to operate the eUSCI in SPI slave mode while the device is in LPM4 and all clock sources are disabled. The receive or transmit interrupt can wake up the CPU from any low-power mode.

31.3.8 eUSCI Interrupts in SPI Mode

The eUSCI has only one interrupt vector that is shared for transmission and for reception. eUSCI_Ax and eUSCI_Bx do not share the same interrupt vector.

31.3.8.1 SPI Transmit Interrupt Operation

The UCTXIFG interrupt flag is set by the transmitter to indicate that UCxTXBUF is ready to accept another character. An interrupt request is generated if UCTXIE and GIE are also set. UCTXIFG is automatically reset if a character is written to UCxTXBUF. UCTXIFG is set after a PUC or when UCSWRST = 1. UCTXIE is reset after a PUC or when UCSWRST = 1.

NOTE: Writing to UCxTXBUF in SPI mode

Data written to UCxTXBUF when UCTXIFG = 0 may result in erroneous data transmission.

31.3.8.2 SPI Receive Interrupt Operation

The UCRXIFG interrupt flag is set each time a character is received and loaded into UCxRXBUF. An interrupt request is generated if UCRXIE and GIE are also set. UCRXIFG and UCRXIE are reset by a system reset PUC signal or when UCSWRST = 1. UCRXIFG is automatically reset when UCxRXBUF is read.

31.3.8.3 UCxIV, Interrupt Vector Generator

The eUSCI interrupt flags are prioritized and combined to source a single interrupt vector. The interrupt vector register UCxIV is used to determine which flag requested an interrupt. The highest-priority enabled interrupt generates a number in the UCxIV register that can be evaluated or added to the program counter (PC) to automatically enter the appropriate software routine. Disabled interrupts do not affect the UCxIV value.

Any access, read or write, of the UCxIV register automatically resets the highest-pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt.

31.3.8.3.1 UCxIV Software Example

The following software example shows the recommended use of UCxIV. The UCxIV value is added to the PC to automatically jump to the appropriate routine. The following example is given for eUSCI_B0.

```
USCI_SPI_ISR
    ADD    &UCB0IV, PC    ; Add offset to jump table
    RETI                                ; Vector 0: No interrupt
    JMP    RXIFG_ISR      ; Vector 2: RXIFG
TXIFG_ISR                                ; Vector 4: TXIFG
    ...                            ; Task starts here
    RETI                                ; Return
RXIFG_ISR                                ; Vector 2
    ...                            ; Task starts here
    RETI                                ; Return
```

31.4 eUSCI_A SPI Registers

The eUSCI_A registers applicable in SPI mode and their address offsets are listed in [Table 31-2](#). The base addresses can be found in the device-specific data sheet.

Table 31-2. eUSCI_A SPI Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	UCAxCTLW0	eUSCI_Ax Control Word 0	Read/write	Word	0001h	Section 31.4.1
00h	UCAxCTL1	eUSCI_Ax Control 1	Read/write	Byte	01h	
01h	UCAxCTL0	eUSCI_Ax Control 0	Read/write	Byte	00h	
06h	UCAxBRW	eUSCI_Ax Bit Rate Control Word	Read/write	Word	0000h	Section 31.4.2
06h	UCAxBR0	eUSCI_Ax Bit Rate Control 0	Read/write	Byte	00h	
07h	UCAxBR1	eUSCI_Ax Bit Rate Control 1	Read/write	Byte	00h	
0Ah	UCAxSTATW	eUSCI_Ax Status	Read/write	Word	00h	Section 31.4.3
0Ch	UCAxRXBUF	eUSCI_Ax Receive Buffer	Read/write	Word	00h	Section 31.4.4
0Eh	UCAxTXBUF	eUSCI_Ax Transmit Buffer	Read/write	Word	00h	Section 31.4.5
1Ah	UCAxIE	eUSCI_Ax Interrupt Enable	Read/write	Word	00h	Section 31.4.6
1Ch	UCAxIFG	eUSCI_Ax Interrupt Flag	Read/write	Word	02h	Section 31.4.7
1Eh	UCAxIV	eUSCI_Ax Interrupt Vector	Read	Word	0000h	Section 31.4.8

31.4.1 UCxCTLW0 Register

eUSCI_Ax Control Register 0

Figure 31-5. UCxCTLW0 Register

15	14	13	12	11	10	9	8
UCCKPH	UCCKPL	UCMSB	UC7BIT	UCMST	UCMODEx		UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCSSELx		Reserved				UCSTEM	UCSWRST
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

Can be modified only when UCSWRST = 1.

Table 31-3. UCxCTLW0 Register Description

Bit	Field	Type	Reset	Description
15	UCCKPH	RW	0h	Clock phase select 0b = Data is changed on the first UCLK edge and captured on the following edge. 1b = Data is captured on the first UCLK edge and changed on the following edge.
14	UCCKPL	RW	0h	Clock polarity select 0b = The inactive state is low. 1b = The inactive state is high.
13	UCMSB	RW	0h	MSB first select. Controls the direction of the receive and transmit shift register. 0b = LSB first 1b = MSB first
12	UC7BIT	RW	0h	Character length. Selects 7-bit or 8-bit character length. 0b = 8-bit data 1b = 7-bit data
11	UCMST	RW	0h	Master mode select 0b = Slave mode 1b = Master mode
10-9	UCMODEx	RW	0h	eUSCI mode. The UCMODEx bits select the synchronous mode when UCSYNC = 1. 00b = 3-pin SPI 01b = 4-pin SPI with UCxSTE active high: Slave enabled when UCxSTE = 1 10b = 4-pin SPI with UCxSTE active low: Slave enabled when UCxSTE = 0 11b = Reserved
8	UCSYNC	RW	0h	Synchronous mode enable 0b = Asynchronous mode 1b = Synchronous mode
7-6	UCSSELx	RW	0h	eUSCI clock source select. These bits select the BRCLK source clock. 00b = UCxCLK in slave mode. Do not use in master mode. 01b = ACLK in master mode. Do not use in slave mode. 10b = SMCLK in master mode. Do not use in slave mode. 11b = SMCLK in master mode. Do not use in slave mode.
5-2	Reserved	R	0h	Reserved
1	UCSTEM	RW	0h	STE mode select in master mode. This byte is ignored in slave or 3-wire mode. 0b = STE pin is used to prevent conflicts with other masters 1b = STE pin is used to generate the enable signal for a 4-wire slave
0	UCSWRST	RW	1h	Software reset enable 0b = Disabled. eUSCI reset released for operation. 1b = Enabled. eUSCI logic held in reset state.

31.4.2 UCxBRW Register

eUSCI_Ax Bit Rate Control Register 1

Figure 31-6. UCxBRW Register

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when UCSWRST = 1.

Table 31-4. UCxBRW Register Description

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Bit clock prescaler setting. $f_{\text{BitClock}} = f_{\text{BRCLK}} / \text{UCBRx}$ If UCBRx = 0, $f_{\text{BitClock}} = f_{\text{BRCLK}}$

31.4.3 UCxSTATW Register

eUSCI_Ax Status Register

Figure 31-7. UCxSTATW Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCLISTEN	UCFE	UCOE	Reserved				UCBUSY
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r-0

Can be modified only when UCSWRST = 1.

Table 31-5. UCxSTATW Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7	UCLISTEN	RW	0h	Listen enable. The UCLISTEN bit selects loopback mode. 0b = Disabled 1b = Enabled. The transmitter output is internally fed back to the receiver.
6	UCFE	RW	0h	Framing error flag. This bit indicates a bus conflict in 4-wire master mode. UCFE is not used in 3-wire master or any slave mode. 0b = No error 1b = Bus conflict occurred
5	UCOE	RW	0h	Overrun error flag. This bit is set when a character is transferred into UCxRXBUF before the previous character was read. UCOE is cleared automatically when UCxRXBUF is read, and must not be cleared by software. Otherwise, it does not function correctly. 0b = No error 1b = Overrun error occurred
4-1	Reserved	RW	0h	Reserved
0	UCBUSY	R	0h	eUSCI busy. This bit indicates if a transmit or receive operation is in progress. 0b = eUSCI inactive 1b = eUSCI transmitting or receiving

31.4.4 UCxRXBUF Register

eUSCI_Ax Receive Buffer Register

Figure 31-8. UCxRXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCRXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 31-6. UCxRXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCRXBUFx	R	0h	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCxRXBUF resets the receive-error bits and UCRXIFG. In 7-bit data mode, UCxRXBUF is LSB justified and the MSB is always reset.

31.4.5 UCxTXBUF Register

eUSCI_Ax Transmit Buffer Register

Figure 31-9. UCxTXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 31-7. UCxTXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTXBUFx	RW	0h	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted. Writing to the transmit data buffer clears UCTXIFG. The MSB of UCxTXBUF is not used for 7-bit data and is reset.

31.4.6 UCxIE Register

eUSCI_Ax Interrupt Enable Register

Figure 31-10. UCxIE Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved						UCTXIE	UCRXIE
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0

Table 31-8. UCxIE Register Description

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIE	RW	0h	Transmit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE	RW	0h	Receive interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

31.4.7 UCAXIFG Register

eUSCI_Ax Interrupt Flag Register

Figure 31-11. UCAXIFG Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved						UCTXIFG	UCRXIFG
r-0	r-0	r-0	r-0	r-0	r-0	rw-1	rw-0

Table 31-9. UCAXIFG Register Description

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIFG	RW	1h	Transmit interrupt flag. UCTXIFG is set when UCxxTXBUF empty. 0b = No interrupt pending 1b = Interrupt pending
0	UCRXIFG	RW	0h	Receive interrupt flag. UCRXIFG is set when UCxxRXBUF has received a complete character. 0b = No interrupt pending 1b = Interrupt pending

31.4.8 UCAXIV Register

eUSCI_Ax Interrupt Vector Register

Figure 31-12. UCAXIV Register

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCIVx							
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Table 31-10. UCAXIV Register Description

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI interrupt vector value 000h = No interrupt pending 002h = Interrupt Source: Data received; Interrupt Flag: UCRXIFG; Interrupt Priority: Highest 004h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG; Interrupt Priority: Lowest

31.5 eUSCI_B SPI Registers

The eUSCI_B registers applicable in SPI mode and their address offsets are listed in [Table 31-11](#). The base addresses can be found in the device-specific data sheet.

Table 31-11. eUSCI_B SPI Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	UCBxCTLW0	eUSCI_Bx Control Word 0	Read/write	Word	01C1h	Section 31.5.1
00h	UCBxCTL1	eUSCI_Bx Control 1	Read/write	Byte	C1h	
01h	UCBxCTL0	eUSCI_Bx Control 0	Read/write	Byte	01h	
06h	UCBxBRW	eUSCI_Bx Bit Rate Control Word	Read/write	Word	0000h	Section 31.5.2
06h	UCBxBR0	eUSCI_Bx Bit Rate Control 0	Read/write	Byte	00h	
07h	UCBxBR1	eUSCI_Bx Bit Rate Control 1	Read/write	Byte	00h	
08h	UCBxSTATW	eUSCI_Bx Status	Read/write	Word	00h	Section 31.5.3
0Ch	UCBxRXBUF	eUSCI_Bx Receive Buffer	Read/write	Word	00h	Section 31.5.4
0Eh	UCBxTXBUF	eUSCI_Bx Transmit Buffer	Read/write	Word	00h	Section 31.5.5
2Ah	UCBxIE	eUSCI_Bx Interrupt Enable	Read/write	Word	00h	Section 31.5.6
2Ch	UCBxIFG	eUSCI_Bx Interrupt Flag	Read/write	Word	02h	Section 31.5.7
2Eh	UCBxIV	eUSCI_Bx Interrupt Vector	Read	Word	0000h	Section 31.5.8

31.5.1 UCBxCTLW0 Register

eUSCI_Bx Control Register 0

Figure 31-13. UCBxCTLW0 Register

15	14	13	12	11	10	9	8
UCCKPH	UCCKPL	UCMSB	UC7BIT	UCMST	UCMODEx		UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1
7	6	5	4	3	2	1	0
UCSSELx		Reserved				UCSTEM	UCSWRST
rw-1	rw-1	r0	rw-0	rw-0	rw-0	rw-0	rw-1

Can be modified only when UCSWRST = 1.

Table 31-12. UCBxCTLW0 Register Description

Bit	Field	Type	Reset	Description
15	UCCKPH	RW	0h	Clock phase select 0b = Data is changed on the first UCLK edge and captured on the following edge. 1b = Data is captured on the first UCLK edge and changed on the following edge.
14	UCCKPL	RW	0h	Clock polarity select 0b = The inactive state is low. 1b = The inactive state is high.
13	UCMSB	RW	0h	MSB first select. Controls the direction of the receive and transmit shift register. 0b = LSB first 1b = MSB first
12	UC7BIT	RW	0h	Character length. Selects 7-bit or 8-bit character length. 0b = 8-bit data 1b = 7-bit data
11	UCMST	RW	0h	Master mode select 0b = Slave mode 1b = Master mode
10-9	UCMODEx	RW	0h	eUSCI mode. The UCMODEx bits select the synchronous mode when UCSYNC = 1. 00b = 3-pin SPI 01b = 4-pin SPI with UCxSTE active high: Slave enabled when UCxSTE = 1 10b = 4-pin SPI with UCxSTE active low: Slave enabled when UCxSTE = 0 11b = I2C mode
8	UCSYNC	RW	1h	Synchronous mode enable 0b = Asynchronous mode 1b = Synchronous mode
7-6	UCSSELx	RW	3h	eUSCI clock source select. These bits select the BRCLK source clock. 00b = UCxCLK in slave mode. Don't use in master mode. 01b = ACLK in master mode. Don't use in slave mode. 10b = SMCLK in master mode. Don't use in slave mode. 11b = SMCLK in master mode. Don't use in slave mode.
5-2	Reserved	R	0h	Reserved
1	UCSTEM	RW	0h	STE mode select in master mode. This byte is ignored in slave or 3-wire mode. 0b = STE pin is used to prevent conflicts with other masters 1b = STE pin is used to generate the enable signal for a 4-wire slave
0	UCSWRST	RW	1h	Software reset enable 0b = Disabled. eUSCI reset released for operation. 1b = Enabled. eUSCI logic held in reset state.

31.5.2 UCBxBRW Register

eUSCI_Bx Bit Rate Control Register 1

Figure 31-14. UCBxBRW Register

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when UCSWRST = 1.

Table 31-13. UCBxBRW Register Description

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Bit clock prescaler setting. $f_{\text{BitClock}} = f_{\text{BRCLK}} / \text{UCBRx}$ If UCBRx = 0, $f_{\text{BitClock}} = f_{\text{BRCLK}}$

31.5.3 UCBxSTATW Register

eUSCI_Bx Status Register

Figure 31-15. UCBxSTATW Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCLISTEN	UCFE	UCOE	Reserved				UCBUSY
rw-0	rw-0	rw-0	r0	r0	r0	r0	r-0

Can be modified only when UCSWRST = 1.

Table 31-14. UCBxSTATW Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7	UCLISTEN	RW	0h	Listen enable. The UCLISTEN bit selects loopback mode. 0b = Disabled 1b = Enabled. The transmitter output is internally fed back to the receiver.
6	UCFE	RW	0h	Framing error flag. This bit indicates a bus conflict in 4-wire master mode. UCFE is not used in 3-wire master or any slave mode. 0b = No error 1b = Bus conflict occurred
5	UCOE	RW	0h	Overrun error flag. This bit is set when a character is transferred into UCxRXBUF before the previous character was read. UCOE is cleared automatically when UCxRXBUF is read, and must not be cleared by software. Otherwise, it does not function correctly. 0b = No error 1b = Overrun error occurred
4-1	Reserved	R	0h	Reserved
0	UCBUSY	R	0h	eUSCI busy. This bit indicates if a transmit or receive operation is in progress. 0b = eUSCI inactive 1b = eUSCI transmitting or receiving

31.5.4 UCBxRXBUF Register

eUSCI_Bx Receive Buffer Register

Figure 31-16. UCBxRXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCRXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 31-15. UCBxRXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCRXBUFx	R	0h	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCxRXBUF resets the receive-error bits and UCRXIFG. In 7-bit data mode, UCxRXBUF is LSB justified and the MSB is always reset.

31.5.5 UCBxTXBUF Register

eUSCI_Bx Transmit Buffer Register

Figure 31-17. UCBxTXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 31-16. UCBxTXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTXBUFx	RW	0h	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted. Writing to the transmit data buffer clears UCTXIFG. The MSB of UCxTXBUF is not used for 7-bit data and is reset.

31.5.6 UCBxIE Register

eUSCI_Bx Interrupt Enable Register

Figure 31-18. UCBxIE Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved						UCTXIE	UCRXIE
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0

Table 31-17. UCBxIE Register Description

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIE	RW	0h	Transmit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE	RW	0h	Receive interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

31.5.7 UCBxIFG Register

eUSCI_Bx Interrupt Flag Register

Figure 31-19. UCBxIFG Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved						UCTXIFG	UCRXIFG
r-0	r-0	r-0	r-0	r-0	r-0	rw-1	rw-0

Table 31-18. UCBxIFG Register Description

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIFG	RW	1h	Transmit interrupt flag. UCTXIFG is set when UCxxTXBUF empty. 0b = No interrupt pending 1b = Interrupt pending
0	UCRXIFG	RW	0h	Receive interrupt flag. UCRXIFG is set when UCxxRXBUF has received a complete character. 0b = No interrupt pending 1b = Interrupt pending

31.5.8 UCBxIV Register

eUSCI_Bx Interrupt Vector Register

Figure 31-20. UCBxIV Register

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCIVx							
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Table 31-19. UCBxIV Register Description

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI interrupt vector value 0000h = No interrupt pending 0002h = Interrupt Source: Data received; Interrupt Flag: UCRXIFG; Interrupt Priority: Highest 0004h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG; Interrupt Priority: Lowest

Enhanced Universal Serial Communication Interface (eUSCI) – I²C Mode

The enhanced universal serial communication interface B (eUSCI_B) supports multiple serial communication modes with one hardware module. This chapter discusses the operation of the I²C mode.

Topic	Page
32.1 Enhanced Universal Serial Communication Interface B (eUSCI_B) Overview	822
32.2 eUSCI_B Introduction – I²C Mode	822
32.3 eUSCI_B Operation – I²C Mode.....	823
32.4 eUSCI_B I2C Registers.....	844

32.1 Enhanced Universal Serial Communication Interface B (eUSCI_B) Overview

The eUSCI_B module supports two serial communication modes:

- I²C mode
- SPI mode

If more than one eUSCI_B module is implemented on one device, those modules are named with incrementing numbers. For example, if one device has two eUSCI_B modules, they are named eUSCI0_B and eUSCI1_B.

32.2 eUSCI_B Introduction – I²C Mode

In I²C mode, the eUSCI_B module provides an interface between the device and I²C-compatible devices connected by the two-wire I²C serial bus. External components attached to the I²C bus serially transmit or receive serial data to or from the eUSCI_B module through the 2-wire I²C interface.

The eUSCI_B I²C mode features include:

- 7-bit and 10-bit device addressing modes
- General call
- START, RESTART, STOP
- Multi-master transmitter or receiver mode
- Slave receiver or transmitter mode
- Standard mode up to 100 kbps and fast mode up to 400 kbps support
- Programmable UCxCLK frequency in master mode
- Designed for low power
- 8-bit byte counter with interrupt capability and automatic STOP assertion
- Up to four hardware slave addresses, each having its own interrupt and DMA trigger
- Mask register for slave address and address received interrupt
- Clock low time-out interrupt to avoid bus stalls
- Slave operation in LPM4
- Slave receiver START detection for auto wake-up from LPMx modes (not LPM3.5 and LPM4.5)

[Figure 32-1](#) shows the eUSCI_B when configured in I²C mode.

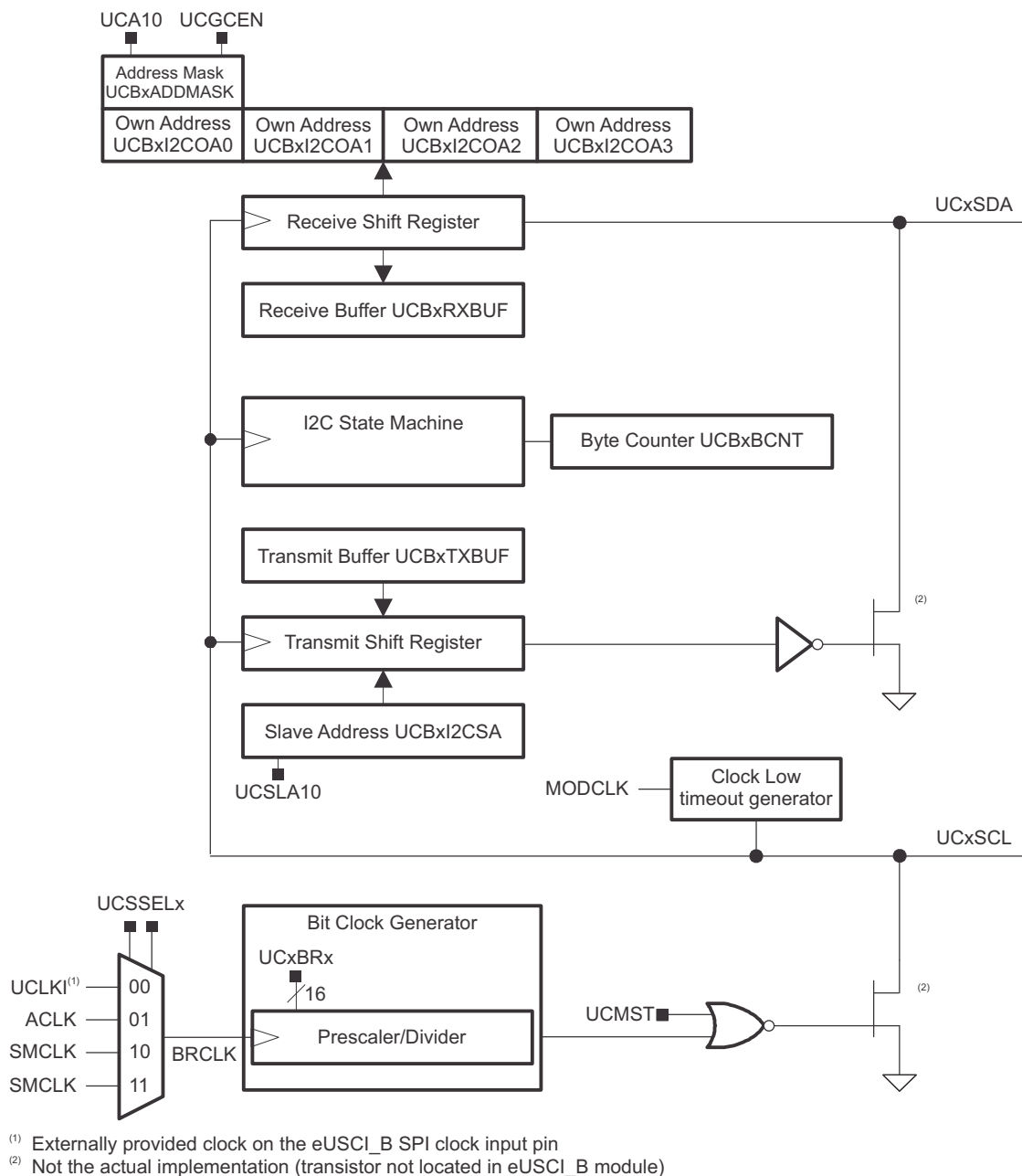


Figure 32-1. eUSCI_B Block Diagram – I²C Mode

32.3 eUSCI_B Operation – I²C Mode

The I²C mode supports any slave or master I²C-compatible device. [Figure 32-2](#) shows an example of an I²C bus. Each I²C device is recognized by a unique address and can operate as either a transmitter or a receiver. A device connected to the I²C bus can be considered as the master or the slave when performing data transfers. A master initiates a data transfer and generates the clock signal SCL. Any device addressed by a master is considered a slave.

I²C data is communicated using the serial data (SDA) pin and the serial clock (SCL) pin. Both SDA and SCL are bidirectional and must be connected to a positive supply voltage using a pullup resistor.

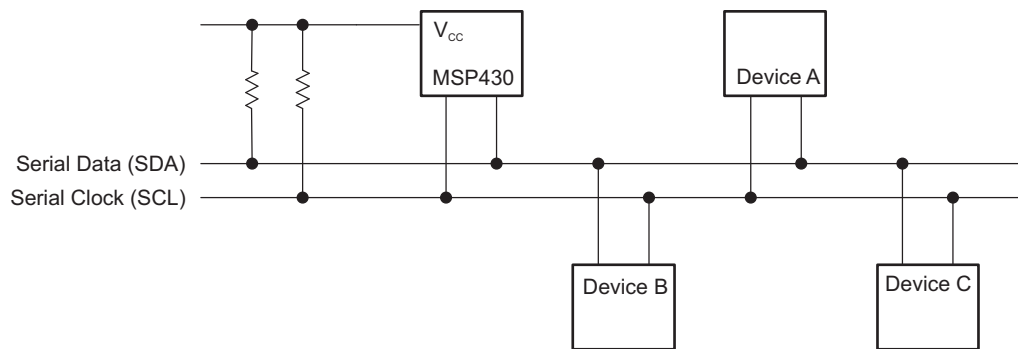


Figure 32-2. I²C Bus Connection Diagram

NOTE: SDA and SCL levels

The SDA and SCL pins must not be pulled up above the device V_{CC} level.

32.3.1 eUSCI_B Initialization and Reset

The eUSCI_B is reset by a PUC or by setting the UCSWRST bit. After a PUC, the UCSWRST bit is automatically set, keeping the eUSCI_B in a reset condition. To select I²C operation, the UCMODEx bits must be set to 11b. After module initialization, it is ready for transmit or receive operation. Clear UCSWRST to release the eUSCI_B for operation.

To avoid unpredictable behavior, configure or reconfigure the eUSCI_B module only when UCSWRST is set. Setting UCSWRST in I²C mode has the following effects:

- I²C communication stops.
- SDA and SCL are high impedance.
- UCBxSTAT, bits 15-8 and 6-4 are cleared.
- Registers UCBxIE and UCBxIFG are cleared.
- All other bits and registers remain unchanged.

NOTE: Initializing or reconfiguring the eUSCI_B module

The recommended eUSCI_B initialization/reconfiguration process is:

1. Set UCSWRST (BIS.B
#UCSWRST, &UCxCTL1).
 2. Initialize all eUSCI_B registers with UCSWRST = 1 (including UCxCTL1).
 3. Configure ports.
 4. Clear UCSWRST through software (BIC.B
#UCSWRST, &UCxCTL1).
 5. Enable interrupts (optional).
-

32.3.2 I²C Serial Data

One clock pulse is generated by the master device for each data bit transferred. The I²C mode operates with byte data. Data is transferred MSB first as shown in [Figure 32-3](#).

The first byte after a START condition consists of a 7-bit slave address and the R/ $\overline{W}$ bit. When R/ $\overline{W}$ = 0, the master transmits data to a slave. When R/ $\overline{W}$ = 1, the master receives data from a slave. The ACK bit is sent from the receiver after each byte on the ninth SCL clock.

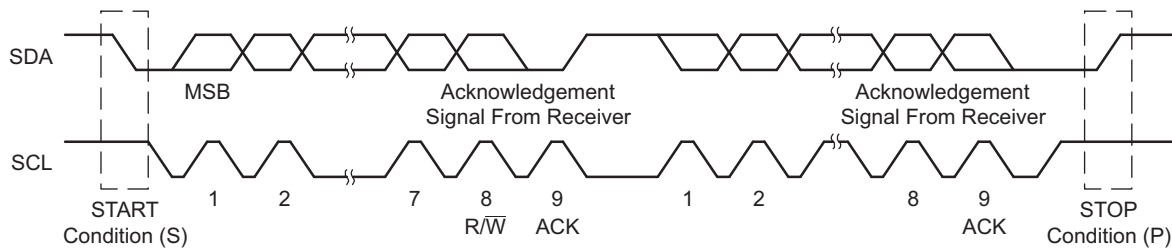


Figure 32-3. I²C Module Data Transfer

START and STOP conditions are generated by the master and are shown in [Figure 32-3](#). A START condition is a high-to-low transition on the SDA line while SCL is high. A STOP condition is a low-to-high transition on the SDA line while SCL is high. The bus busy bit, UCBBUSY, is set after a START and cleared after a STOP.

Data on SDA must be stable during the high period of SCL (see [Figure 32-4](#)). The high and low state of SDA can change only when SCL is low, otherwise START or STOP conditions are generated.

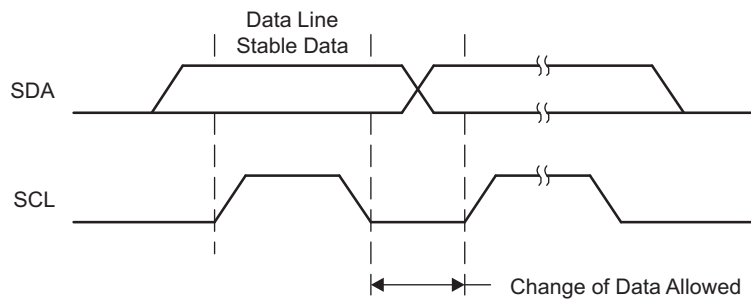


Figure 32-4. Bit Transfer on I²C Bus

32.3.3 I²C Addressing Modes

The I²C mode supports 7-bit and 10-bit addressing modes.

32.3.3.1 7-Bit Addressing

In the 7-bit addressing format (see [Figure 32-5](#)), the first byte is the 7-bit slave address and the $R/\overline{W}$ bit. The ACK bit is sent from the receiver after each byte.

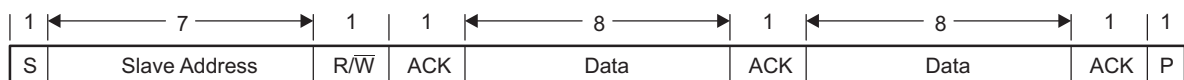
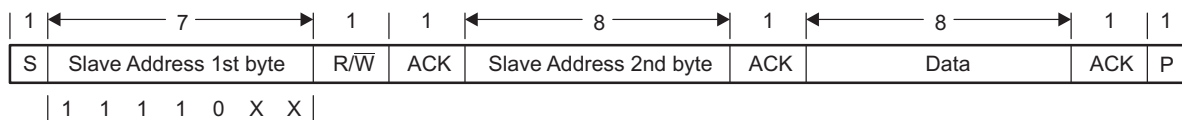


Figure 32-5. I²C Module 7-Bit Addressing Format

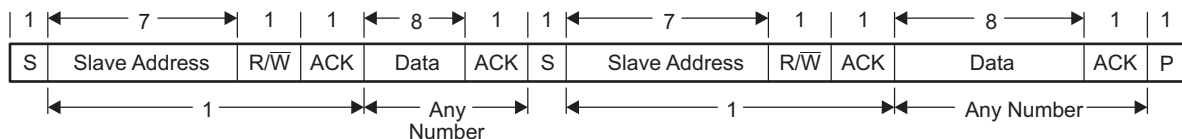
32.3.3.2 10-Bit Addressing

In the 10-bit addressing format (see [Figure 32-6](#)), the first byte is made up of 11110b plus the two MSBs of the 10-bit slave address and the R/W bit. The ACK bit is sent from the receiver after each byte. The next byte is the remaining eight bits of the 10-bit slave address, followed by the ACK bit and the 8-bit data. See [I2C Slave 10-bit Addressing Mode](#) and [I2C Master 10-bit Addressing Mode](#) for details how to use the 10-bit addressing mode with the eUSCI_B module.


Figure 32-6. I²C Module 10-Bit Addressing Format

32.3.3.3 Repeated Start Conditions

The direction of data flow on SDA can be changed by the master, without first stopping a transfer, by issuing a repeated START condition. This is called a RESTART. After a RESTART is issued, the slave address is again sent out with the new data direction specified by the R/W bit. The RESTART condition is shown in [Figure 32-7](#).


Figure 32-7. I²C Module Addressing Format With Repeated START Condition

32.3.4 I²C Quick Setup

This section gives a quick introduction into the operation of the eUSCI_B in I2C mode. The basic steps to start communication are described and shown as a software example. More detailed information about the possible configurations and details can be found in [Section 32.3.5](#).

The latest code examples can be found on the MSP430 web under "Code Examples".

To set up the eUSCI_B as a master transmitter that transmits to a slave with the address 0x12h, only a few steps are needed (see [Example 32-1](#)).

Example 32-1. Master TX With 7-Bit Address

```
UCBxCTL1 |= UCSWRST;           // put eUSCI_B in reset state
UCBxCTLW0 |= UCMODE_3 + UCMST; // I2C master mode
UCBxBRW = 0x0008;              // baud rate = SMCLK / 8
UCBxCTLW1 = UCASTP_2;          // automatic STOP assertion
UCBxTBCNT = 0x07;              // TX 7 bytes of data
UCBxI2CSA = 0x0012;            // address slave is 12hex
P2SEL |= 0x03;                 // configure I2C pins (device specific)
UCBxCTL1 &= ^UCSWRST;          // eUSCI_B in operational state
UCBxIE |= UCTXIE;              // enable TX-interrupt
GIE;                           // general interrupt enable
...
// inside the eUSCI_B TX interrupt service routine
UCBxTXBUF = 0x77;              // fill TX buffer
```

As shown in the code example, all configurations must be done while UCSWRST is set. To select the I²C operation of the eUSCI_B, UCMODE must be set accordingly. The baud rate of the transmission is set by writing the correct divider in the UCBxBRW register. The default clock selected is SMCLK. How many bytes are transmitted in one frame is controlled by the byte counter threshold register UCBxTBCNT together with the UCASTPx bits.

The slave address to send to is specified in the UCBxI2CSA register. Finally, the ports must be configured. This step is device dependent; see the data sheet for the pins that must be used.

Each byte that is to be transmitted must be written to the UCBxTXBUF inside the interrupt service routine. The recommended structure of the interrupt service routine can be found in [Example 32-3](#).

[Example 32-2](#) shows the steps needed to set up the eUSCI_B as a slave with the address 0x12h that is able to receive and transmit data to the master.

Example 32-2. Slave RX With 7-Bit Address

```
UCBxCTL1 |= UCSWRST;           // eUSCI_B in reset state
UCBxCTLW0 |= UCMODE_3;         // I2C slave mode
UCBxI2COA0 = 0x0412;          // own address is 12hex
P2SEL |= 0x03;                // configure I2C pins (device specific)
UCBxCTL1 &= ^UCSWRST;         // eUSCI_B in operational state
UCBxIE |= UCTXIE + UCRXIE;     // enable TX&RX-interrupt
GIE;                           // general interrupt enable
...
// inside the eUSCI_B TX interrupt service routine
UCBxTXBUF = 0x77;              // send 077h
...
// inside the eUSCI_B RX interrupt service routine
data = UCBxRXBUF;              // data is the internal variable
```

As shown in [Example 32-2](#), all configurations must be done while UCSWRST is set. For the slave, I²C operation is selected by setting UCMODE. The slave address is specified in the UCBxI2COA0 register. To enable the interrupts for receive and transmit requests, the according bits in UCBxIE and, at the end, GIE need to be set. Finally the ports must be configured. This step is device dependent; see the data sheet for the pins that are used.

The RX interrupt service routine is called for every byte received by a master device. The TX interrupt service routine is executed each time the master requests a byte. The recommended structure of the interrupt service routine can be found in [Example 32-3](#).

32.3.5 I²C Module Operating Modes

In I²C mode, the eUSCI_B module can operate in master transmitter, master receiver, slave transmitter, or slave receiver mode. The modes are discussed in the following sections. Time lines are used to illustrate the modes.

[Figure 32-8](#) shows how to interpret the time-line figures. Data transmitted by the master is represented by grey rectangles; data transmitted by the slave is represented by white rectangles. Data transmitted by the eUSCI_B module, either as master or slave, is shown by rectangles that are taller than the others.

Actions taken by the eUSCI_B module are shown in grey rectangles with an arrow indicating where in the data stream the action occurs. Actions that must be handled with software are indicated with white rectangles with an arrow pointing to where in the data stream the action must take place.

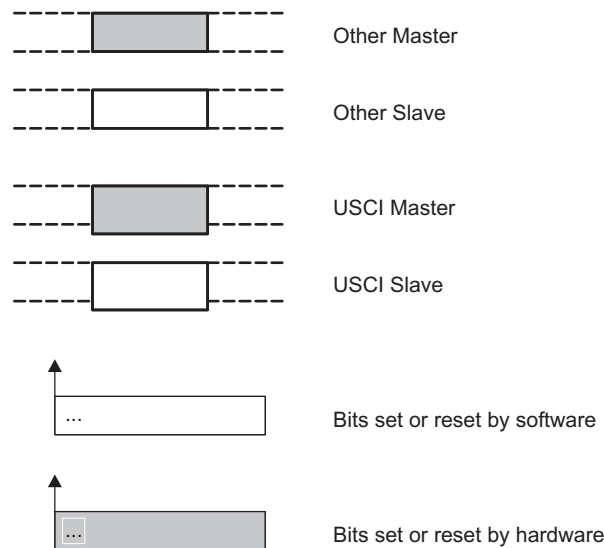


Figure 32-8. I²C Time-Line Legend

32.3.5.1 Slave Mode

The eUSCI_B module is configured as an I²C slave by selecting the I²C mode with UCMODEx = 11 and UCSYNC = 1 and clearing the UCMST bit.

Initially, the eUSCI_B module must be configured in receiver mode by clearing the UCTR bit to receive the I²C address. Afterwards, transmit and receive operations are controlled automatically, depending on the R/W bit received together with the slave address.

The eUSCI_B slave address is programmed with the UCBxI2COA0 register. Support for multiple slave addresses is explained in [Section 32.3.9](#). When UCA10 = 0, 7-bit addressing is selected. When UCA10 = 1, 10-bit addressing is selected. The UCGCEN bit selects if the slave responds to a general call.

When a START condition is detected on the bus, the eUSCI_B module receives the transmitted address and compares it against its own address stored in UCBxI2COA0. The UCSTTIFG flag is set when address received matches the eUSCI_B slave address.

32.3.5.1.1 I²C Slave Transmitter Mode

Slave transmitter mode is entered when the slave address transmitted by the master is identical to its own address with a set R/W bit. The slave transmitter shifts the serial data out on SDA with the clock pulses that are generated by the master device. The slave device does not generate the clock, but it does hold SCL low while intervention of the CPU is required after a byte has been transmitted.

If the master requests data from the slave, the eUSCI_B module is automatically configured as a transmitter and UCTR and UCTXIFG0 become set. The SCL line is held low until the first data to be sent is written into the transmit buffer UCBxTXBUF. Then the address is acknowledged and the data is transmitted. As soon as the data is transferred into the shift register, the UCTXIFG0 is set again. After the data is acknowledged by the master, the next data byte written into UCBxTXBUF is transmitted or, if the buffer is empty, the bus is stalled during the acknowledge cycle by holding SCL low until new data is written into UCBxTXBUF. If the master sends a NACK followed by a STOP condition, the UCSTPIFG flag is set. If the NACK is followed by a repeated START condition, the eUSCI_B I²C state machine returns to its address-reception state.

[Figure 32-9](#) shows the slave transmitter operation.

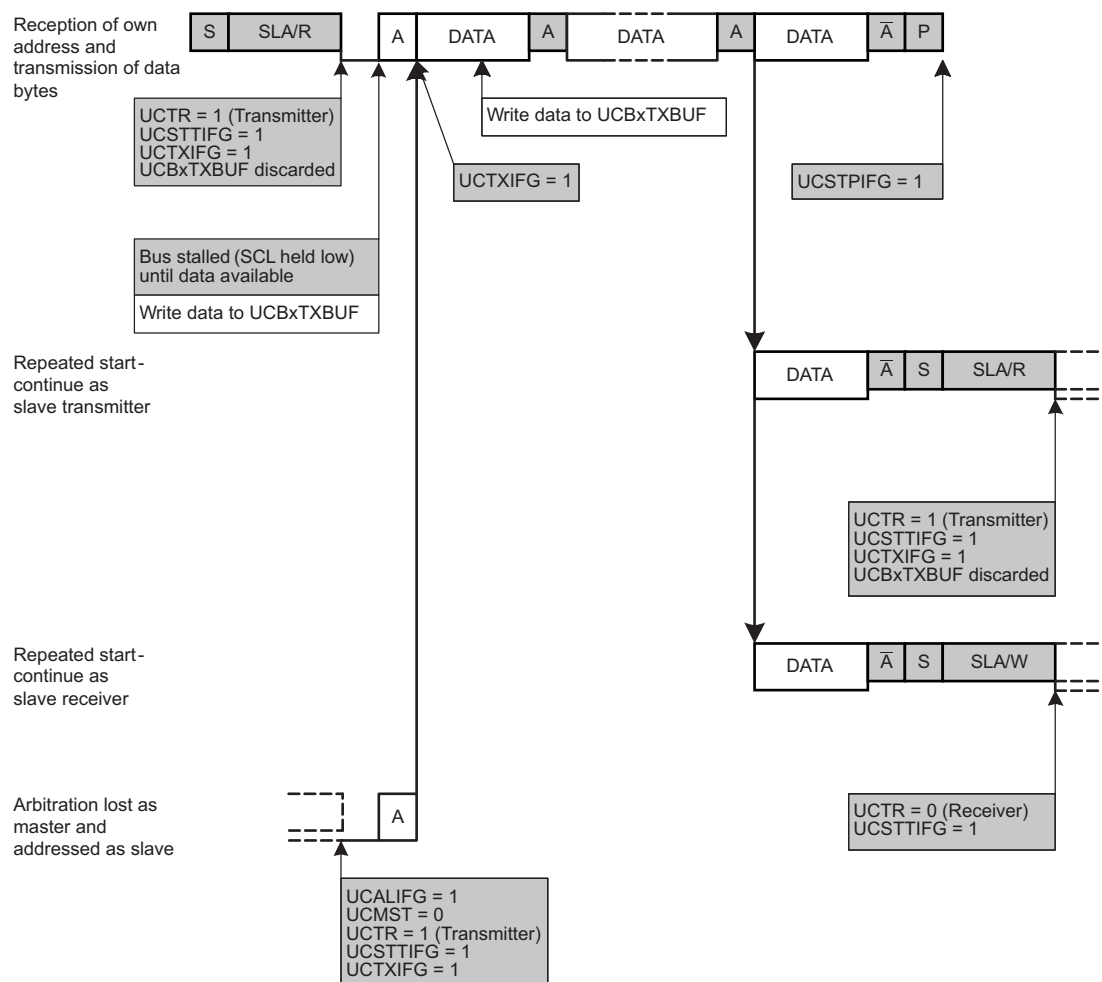


Figure 32-9. I²C Slave Transmitter Mode

32.3.5.1.2 I²C Slave Receiver Mode

Slave receiver mode is entered when the slave address transmitted by the master is identical to its own address and a cleared R/W bit is received. In slave receiver mode, serial data bits received on SDA are shifted in with the clock pulses that are generated by the master device. The slave device does not generate the clock, but it can hold SCL low if intervention of the CPU is required after a byte has been received.

If the slave receives data from the master, the eUSCI_B module is automatically configured as a receiver and UCTR is cleared. After the first data byte is received, the receive interrupt flag UCRXIFG0 is set. The eUSCI_B module automatically acknowledges the received data and can receive the next data byte.

If the previous data was not read from the receive buffer UCBxRXBUF at the end of a reception, the bus is stalled by holding SCL low. As soon as UCBxRXBUF is read, the new data is transferred into UCBxRXBUF, an acknowledge is sent to the master, and the next data can be received.

Setting the UCTXNACK bit causes a NACK to be transmitted to the master during the next acknowledgment cycle. A NACK is sent even if UCBxRXBUF is not ready to receive the latest data. If the UCTXNACK bit is set while SCL is held low, the bus is released, a NACK is transmitted immediately, and UCBxRXBUF is loaded with the last received data. Because the previous data was not read, that data is lost. To avoid loss of data, the UCBxRXBUF must be read before UCTXNACK is set.

When the master generates a STOP condition, the UCSTPIFG flag is set.

If the master generates a repeated START condition, the eUSCI_B I²C state machine returns to its address-reception state.

Figure 32-10 shows the I²C slave receiver operation.

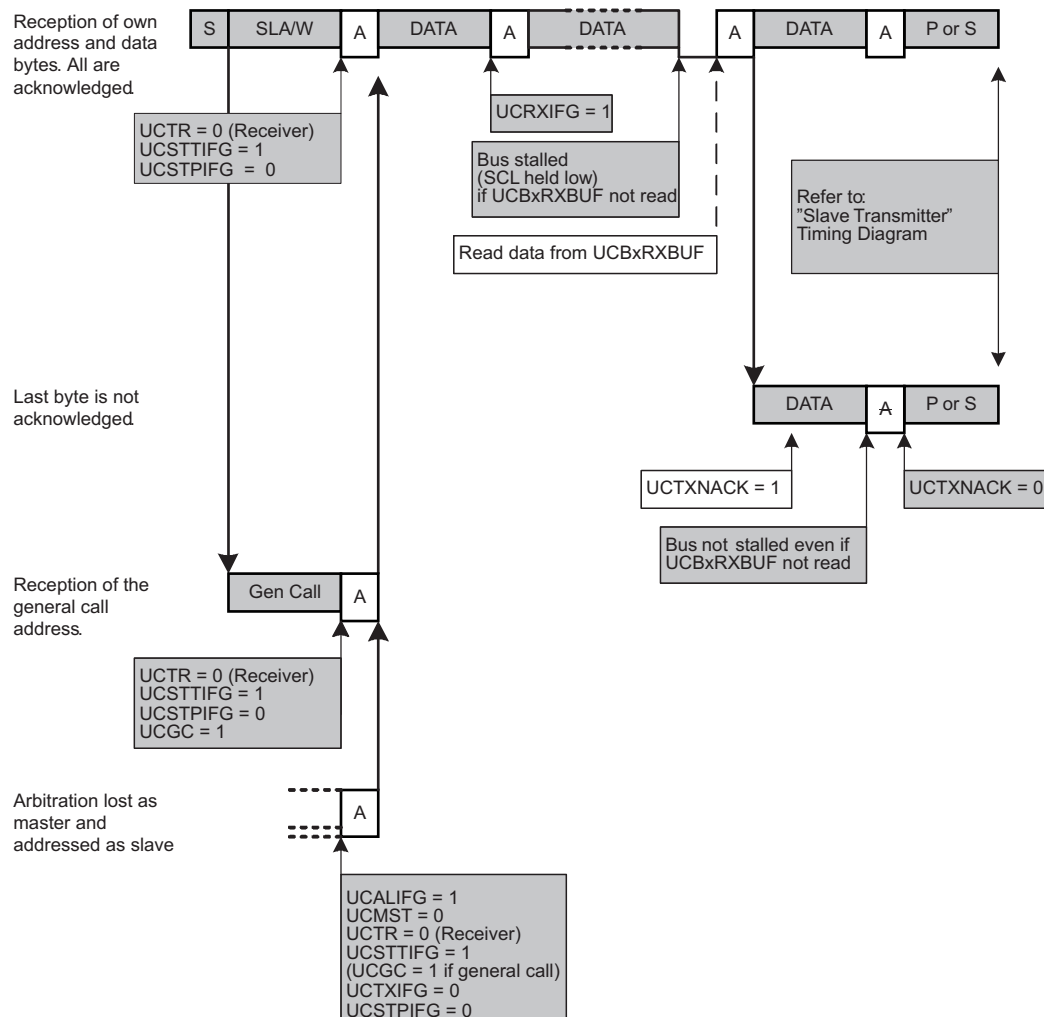
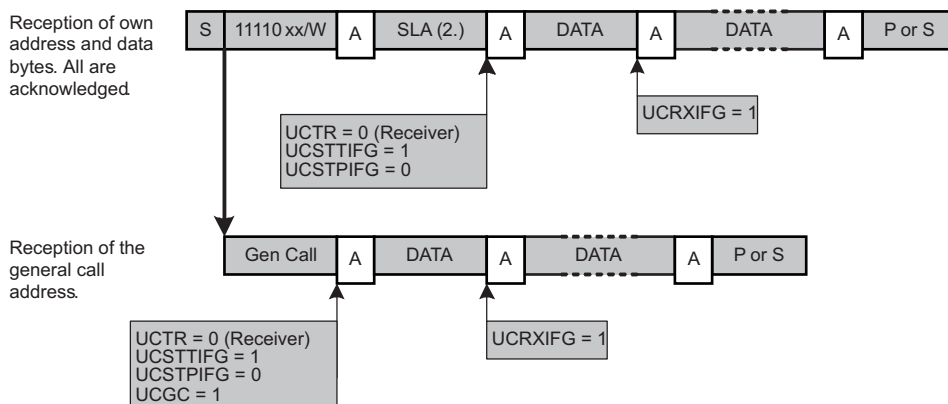


Figure 32-10. I²C Slave Receiver Mode

32.3.5.1.3 I²C Slave 10-Bit Addressing Mode

The 10-bit addressing mode is selected when UCA10 = 1 and is as shown in Figure 32-11. In 10-bit addressing mode, the slave is in receive mode after the full address is received. The eUSCI_B module indicates this by setting the UCSTTIFG flag while the UCTR bit is cleared. To switch the slave into transmitter mode, the master sends a repeated START condition together with the first byte of the address but with the R/W bit set. This sets the UCSTTIFG flag if it was previously cleared by software, and the eUSCI_B module switches to transmitter mode with UCTR = 1.

Slave Receiver



Slave Transmitter

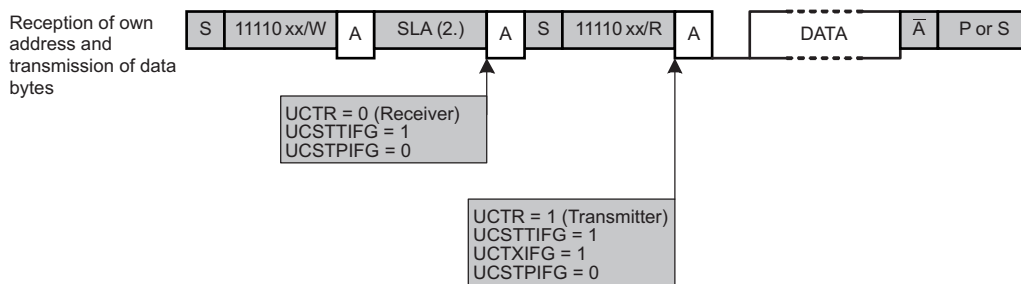


Figure 32-11. I²C Slave 10-Bit Addressing Mode

32.3.5.2 Master Mode

The eUSCI_B module is configured as an I²C master by selecting the I²C mode with UCMODEx = 11 and UCSYNC = 1 and setting the UCMST bit. When the master is part of a multi-master system, UCMM must be set and its own address must be programmed into the UCBxI2COA0 register. Support for multiple slave addresses is explained in [Section 32.3.9](#). When UCA10 = 0, 7-bit addressing is selected. When UCA10 = 1, 10-bit addressing is selected. The UGCEN bit selects if the eUSCI_B module responds to a general call.

NOTE: Addresses and multi-master systems

In master mode with own-address detection enabled (UCOAEN = 1)—especially in multi-master systems—it is not allowed to specify the same address in the own address and slave address register (UCBxI2CSA = UCBxI2COAx). This would mean that the eUSCI_B addresses itself.

The user software must ensure that this situation does not occur. There is no hardware detection for this case, and the consequence is unpredictable behavior of the eUSCI_B.

32.3.5.2.1 I²C Master Transmitter Mode

After initialization, master transmitter mode is initiated by writing the desired slave address to the UCBxI2CSA register, selecting the size of the slave address with the UCSLA10 bit, setting UCTR for transmitter mode, and setting UCTXSTT to generate a START condition.

The eUSCI_B module waits until the bus is available, then generates the START condition, and transmits the slave address. The UCTXIFG0 bit is set when the START condition is generated and the first data to be transmitted can be written into UCBxTXBUF. **The UCTXSTT flag is cleared as soon as the complete address is sent.**

The data written into UCBxTXBUF is transmitted if arbitration is not lost during transmission of the slave address. UCTXIFG0 is set again as soon as the data is transferred from the buffer into the shift register. If there is no data loaded to UCBxTXBUF before the acknowledge cycle, the bus is held during the acknowledge cycle with SCL low until data is written into UCBxTXBUF. Data is transmitted or the bus is held, as long as:

- No automatic STOP is generated
- The UCTXSTP bit is not set
- The UCTXSTT bit is not set

Setting UCTXSTP generates a STOP condition after the next acknowledge from the slave. If UCTXSTP is set during the transmission of the slave address or while the eUSCI_B module waits for data to be written into UCBxTXBUF, a STOP condition is generated, even if no data was transmitted to the slave. **In this case, the UCSTPIFG is set.** When transmitting a single byte of data, the UCTXSTP bit must be set while the byte is being transmitted or any time after transmission begins, without writing new data into UCBxTXBUF. Otherwise, only the address is transmitted. When the data is transferred from the buffer to the shift register, UCTXIFG0 is set, indicating data transmission has begun, and the UCTXSTP bit may be set. When UCASTPx = 10 is set, the byte counter is used for STOP generation and the user does not need to set the UCTXSTP. **This is recommended when transmitting only one byte.**

Setting UCTXSTT generates a repeated START condition. In this case, UCTR may be set or cleared to configure transmitter or receiver, and a different slave address may be written into UCBxI2CSA, if desired.

If the slave does not acknowledge the transmitted data, the not-acknowledge interrupt flag UCNACKIFG is set. The master must react with either a STOP condition or a repeated START condition. If data was already written into UCBxTXBUF, it is discarded. If this data should be transmitted after a repeated START, it must be written into UCBxTXBUF again. Any set UCTXSTT or UCTXSTP is also discarded.

Figure 32-12 shows the I²C master transmitter operation.



32.3.5.2.2 I²C Master Receiver Mode

After initialization, master receiver mode is initiated by writing the desired slave address to the UCBxI2CSA register, selecting the size of the slave address with the UCSLA10 bit, clearing UCTR for receiver mode, and setting UCTXSTT to generate a START condition.

The eUSCI_B module checks if the bus is available, generates the START condition, and transmits the slave address. The UCTXSTT flag is cleared as soon as the complete address is sent.

After the acknowledge of the address from the slave, the first data byte from the slave is received and acknowledged and the UCRXIFG flag is set. Data is received from the slave, as long as:

- No automatic STOP is generated
- The UCTXSTP bit is not set
- The UCTXSTT bit is not set

If a STOP condition was generated by the eUSCI_B module, the UCSTPIFG is set. If UCBxRXBUF is not read, the master holds the bus during reception of the last data bit and until the UCBxRXBUF is read.

If the slave does not acknowledge the transmitted address, the not-acknowledge interrupt flag UCNACKIFG is set. The master must react with either a STOP condition or a repeated START condition.

A STOP condition is either generated by the automatic STOP generation or by setting the UCTXSTP bit. The next byte received from the slave is followed by a NACK and a STOP condition. This NACK occurs immediately if the eUSCI_B module is currently waiting for UCBxRXBUF to be read.

If a RESTART is sent, UCTR may be set or cleared to configure transmitter or receiver, and a different slave address may be written into UCBxI2CSA if desired.

Figure 32-13 shows the I²C master receiver operation.

NOTE: Consecutive master transactions without repeated START

When performing multiple consecutive I²C master transactions without the repeated START feature, the current transaction must be completed before the next one is initiated. This can be done by ensuring that the transmit STOP condition flag UCTXSTP is cleared before the next I²C transaction is initiated with setting UCTXSTT = 1. Otherwise, the current transaction might be affected.

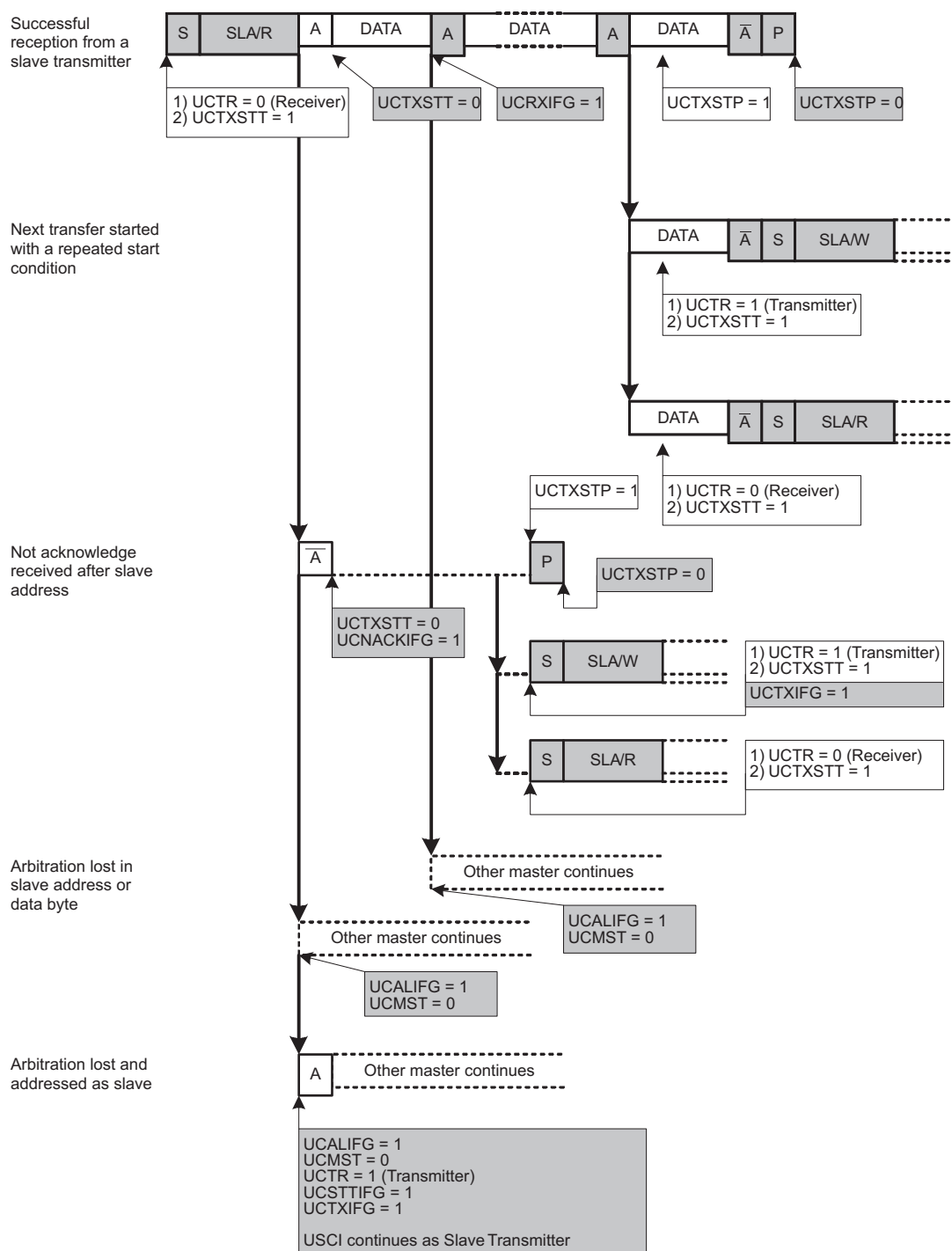


Figure 32-13. I²C Master Receiver Mode

32.3.6 Glitch Filtering

According to the I²C standard, both the SDA and the SCL line need to be glitch filtered. The eUSCI_B module provides the UCGLITx bits to configure the length of this glitch filter:

Table 32-1. Glitch Filter Length Selection Bits

UCGLITx	Corresponding Glitch Filter Length on SDA and SCL	According to I ² C Standard
00	Pulses of max 50-ns length are filtered	yes
01	Pulses of max 25-ns length are filtered.	no
10	Pulses of max 12.5-ns length are filtered.	no
11	Pulses of max 6.25-ns length are filtered.	no

32.3.7 I²C Clock Generation and Synchronization

The I²C clock SCL is provided by the master on the I²C bus. When the eUSCI_B is in master mode, BITCLK is provided by the eUSCI_B bit clock generator and the clock source is selected with the UCSSELx bits. In slave mode, the bit clock generator is not used and the UCSSELx bits are don't care.

The 16-bit value of UCBRx in register UCBxBRW is the division factor of the eUSCI_B clock source, BRCLK. The maximum bit clock that can be used in single master mode is $f_{BRCLK}/4$. In multi-master mode, the maximum bit clock is $f_{BRCLK}/8$. The BITCLK frequency is given by:

$$f_{BitClock} = f_{BRCLK}/UCBRx$$

The minimum high and low periods of the generated SCL are:

$$t_{LOW,MIN} = t_{HIGH,MIN} = (UCBRx/2)/f_{BRCLK} \text{ when UCBRx is even}$$

$$t_{LOW,MIN} = t_{HIGH,MIN} = ((UCBRx - 1)/2)/f_{BRCLK} \text{ when UCBRx is odd}$$

The eUSCI_B clock source frequency and the prescaler setting UCBRx must to be chosen such that the minimum low and high period times of the I²C specification are met.

During the arbitration procedure the clocks from the different masters must be synchronized. A device that first generates a low period on SCL overrules the other devices, forcing them to start their own low periods. SCL is then held low by the device with the longest low period. The other devices must wait for SCL to be released before starting their high periods. Figure 32-16 shows the clock synchronization. This allows a slow slave to slow down a fast master.

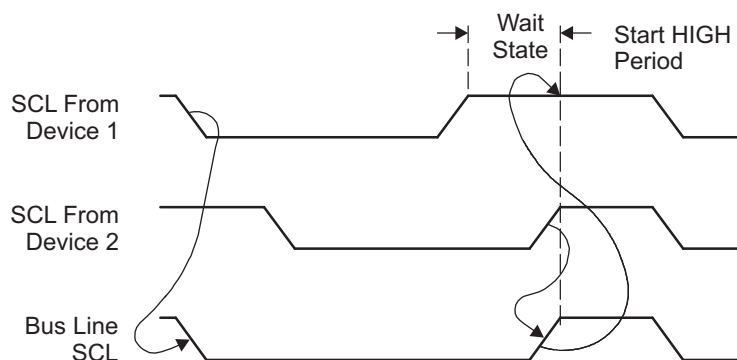


Figure 32-16. Synchronization of Two I²C Clock Generators During Arbitration

32.3.7.1 Clock Stretching

The eUSCI_B module supports clock stretching and also makes use of this feature as described in the Operation Mode sections.

The UCSCLLOW bit can be used to observe if another device pulls SCL low while the eUSCI_B module already released SCL due to the following conditions:

- eUSCI_B is acting as master and a connected slave drives SCL low.

- eUSCI_B is acting as master and another master drives SCL low during arbitration.

The UCSCLLOW bit is also active if the eUSCI_B holds SCL low because it is waiting as transmitter for data being written into UCBxTXBUF or as receiver for the data being read from UCBxRXBUF. The UCSCLLOW bit might be set for a short time with each rising SCL edge because the logic observes the external SCL and compares it to the internally generated SCL.

32.3.7.2 Avoiding Clock Stretching

Even though clock stretching is part of the I2C specification, there are applications in which clock stretching should be avoided.

The clock is stretched by the eUSCI_B under the following conditions:

- The internal shift register is expecting data, but the TXIFG is still pending
- The internal shift register is full, but the RXIFG is still pending
- The arbitration lost interrupt is pending
- UCSWACK is selected and UCBxI2COA0 did cause a match

To avoid clock stretching, all of these situations for clock stretch either need to be avoided or the corresponding interrupt flags need to be processed before the actual clock stretch can occur.

Using the DMA (on devices that contain a DMA) is the most secure way to avoid clock stretching. If no DMA is available, the software must ensure that the corresponding interrupts are serviced in time before the clock is stretched.

In slave transmitter mode, the TXIFG is set only after the reception of the direction bit; therefore, there is only a short amount of time for the software to write the TXBUF before a clock stretch occurs. This situation can be remedied by using the early Transmit Interrupt (see [Section 32.3.11.2](#)).

32.3.7.3 Clock Low Time-out

The UCCLTOIFG interrupt allows the software to react if the clock is low longer than a defined time. It is possible to detect the situation, when a clock is stretched by a master or slave for a too long time. The user can then, for example, reset the eUSCI_B module by using the UCSWRST bit.

The clock low time-out feature is enabled using the UCCLTO bits. It is possible to select one of three predefined times for the clock low time-out. If the clock has been low longer than the time defined with the UCCLTO bits and the eUSCI_B was actively receiving or transmitting, the UCCLTOIFG is set and an interrupt request is generated if UCCLTOIE and GIE are set as well. The UCCLTOIFG is set only once, even if the clock is stretched a multiple of the time defined in UCCLTO.

32.3.8 Byte Counter

The eUSCI_B module supports hardware counting of the bytes received or transmitted. The counter is automatically active and counts up for each byte seen on the bus in both master and slave mode.

The byte counter is incremented at the second bit position of each byte independently of the following ACK or NACK. A START or RESTART condition resets the counter value to zero. Address bytes do not increment the counter. The byte counter is also incremented at the second bit position, if an arbitration lost occurs during the first bit of data.

32.3.8.1 Byte Counter Interrupt

If UCASTPx = 01 or 10 the UCBCNTIFG is set when the byte counter threshold value UCBxTBCNT is reached in both master- and slave-mode. Writing zero to UCBxTBCNT does not generate an interrupt.

Because the UCBCNTIFG has a lower interrupt priority than the UCBTXIFG and UCBRXIFG, TI recommends using it only for protocol control together with the DMA handling the received and transmitted bytes. Otherwise, the application must have enough processor bandwidth to ensure that the UCBCNT interrupt routine is executed in time to generate for example a RESTART.

32.3.8.2 Automatic STOP Generation

When the eUSCI_B module is configured as a master, the byte counter can be used for automatic STOP generation by setting the UCASTPx = 10. Before starting the transmission using UCTXSTT, the byte counter threshold UCBxTBCNT must be set to the number of bytes that are to be transmitted or received. After the number of bytes that are configured in UCBxTBCNT have been transmitted, the eUSCI_B automatically generates a STOP condition.

UCBxTBCNT cannot be used if the user wants to transmit the slave address only without any data. In this case, TI recommends setting UCTXSTT and UCTXSTP at the same time.

32.3.9 Multiple Slave Addresses

The eUSCI_B module supports two different ways of implementing multiple slave addresses at the same time:

- Hardware support for up to 4 different slave addresses, each with its own interrupt flag and DMA trigger
- Software support for up to 2¹⁰ different slave addresses all sharing one interrupt

32.3.9.1 Multiple Slave Address Registers

The registers UCBxI2COA0, UCBxI2COA1, UCBxI2COA2, and UCBxI2COA3 contain four slave addresses. Up to four address registers are compared against a received 7- or 10-bit address. Each slave address must be activated by setting the UCAOEN bit in the corresponding UCBxI2COAx register. Register UCBxI2COA3 has the highest priority if the address received on the bus matches more than one of the slave address registers. The priority decreases with the index number of the address register, so that UCBxI2COA0 in combination with the address mask has the lowest priority.

When one of the slave registers matches the 7- or 10-bit address seen on the bus, the address is acknowledged. In the following the corresponding receive- or transmit-interrupt flag (UCTXIFGx or UCRXIFGx) to the received address is updated. The state change interrupt flags are independent of the address comparison result. They are updated according to the bus condition.

32.3.9.2 Address Mask Register

The address mask register can be used when the eUSCI_B is configured in slave or in multiple-master mode. To activate this feature, at least one bit of the address mask in register UCBxADDMASK must be cleared.

If the received address matches the own address in UCBxI2COA0 on all bit positions that are not masked by UCBxADDMASK, the eUSCI_B module considers the received address as its own address. If UCSWACK = 0, the module sends an acknowledge automatically. If UCSWACK = 1, the user software must evaluate the received address in register UCBxADDRX after the UCSTTIFG is set. To acknowledge the received address, the software must set UCTXACK to 1.

The eUSCI_B module also automatically acknowledges a slave address that is seen on the bus if the address matches any of the enabled slave addresses defined in UCBxI2COA1 to UCBxI2COA3.

NOTE: UCSWACK and slave-transmitter

If the user selects manual acknowledge of slave addresses, TXIFG is set if the slave is addressed as a transmitter. If the software decides not to acknowledge the address, TXIFG0 must be reset.

32.3.10 Using the eUSCI_B Module in I²C Mode With Low-Power Modes

The eUSCI_B module provides automatic clock activation for use with low-power modes. When the eUSCI_B clock source is inactive because the device is in a low-power mode, the eUSCI_B module automatically activates it when needed, regardless of the control-bit settings for the clock source. The clock remains active until the eUSCI_B module returns to its idle condition. After the eUSCI_B module returns to the idle condition, control of the clock source reverts to the settings of its control bits.

In I²C slave mode, no internal clock source is required because the clock is provided by the external master. It is possible to operate the eUSCI_B in I²C slave mode while the device is in LPM4 and all internal clock sources are disabled. The receive or transmit interrupts can wake up the CPU from any low-power mode.

32.3.11 eUSCI_B Interrupts in I²C Mode

The eUSCI_B has only one interrupt vector that is shared for transmission, reception, and the state change.

Each interrupt flag has its own interrupt enable bit. When an interrupt is enabled and the GIE bit is set, the interrupt flag generates an interrupt request. DMA transfers are controlled by the UCTXIFGx and UCRXIFGx flags on devices with a DMA controller. It is possible to react on each slave address with an individual DMA channel.

All interrupt flags are not cleared automatically, but they need to be cleared together by user interactions (for example, reading the UCRXBUF clears UCRXIFGx). If the user wants to use an interrupt flag he needs to ensure that the flag has the correct state before the corresponding interrupt is enabled.

32.3.11.1 I²C Transmit Interrupt Operation

The UCTXIFG0 interrupt flag is set whenever the transmitter is able to accept a new byte. When operating as a slave with multiple slave addresses, the UCTXIFGx flags are set corresponding to which address was received before. If, for example, the slave address specified in register UCBxI2COA3 did match the address seen on the bus, the UCTXIFG3 indicates that the UCBxTXBUF is ready to accept a new byte.

When operating in master mode with automatic STOP generation (UCASTPx = 10), the UCTXIFG0 is set as many times as defined in UCBxTBCNT.

An interrupt request is generated if UCTXIE_x and GIE are also set. UCTXIFG_x is automatically reset if a write to UCBxTXBUF occurs or if the UCAIFG is cleared. UCTXIFG_x is set when:

- Master mode: UCTXSTT was set by the user
- Slave mode: own address was received (UCETXINT = 0) or START was received (UCETXINT = 1)

UCTXIE_x is reset after a PUC or when UCSWRST = 1.

32.3.11.2 Early I²C Transmit Interrupt

Setting the UCETXINT causes UCTXIFG0 to be sent out automatically when a START condition is sent and the eUSCI_B is configured as slave. In this case, it is not allowed to enable the other slave addresses UCBxI2COA1-UCBxI2COA3. This allows the software more time to handle the UCTXIFG0 compared to the normal situation, when UCTXIFG0 is sent out after the slave address match was detected. Situations where the UCTXIFG0 was set and afterward no slave address match occurred need to be handled in software. TI recommends using the byte counter to handle this.

32.3.11.3 I²C Receive Interrupt Operation

The UCRXIFG0 interrupt flag is set when a character is received and loaded into UCBxRXBUF. When operating as a slave with multiple slave addresses, the UCRXIFGx flag is set corresponding to which address was received before.

An interrupt request is generated if UCRXIE_x and GIE are also set. UCRXIFG_x and UCRXIE_x are reset after a PUC signal or when UCSWRST = 1. UCRXIFG_x is automatically reset when UCxRXBUF is read.

32.3.11.4 I²C State Change Interrupt Operation

[Table 32-2](#) describes the I²C state change interrupt flags.

Table 32-2. I²C State Change Interrupt Flags

Interrupt Flag	Interrupt Condition
UCALIFG	Arbitration lost interrupt. Arbitration can be lost when two or more transmitters start a transmission simultaneously, or when the eUSCI_B operates as master but is addressed as a slave by another master in the system. The UCALIFG flag is set when arbitration is lost. When UCALIFG is set, the UCMST bit is cleared and the I ² C controller becomes a slave.
UCNACKIFG	Not acknowledge interrupt. This flag is set when an acknowledge is expected but is not received. UCNACKIFG is used in master mode only.
UCCLTOIFG	Clock low time-out. This interrupt flag is set, if the clock is held low longer than defined by the UCCLTO bits.
UCBIT9IFG	This interrupt flag is generated each time the eUSCI_B is transferring the ninth clock cycle of a byte of data. This gives the user the ability to follow the I ² C communication in software if wanted. UCBIT9IFG is not set for address information.
UCBCNTIFG	Byte counter interrupt. This flag is set when the byte counter value reaches the value defined in UCBxTBCNT and UCASTPx = 01 or 10. This bit allows to organize following communications, especially if a RESTART will be issued.
UCSTTIFG	START condition detected interrupt. This flag is set when the I ² C module detects a START condition together with its own address ⁽¹⁾ . UCSTTIFG is used in slave mode only.
UCSTPIFG	STOP condition detected interrupt. This flag is set when the I ² C module detects a STOP condition on the bus. UCSTPIFG is used in slave and master mode.

⁽¹⁾ The address evaluation includes the address mask register if it is used.

32.3.11.5 UCBxIV, Interrupt Vector Generator

The eUSCI_B interrupt flags are prioritized and combined to source a single interrupt vector. The interrupt vector register UCBxIV is used to determine which flag requested an interrupt. The highest-priority enabled interrupt generates a number in the UCBxIV register that can be evaluated or added to the PC to automatically enter the appropriate software routine. Disabled interrupts do not affect the UCBxIV value.

Read access of the UCBxIV register automatically resets the highest-pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt.

Write access of the UCBxIV register clears all pending Interrupt conditions and flags.

[Example 32-3](#) shows the recommended use of UCBxIV. The UCBxIV value is added to the PC to automatically jump to the appropriate routine. The example is given for eUSCI0_B.

Example 32-3. UCBxIV Software Example

```
#pragma vector = USCI_B0_VECTOR __interrupt void USCI_B0_ISR(void) {
    switch(__even_in_range(UCB0IV,0x1e))    {
        case 0x00:    // Vector 0: No interrupts
            break;
        case 0x02:    ... // Vector 2: ALIFG
            break;
        case 0x04:    ... // Vector 4: NACKIFG
            break;
        case 0x06:    ... // Vector 6: STTIFG
            break;
        case 0x08:    ... // Vector 8: STPIFG
            break;
        case 0x0a:    ... // Vector 10: RXIFG3
            break;
        case 0x0c:    ... // Vector 12: TXIFG3
            break;
        case 0x0e:    ... // Vector 14: RXIFG2
            break;
        case 0x10:    ... // Vector 16: TXIFG2
            break;
        case 0x12:    ... // Vector 18: RXIFG1
            break;
        case 0x14:    ... // Vector 20: TXIFG1
            break;
        case 0x16:    ... // Vector 22: RXIFG0
            break;
        case 0x18:    ... // Vector 24: TXIFG0
            break;
        case 0x1a:    ... // Vector 26: BCNTIFG
            break;
        case 0x1c:    ... // Vector 28: clock low time-out
            break;
        case 0x1e:    ... // Vector 30: 9th bit
            break;
        default:    break;
    }
}
```

32.4 eUSCI_B I2C Registers

The eUSCI_B registers applicable in I²C mode and their address offsets are listed in [Table 32-3](#). The base address can be found in the device-specific data sheet.

Table 32-3. eUSCI_B Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	UCBxCTLW0	eUSCI_Bx Control Word 0	Read/write	Word	01C1h	Section 32.4.1
00h	UCBxCTL1	eUSCI_Bx Control 1	Read/write	Byte	C1h	
01h	UCBxCTL0	eUSCI_Bx Control 0	Read/write	Byte	01h	
02h	UCBxCTLW1	eUSCI_Bx Control Word 1	Read/write	Word	0000h	Section 32.4.2
06h	UCBxBRW	eUSCI_Bx Bit Rate Control Word	Read/write	Word	0000h	Section 32.4.3
06h	UCBxBR0	eUSCI_Bx Bit Rate Control 0	Read/write	Byte	00h	
07h	UCBxBR1	eUSCI_Bx Bit Rate Control 1	Read/write	Byte	00h	
08h	UCBxSTATW	eUSCI_Bx Status Word	Read	Word	0000h	Section 32.4.4
08h	UCBxSTAT	eUSCI_Bx Status	Read	Byte	00h	
09h	UCBxBCNT	eUSCI_Bx Byte Counter Register	Read	Byte	00h	
0Ah	UCBxTBCNT	eUSCI_Bx Byte Counter Threshold Register	Read/Write	Word	00h	Section 32.4.5
0Ch	UCBxRXBUF	eUSCI_Bx Receive Buffer	Read/write	Word	00h	Section 32.4.6
0Eh	UCBxTXBUF	eUSCI_Bx Transmit Buffer	Read/write	Word	00h	Section 32.4.7
14h	UCBxI2COA0	eUSCI_Bx I2C Own Address 0	Read/write	Word	0000h	Section 32.4.8
16h	UCBxI2COA1	eUSCI_Bx I2C Own Address 1	Read/write	Word	0000h	Section 32.4.9
18h	UCBxI2COA2	eUSCI_Bx I2C Own Address 2	Read/write	Word	0000h	Section 32.4.10
1Ah	UCBxI2COA3	eUSCI_Bx I2C Own Address 3	Read/write	Word	0000h	Section 32.4.11
1Ch	UCBxADDRX	eUSCI_Bx Received Address Register	Read	Word		Section 32.4.12
1Eh	UCBxADDMASK	eUSCI_Bx Address Mask Register	Read/write	Word	03FFh	Section 32.4.13
20h	UCBxI2CSA	eUSCI_Bx I2C Slave Address	Read/write	Word	0000h	Section 32.4.14
2Ah	UCBxIE	eUSCI_Bx Interrupt Enable	Read/write	Word	0000h	Section 32.4.15
2Ch	UCBxIFG	eUSCI_Bx Interrupt Flag	Read/write	Word	0002h	Section 32.4.16
2Eh	UCBxIV	eUSCI_Bx Interrupt Vector	Read	Word	0000h	Section 32.4.17

32.4.1 UCBxCTLW0 Register

eUSCI_Bx Control Word Register 0

Figure 32-17. UCBxCTLW0 Register

15	14	13	12	11	10	9	8
UCA10	UCSLA10	UCMM	Reserved	UCMST	UCMODEx		UCSYNC
rw-0	rw-0	rw-0	r0	rw-0	rw-0	rw-0	r1
7	6	5	4	3	2	1	0
UCSSELx		UCTXACK	UCTR	UCTXNACK	UCTXSTP	UCTXSTT	UCSWRST
rw-1	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

Can be modified only when UCSWRST = 1.

Table 32-4. UCBxCTLW0 Register Description

Bit	Field	Type	Reset	Description
15	UCA10	RW	0h	Own addressing mode select. Modify only when UCSWRST = 1. 0b = Own address is a 7-bit address. 1b = Own address is a 10-bit address.
14	UCSLA10	RW	0h	Slave addressing mode select 0b = Address slave with 7-bit address 1b = Address slave with 10-bit address
13	UCMM	RW	0h	Multi-master environment select. Modify only when UCSWRST = 1. 0b = Single master environment. There is no other master in the system. The address compare unit is disabled. 1b = Multi-master environment
12	Reserved	R	0h	Reserved
11	UCMST	RW	0h	Master mode select. When a master loses arbitration in a multi-master environment (UCMM = 1), the UCMST bit is automatically cleared and the module acts as slave. 0b = Slave mode 1b = Master mode
10-9	UCMODEx	RW	0h	eUSCI_B mode. The UCMODEx bits select the synchronous mode when UCSYNC = 1. Modify only when UCSWRST = 1. 00b = 3-pin SPI 01b = 4-pin SPI (master or slave enabled if STE = 1) 10b = 4-pin SPI (master or slave enabled if STE = 0) 11b = I2C mode
8	UCSYNC	RW	1h	Synchronous mode enable. For eUSCI_B always read and write as 1.
7-6	UCSSELx	RW	3h	eUSCI_B clock source select. These bits select the BRCLK source clock. These bits are ignored in slave mode. Modify only when UCSWRST = 1. 00b = UCLKI 01b = ACLK 10b = SMCLK 11b = SMCLK
5	UCTXACK	RW	0h	Transmit ACK condition in slave mode with enabled address mask register. After the UCSTTIFG has been set, the user needs to set or reset the UCTXACK flag to continue with the I2C protocol. The clock is stretched until the UCBxCTL1 register has been written. This bit is cleared automatically after the ACK has been send. 0b = Do not acknowledge the slave address 1b = Acknowledge the slave address

Table 32-4. UCBxCTLW0 Register Description (continued)

Bit	Field	Type	Reset	Description
4	UCTR	RW	0h	Transmitter/receiver 0b = Receiver 1b = Transmitter
3	UCTXNACK	RW	0h	Transmit a NACK. UCTXNACK is automatically cleared after a NACK is transmitted. Only for slave receiver mode. 0b = Acknowledge normally 1b = Generate NACK
2	UCTXSTP	RW	0h	Transmit STOP condition in master mode. Ignored in slave mode. In master receiver mode, the STOP condition is preceded by a NACK. UCTXSTP is automatically cleared after STOP is generated. This bit is a don't care, if automatic UCASTPx is different from 01 or 10. 0b = No STOP generated 1b = Generate STOP
1	UCTXSTT	RW	0h	Transmit START condition in master mode. Ignored in slave mode. In master receiver mode, a repeated START condition is preceded by a NACK. UCTXSTT is automatically cleared after START condition and address information is transmitted. Ignored in slave mode. 0b = Do not generate START condition 1b = Generate START condition
0	UCSWRST	RW	1h	Software reset enable. 0b = Disabled. eUSCI_B released for operation. 1b = Enabled. eUSCI_B logic held in reset state.

32.4.2 UCBxCTLW1 Register

eUSCI_Bx Control Word Register 1

Figure 32-18. UCBxCTLW1 Register

15	14	13	12	11	10	9	8
Reserved							UCETXINT
r0	r0	r0	r0	r0	r0	r0	rw-0
7	6	5	4	3	2	1	0
UCCLTO		UCSTPNACK	UCSWACK	UCASTPx		UCGLITx	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when UCSWRST = 1.

Table 32-5. UCBxCTLW1 Register Description

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0h	Reserved
8	UCETXINT	RW	0h	Early UCTXIFG0. Only in slave mode. When this bit is set, the slave addresses defined in UCxI2COA1 to UCxI2COA3 must be disabled. Modify only when UCSWRST = 1. 0b = UCTXIFGx is set after an address match with UCxI2COAx and the direction bit indicating slave transmit 1b = UCTXIFG0 is set for each START condition
7-6	UCCLTO	RW	0h	Clock low time-out select. Modify only when UCSWRST = 1. 00b = Disable clock low time-out counter 01b = 135000 MODCLK cycles (approximately 28 ms) 10b = 150000 MODCLK cycles (approximately 31 ms) 11b = 165000 MODCLK cycles (approximately 34 ms)
5	UCSTPNACK	RW	0h	The UCSTPNACK bit allows to make the eUSCI_B master acknowledge the last byte in master receiver mode as well. This does not conform to the I2C specification and should only be used for slaves that automatically release the SDA after a fixed packet length. Modify only when UCSWRST = 1. 0b = Send a not acknowledge before the STOP condition as a master receiver (conform to I2C standard) 1b = All bytes are acknowledged by the eUSCI_B when configured as master receiver
4	UCSWACK	RW	0h	This bit selects whether sending an ACK of the address is triggered by the eUSCI_B module or is controlled by software. 0b = The address acknowledge of the slave is controlled by the eUSCI_B module 1b = The user needs to trigger the sending of the address ACK by issuing UCTXACK
3-2	UCASTPx	RW	0h	Automatic STOP condition generation. In slave mode, only settings 00b and 01b are available. Modify only when UCSWRST = 1. 00b = No automatic STOP generation. The STOP condition is generated after the user sets the UCTXSTP bit. The value in UCBxTBCNT is a don't care. 01b = UCBCNTIFG is set with the byte counter reaches the threshold defined in UCBxTBCNT 10b = A STOP condition is generated automatically after the byte counter value reached UCBxTBCNT. UCBCNTIFG is set with the byte counter reaching the threshold. 11b = Reserved

Table 32-5. UCBxCTLW1 Register Description (continued)

Bit	Field	Type	Reset	Description
1-0	UCGLITx	RW	0h	Deglitch time 00b = 50 ns 01b = 25 ns 10b = 12.5 ns 11b = 6.25 ns

32.4.3 UCBxBRW Register

eUSCI_Bx Bit Rate Control Word Register

Figure 32-19. UCBxBRW Register

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when UCSWRST = 1.

Table 32-6. UCBxBRW Register Description

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Bit clock prescaler. Modify only when UCSWRST = 1.

32.4.4 UCBxSTATW

eUSCI_Bx Status Word Register

Figure 32-20. UCBxSTATW Register

15	14	13	12	11	10	9	8
UCBCNTx							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved	UCSCLLOW	UCGC	UCBBUSY	Reserved			
r0	r-0	r-0	r-0	r-0	r0	r0	r0

Table 32-7. UCBxSTATW Register Description

Bit	Field	Type	Reset	Description
15-8	UCBCNTx	R	0h	Hardware byte counter value. Reading this register returns the number of bytes received or transmitted on the I2C-Bus since the last START or RESTART. There is no synchronization of this register done. When reading UCBxBCNT during the first bit position, a faulty read back can occur.
7	Reserved	R	0h	Reserved
6	UCSCLLOW	R	0h	SCL low 0b = SCL is not held low 1b = SCL is held low
5	UCGC	R	0h	General call address received. UCGC is automatically cleared when a START condition is received. 0b = No general call address received 1b = General call address received
4	UCBBUSY	R	0h	Bus busy 0b = Bus inactive 1b = Bus busy
3-0	Reserved	R	0h	Reserved

32.4.5 UCBxTBCNT Register

eUSCI_Bx Byte Counter Threshold Register

Figure 32-21. UCBxTBCNT Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCTBCNTx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when UCSWRST = 1.

Table 32-8. UCBxTBCNT Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTBCNTx	RW	0h	The byte counter threshold value is used to set the number of I2C data bytes after which the automatic STOP or the UCSTPIFG should occur. This value is evaluated only if UCASTPx is different from 00. Modify only when UCSWRST = 1.

32.4.6 UCBxRXBUF Register

eUSCI_Bx Receive Buffer Register

Figure 32-22. UCBxRXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCRXBUFx							
r	r	r	r	r	r	r	r

Table 32-9. UCBxRXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCRXBUFx	R	0h	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCBxRXBUF resets the UCRXIFGx flags.

32.4.7 UCBxTXBUF

eUSCI_Bx Transmit Buffer Register

Figure 32-23. UCBxTXBUF Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Table 32-10. UCBxTXBUF Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTXBUFx	RW	0h	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted. Writing to the transmit data buffer clears the UCTXIFGx flags.

32.4.8 UCBxI2COA0 Register

eUSCI_Bx I2C Own Address 0 Register

Figure 32-24. UCBxI2COA0 Register

15	14	13	12	11	10	9	8
UCGCEN	Reserved				UCOAEN	I2COA0	
rw-0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COA0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when UCSWRST = 1.

Table 32-11. UCBxI2COA0 Register Description

Bit	Field	Type	Reset	Description
15	UCGCEN	RW	0h	General call response enable. This bit is only available in UCBxI2COA0. Modify only when UCSWRST = 1. 0b = Do not respond to a general call 1b = Respond to a general call
14-11	Reserved	R	0h	Reserved
10	UCOAEN	RW	0h	Own Address enable register. With this register it can be selected if the I2C slave-address related to this register UCBxI2COA0 is evaluated or not. Modify only when UCSWRST = 1. 0b = The slave address defined in I2COA0 is disabled 1b = The slave address defined in I2COA0 is enabled
9-0	I2COAx	RW	0h	I2C own address. The I2COA0 bits contain the local address of the eUSCIx_B I2C controller. The address is right justified. In 7-bit addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB. Modify only when UCSWRST = 1.

32.4.9 UCBxI2COA1 Register

eUSCI_Bx I2C Own Address 1 Register

Figure 32-25. UCBxI2COA1 Register

15	14	13	12	11	10	9	8
Reserved					UCOAEN	I2COA1	
rw-0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COA1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when UCSWRST = 1.

Table 32-12. UCBxI2COA1 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved
10	UCOAEN	RW	0h	Own Address enable register. With this register it can be selected if the I2C slave-address related to this register UCBxI2COA1 is evaluated or not. Modify only when UCSWRST = 1. 0b = The slave address defined in I2COA1 is disabled 1b = The slave address defined in I2COA1 is enabled
9-0	I2COA1	RW	0h	I2C own address. The I2COAx bits contain the local address of the eUSCIx_B I2C controller. The address is right justified. In 7-bit addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB. Modify only when UCSWRST = 1.

32.4.10 UCBxI2COA2 Register

eUSCI_Bx I2C Own Address 2 Register

Figure 32-26. UCBxI2COA2 Register

15	14	13	12	11	10	9	8
Reserved					UCOAEN	I2COA2	
rw-0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COA2							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when UCSWRST = 1.

Table 32-13. UCBxI2COA2 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved
10	UCOAEN	RW	0h	Own Address enable register. With this register it can be selected if the I2C slave-address related to this register UCBxI2COA2 is evaluated or not. Modify only when UCSWRST = 1. 0b = The slave address defined in I2COA2 is disabled 1b = The slave address defined in I2COA2 is enabled
9-0	I2COA2	RW	0h	I2C own address. The I2COAx bits contain the local address of the eUSCIx_B I2C controller. The address is right justified. In 7-bit addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB. Modify only when UCSWRST = 1.

32.4.11 UCBxI2COA3 Register

eUSCI_Bx I2C Own Address 3 Register

Figure 32-27. UCBxI2COA3 Register

15	14	13	12	11	10	9	8
Reserved					UCOAEN	I2COA3	
rw-0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COA3							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Can be modified only when UCSWRST = 1.

Table 32-14. UCBxI2COA3 Register Description

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved
10	UCOAEN	RW	0h	Own Address enable register. With this register it can be selected if the I2C slave-address related to this register UCBxI2COA3 is evaluated or not. Modify only when UCSWRST = 1. 0b = The slave address defined in I2COA3 is disabled 1b = The slave address defined in I2COA3 is enabled
9-0	I2COA3	RW	0h	I2C own address. The I2COA3 bits contain the local address of the eUSCIx_B I2C controller. The address is right justified. In 7-bit addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB. Modify only when UCSWRST = 1.

32.4.12 UCBxADDRX Register

eUSCI_Bx I2C Received Address Register

Figure 32-28. UCBxADDRX Register

15	14	13	12	11	10	9	8
Reserved					ADDRXx		
r-0	r0	r0	r0	r0	r0	r-0	r-0
7	6	5	4	3	2	1	0
ADDRXx							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 32-15. UCBxADDRX Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ADDRXx	R	0h	Received Address Register. This register contains the last received slave address on the bus. Using this register and the address mask register it is possible to react on more than one slave address using one eUSCI_B module.

32.4.13 UCBxADDMASK Register

eUSCI_Bx I2C Address Mask Register

Figure 32-29. UCBxADDMASK Register

15	14	13	12	11	10	9	8
Reserved						ADDMASKx	
r-0	r0	r0	r0	r0	r0	rw-1	rw-1
7	6	5	4	3	2	1	0
ADDMASKx							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

Can be modified only when UCSWRST = 1.

Table 32-16. UCBxADDMASK Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ADDMASKx	RW	3FFh	Address Mask Register. By clearing the corresponding bit of the own address, this bit is a don't care when comparing the address on the bus to the own address. Using this method, it is possible to react on more than one slave address. When all bits of ADDMASKx are set, the address mask feature is deactivated. Modify only when UCSWRST = 1.

32.4.14 UCBxI2CSA Register

eUSCI_Bx I2C Slave Address Register

Figure 32-30. UCBxI2CSA Register

15	14	13	12	11	10	9	8
Reserved						I2CSAx	
r-0	r0	r0	r0	r0	r0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2CSAx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 32-17. UCBxI2CSA Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	I2CSAx	RW	0h	I2C slave address. The I2CSAx bits contain the slave address of the external device to be addressed by the eUSCIx_B module. It is only used in master mode. The address is right justified. In 7-bit slave addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit slave addressing mode, bit 9 is the MSB.

32.4.15 UCBxIE Register

eUSCI_Bx I2C Interrupt Enable Register

Figure 32-31. UCBxIE Register

15	14	13	12	11	10	9	8
Reserved	UCBIT9IE	UCTXIE3	UCRXIE3	UCTXIE2	UCRXIE2	UCTXIE1	UCRXIE1
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCCLTOIE	UCBCNTIE	UCNACKIE	UCALIE	UCSTPIE	UCSTTIE	UCTXIE0	UCRXIE0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 32-18. UCBxIE Register Description

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved
14	UCBIT9IE	RW	0h	Bit position 9 interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
13	UCTXIE3	RW	0h	Transmit interrupt enable 3 0b = Interrupt disabled 1b = Interrupt enabled
12	UCRXIE3	RW	0h	Receive interrupt enable 3 0b = Interrupt disabled 1b = Interrupt enabled
11	UCTXIE2	RW	0h	Transmit interrupt enable 2 0b = Interrupt disabled 1b = Interrupt enabled
10	UCRXIE2	RW	0h	Receive interrupt enable 2 0b = Interrupt disabled 1b = Interrupt enabled
9	UCTXIE1	RW	0h	Transmit interrupt enable 1 0b = Interrupt disabled 1b = Interrupt enabled
8	UCRXIE1	RW	0h	Receive interrupt enable 1 0b = Interrupt disabled 1b = Interrupt enabled
7	UCCLTOIE	RW	0h	Clock low time-out interrupt enable. 0b = Interrupt disabled 1b = Interrupt enabled
6	UCBCNTIE	RW	0h	Byte counter interrupt enable. 0b = Interrupt disabled 1b = Interrupt enabled
5	UCNACKIE	RW	0h	Not-acknowledge interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
4	UCALIE	RW	0h	Arbitration lost interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
3	UCSTPIE	RW	0h	STOP condition interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

Table 32-18. UCBxIE Register Description (continued)

Bit	Field	Type	Reset	Description
2	UCSTTIE	RW	0h	START condition interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
1	UCTXIE0	RW	0h	Transmit interrupt enable 0 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE0	RW	0h	Receive interrupt enable 0 0b = Interrupt disabled 1b = Interrupt enabled

32.4.16 UCBxIFG Register

eUSCI_Bx I2C Interrupt Flag Register

Figure 32-32. UCBxIFG Register

15	14	13	12	11	10	9	8
Reserved	UCBIT9IFG	UCTXIFG3	UCRXIFG3	UCTXIFG2	UCRXIFG2	UCTXIFG1	UCRXIFG1
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCCLTOIFG	UCBCNTIFG	UCNACKIFG	UCALIFG	UCSTPIFG	UCSTTIFG	UCTXIFG0	UCRXIFG0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1	rw-0

Table 32-19. UCBxIFG Register Description

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved
14	UCBIT9IFG	RW	0h	Bit position 9 interrupt flag 0b = No interrupt pending 1b = Interrupt pending
13	UCTXIFG3	RW	0h	eUSCI_B transmit interrupt flag 3. UCTXIFG3 is set when UCBxTXBUF is empty in slave mode, if the slave address defined in UCBxI2COA3 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
12	UCRXIFG3	RW	0h	Receive interrupt flag 3. UCRXIFG3 is set when UCBxRXBUF has received a complete byte in slave mode and if the slave address defined in UCBxI2COA3 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
11	UCTXIFG2	RW	0h	eUSCI_B transmit interrupt flag 2. UCTXIFG2 is set when UCBxTXBUF is empty in slave mode, if the slave address defined in UCBxI2COA2 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
10	UCRXIFG2	RW	0h	Receive interrupt flag 2. UCRXIFG2 is set when UCBxRXBUF has received a complete byte in slave mode and if the slave address defined in UCBxI2COA2 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
9	UCTXIFG1	RW	0h	eUSCI_B transmit interrupt flag 1. UCTXIFG1 is set when UCBxTXBUF is empty in slave mode, if the slave address defined in UCBxI2COA1 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
8	UCRXIFG1	RW	0h	Receive interrupt flag 1. UCRXIFG1 is set when UCBxRXBUF has received a complete byte in slave mode and if the slave address defined in UCBxI2COA1 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
7	UCCLTOIFG	RW	0h	Clock low time-out interrupt flag 0b = No interrupt pending 1b = Interrupt pending
6	UCBCNTIFG	RW	0h	Byte counter interrupt flag. When using this interrupt the user needs to ensure enough processing bandwidth (see the Byte Counter Interrupt section). 0b = No interrupt pending 1b = Interrupt pending

Table 32-19. UCBxIFG Register Description (continued)

Bit	Field	Type	Reset	Description
5	UCNACKIFG	RW	0h	Not-acknowledge received interrupt flag. This flag only is updated when operating in master mode. 0b = No interrupt pending 1b = Interrupt pending
4	UCALIFG	RW	0h	Arbitration lost interrupt flag 0b = No interrupt pending 1b = Interrupt pending
3	UCSTPIFG	RW	0h	STOP condition interrupt flag 0b = No interrupt pending 1b = Interrupt pending
2	UCSTTIFG	RW	0h	START condition interrupt flag 0b = No interrupt pending 1b = Interrupt pending
1	UCTXIFG0	RW	0h	eUSCI_B transmit interrupt flag 0. UCTXIFG0 is set when UCBxTXBUF is empty in master mode or in slave mode, if the slave address defined in UCBxI2COA0 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
0	UCRXIFG0	RW	0h	eUSCI_B receive interrupt flag 0. UCRXIFG0 is set when UCBxRXBUF has received a complete character in master mode or in slave mode, if the slave address defined in UCBxI2COA0 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending

32.4.17 UCBxIV Register

eUSCI_Bx Interrupt Vector Register

Figure 32-33. UCBxIV Register

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCIVx							
r0	r0	r0	r0	r-0	r-0	r-0	r0

Table 32-20. UCBxIV Register Description

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI_B interrupt vector value. It generates an value that can be used as address offset for fast interrupt service routine handling. Writing to this register clears all pending interrupt flags. 00h = No interrupt pending 02h = Interrupt Source: Arbitration lost; Interrupt Flag: UCALIFG; Interrupt Priority: Highest 04h = Interrupt Source: Not acknowledgment; Interrupt Flag: UCNACKIFG 06h = Interrupt Source: Start condition received; Interrupt Flag: UCSTTIFG 08h = Interrupt Source: Stop condition received; Interrupt Flag: UCSTPIFG 0Ah = Interrupt Source: Slave 3 Data received; Interrupt Flag: UCRXIFG3 0Ch = Interrupt Source: Slave 3 Transmit buffer empty; Interrupt Flag: UCTXIFG3 0Eh = Interrupt Source: Slave 2 Data received; Interrupt Flag: UCRXIFG2 10h = Interrupt Source: Slave 2 Transmit buffer empty; Interrupt Flag: UCTXIFG2 12h = Interrupt Source: Slave 1 Data received; Interrupt Flag: UCRXIFG1 14h = Interrupt Source: Slave 1 Transmit buffer empty; Interrupt Flag: UCTXIFG1 16h = Interrupt Source: Data received; Interrupt Flag: UCRXIFG0 18h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG0 1Ah = Interrupt Source: Byte counter zero; Interrupt Flag: UCBCNTIFG 1Ch = Interrupt Source: Clock low time-out; Interrupt Flag: UCCLTOIFG 1Eh = Interrupt Source: 9th bit position; Interrupt Flag: UCBIT9IFG; Priority: Lowest

REF_A

The REF_A module is a general-purpose reference system that generates the voltage references required for other subsystems such as digital-to-analog converters, analog-to-digital converters, or comparators. This chapter describes the REF_A module.

Topic	Page
33.1 REF_A Introduction	862
33.2 Principle of Operation	863
33.3 REF_A Registers	865

33.2 Principle of Operation

The REF_A module provides all of the necessary voltage references for use by various peripheral modules throughout the system.

The REFGEN subsystem contains a high-performance bandgap. This bandgap has very good accuracy (factory trimmed), low temperature coefficient, and high PSRR even while operating at low power. The bandgap voltage is used to generate three voltages (1.2 V, 2.0 V, and 2.5 V) through a noninverting amplifier stage. One voltage can be selected at a time. One output of the REFGEN subsystem is the variable reference line. The variable reference line provides either 1.2 V, 2.0 V, or 2.5 V to the rest of the system. The second output of the REFGEN subsystem is the buffered bandgap reference line. Additionally, the REFGEN supports the voltage references that are required for a DAC module, if available. Lastly, the REFGEN subsystem also includes the temperature sensor circuitry, which is derived from the bandgap. The temperature sensor is used by an ADC to measure a voltage proportional to temperature.

33.2.1 Low-Power Operation

The REF_A module can support low-power applications such as LCD generation. Many of these applications do not require a very accurate reference, compared to data conversion, yet power is of prime concern. To support these kinds of applications, the bandgap can be used in a sampled mode. In sampled mode, the bandgap circuitry is clocked by the VLO at an appropriate duty cycle. This reduces the average power of the bandgap circuitry significantly, at the cost of accuracy. When not in sampled mode, the bandgap is in static mode. Its power is at its highest, but so is its accuracy.

Modules can request static mode or sampled mode through their own individual request lines. In this way, the particular module determines which mode is appropriate for its proper operation and performance. Any one active module that requests static mode causes all other modules to use static mode, even if another module requests sampled mode. For example, any module using the gray box in the [block diagram](#) requests static mode and causes all other modules to use static mode. In other words, static mode always has higher priority than sampled mode.

33.2.2 Reference System Requests

There are three basic reference system requests that are used by the reference system. Each module can use these requests to obtain the proper response from the reference system. The three basic requests are REFGENREQ, REFBGREQ, and REFMODEREQ. No interaction is required by the user code. The modules automatically select the proper request.

A reference request signal, REFGENREQ, is available as an input into the REFGEN subsystem. This signal represents a logical OR of individual requests coming from the various modules in the system that require a voltage reference to be available on the variable reference line. When a module requires a voltage reference, it asserts its corresponding REFGENREQ signal. When the REFGENREQ is asserted, the REFGEN subsystem is enabled. After the specified settling time, the variable reference line voltage is stable and ready for use. The REFVSEL settings determine which voltage is generated on the variable reference line.

After the specified settling time of the REFGEN subsystem, the REF_A module sets the REFGENRDY signal. This signal can be used by each module to wait, for example, before a conversion is started after a REFGENREQ was set. The generation of the reference voltage can be triggered by a timer or by software to make sure that the reference voltage is ready when a module requires it.

In addition to the REFGENREQ, a second reference request signal, REFBGREQ, is available. The REFBGREQ signal represents a logical OR of requests coming from the various modules that require the bandgap reference line. When the REFBGREQ is asserted, the bandgap and its bias circuitry and local buffer are enabled, if not already enabled by a prior request.

After the specified settling time of the REFBGREQ subsystem, the REF_A module sets the REFBGRDY signal. This signal can be used by each module to delay operation while the bandgap reference voltage is settling. The generation of the buffered bandgap voltage can be triggered by a timer or by software to make sure that the reference voltage is ready when a module requires it.

The REFMODEREQ request signal configures the bandgap and its bias circuitry to operate in either sampled or static mode of operation. The REFMODEREQ signal represents a logical AND of individual requests coming from the various analog modules. A REFMODEREQ occurs only if a REFGENREQ or REFBGQ is also asserted by a module, otherwise it is a don't care. When REFMODEREQ = 1, the bandgap operates in sampled mode. When a module asserts its corresponding REFMODEREQ signal, it is requesting that the bandgap operate in sampled mode. Because REMODEREQ is a logical AND of all individual requests, any modules that request static mode cause the bandgap to operate in static mode. The BGMODE bit can be read for use as an indicator of static or sampled mode of operation.

33.2.2.1 REFBGACT, REFGENACT, REFGENBUSY

Any module that is using the variable reference line causes REFGENACT to be set inside the REFCTL register. This bit is read only and indicates to the user whether the REFGEN is active or off. Similarly, the REFBGACT is active any time one or more modules are actively using the bandgap reference line and, therefore, indicates to the user whether the REFBG is active or off.

The REFGENBUSY signal, when asserted, indicates that a module is using the reference and that no changes should be made to the reference settings. For example, during an active ADC12_B conversion, the reference voltage level should not be changed. REFGENBUSY is asserted when there is an active ADC12_B conversion (ADC12BUSY = 1). REFGENBUSY when asserted, write protects the REFCTL register. This prevents the reference from being disabled or its level changed during any active conversion.

33.2.2.2 ADC12_B

For devices that contain an ADC12_B module, there are two buffers. The larger buffer can be used to drive the reference voltage, which is present on the variable reference line. This buffer has larger power consumption to drive larger DC loads that may be present outside the device. The large buffer is enabled continuously when REFON = 1 and REFOUT = 1. In addition, when REFON = 1 and REFOUT = 1, the second smaller buffer is automatically disabled. In this case, the output of the large buffer is connected to the capacitor array through an internal analog switch. This makes sure that the same reference is used throughout the system. If REFON = 1 and REFOUT = 0, the internal buffer is used for ADC conversion, and the large buffer remains disabled.

33.2.2.3 LCD Modules

On devices that contain an LCD module, this module requires a reference to generate the proper LCD voltages. The bandgap reference line from the REFGEN subsystem is used for this purpose. Enabling the LCD module in a mode that requires a reference voltage causes a REFBGREQ from the LCD module to be asserted. The buffered bandgap is made available on the bandgap reference line for use inside the LCD module.

33.3 REF_A Registers

The REF_A registers are listed in [Table 33-1](#). The base address can be found in the device specific datasheet. The address offset is listed in [Table 33-1](#).

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 33-1. REF_A Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	REFCTL0	REFCTL0	Read/write	Word	0000h	Section 33.3.1
00h	REFCTL0_L		Read/write	Byte	00h	
01h	REFCTL0_H		Read/write	Byte	00h	

33.3.1 REFCTL0 Register (offset = 00h) [reset = 0000h]

REF_A Control Register 0

Figure 33-2. REFCTL0 Register

15	14	13	12	11	10	9	8
Reserved		REFBGRDY	REFGENRDY	BGMODE	REFGENBUSY	REFBGACT	REFGENACT
r0	r0	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)
7	6	5	4	3	2	1	0
REFBGOT	REFGENOT	REFVSEL	REFTCOFF	Reserved	REFOUT	REFON	
rw-0	rw-0	rw-(0)	rw-(0)	rw-(0)	r0	rw-(0)	rw-(0)
Can be modified only when REFGENBUSY = 0.							

Table 33-2. REFCTL0 Register Description

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0h	Reserved. Always reads as 0.
13	REFBGRDY	R	0h	Buffered bandgap voltage is ready to be used. Both the bandgap and the bandgap buffer are active, and the reference voltage is settled for use by the comparator and the LCD. 0b = Buffered bandgap voltage is not ready to be used 1b = Buffered bandgap voltage is ready to be used
12	REFGENRDY	R	0h	Variable reference voltage ready status. Variable reference voltage is ready to be used. Both the bandgap and the reference voltage amplifier are active and the variable reference voltage is settled; for example, for use by the ADC. 0b = Reference voltage output is not ready to be used 1b = Reference voltage output is ready to be used
11	BGMODE	R	0h	Bandgap mode. Read only. 0b = Static mode 1b = Sampled mode
10	REFGENBUSY	R	0h	Reference generator busy. Read only. 0b = Reference generator not busy 1b = Reference generator busy
9	REFBGACT	R	0h	Reference bandgap active. Read only. 0b = Reference bandgap buffer not active 1b = Reference bandgap buffer active
8	REFGENACT	R	0h	Reference generator active. Read only. 0b = Reference generator not active 1b = Reference generator active
7	REFBGOT	RW	0h	Bandgap and bandgap buffer one-time trigger. If written with a 1, the generation of the buffered bandgap voltage is started. When the bandgap buffer voltage request is set, this bit is cleared by hardware. 0b = No trigger 1b = Generation of the bandgap voltage is started by writing 1 or by a hardware trigger
6	REFGENOT	RW	0h	Reference generator one-time trigger. If written with a 1, the generation of the variable reference voltage is started. When the reference voltage request is set, this bit is cleared by hardware. 0b = No trigger 1b = Generation of the reference voltage is started by writing 1 or by a hardware trigger
5-4	REFVSEL	RW	0h	Reference voltage level select. Can be modified only when REFGENBUSY = 0. 00b = 1.2 V available when reference requested or REFON = 1 01b = 2.0 V available when reference requested or REFON = 1 10b = 2.5 V available when reference requested or REFON = 1 11b = 2.5 V available when reference requested or REFON = 1

Table 33-2. REFCTL0 Register Description (continued)

Bit	Field	Type	Reset	Description
3	REFTCOFF	RW	0h	Temperature sensor disable. The temperature sensor is disabled if the ADC on the device is not enabled independent of this control bit. Can be modified only when REFGENBUSY = 0. 0b = Temperature sensor enabled 1b = Temperature sensor disabled to save power
2	Reserved	R	0h	Reserved. Always reads as 0.
1	REFOUT	RW	0h	Reference output buffer. On devices with an ADC10_A, this bit must be written with 0. Can be modified only when REFGENBUSY = 0. 0b = Reference output not available externally 1b = Reference output available externally
0	REFON	RW	0h	Reference enable. Can be modified only when REFGENBUSY = 0. 0b = Disables reference if no other reference requests are pending 1b = Enables reference

ADC12_B

The ADC12_B module is a high-performance 12-bit analog-to-digital converter (ADC). This chapter describes the operation of the ADC12_B module.

Topic	Page
34.1 ADC12_B Introduction.....	869
34.2 ADC12_B Operation.....	871
34.3 ADC12_B Registers	887

34.1 ADC12_B Introduction

The ADC12_B module supports fast 12-bit analog-to-digital conversions. The module implements a 12-bit SAR core, sample select control, and up to 32 independent conversion-and-control buffers. The conversion-and-control buffer allows up to 32 independent analog-to-digital converter (ADC) samples to be converted and stored without any CPU intervention.

ADC12_B features include:

- 200-kSPS maximum conversion rate at maximum resolution of 12 bits
- Monotonic 12-bit converter with no missing codes
- Sample-and-hold with programmable sampling periods controlled by software or timers
- Conversion initiation by software or timers
- Software-selectable on-chip reference voltage generation (1.2 V, 2.0 V, or 2.5 V) with option to make available externally
- Software-selectable internal or external reference
- Up to 32 individually configurable external input channels with single-ended or differential input selection available
- Internal conversion channels for internal temperature sensor and $1/2 \times AV_{CC}$ and four more internal channels available on select devices (see the device-specific data sheet for availability and function)
- Independent channel-selectable reference sources for both positive and negative references
- Selectable conversion clock source
- Single-channel, repeat-single-channel, sequence (autoscan), and repeat-sequence (repeated autoscan) conversion modes
- Interrupt vector register for fast decoding of 38 ADC interrupts
- 32 conversion-result storage registers
- Window comparator for low-power monitoring of input signals of conversion-result registers

[Figure 34-1](#) shows the block diagram of ADC12_B. The reference generation is located in the reference module (REF) (see the device-specific data sheet).

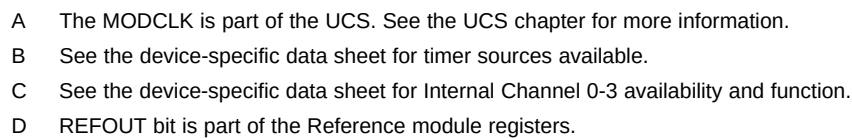


Figure 34-1. ADC12_B Block Diagram

34.2 ADC12_B Operation

The ADC12_B module is configured with user software. The following sections describe the setup and operation of the ADC12_B.

34.2.1 12-Bit ADC Core

The ADC core converts an analog input to its 12-bit digital representation. The core uses two programmable and selectable voltage levels (V_{R+} and V_{R-}) to define the upper and lower limits of the conversion. The digital output (N_{ADC}) is full scale (0FFFh) when the input signal is equal to or higher than V_{R+} , and is zero when the input signal is equal to or lower than V_{R-} . The input channel and the reference voltage levels (V_{R+} and V_{R-}) are defined in the conversion-control memory.

Equation 13 shows the conversion formula for the ADC result N_{ADC} for single-ended mode.

$$N_{ADC} = 4096 \times \frac{(V_{in+} + \frac{1}{2} \text{LSB}) - V_{R-}}{V_{R+} - V_{R-}} \quad \text{Where, } 1 \text{ LSB} = \frac{V_{R+} - V_{R-}}{4096} \quad (13)$$

Equation 14 shows the conversion formula for the ADC result N_{ADC} for differential mode.

$$N_{ADC} = \left(2048 \times \frac{V_{in+} - V_{in-} + \frac{1}{2} \text{LSB}}{V_{R+} - V_{R-}} \right) + 2048 \quad \text{Where, } 1 \text{ LSB} = \frac{V_{R+} - V_{R-}}{2048} \quad (14)$$

Equation 15 describes the input voltage at which the ADC output saturates for single-ended mode.

$$V_{in+} = V_{R+} - V_{R-} - 1.5 \text{ LSB} \quad (15)$$

Equation 16 describes the input voltage at which the ADC output saturates for differential mode.

$$V_{in+} - V_{in-} = V_{R+} - V_{R-} - 1.5 \text{ LSB}$$

where

- $V_{R-} < V_{in+} < V_{R+}$
 - $V_{R-} < V_{in-} < V_{R+}$
- (16)

Four control registers configure the ADC12_B core: ADC12CTL0, ADC12CTL1, ADC12CTL2, and ADC12CTL3. The ADC12ON bit enables or disables the core. The ADC12_B can be turned off when it is not in use to save power. If the ADC12ON bit is set to 0 during a conversion, the conversion is abruptly exited and the module is powered down. With few exceptions, an application can modify the ADC12_B control bits only when ADC12ENC = 0. ADC12ENC must be set to 1 before any conversion can take place.

The conversion results are always stored in binary unsigned format. For differential input, this means that an offset of 2048 is added to the result to make the number positive. The data format bit ADC12DF in ADC12CTL2 allows the user to read the conversion results as binary unsigned or signed binary (2's complement).

34.2.1.1 Conversion Clock Selection

The ADC12CLK operates as the conversion clock and also generates the sampling period when the pulse sampling mode is selected. The ADC12SSELx bit selects the ADC12_B source clock. SMCLK, MCLK, ACLK, and the MODCLK are the possible ADC12CLK sources. The ADC12PDIV bits set the initial divider on the input clock (1, 4, 32, or 64), and then ADC12DIV bits set an additional divider of 1 to 8.

The user must ensure that the clock that is used for ADC12CLK remains active until the end of a conversion. If the clock is removed during a conversion, the operation does not complete and any result is invalid.

34.2.2 ADC12_B Inputs and Multiplexer

Up to 32 external and up to 6 internal analog signals are selected as the channel for conversion by the analog input multiplexer based on the ADC12INCHx bit and for A₂₆- A₃₁ the ADC12CTL3 register. The number of channels that are available as well as internal channel 0-3 is device specific and is shown in the device-specific data sheet. The input multiplexer is a break-before-make type to reduce input-to-input noise injection that can result from channel switching (see Figure 34-2). The input multiplexer is also a T-switch to minimize the coupling between channels. Channels that are not selected are isolated from the ADC, and the intermediate node is connected to analog ground (AVSS), so that the stray capacitance is grounded to eliminate crosstalk.

The ADC12_B supports single-ended input or differential inputs configurable for each conversion memory with the ADC12DIF bit in the ADC12_B Conversion Memory Control x Register (ADC12MCTLx). Differential input mode should be selected for differential input signals and can also be used for single-ended signals by tying the negative input to AVSS. The advantage of using differential mode is increased common mode noise rejection at the cost of a small increase in current consumption.

The ADC12_B uses the charge redistribution method. When the inputs are internally switched, the switching action may cause transients on the input signal. These transients decay and settle before causing errant conversion.

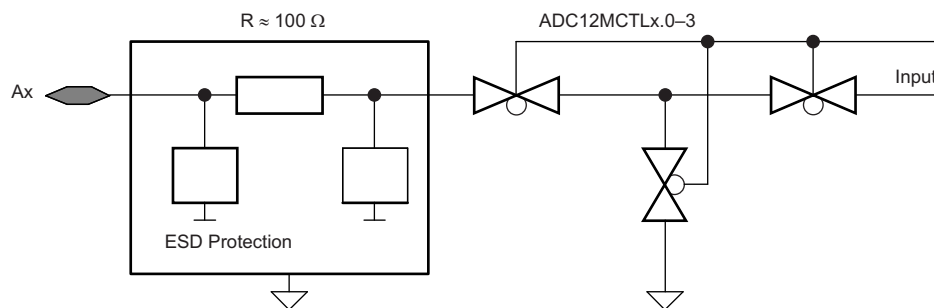


Figure 34-2. Analog Multiplexer T-Switch

34.2.2.1 Analog Port Selection

The ADC12_B inputs are multiplexed with digital port pins. When analog signals are applied to digital gates, parasitic current can flow from V_{CC} to GND. This parasitic current occurs if the input voltage is near the transition level of the gate. Disabling the digital part of the port pin eliminates the parasitic current flow and, therefore, reduces overall current consumption. The PySELx bits can disable the port pin input and output buffers. Refer to the device specific port x input/output schematic and table for the ADC12_B input pin for PySELx details.

34.2.3 Voltage References

The ADC12_B module may use an on-chip shared reference module that supplies three selectable voltage levels of 1.2 V, 2.0 V, and 2.5 V (see the reference module for proper configuration details) to supply V_{R+} and V_{R-}. These reference voltages may be used internally and externally on pin VREF+ if REFOUT=1. Alternatively, external references may be supplied for V_{R+} and V_{R-} through pins VREF+/VeREF+ and VeREF-. The ADC12_B module reference selection is through the ADC12VRSEL bits. For pin flexibility VR+ and VR- are not restricted to VeREF+ and VeREF- respectively. Care must be taken that ADC12VRSEL does not conflict with REFOUT bit settings as only one buffer is available for internal reference with REFOUT=1 or ADC12_B module reference when external reference with internal buffer is selected. So if REFOUT=1, VeREF+ buffered should not be selected with ADC12VRSEL = 0x3, 0x5, or 0xF.

34.2.4 Auto Power Down

The ADC12_B is designed for low-power applications. When the ADC12_B is not actively converting, the core is automatically disabled and automatically reenabled when needed. The MODOSC that sources MODCLK is also automatically enabled when needed and disabled when not needed, if the ADC12VRSEL selects the internal reference for the ADC.

If REFON=1, the internal reference is on continually; otherwise, it is only requested when an ADC conversion is triggered. The REF buffer is powered down between conversions to save power unless REFOUT=1, or pulse sample mode is used with ADC12MSC=1, or a conversion mode other than single-channel single conversion is used. When the REF buffer is powered down in pulse sample mode, the ADC sample time does not start until the REF buffer is ready (ADC12RDYIFFG=1), so the user does not need to do anything. When the REF buffer is powered down in extended sample mode, the user must account for the REF buffer settle/ready time by using the ADC12RDYIFFG=1 in calculating the time the trigger should be asserted to make sure that the application meets the required sample time or ADC12_B minimum sample time.

34.2.5 Sample Frequency Mode Selection

The ADC12PWRMD bit optimizes the ADC12_B power consumption at two ADC12CLK ranges. Select the lowest ADC12CLK frequency that meets or exceeds the application requirements. If ADC12CLK is 1/4 or less of data sheet specified maximum for ADC12PWRMD=0, ADC12PWRMD=1 may be set to save power.

34.2.6 Sample and Conversion Timing

A rising edge of the sample input signal (SHI) initiates an analog-to-digital conversion. The sample input signal can be inverted with the ADC12ISSH bit. The SHSx bits select the source for SHI and include the following:

- ADC12SC bit
- Up to seven other sources that may include timer output (see to the device-specific data sheet for available sources).

The ADC12_B supports 8-bit, 10-bit, and 12-bit resolution modes, and the ADC12RES bits select the current mode. The analog-to-digital conversion requires 10, 12, and 14 ADC12CLK cycles, respectively. The ADC12ISSH bit can invert the polarity of the SHI signal source. The SAMPCON signal controls the sample period and start of conversion. When SAMPCON is high, sampling is active. The high-to-low SAMPCON transition starts the analog-to-digital conversion after one clock cycle for pulse sample mode and after one clock cycle plus a clock sync in extended sample mode. Control bit ADC12SHP defines the sample-timing method, either extended sample mode or pulse mode. See the device-specific data sheet for timers that are available for SHI sources.

34.2.6.1 Extended Sample Mode

ADC12SHP = 0 selects the extended sample mode. The SHI signal directly controls SAMPCON and defines the length of the sample period t_{sample} . If an ADC local reference buffer is used, the user should assert the sample trigger, wait for the ADC12RDYIFG flag to be set (which indicates that the ADC12_B local reference buffer is settled, and the flag does not occur if the sample trigger has not been asserted), and then keep the sample trigger asserted for the desired sample period before de-asserting. Alternately, if a local reference buffer is used, the user may assert the sample trigger for the desired sample time plus the maximum time for the reference and buffers to settle (reference and buffer settling times are provided in the device-specific data sheet). An ADC local reference buffer is used when ADC12VRSEL= 0001, 0011, 0101, 0111, 1001, 1011, 1101, or 1111. When SAMPCON is high, sampling is active. The high-to-low SAMPCON transition starts the conversion after synchronization with ADC12CLK plus one clock cycle (see [Figure 34-3](#) and [Figure 34-4](#)).

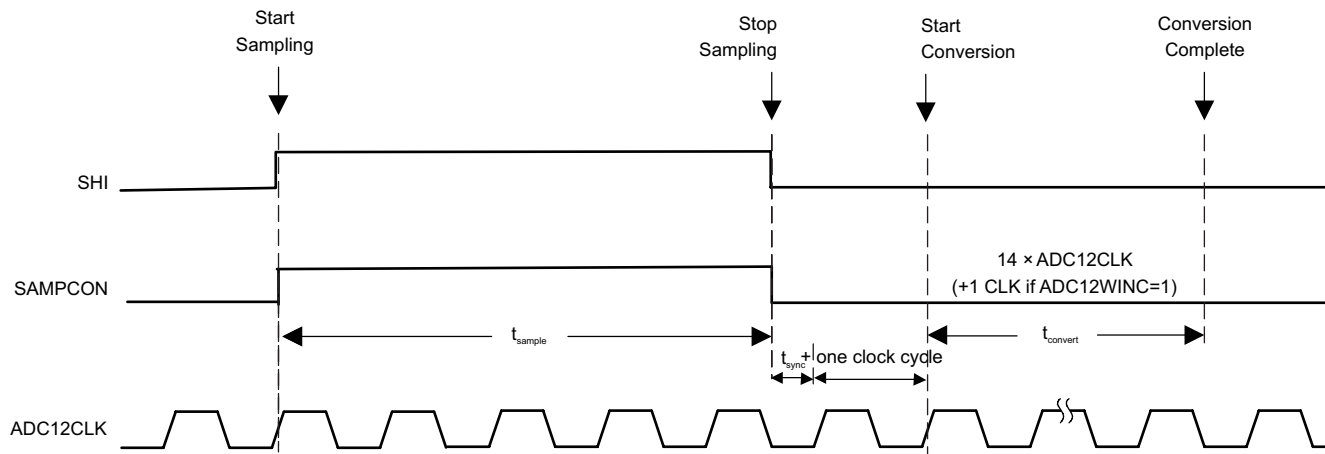


Figure 34-3. Extended Sample Mode Without Internal Reference in 12-Bit Mode

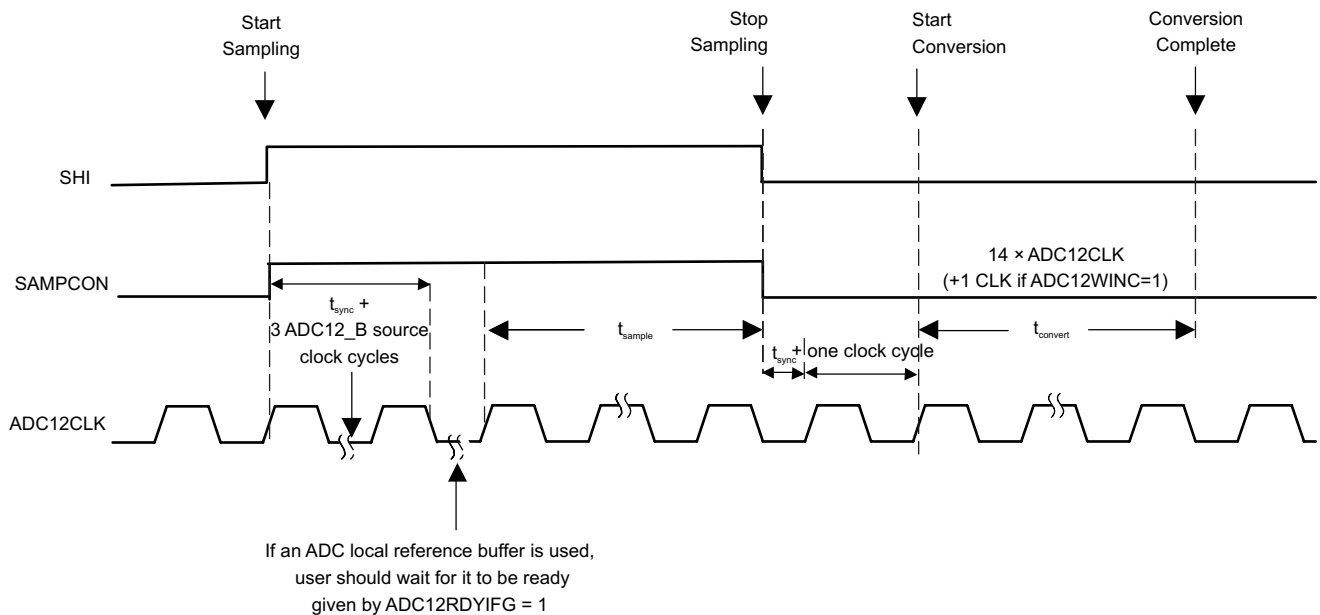


Figure 34-4. Extended Sample Mode With Internal Reference in 12-Bit Mode

34.2.6.2 Pulse Sample Mode

ADC12SHP = 1 selects the pulse sample mode. The SHI signal triggers the sampling timer. The ADC12SHT0x and ADC12SHT1x bits in ADC12CTL0 control the interval of the sampling timer that defines the SAMPCON sample period t_{sample} . The sampling timer keeps SAMPCON high while waiting for reference and ADC local reference buffer to settle (if the internal reference is used), synchronization with ADC12CLK, and for the programmed interval t_{sample} . The exception is for the first conversion or where ADC12MSC=0 where an extra 3 ADC12_B source clock cycles is required when SAMPCON goes high. (see [Figure 34-5](#) and [Figure 34-6](#)).

The ADC12SHTx bits select the sampling time in 4x multiples of ADC12CLK. ADC12SHT1x selects the sampling time for ADC12MEM8 to ADC12MEM23, and ADC12SHT0x selects the sampling time for ADC12MEM0 to ADC12MEM7 and ADC12MEM24 to ADC12MEM31.

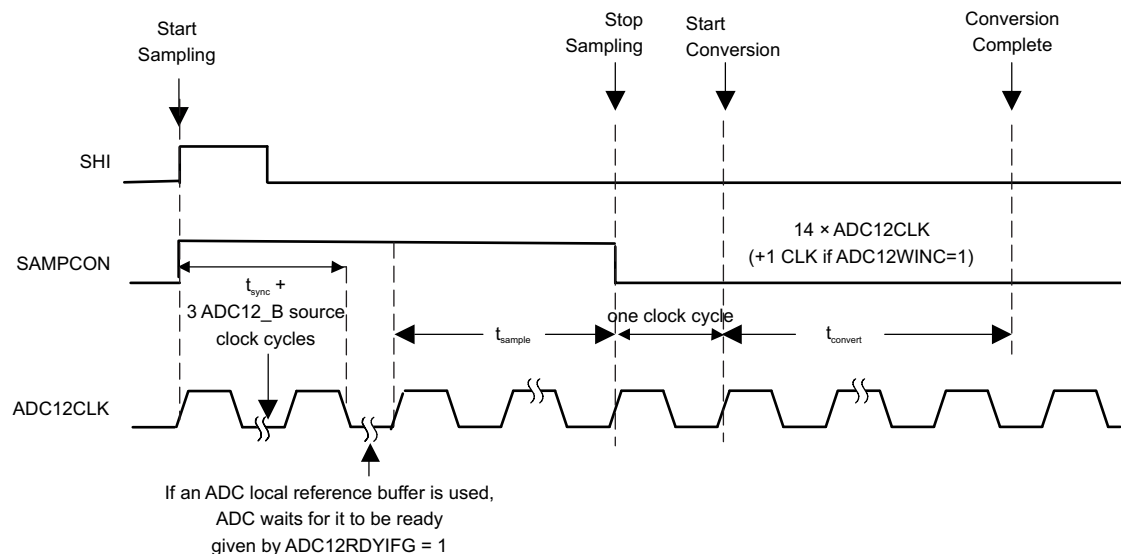


Figure 34-5. Pulse Sample Mode First Conversion or Where ADC12MSC = 0 in 12-Bit Mode

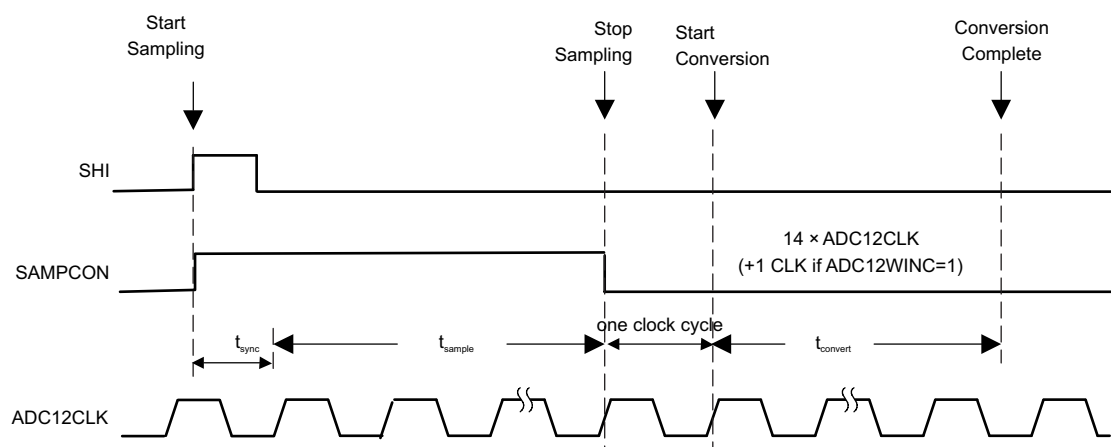


Figure 34-6. Pulse Sample Mode Subsequent Conversions in 12-Bit Mode

34.2.6.3 Sample Timing Considerations

When SAMPCON = 0, all Ax inputs are high impedance. When SAMPCON = 1, the selected Ax input can be modeled as an RC low-pass filter during the sampling time t_{sample} (see Figure 34-7). An internal MUX-on input resistance R_i (see the device-specific data sheet) in series with capacitor C_i (see the device-specific data sheet) is seen by the source. The capacitor C_i voltage (V_c) must be charged to within one-half LSB of the source voltage (V_s) for an accurate n-bit conversion, where n is the bits of resolution required.

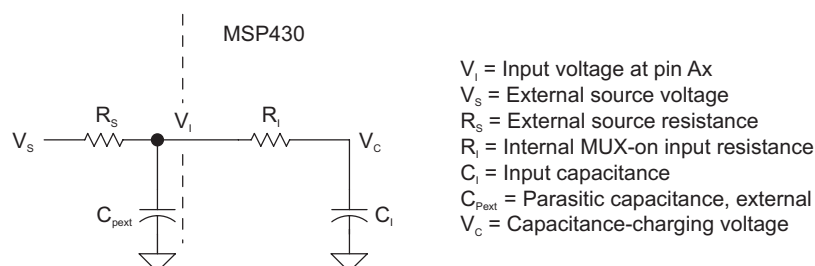


Figure 34-7. Analog Input Equivalent Circuit

The resistance of the source R_S and R_I affect t_{sample} . Use Equation 17 to calculate the minimum sampling time t_{sample} for a n-bit conversion, where n equals the bits of resolution.

$$t_{\text{sample}} \geq (R_S + R_I) \times \ln(2^{n+2}) \times (C_I + C_{\text{pext}}), R_S < 10 \text{ k}\Omega \quad (17)$$

See the device-specific data sheet for R_I and C_I values.

34.2.7 Conversion Memory

32 ADC12MEMx conversion memory registers store the conversion results. Each ADC12MEMx is configured with an associated ADC12MCTLx control register. The ADC12VRSEL bits define the voltage reference, and the ADC12INCHx and ADC12DIF bits select the input channels. The ADC12EOS bit defines the end of sequence when a sequential conversion mode is used. A sequence rolls over from ADC12MEM31 to ADC12MEM0 when the ADC12EOS bit in ADC12MCTL31 is not set.

The CSTARTADDx bits define the first ADC12MCTLx used for any conversion. If the conversion mode is single-channel or repeat-single-channel, the CSTARTADDx points to the single ADC12MCTLx to be used.

If the conversion mode selected is either sequence-of-channels or repeat-sequence-of-channels, CSTARTADDx points to the first ADC12MCTLx location to be used in a sequence. A pointer, not visible to software, is incremented automatically to the next ADC12MCTLx in sequence when each conversion completes. The sequence continues until an ADC12EOS bit in ADC12MCTLx is processed; this is, the last control byte processed.

When conversion results are written to a selected ADC12MEMx, the corresponding flag in the ADC12IFGRx register is set.

There are two formats available to read the conversion result from ADC12MEMx. When ADC12DF = 0, the conversion is right justified and unsigned. For ADC12DF = 0 with ADC12DIF = 0 and 8-bit, 10-bit, and 12-bit resolutions, the upper 8, 6, and 4 bits, respectively, of an ADC12MEMx read are always zeros. To convert a ADC12DIF = 1 to binary unsigned, the maximum negative value is added to the conversion. Therefore, 128 is added for 8-bit conversions, 512 is added for 10-bit conversions, and 2048 is added for 12-bit conversions.

When ADC12DF = 1, the conversion result is left justified and two's complement. For 8-bit, 10-bit, and 12-bit resolutions, the lower 8, 6, and 4 bits, respectively, of a ADC12MEMx read are always zeros.

Table 34-1 summarizes the output data formats.

Table 34-1. ADC12_B Conversion Result Formats

Analog Input Voltage Range	ADC12DIF	ADC12DF	ADC12RES	Ideal Conversion Results (With Offset Added When ADC12DIF = 1)	ADC12MEMx Read Value
Vin to VR-: VR- to +VR+	0	0	00	0 to 255	0000h to 00FFh
	0	0	01	0 to 1023	0000h to 03FFh
	0	0	10	0 to 4095	0000h to 0FFFh
	0	1	00	-128 to 127	8000h to 7F00h
	0	1	01	-512 to 511	8000h to 7FC0h
	0	1	10	-2048 to 2047	8000h to 7FF0h
Vin+ to Vin-: VR- to +VR+	1	0	00	-128 to 127 (0 to 255)	0000h to 00FFh
	1	0	01	-512 to 511 (0 to 1023)	0000h to 03FFh
	1	0	10	-2048 to 2047 (0 to 4095)	0000h to 0FFFh
	1	1	00	-128 to 127	8000h to 7F00h
	1	1	01	-512 to 511	8000h to 7FC0h
	1	1	10	-2048 to 2047	8000h to 7FF0h

34.2.8 ADC12_B Conversion Modes

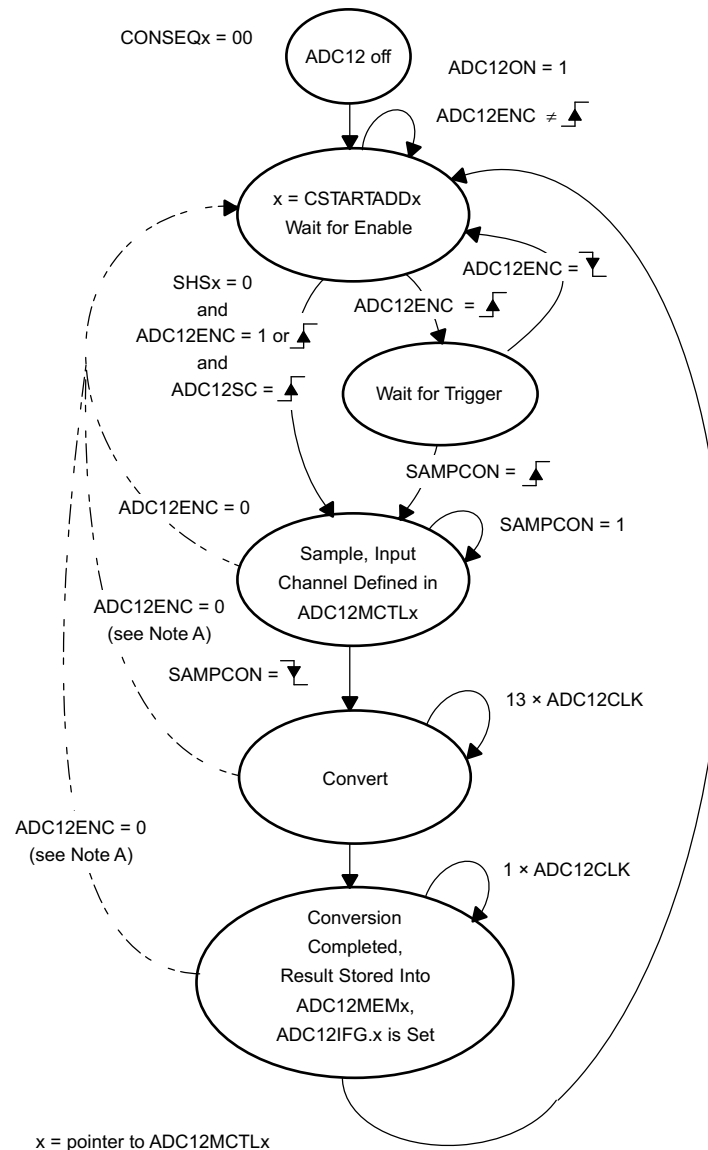
Table 34-2 shows the four operating modes that are selected by the CONSEQx bits. All state diagrams assume a 12-bit resolution setting.

Table 34-2. Conversion Mode Summary

ADC12CONSEQx	Mode	Operation
00	Single-channel single-conversion	A single channel is converted once.
01	Sequence-of-channels (autoscan)	A sequence of channels is converted once.
10	Repeat-single-channel	A single channel is converted repeatedly.
11	Repeat-sequence-of-channels (repeated autoscan)	A sequence of channels is converted repeatedly.

34.2.8.1 Single-Channel Single-Conversion Mode

A single channel is sampled and converted once. The ADC result is written to the ADC12MEMx that is defined by the CSTARTADDx bits. Figure 34-8 shows the flow of the single-channel single-conversion mode when RES = 0x2 for 12-bit mode. When ADC12SC triggers a conversion, the ADC12SC bit can trigger successive conversions. When any other trigger source is used, ADC12ENC must be toggled between each conversion. When there are multiple triggers then ADC12ENC bit must be toggled after the additional trigger(s) for lowest power (otherwise clocks are still requested even after conversion is complete).



x = pointer to ADC12MCTLx

A Conversion result is unpredictable.

Figure 34-8. Single-Channel Single-Conversion Mode, ADC12ISSH = 0

34.2.8.2 Sequence-of-Channels Mode (Autoscan Mode)

In sequence-of-channels mode, also called autoscan mode, a sequence of channels is sampled and converted once. The ADC results are written to the conversion memories starting with the ADC12MEMx that is defined by the CSTARTADDx bits. The sequence stops after the measurement of the channel with a set ADC12EOS bit. Figure 34-9 shows the sequence-of-channels mode when RES = 0x02 for 12-bit mode. When ADC12SC triggers a sequence, the ADC12SC bit can trigger successive sequences. When any other trigger source is used, ADC12ENC must be toggled between each sequence. When there are multiple triggers then ADC12ENC bit must be toggled after the additional trigger(s) for lowest power (otherwise clocks are still requested even after conversion sequence is complete).

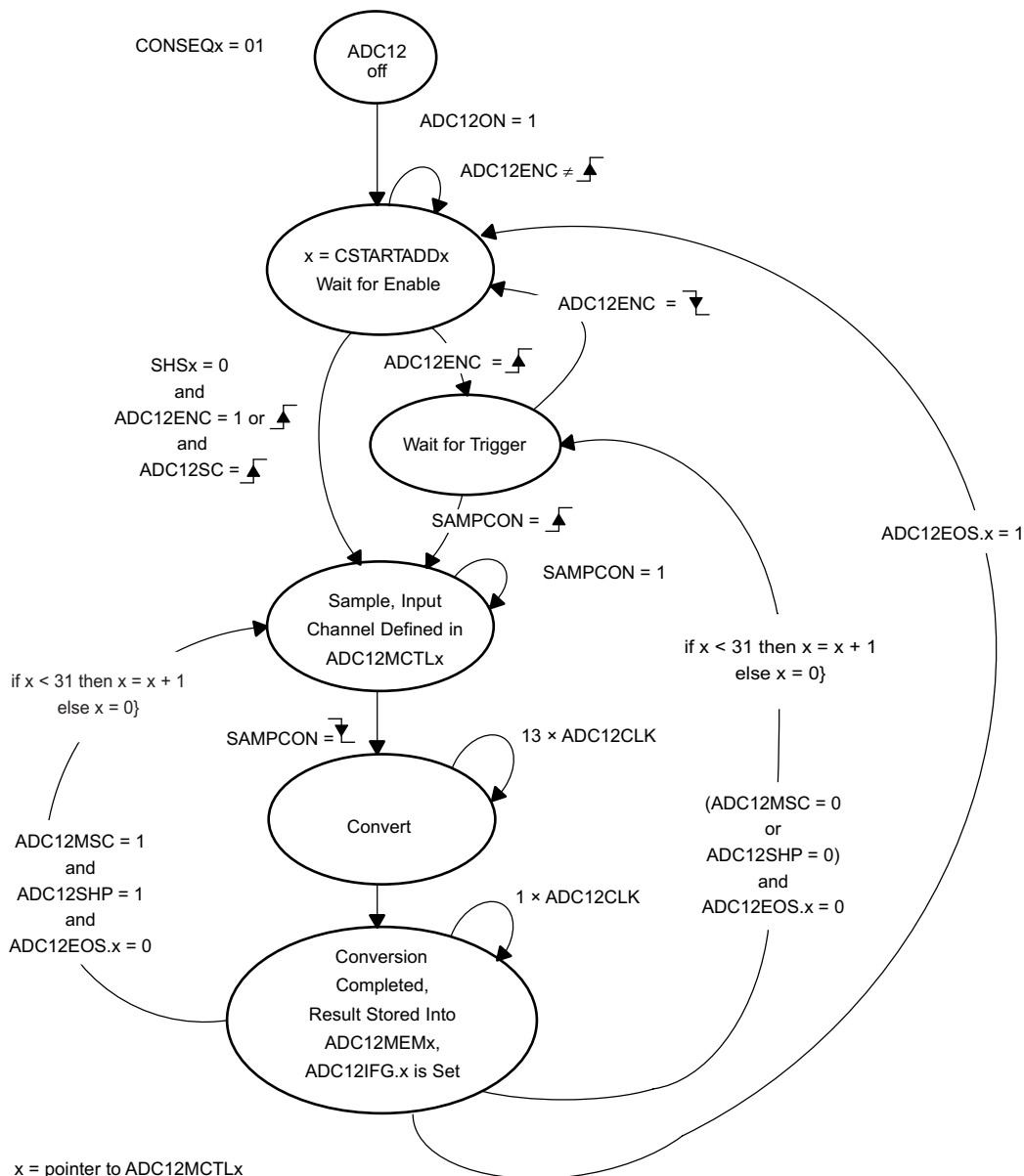


Figure 34-9. Sequence-of-Channels Mode, ADC12ISSH = 0

34.2.8.3 Repeat-Single-Channel Mode

In repeat-single-channel mode, a single channel is sampled and converted continuously. The ADC results are written to the ADC12MEMx defined by the CSTARTADDx bits. It is necessary to read the result after the completed conversion, because only one ADC12MEMx memory is used and is overwritten by the next conversion. Figure 34-10 shows the repeat-single-channel mode when RES = 0x2 for 12-bit mode.

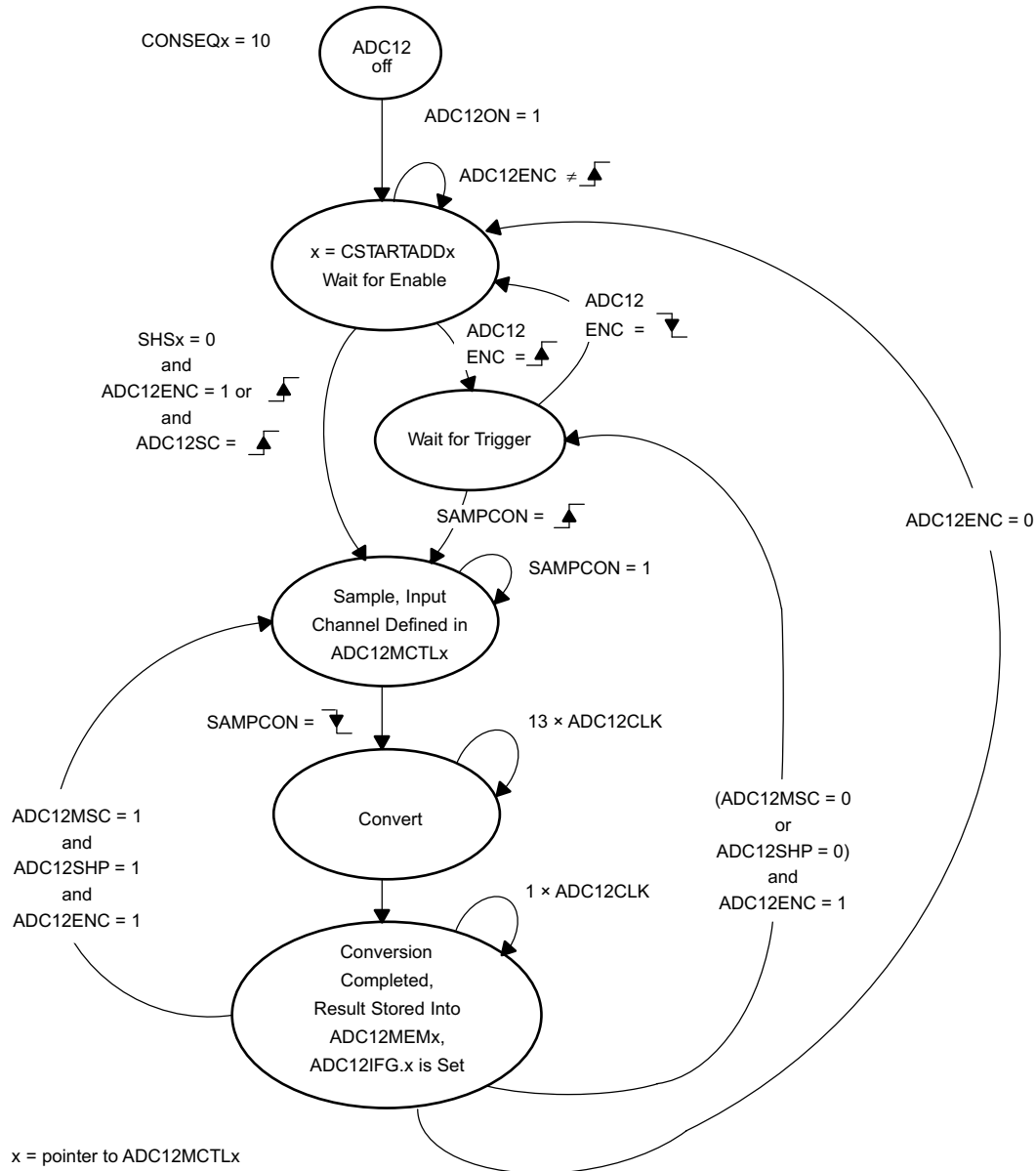


Figure 34-10. Repeat-Single-Channel Mode, ADC12ISSH = 0

34.2.8.4 Repeat-Sequence-of-Channels Mode (Repeated Autoscan Mode)

In repeat-sequence-of-channels mode, a sequence of channels is sampled and converted repeatedly. This mode is also called repeated autoscan mode. The ADC results are written to the conversion memories starting with the ADC12MEMx that is defined by the CSTARTADDx bits. The sequence ends after the measurement of the channel with a set ADC12EOS bit, and the next trigger signal restarts the sequence. Figure 34-11 shows the repeat-sequence-of-channels mode.

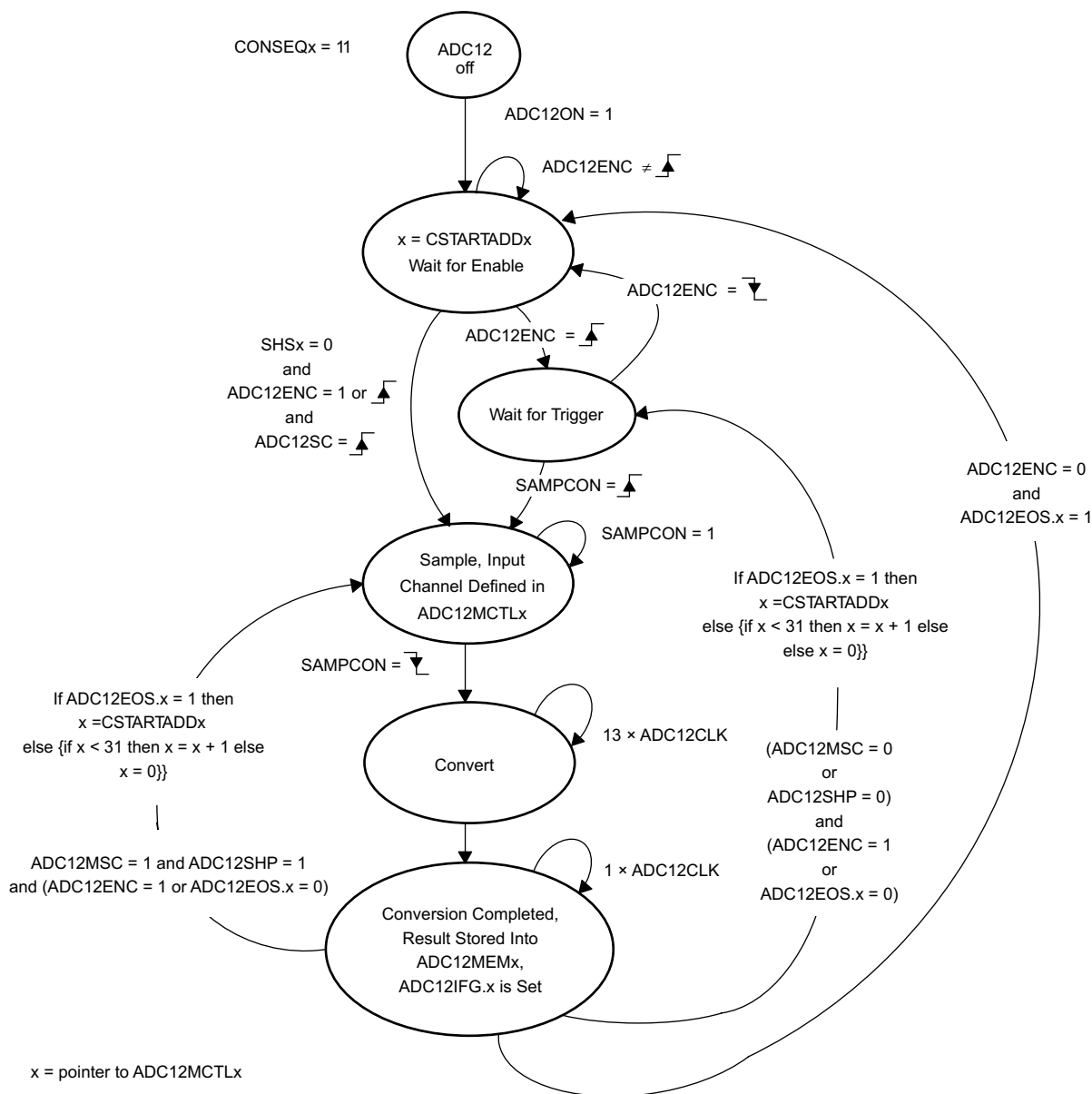


Figure 34-11. Repeat-Sequence-of-Channels Mode, ADC12ISSH = 0

34.2.8.5 Using the Multiple Sample and Convert (ADC12MSC) Bit

To configure the converter to perform successive conversions automatically and as quickly as possible, a multiple sample and convert function is available. When ADC12MSC = 1, CONSEQx > 0, and the sample timer is used (pulse sample mode, ADC12SHP = 1), the first rising edge of SHI signal triggers the first conversion. Successive conversions are triggered automatically as soon as the prior conversion is completed (if the ADC local reference buffer is used, ADC12VRSEL = 0001, 0011, 0101, 0111, 1001, 1011, 1101, or 1111, there is one clock cycle before the successive conversion is triggered). Additional SHI triggers are ignored until the sequence is completed in the single-sequence mode, or until the ADC12ENC bit is toggled in repeat-single-channel or repeated-sequence modes. The function of the ADC12ENC bit is unchanged when using the ADC12MSC bit.

34.2.8.6 Stopping Conversions

Stopping ADC12_B activity depends on the mode of operation. The recommended ways to stop an active conversion or conversion sequence are:

- Reset ADC12ENC in single-channel single-conversion mode to stop a conversion immediately. The results are unreliable. For correct results, poll the busy bit until it is reset before clearing ADC12ENC.
- Reset ADC12ENC during repeat-single-channel operation to stop the converter at the end of the current conversion.
- Reset ADC12ENC during a sequence or repeat-sequence mode to stop the converter at the end of the current conversion.
- Stop any conversion mode immediately by setting the CONSEQx = 0 and resetting the ADC12ENC and ADC12ON bit. Conversion data are unreliable.

NOTE: No ADC12EOS bit set for sequence

If no ADC12EOS bit is set and a sequence mode is selected, resetting the ADC12ENC bit does not stop the sequence. To stop the sequence, first select a single-channel mode and then reset ADC12ENC.

34.2.9 Operation in LPM3 and LPM4

The ADC remains active in LPM3 if the following are all true:

- ADC is on (ADC12ON = 1).
- Conversion is enabled (ADC12ENC = 1).
- External triggers are selected (ADC12SHSx ≠ 0) OR ACLK is ADC12B source clock (ADC12SSELx = 01b).

The ADC remains active in LPM4 if the following are all true:

- ADC is on (ADC12ON = 1).
- Conversion is enabled (ADC12ENC = 1).
- External triggers are selected (ADC12SHSx ≠ 0).

34.2.10 Window Comparator

The window comparator allows to monitor analog signals without any CPU interaction. It is enabled for the desired ADC12MEMx conversion with the ADC12WINC bit in the ADC12MCTLx register. In the following the window comparator interrupts are listed:

- The ADC12LO interrupt flag (ADC12LOIFG) is set if the current result of the ADC12_B conversion is below the low threshold defined in register ADC12LO.
- The ADC12HI interrupt flag (ADC12HIIFG) is set if the current result of the ADC12_B conversion is greater than the high threshold defined in the register ADC12HI.
- The ADC12IN interrupt flag (ADC12INIFG) is set if the current result of the ADC12_B conversion is greater than the low threshold defined in register ADC12LO and less than the high threshold defined in ADC12HI.

These interrupts are generated independently of the conversion mode selected by the user. The update of the window comparator interrupt flags happen after the ADC12IFGx.

The lower and higher threshold in the ADC12LO and ADC12HI registers have to be given in the correct data format. If the binary unsigned data format is selected by ADC12DF = 0, then the thresholds in the registers ADC12LO and ADC12HI must be written as binary unsigned values. If the signed binary (2s complement) data format is selected by ADC12DF = 1, then the thresholds in the registers ADC12LO and ADC12HI must be written as signed binary (2s complement). Altering the ADC12DF register or the ADC12RES register resets the threshold registers.

The interrupt flags are reset by the user software. The ADC12_B sets the interrupt flags each time a new conversion result is available in the ADC12MEMx register if applicable. Interrupt flags are not cleared by hardware. The user software resets the window comparator interrupt flags per the application needs.

34.2.11 Using the Integrated Temperature Sensor

To use the on-chip temperature sensor, the user must enable the temperature sensor input channel by setting the ADC12TCMAP bit equal to 1 in the ADC12CTL3 register. The user must then select the analog input channel ADC12INCHx = 0x1E for the temperature sensor. Any other configuration is done as if an external channel were selected, including reference selection, conversion-memory selection, and so on. The temperature sensor is in the REF module.

A typical temperature sensor transfer function is shown in [Figure 34-12](#). The transfer function shown is only an example. Calibration is required to determine the corresponding voltages for a specific device. When using the temperature sensor, the sample period must be greater than 30 μ s. The temperature sensor offset error can be large and may need to be calibrated for most applications. Temperature calibration values are available for use in the TLV descriptors (see the device-specific data sheet for locations). Some MSP430 devices include calibration data that can be used to compute temperature more accurately. For more information, refer to [Section 1.14.3.3](#).

Selecting the temperature sensor automatically turns on the on-chip reference generator as a voltage source for the temperature sensor. However, it does not enable the VREF+ output or affect the reference selections for the conversion. The reference choices for converting the temperature sensor are the same as with any other channel.

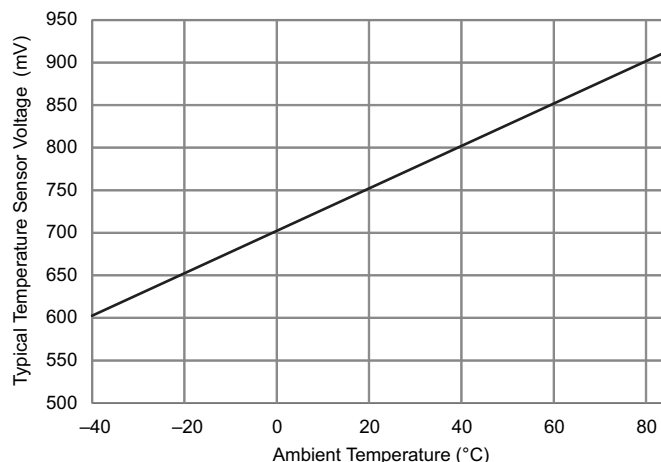


Figure 34-12. Typical Temperature Sensor Transfer Function

34.2.12 ADC12_B Grounding and Noise Considerations

As with any high-resolution ADC, appropriate printed-circuit-board layout and grounding techniques should be followed to eliminate ground loops, unwanted parasitic effects, and noise.

Ground loops are formed when return current from the ADC flows through paths that are common with other analog or digital circuitry. If care is not taken, this current can generate small unwanted offset voltages that can add to or subtract from the reference or input voltages of the ADC. The connections shown in Figure 34-13 prevent this.

In addition to grounding, ripple and noise spikes on the power-supply lines due to digital switching or switching power supplies can corrupt the conversion result. A noise-free design using separate analog and digital ground planes with a single-point connection is recommend to achieve high accuracy.

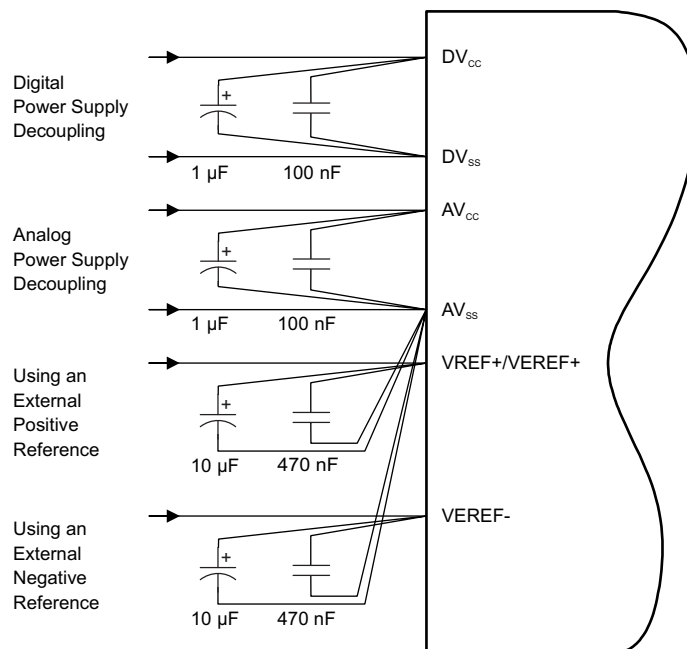


Figure 34-13. ADC12_B Grounding and Noise Considerations

34.2.13 ADC12_B Calibration

The device TLV structure contains calibration values that can be used to improve the measurement capability of the ADC12_B. Refer to [Section 1.14](#) of the *System Resets, Interrupts, and Operating Modes, System Control Module (SYS)* chapter for more details.

34.2.14 ADC12_B Interrupts

The ADC12_B has 38 interrupt sources:

- ADC12IFG0 to ADC12IFG31
- ADC12OVIFG: ADC12MEMx overflow
- ADC12TOVIFG: ADC12_B conversion time overflow
- ADC12LOIFG, ADC12INIFG, and ADC12HIIFG for ADC12MEMx
- ADC12RDYIFG: ADC12_B local reference buffer ready

The ADC12IFGx bits are set when their corresponding ADC12MEMx memory register is loaded with a conversion result. An interrupt request is generated if the corresponding ADC12IEx bit and the GIE bit are set. The conversion result written into ADC12MEMx result register also sets the ADC12LOIFG, ADC12INIFG or ADC12HIIFG if applicable. The ADC12OVIFG condition occurs when a conversion result is written to any ADC12MEMx before its previous conversion result was read. The ADC12TOVIFG condition is generated when another sample-and-conversion is requested before the current conversion is completed. The DMA is triggered after the conversion in single-channel conversion mode or after the completion of a sequence of channel conversions in sequence-of-channels conversion mode. See [Section 11.2.11](#) for additional details. The ADC12RDYIFG is set after the sample trigger is asserted when the ADC12_B local reference buffer is ready. Note the ADC12RDYIFG will be set even when the ADC12B does not select the buffered reference. It can be used during extended sample mode instead of adding the max ADC12_B local reference buffer settle time to the sample signal time.

34.2.14.1 ADC12IV, Interrupt Vector Generator

All ADC12_B interrupt sources are prioritized and combined to source a single interrupt vector. The interrupt vector register ADC12IV is used to determine which ADC12_B interrupt source requested an interrupt.

The highest-priority enabled ADC12_B interrupt generates a number in the ADC12IV register (see [Section 34.3.15](#)). This number can be evaluated or added to the program counter (PC) to automatically enter the appropriate software routine. ADC12_B interrupts that are disabled do not affect the ADC12IV value.

Read access of the ADC12IV register automatically resets the highest pending interrupt condition and flag except the ADC12IFGx flags. ADC12IFGx bits are reset automatically by accessing their associated ADC12MEMx register or may be reset with software.

Write access of the ADC12IV register clears all pending interrupt conditions and flags.

If another interrupt is pending after servicing of an interrupt, another interrupt is generated. For example, if the ADC12OV and ADC12IFG3 interrupts are pending when the interrupt service routine accesses the ADC12IV register, the ADC12OV interrupt condition is reset automatically. After the RETI instruction of the interrupt service routine is executed, the ADC12IFG3 generates another interrupt.

34.2.14.2 ADC12_B Interrupt Handling Software Example

The following software example shows the recommended use of the ADC12IV and handling overhead. The ADC12IV value is added to the PC to automatically jump to the appropriate routine.

The software overhead for different interrupt sources includes interrupt latency and return-from-interrupt cycles, but not the task handling itself, are:

- ADC12IFG0 through ADC12IFG30, ADC12TOV, ADC12OV, ADC12LO, ADC12HI, ADC12IN, ADC12RDY: 16 cycles
- ADC12IFG31: 14 cycles

The interrupt handler for ADC12IFG31 shows a way to check immediately if a higher-prioritized interrupt occurred during the processing of ADC12IFG31. This saves nine cycles if another ADC12_B interrupt is pending.

```
; Interrupt handler for ADC12.
INT_ADC12      ; Enter Interrupt Service Routine
    ADD        &ADC12IV,PC    ; Add offset to PC
    RETI        ; Vector 0: No interrupt
    JMP        ADOV           ; Vector 2: ADC overflow
    JMP        ADTOV          ; Vector 4: ADC timing overflow
    JMP        ADHI           ; Vector 6: ADC12HIIFG
    JMP        ADLO           ; Vector 8: ADC12LOIFG
    JMP        ADIN           ; Vector A: ADC12INIFG
    JMP        ADM0           ; Vector C: ADC12IFG0
    ...           ; Vectors E-70
    JMP        ADM30          ; Vector 72: ADC12IFG30
    ...
    JMP        ADRDY          ; Vector 76: ADC12RDYIFG
;
; Handler for ADC12IFG31 starts here. No JMP required.
;
;
ADM31          MOV        &ADC12MEM31,xxx    ; Move result, flag is reset
    ...           ; Other instruction needed?
    JMP        INT_ADC12    ; Check other int pending
;
; ADC12IFG30-ADC12IFG1 handlers go here
;
ADM0           MOV        &ADC12MEM0,xxx    ; Move result, flag is reset
    ...           ; Other instruction needed?
    RETI          ; Return;
ADTOV          ...           ; Handle Conv. time overflow
    RETI          ; Return;
ADOV           ...           ; Handle ADC12MEMx overflow
    RETI          ; Return;
ADHI           ...           ; Handle window comparator high Interrupt
    RETI          ; Return;
ADLO           ...           ; Handle window comparator low Interrupt
    RETI          ; Return;
ADIN           ...           ; Handle window comparator in window Interrupt
    RETI          ; Return;
ADRDY          ...           ; Handle window comparator in window Interrupt
    RETI          ; Return;
```


34.3 ADC12_B Registers

Table 34-3 lists the memory-mapped registers for the ADC12_B. See the device-specific data sheet for the base memory address of these registers. All other register offset addresses not listed in **Table 34-3** should be considered as reserved locations, and the register contents should not be modified.

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 34-3. ADC12_B Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	ADC12CTL0	ADC12_B Control 0	Read/write	Word	0000h	Section 34.3.1
00h	ADC12CTL0_L		Read/write	Byte	00h	
01h	ADC12CTL0_H		Read/write	Byte	00h	
02h	ADC12CTL1	ADC12_B Control 1	Read/write	Word	0000h	Section 34.3.2
02h	ADC12CTL1_L		Read/write	Byte	00h	
03h	ADC12CTL1_H		Read/write	Byte	00h	
04h	ADC12CTL2	ADC12_B Control 2	Read/write	Word	0020h	Section 34.3.3
04h	ADC12CTL2_L		Read/write	Byte	20h	
05h	ADC12CTL2_H		Read/write	Byte	00h	
06h	ADC12CTL3	ADC12_B Control 3	Read/write	Byte	0000h	Section 34.3.4
06h	ADC12CTL3_L		Read/write	Byte	00h	
07h	ADC12CTL3_H		Read/write		00h	
08h	ADC12LO	ADC12_B Window Comparator Low Threshold Register	Read/write	Word	0000h	Section 34.3.8
08h	ADC12LO_L		Read/write	Byte	00h	
09h	ADC12LO_H		Read/write	Byte	00h	
0Ah	ADC12HI	ADC12_B Window Comparator High Threshold Register	Read/write	Word	0FFFh	Section 34.3.7
0Ah	ADC12HI_L		Read/write	Byte	FFh	
0Bh	ADC12HI_H		Read/write	Byte	0Fh	
0Ch	ADC12IFGR0	ADC12_B Interrupt Flag 0	Read/write	Word	0000h	Section 34.3.12
0Ch	ADC12IFGR0_L		Read/write	Byte	00h	
0Dh	ADC12IFGR0_H		Read/write	Byte	00h	
0Eh	ADC12IFGR1	ADC12_B Interrupt Flag 1	Read/write	Word	0000h	Section 34.3.13
0Eh	ADC12IFGR1_L		Read/write	Byte	00h	
0Fh	ADC12IFGR1_H		Read/write	Byte	00h	
10h	ADC12IFGR2	ADC12_B Interrupt Flag 2	Read/write	Word	0000h	Section 34.3.14
10h	ADC12IFGR2_L		Read/write	Byte	00h	
11h	ADC12IFGR2_H		Read/write	Byte	00h	
12h	ADC12IER0	ADC12_B Interrupt Enable 0	Read/write	Word	0000h	Section 34.3.9
12h	ADC12IER0_L		Read/write	Byte	00h	
13h	ADC12IER0_H		Read/write	Byte	00h	
14h	ADC12IER1	ADC12_B Interrupt Enable 1	Read/write	Word	0000h	Section 34.3.10
14h	ADC12IER1_L		Read/write	Byte	00h	
15h	ADC12IER1_H		Read/write	Byte	00h	
16h	ADC12IER2	ADC12_B Interrupt Enable 2	Read/write	Word	0000h	Section 34.3.11
16h	ADC12IER2_L		Read/write	Byte	00h	
17h	ADC12IER2_H		Read/write	Byte	00h	

Table 34-3. ADC12_B Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
18h	ADC12IV	ADC12_B Interrupt Vector	Read/write	Word	0000h	Section 34.3.15
18h	ADC12IV_L		Read/write	Byte	00h	
19h	ADC12IV_H		Read	Byte	00h	
20h	ADC12MCTL0	ADC12_B Memory Control 0	Read/write	Word	0000h	Section 34.3.6
20h	ADC12MCTL0_L		Read/write	Byte	00h	
21h	ADC12MCTL0_H		Read/write	Byte	00h	
22h	ADC12MCTL1	ADC12_B Memory Control 1	Read/write	Word	0000h	Section 34.3.6
22h	ADC12MCTL1_L		Read/write	Byte	00h	
23h	ADC12MCTL1_H		Read/write	Byte	00h	
24h	ADC12MCTL2	ADC12_B Memory Control 2	Read/write	Word	0000h	Section 34.3.6
24h	ADC12MCTL2_L		Read/write	Byte	00h	
25h	ADC12MCTL2_H		Read/write	Byte	00h	
26h	ADC12MCTL3	ADC12_B Memory Control 3	Read/write	Word	0000h	Section 34.3.6
26h	ADC12MCTL3_L		Read/write	Byte	00h	
27h	ADC12MCTL3_H		Read/write	Byte	00h	
28h	ADC12MCTL4	ADC12_B Memory Control 4	Read/write	Word	0000h	Section 34.3.6
28h	ADC12MCTL4_L		Read/write	Byte	00h	
29h	ADC12MCTL4_H		Read/write	Byte	00h	
2Ah	ADC12MCTL5	ADC12_B Memory Control 5	Read/write	Word	0000h	Section 34.3.6
2Ah	ADC12MCTL5_L		Read/write	Byte	00h	
2Bh	ADC12MCTL5_H		Read/write	Byte	00h	
2Ch	ADC12MCTL6	ADC12_B Memory Control 6	Read/write	Word	0000h	Section 34.3.6
2Ch	ADC12MCTL6_L		Read/write	Byte	00h	
2Dh	ADC12MCTL6_H		Read/write	Byte	00h	
2Eh	ADC12MCTL7	ADC12_B Memory Control 7	Read/write	Word	0000h	Section 34.3.6
2Eh	ADC12MCTL7_L		Read/write	Byte	00h	
2Fh	ADC12MCTL7_H		Read/write	Byte	00h	
30h	ADC12MCTL8	ADC12_B Memory Control 8	Read/write	Word	0000h	Section 34.3.6
30h	ADC12MCTL8_L		Read/write	Byte	00h	
31h	ADC12MCTL8_H		Read/write	Byte	00h	
32h	ADC12MCTL9	ADC12_B Memory Control 9	Read/write	Word	0000h	Section 34.3.6
32h	ADC12MCTL9_L		Read/write	Byte	00h	
33h	ADC12MCTL9_H		Read/write	Byte	00h	
34h	ADC12MCTL10	ADC12_B Memory Control 10	Read/write	Word	0000h	Section 34.3.6
34h	ADC12MCTL10_L		Read/write	Byte	00h	
35h	ADC12MCTL10_H		Read/write	Byte	00h	
36h	ADC12MCTL11	ADC12_B Memory Control 11	Read/write	Word	0000h	Section 34.3.6
36h	ADC12MCTL11_L		Read/write	Byte	00h	
37h	ADC12MCTL11_H		Read/write	Byte	00h	
38h	ADC12MCTL12	ADC12_B Memory Control 12	Read/write	Word	0000h	Section 34.3.6
38h	ADC12MCTL12_L		Read/write	Byte	00h	
39h	ADC12MCTL12_H		Read/write	Byte	00h	
3Ah	ADC12MCTL13	ADC12_B Memory Control 13	Read/write	Word	0000h	Section 34.3.6
3Ah	ADC12MCTL13_L		Read/write	Byte	00h	
3Bh	ADC12MCTL13_H		Read/write	Byte	00h	

Table 34-3. ADC12_B Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
3Ch	ADC12MCTL14	ADC12_B Memory Control 14	Read/write	Word	0000h	Section 34.3.6
3Ch	ADC12MCTL14_L		Read/write	Byte	00h	
3Dh	ADC12MCTL14_H		Read/write	Byte	00h	
3Eh	ADC12MCTL15	ADC12_B Memory Control 15	Read/write	Word	0000h	Section 34.3.6
3Eh	ADC12MCTL15_L		Read/write	Byte	00h	
3Fh	ADC12MCTL15_H		Read/write	Byte	00h	
40h	ADC12MCTL16	ADC12_B Memory Control 16	Read/write	Word	0000h	Section 34.3.6
40h	ADC12MCTL16_L		Read/write	Byte	00h	
41h	ADC12MCTL16_H		Read/write	Byte	00h	
42h	ADC12MCTL17	ADC12_B Memory Control 17	Read/write	Word	0000h	Section 34.3.6
42h	ADC12MCTL17_L		Read/write	Byte	00h	
43h	ADC12MCTL17_H		Read/write	Byte	00h	
44h	ADC12MCTL18	ADC12_B Memory Control 18	Read/write	Word	0000h	Section 34.3.6
44h	ADC12MCTL18_L		Read/write	Byte	00h	
45h	ADC12MCTL18_H		Read/write	Byte	00h	
46h	ADC12MCTL19	ADC12_B Memory Control 19	Read/write	Word	0000h	Section 34.3.6
46h	ADC12MCTL19_L		Read/write	Byte	00h	
47h	ADC12MCTL19_H		Read/write	Byte	00h	
48h	ADC12MCTL20	ADC12_B Memory Control 20	Read/write	Word	0000h	Section 34.3.6
48h	ADC12MCTL20_L		Read/write	Byte	00h	
49h	ADC12MCTL20_H		Read/write	Byte	00h	
4Ah	ADC12MCTL21	ADC12_B Memory Control 21	Read/write	Word	0000h	Section 34.3.6
4Ah	ADC12MCTL21_L		Read/write	Byte	00h	
4Bh	ADC12MCTL21_H		Read/write	Byte	00h	
4Ch	ADC12MCTL22	ADC12_B Memory Control 22	Read/write	Word	0000h	Section 34.3.6
4Ch	ADC12MCTL22_L		Read/write	Byte	00h	
4Dh	ADC12MCTL22_H		Read/write	Byte	00h	
4Eh	ADC12MCTL23	ADC12_B Memory Control 23	Read/write	Word	0000h	Section 34.3.6
4Eh	ADC12MCTL23_L		Read/write	Byte	00h	
4Fh	ADC12MCTL23_H		Read/write	Byte	00h	
50h	ADC12MCTL24	ADC12_B Memory Control 24	Read/write	Word	0000h	Section 34.3.6
50h	ADC12MCTL24_L		Read/write	Byte	00h	
51h	ADC12MCTL24_H		Read/write	Byte	00h	
52h	ADC12MCTL25	ADC12_B Memory Control 25	Read/write	Word	0000h	Section 34.3.6
52h	ADC12MCTL25_L		Read/write	Byte	00h	
53h	ADC12MCTL25_H		Read/write	Byte	00h	
54h	ADC12MCTL26	ADC12_B Memory Control 26	Read/write	Word	0000h	Section 34.3.6
54h	ADC12MCTL26_L		Read/write	Byte	00h	
55h	ADC12MCTL26_H		Read/write	Byte	00h	
56h	ADC12MCTL27	ADC12_B Memory Control 27	Read/write	Word	0000h	Section 34.3.6
56h	ADC12MCTL27_L		Read/write	Byte	00h	
57h	ADC12MCTL27_H		Read/write	Byte	00h	
58h	ADC12MCTL28	ADC12_B Memory Control 28	Read/write	Word	0000h	Section 34.3.6
58h	ADC12MCTL28_L		Read/write	Byte	00h	
59h	ADC12MCTL28_H		Read/write	Byte	00h	

Table 34-3. ADC12_B Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
5Ah	ADC12MCTL29	ADC12_B Memory Control 29	Read/write	Word	0000h	Section 34.3.6
5Ah	ADC12MCTL29_L		Read/write	Byte	00h	
5Bh	ADC12MCTL29_H		Read/write	Byte	00h	
5Ch	ADC12MCTL30	ADC12_B Memory Control 30	Read/write	Word	0000h	Section 34.3.6
5Ch	ADC12MCTL30_L		Read/write	Byte	00h	
5Dh	ADC12MCTL30_H		Read/write	Byte	00h	
5Eh	ADC12MCTL31	ADC12_B Memory Control 31	Read/write	Word	0000h	Section 34.3.6
5Eh	ADC12MCTL31_L		Read/write	Byte	00h	
5Fh	ADC12MCTL31_H		Read/write	Byte	00h	
60h	ADC12MEM0	ADC12_B Memory 0	Read/write	Word	undefined	Section 34.3.5
60h	ADC12MEM0_L		Read/write	Byte	undefined	
61h	ADC12MEM0_H		Read/write	Byte	undefined	
62h	ADC12MEM1	ADC12_B Memory 1	Read/write	Word	undefined	Section 34.3.5
62h	ADC12MEM1_L		Read/write	Byte	undefined	
63h	ADC12MEM1_H		Read/write	Byte	undefined	
64h	ADC12MEM2	ADC12_B Memory 2	Read/write	Word	undefined	Section 34.3.5
64h	ADC12MEM2_L		Read/write	Byte	undefined	
65h	ADC12MEM2_H		Read/write	Byte	undefined	
66h	ADC12MEM3	ADC12_B Memory 3	Read/write	Word	undefined	Section 34.3.5
66h	ADC12MEM3_L		Read/write	Byte	undefined	
67h	ADC12MEM3_H		Read/write	Byte	undefined	
68h	ADC12MEM4	ADC12_B Memory 4	Read/write	Word	undefined	Section 34.3.5
68h	ADC12MEM4_L		Read/write	Byte	undefined	
69h	ADC12MEM4_H		Read/write	Byte	undefined	
6Ah	ADC12MEM5	ADC12_B Memory 5	Read/write	Word	undefined	Section 34.3.5
6Ah	ADC12MEM5_L		Read/write	Byte	undefined	
6Bh	ADC12MEM5_H		Read/write	Byte	undefined	
6Ch	ADC12MEM6	ADC12_B Memory 6	Read/write	Word	undefined	Section 34.3.5
6Ch	ADC12MEM6_L		Read/write	Byte	undefined	
6Dh	ADC12MEM6_H		Read/write	Byte	undefined	
6Eh	ADC12MEM7	ADC12_B Memory 7	Read/write	Word	undefined	Section 34.3.5
6Eh	ADC12MEM7_L		Read/write	Byte	undefined	
6Fh	ADC12MEM7_H		Read/write	Byte	undefined	
70h	ADC12MEM8	ADC12_B Memory 8	Read/write	Word	undefined	Section 34.3.5
70h	ADC12MEM8_L		Read/write	Byte	undefined	
71h	ADC12MEM8_H		Read/write	Byte	undefined	
72h	ADC12MEM9	ADC12_B Memory 9	Read/write	Word	undefined	Section 34.3.5
72h	ADC12MEM9_L		Read/write	Byte	undefined	
73h	ADC12MEM9_H		Read/write	Byte	undefined	
74h	ADC12MEM10	ADC12_B Memory 10	Read/write	Word	undefined	Section 34.3.5
74h	ADC12MEM10_L		Read/write	Byte	undefined	
75h	ADC12MEM10_H		Read/write	Byte	undefined	
76h	ADC12MEM11	ADC12_B Memory 11	Read/write	Word	undefined	Section 34.3.5
76h	ADC12MEM11_L		Read/write	Byte	undefined	
77h	ADC12MEM11_H		Read/write	Byte	undefined	

Table 34-3. ADC12_B Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
78h	ADC12MEM12	ADC12_B Memory 12	Read/write	Word	undefined	Section 34.3.5
78h	ADC12MEM12_L		Read/write	Byte	undefined	
79h	ADC12MEM12_H		Read/write	Byte	undefined	
7Ah	ADC12MEM13	ADC12_B Memory 13	Read/write	Word	undefined	Section 34.3.5
7Ah	ADC12MEM13_L		Read/write	Byte	undefined	
7Bh	ADC12MEM13_H		Read/write	Byte	undefined	
7Ch	ADC12MEM14	ADC12_B Memory 14	Read/write	Word	undefined	Section 34.3.5
7Ch	ADC12MEM14_L		Read/write	Byte	undefined	
7Dh	ADC12MEM14_H		Read/write	Byte	undefined	
7Eh	ADC12MEM15	ADC12_B Memory 15	Read/write	Word	undefined	Section 34.3.5
7Eh	ADC12MEM15_L		Read/write	Byte	undefined	
7Fh	ADC12MEM15_H		Read/write	Byte	undefined	
80h	ADC12MEM16	ADC12_B Memory 16	Read/write	Word	undefined	Section 34.3.5
80h	ADC12MEM16_L		Read/write	Byte	undefined	
81h	ADC12MEM16_H		Read/write	Byte	undefined	
82h	ADC12MEM17	ADC12_B Memory 17	Read/write	Word	undefined	Section 34.3.5
82h	ADC12MEM17_L		Read/write	Byte	undefined	
83h	ADC12MEM17_H		Read/write	Byte	undefined	
84h	ADC12MEM18	ADC12_B Memory 18	Read/write	Word	undefined	Section 34.3.5
84h	ADC12MEM18_L		Read/write	Byte	undefined	
85h	ADC12MEM18_H		Read/write	Byte	undefined	
86h	ADC12MEM19	ADC12_B Memory 19	Read/write	Word	undefined	Section 34.3.5
86h	ADC12MEM19_L		Read/write	Byte	undefined	
87h	ADC12MEM19_H		Read/write	Byte	undefined	
88h	ADC12MEM20	ADC12_B Memory 20	Read/write	Word	undefined	Section 34.3.5
88h	ADC12MEM20_L		Read/write	Byte	undefined	
89h	ADC12MEM20_H		Read/write	Byte	undefined	
8Ah	ADC12MEM21	ADC12_B Memory 21	Read/write	Word	undefined	Section 34.3.5
8Ah	ADC12MEM21_L		Read/write	Byte	undefined	
8Bh	ADC12MEM21_H		Read/write	Byte	undefined	
8Ch	ADC12MEM22	ADC12_B Memory 22	Read/write	Word	undefined	Section 34.3.5
8Ch	ADC12MEM22_L		Read/write	Byte	undefined	
8Dh	ADC12MEM22_H		Read/write	Byte	undefined	
8Eh	ADC12MEM23	ADC12_B Memory 23	Read/write	Word	undefined	Section 34.3.5
8Eh	ADC12MEM23_L		Read/write	Byte	undefined	
8Fh	ADC12MEM23_H		Read/write	Byte	undefined	
90h	ADC12MEM24	ADC12_B Memory 24	Read/write	Word	undefined	Section 34.3.5
90h	ADC12MEM24_L		Read/write	Byte	undefined	
91h	ADC12MEM24_H		Read/write	Byte	undefined	
92h	ADC12MEM25	ADC12_B Memory 25	Read/write	Word	undefined	Section 34.3.5
92h	ADC12MEM25_L		Read/write	Byte	undefined	
93h	ADC12MEM25_H		Read/write	Byte	undefined	
94h	ADC12MEM26	ADC12_B Memory 26	Read/write	Word	undefined	Section 34.3.5
94h	ADC12MEM26_L		Read/write	Byte	undefined	
95h	ADC12MEM26_H		Read/write	Byte	undefined	

Table 34-3. ADC12_B Registers (continued)

Offset	Acronym	Register Name	Type	Access	Reset	Section
96h	ADC12MEM27	ADC12_B Memory 27	Read/write	Word	undefined	Section 34.3.5
96h	ADC12MEM27_L		Read/write	Byte	undefined	
97h	ADC12MEM27_H		Read/write	Byte	undefined	
98h	ADC12MEM28	ADC12_B Memory 28	Read/write	Word	undefined	Section 34.3.5
98h	ADC12MEM28_L		Read/write	Byte	undefined	
99h	ADC12MEM28_H		Read/write	Byte	undefined	
9Ah	ADC12MEM29	ADC12_B Memory 29	Read/write	Word	undefined	Section 34.3.5
9Ah	ADC12MEM29_L		Read/write	Byte	undefined	
9Bh	ADC12MEM29_H		Read/write	Byte	undefined	
9Ch	ADC12MEM30	ADC12_B Memory 30	Read/write	Word	undefined	Section 34.3.5
9Ch	ADC12MEM30_L		Read/write	Byte	undefined	
9Dh	ADC12MEM30_H		Read/write	Byte	undefined	
9Eh	ADC12MEM31	ADC12_B Memory 31	Read/write	Word	undefined	Section 34.3.5
9Eh	ADC12MEM31_L		Read/write	Byte	undefined	
9Fh	ADC12MEM31_H		Read/write	Byte	undefined	

34.3.1 ADC12CTL0 Register (offset = 00h) [reset = 0000h]

ADC12_B Control 0 Register

Figure 34-14. ADC12CTL0 Register

15	14	13	12	11	10	9	8
ADC12SHT1x				ADC12SHT0x			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12MSC	Reserved		ADC12ON	Reserved		ADC12ENC	ADC12SC
rw-(0)	r-0	r-0	rw-(0)	r-0	r-0	rw-(0)	rw-(0)
Can be modified only when ADC12ENC = 0.							

Table 34-4. ADC12CTL0 Register Description

Bit	Field	Type	Reset	Description
15-12	ADC12SHT1x	RW	0	ADC12_B sample-and-hold time. These bits define the number of ADC12CLK cycles in the sampling period for registers ADC12MEM8 to ADC12MEM23. Can be modified only when ADC12ENC = 0. 0000b = 4 ADC12CLK cycles 0001b = 8 ADC12CLK cycles 0010b = 16 ADC12CLK cycles 0011b = 32 ADC12CLK cycles 0100b = 64 ADC12CLK cycles 0101b = 96 ADC12CLK cycles 0110b = 128 ADC12CLK cycles 0111b = 192 ADC12CLK cycles 1000b = 256 ADC12CLK cycles 1001b = 384 ADC12CLK cycles 1010b = 512 ADC12CLK cycles 1011b = Reserved 1100b = Reserved 1101b = Reserved 1110b = Reserved 1111b = Reserved
11-8	ADC12SHT0x	RW	0	ADC12_B sample-and-hold time. These bits define the number of ADC12CLK cycles in the sampling period for registers ADC12MEM0 to ADC12MEM7 and ADC12MEM24 to ADC12MEM31. Can be modified only when ADC12ENC = 0. 0000b = 4 ADC12CLK cycles 0001b = 8 ADC12CLK cycles 0010b = 16 ADC12CLK cycles 0011b = 32 ADC12CLK cycles 0100b = 64 ADC12CLK cycles 0101b = 96 ADC12CLK cycles 0110b = 128 ADC12CLK cycles 0111b = 192 ADC12CLK cycles 1000b = 256 ADC12CLK cycles 1001b = 384 ADC12CLK cycles 1010b = 512 ADC12CLK cycles 1011b = Reserved 1100b = Reserved 1101b = Reserved 1110b = Reserved 1111b = Reserved

Table 34-4. ADC12CTL0 Register Description (continued)

Bit	Field	Type	Reset	Description
7	ADC12MSC	RW	0	ADC12_B multiple sample and conversion. Valid only for sequence or repeated modes. Can be modified only when ADC12ENC = 0. 0b = The sampling timer requires a rising edge of the SHI signal to trigger each sample-and-convert. 1b = The incidence of the first rising edge of the SHI signal triggers the sampling timer, but further sample-and-conversions are performed automatically as soon as the prior conversion is completed.
6-5	Reserved	R	0	Reserved. Always reads as 0.
4	ADC12ON	RW	0	ADC12_B on. Can be modified only when ADC12ENC = 0. 0b = ADC12_B off 1b = ADC12_B on
3-2	Reserved	R	0	Reserved. Always reads as 0.
1	ADC12ENC	RW	0	ADC12_B enable conversion. 0b = ADC12_B disabled 1b = ADC12_B enabled
0	ADC12SC	RW	0	ADC12_B start conversion. Software-controlled sample-and-conversion start. ADC12SC and ADC12ENC may be set together with one instruction. ADC12SC is reset automatically. 0b = No sample-and-conversion-start 1b = Start sample-and-conversion

34.3.2 ADC12CTL1 Register (offset = 02h) [reset = 0000h]

ADC12_B Control 1 Register

Figure 34-15. ADC12CTL1 Register

15	14	13	12	11	10	9	8
Reserved	ADC12PDIV		ADC12SHSx			ADC12SHP	ADC12ISSH
r-0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12DIVx			ADC12SSELx		ADC12CONSEQx		ADC12BUSY
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)
Can be modified only when ADC12ENC = 0.							

Table 34-5. ADC12CTL1 Register Description

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved. Always reads as 0.
14-13	ADC12PDIV	RW	0h	ADC12_B predivider. This bit predivides the selected ADC12_B clock source. 00b = Predivide by 1 01b = Predivide by 4 10b = Predivide by 32 11b = Predivide by 64
12-10	ADC12SHSx	RW	0h	ADC12_B sample-and-hold source select 000b = ADC12SC bit 001b = see the device-specific data sheet for source 010b = see the device-specific data sheet for source 011b = see the device-specific data sheet for source 100b = see the device-specific data sheet for source 101b = see the device-specific data sheet for source 110b = see the device-specific data sheet for source 111b = see the device-specific data sheet for source
9	ADC12SHP	RW	0h	ADC12_B sample-and-hold pulse-mode select. This bit selects the source of the sampling signal (SAMPCON) to be either the output of the sampling timer or the sample-input signal directly. 0b = SAMPCON signal is sourced from the sample-input signal. 1b = SAMPCON signal is sourced from the sampling timer.
8	ADC12ISSH	RW	0h	ADC12_B invert signal sample-and-hold. 0b = The sample-input signal is not inverted. 1b = The sample-input signal is inverted.
7-5	ADC12DIVx	RW	0h	ADC12_B clock divider 000b = /1 001b = /2 010b = /3 011b = /4 100b = /5 101b = /6 110b = /7 111b = /8
4-3	ADC12SSELx	RW	0h	ADC12_B clock source select 00b = ADC12OSC (MODOSC) 01b = ACLK 10b = MCLK 11b = SMCLK

Table 34-5. ADC12CTL1 Register Description (continued)

Bit	Field	Type	Reset	Description
2-1	ADC12CONSEQx	RW	0h	ADC12_B conversion sequence mode select. This bit should only be modified when ADC12ENC = 0 except to stop a conversion immediately by setting ADC12CONSEQx = 00 when ADC12ENC = 1. 00b = Single-channel, single-conversion 01b = Sequence-of-channels 10b = Repeat-single-channel 11b = Repeat-sequence-of-channels
0	ADC12BUSY	R	0h	ADC12_B busy. This bit indicates an active sample or conversion operation. 0b = No operation is active. 1b = A sequence, sample, or conversion is active.

34.3.3 ADC12CTL2 Register (offset = 04h) [reset = 0020h]

ADC12_B Control 2 Register

Figure 34-16. ADC12CTL2 Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved		ADC12RES		ADC12DF	Reserved		ADC12PWRMD
r0	r0	rw-(1)		rw-(0)		r0	rw-(0)

Table 34-6. ADC12CTL2 Register Description

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved. Always reads as 0.
5-4	ADC12RES	RW	2h	ADC12_B resolution. This bit defines the conversion result resolution. This bit should only be modified when ADC12ENC=0. 00b = 8 bit (10 clock cycle conversion time) 01b = 10 bit (12 clock cycle conversion time) 10b = 12 bit (14 clock cycle conversion time) 11b = Reserved
3	ADC12DF	RW	0h	ADC12_B data read-back format. Data is always stored in the binary unsigned format. 0b = Binary unsigned. Theoretically for ADC12DIF = 0 and 12-bit mode the analog input voltage – VREF results in 0000h, the analog input voltage + VREF results in 0FFFh. 1b = Signed binary (2s complement), left aligned. Theoretically, for ADC12DIF = 0 and 12-bit mode, the analog input voltage – VREF results in 8000h, the analog input voltage + VREF results in 7FF0h.
2-1	Reserved	R	0h	Reserved. Always reads as 0.
0	ADC12PWRMD	RW	0h	Enables ADC low-power mode for ADC12CLK with 1/4 the specified maximum for ADC12PWRMD = 0. This bit should only be modified when ADC12ENC = 0. 0b = Regular power mode where sample rate is not restricted 1b = Low power mode enable, ADC12CLK can not be greater than 1/4 the device-specific data sheet specified maximum for ADC12PWRMD = 0

34.3.4 ADC12CTL3 Register (offset = 06h) [reset = 0000h]

ADC12_B Control 3 Register

Figure 34-17. ADC12CTL3 Register

15	14	13	12	11	10	9	8
Reserved				ADC12ICH3MAP	ADC12ICH2MAP	ADC12ICH1MAP	ADC12ICH0MAP
r0	r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12TCMAP	ADC12BATMAP	Reserved	ADC12CSTARTADDx				
rw-(0)	rw-(0)	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
Can be modified only when ADC12ENC = 0.							

Table 34-7. ADC12CTL3 Register Description

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved. Always reads as 0.
11	ADC12ICH3MAP	RW	0h	Controls internal channel 3 selection to ADC input channel A26. Can be modified only when ADC12ENC = 0. 0b = external pin is selected for ADC input channel A26 1b = ADC input channel internal 3 is selected for ADC input channel A26, see device-specific data sheet for availability
10	ADC12ICH2MAP	RW	0h	Controls internal channel 2 selection to ADC input channel A27. Can be modified only when ADC12ENC = 0. 0b = external pin is selected for ADC input channel A27 1b = ADC input channel internal 2 is selected for ADC input channel A27, see device-specific data sheet for availability
9	ADC12ICH1MAP	RW	0h	Controls internal channel 1 selection to ADC input channel A28. Can be modified only when ADC12ENC = 0. 0b = external pin is selected for ADC input channel A28 1b = ADC input channel internal 1 is selected for ADC input channel A28, see device-specific data sheet for availability
8	ADC12ICH0MAP	RW	0h	Controls internal channel 0 selection to ADC input channel A29. Can be modified only when ADC12ENC = 0. 0b = external pin is selected for ADC input channel A29 1b = ADC input channel internal 0 is selected for ADC input channel A29, see device-specific data sheet for availability
7	ADC12TCMAP	RW	0h	Controls temperature sensor ADC input channel selection. Can be modified only when ADC12ENC = 0. 0b = external pin is selected for ADC input channel A30 1b = ADC internal temperature sensor channel is selected for ADC input channel A30
6	ADC12BATMAP	RW	0h	Controls 1/2 AVCC ADC input channel selection. Can be modified only when ADC12ENC = 0. 0b = external pin is selected for ADC input channel A31 1b = ADC internal 1/2 x AVCC channel is selected for ADC input channel A31
5	Reserved	R	0h	Reserved. Always reads as 0.
4-0	ADC12CSTARTADDx	RW	0h	ADC12_B conversion start address. These bits select which ADC12_B conversion memory register is used for a single conversion or for the first conversion in a sequence. The value of CSTARTADDx is 0h to 1Fh, corresponding to ADC12MEM0 to ADC12MEM31. Can be modified only when ADC12ENC = 0.

34.3.5 ADC12MEMx Register (x = 0 to 31)

ADC12_B Conversion Memory x Register (x = 0 to 31)

Figure 34-18. ADC12MEMx Register

15	14	13	12	11	10	9	8
Conversion Results							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
Conversion Results							
rw	rw	rw	rw	rw	rw	rw	rw

Table 34-8. ADC12MEMx Register Description

Bit	Field	Type	Reset	Description
15-0	Conversion Results	RW	undefined	If ADC12DF = 0: The 12-bit conversion results are right justified. Bit 11 is the MSB. Bits 15-12 are 0 in 12-bit mode, bits 15-10 are 0 in 10-bit mode, and bits 15-8 are 0 in 8-bit mode. If the user writes to the conversion memory registers, the results are corrupted. If ADC12DF = 1: The 12-bit conversion results are left-justified 2s-complement format. Bit 15 is the MSB. Bits 3-0 are 0 in 12-bit mode, bits 5-0 are 0 in 10-bit mode, and bits 7-0 are 0 in 8-bit mode. The data is stored in the right-justified format and is converted to the left-justified 2s-complement format during read back. If the user writes to the conversion memory registers, the results are corrupted.

34.3.6 ADC12MCTLx Register (x = 0 to 31)

ADC12_B Conversion Memory Control x Register (x = 0 to 31)

Figure 34-19. ADC12MCTLx Register

15	14	13	12	11	10	9	8
Reserved	ADC12WINC	ADC12DIF	Reserved	ADC12VRSEL			
r0	rw-(0)	rw-(0)	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12EOS	Reserved		ADC12INCHx				
rw-(0)	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
Can be modified only when ADC12ENC = 0.							

Table 34-9. ADC12MCTLx Register Description

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved. Always reads as 0.
14	ADC12WINC	RW	0h	Comparator window enable. Can be modified only when ADC12ENC = 0. 0b = Comparator window disabled 1b = Comparator window enabled
13	ADC12DIF	RW	0h	Differential mode. Can be modified only when ADC12ENC = 0. 0b = Single-ended mode enabled 1b = Differential mode enabled
12	Reserved	R	0h	Reserved. Always reads as 0.
11-8	ADC12VRSEL	RW	0h	Selects combinations of VR+ and VR- sources as well as the buffer selection. Note: there is only one buffer so it can be used for either VR+ or VR-, but not both. Can be modified only when ADC12ENC = 0. 0000b = VR+ = AVCC, VR- = AVSS 0001b = VR+ = VREF buffered, VR- = AVSS 0010b = VR+ = VeREF-, VR- = AVSS 0011b = VR+ = VeREF+ buffered, VR- = AVSS 0100b = VR+ = VeREF+, VR- = AVSS 0101b = VR+ = AVCC, VR- = VeREF+ buffered 0110b = VR+ = AVCC, VR- = VeREF+ 0111b = VR+ = VREF buffered, VR- = VeREF+ 1000b = Reserved 1001b = VR+ = AVCC, VR- = VREF buffered 1010b = Reserved 1011b = VR+ = VeREF+, VR- = VREF buffered 1100b = VR+ = AVCC, VR- = VeREF- 1101b = VR+ = VREF buffered, VR- = VeREF- 1110b = VR+ = VeREF+, VR- = VeREF- 1111b = VR+ = VeREF+ buffered, VR- = VeREF-
7	ADC12EOS	RW	0h	End of sequence. Indicates the last conversion in a sequence. Can be modified only when ADC12ENC = 0. 0b = Not end of sequence 1b = End of sequence
6-5	Reserved	R	0h	Reserved. Always reads as 0.

Table 34-9. ADC12MCTLx Register Description (continued)

Bit	Field	Type	Reset	Description
4-0	ADC12INCHx	RW	0h	Input channel select. If even channels are set as differential, then odd channel configuration is ignored. Can be modified only when ADC12ENC = 0. 00000b = If ADC12DIF = 0: A0; If ADC12DIF = 1: Ain+ = A0, Ain- = A1 00001b = If ADC12DIF = 0: A1; If ADC12DIF = 1: Ain+ = A0, Ain- = A1 00010b = If ADC12DIF = 0: A2; If ADC12DIF = 1: Ain+ = A2, Ain- = A3 00011b = If ADC12DIF = 0: A3; If ADC12DIF = 1: Ain+ = A2, Ain- = A3 00100b = If ADC12DIF = 0: A4; If ADC12DIF = 1: Ain+ = A4, Ain- = A5 00101b = If ADC12DIF = 0: A5; If ADC12DIF = 1: Ain+ = A4, Ain- = A5 00110b = If ADC12DIF = 0: A6; If ADC12DIF = 1: Ain+ = A6, Ain- = A7 00111b = If ADC12DIF = 0: A7; If ADC12DIF = 1: Ain+ = A6, Ain- = A7 01000b = If ADC12DIF = 0: A8; If ADC12DIF = 1: Ain+ = A8, Ain- = A9 01001b = If ADC12DIF = 0: A9; If ADC12DIF = 1: Ain+ = A8, Ain- = A9 01010b = If ADC12DIF = 0: A10; If ADC12DIF = 1: Ain+ = A10, Ain- = A11 01011b = If ADC12DIF = 0: A11; If ADC12DIF = 1: Ain+ = A10, Ain- = A11 01100b = If ADC12DIF = 0: A12; If ADC12DIF = 1: Ain+ = A12, Ain- = A13 01101b = If ADC12DIF = 0: A13; If ADC12DIF = 1: Ain+ = A12, Ain- = A13 01110b = If ADC12DIF = 0: A14; If ADC12DIF = 1: Ain+ = A14, Ain- = A15 01111b = If ADC12DIF = 0: A15; If ADC12DIF = 1: Ain+ = A14, Ain- = A15 10000b = If ADC12DIF = 0: A16; If ADC12DIF = 1: Ain+ = A16, Ain- = A17 10001b = If ADC12DIF = 0: A17; If ADC12DIF = 1: Ain+ = A16, Ain- = A17 10010b = If ADC12DIF = 0: A18; If ADC12DIF = 1: Ain+ = A18, Ain- = A19 10011b = If ADC12DIF = 0: A19; If ADC12DIF = 1: Ain+ = A18, Ain- = A19 10100b = If ADC12DIF = 0: A20; If ADC12DIF = 1: Ain+ = A20, Ain- = A21 10101b = If ADC12DIF = 0: A21; If ADC12DIF = 1: Ain+ = A20, Ain- = A21 10110b = If ADC12DIF = 0: A22; If ADC12DIF = 1: Ain+ = A22, Ain- = A23 10111b = If ADC12DIF = 0: A23; If ADC12DIF = 1: Ain+ = A22, Ain- = A23 11000b = If ADC12DIF = 0: A24; If ADC12DIF = 1: Ain+ = A24, Ain- = A25 11001b = If ADC12DIF = 0: A25; If ADC12DIF = 1: Ain+ = A24, Ain- = A25 11010b = If ADC12DIF = 0: A26; If ADC12DIF = 1: Ain+ = A26, Ain- = A27 11011b = If ADC12DIF = 0: A27; If ADC12DIF = 1: Ain+ = A26, Ain- = A27 11100b = If ADC12DIF = 0: A28; If ADC12DIF = 1: Ain+ = A28, Ain- = A29 11101b = If ADC12DIF = 0: A29; If ADC12DIF = 1: Ain+ = A28, Ain- = A29 11110b = If ADC12DIF = 0: A30; If ADC12DIF = 1: Ain+ = A30, Ain- = A31 11111b = If ADC12DIF = 0: A31; If ADC12DIF = 1: Ain+ = A30, Ain- = A31

34.3.7 ADC12HI Register (offset = 0Ah) [reset = 0FFFh]

ADC12_B Window Comparator High Threshold Register

Figure 34-20. ADC12HI Register

15	14	13	12	11	10	9	8
High Threshold							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(1)	rw-(1)	rw-(1)	rw-(1)
7	6	5	4	3	2	1	0
High Threshold							
rw-(1)	rw-(1)	rw-(1)	rw-(1)	rw-(1)	rw-(1)	rw-(1)	rw-(1)

Table 34-10. ADC12HI Register Description

Bit	Field	Type	Reset	Description
15-0	High Threshold	RW	0FFFh	Window comparator high threshold should only be modified when ADC12ENC=0. If ADC12DF = 0: The 12-bit threshold value is right justified when ADC12DF = 0. Bits 15-12 are 0. Bit 11 is the MSB. Bits 11-10 are 0 in 10-bit mode, and bits 11-8 are 0 in 8-bit mode. If ADC12DF = 1: The 12-bit threshold value is left justified when ADC12DF = 1, 2s-complement format. Bit 15 is the MSB. Bits 3-0 are 0 in 12-bit mode, bits 5-0 are 0 in 10-bit mode, and bits 7-0 are 0 in 8-bit mode.

34.3.8 ADC12LO Register (offset = 08h) [reset = 0000h]

ADC12_B Window Comparator Low Threshold Register

Figure 34-21. ADC12LO Register

15	14	13	12	11	10	9	8
Low Threshold							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
Low Threshold							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 34-11. ADC12LO Register Description

Bit	Field	Type	Reset	Description
15-0	Low Threshold	RW	0h	Window comparator low threshold should only be modified when ADC12ENC=0. If ADC12DF = 0: The 12-bit threshold value is right justified when ADC12DF = 0. Bits 15-12 are 0. Bit 11 is the MSB. Bits 11-10 are 0 in 10-bit mode, and bits 11-8 are 0 in 8-bit mode. If ADC12DF = 1: The 12-bit threshold value is left justified when ADC12DF = 1, 2s-complement format. Bit 15 is the MSB. Bits 3-0 are 0 in 12-bit mode, bits 5-0 are 0 in 10-bit mode, and bits 7-0 are 0 in 8-bit mode.

34.3.9 ADC12IER0 Register (offset = 12h) [reset = 0000h]

ADC12_B Interrupt Enable 0 Register

Figure 34-22. ADC12IER0 Register

15	14	13	12	11	10	9	8
ADC12IE15	ADC12IE14	ADC12IE13	ADC12IE12	ADC12IE11	ADC12IE10	ADC12IE9	ADC12IE8
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12IE7	ADC12IE6	ADC12IE5	ADC12IE4	ADC12IE3	ADC12IE2	ADC12IE1	ADC12IE0
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 34-12. ADC12IER0 Register Description

Bit	Field	Type	Reset	Description
15	ADC12IE15	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG15 bit. 0b = Interrupt disabled 1b = Interrupt enabled
14	ADC12IE14	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG14 bit. 0b = Interrupt disabled 1b = Interrupt enabled
13	ADC12IE13	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG13 bit. 0b = Interrupt disabled 1b = Interrupt enabled
12	ADC12IE12	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG12 bit. 0b = Interrupt disabled 1b = Interrupt enabled
11	ADC12IE11	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG11 bit. 0b = Interrupt disabled 1b = Interrupt enabled
10	ADC12IE10	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG10 bit. 0b = Interrupt disabled 1b = Interrupt enabled
9	ADC12IE9	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG9 bit. 0b = Interrupt disabled 1b = Interrupt enabled
8	ADC12IE8	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG8 bit. 0b = Interrupt disabled 1b = Interrupt enabled
7	ADC12IE7	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG7 bit. 0b = Interrupt disabled 1b = Interrupt enabled
6	ADC12IE6	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG6 bit. 0b = Interrupt disabled 1b = Interrupt enabled
5	ADC12IE5	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG5 bit. 0b = Interrupt disabled 1b = Interrupt enabled
4	ADC12IE4	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG4 bit. 0b = Interrupt disabled 1b = Interrupt enabled
3	ADC12IE3	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG3 bit. 0b = Interrupt disabled 1b = Interrupt enabled

Table 34-12. ADC12IER0 Register Description (continued)

Bit	Field	Type	Reset	Description
2	ADC12IE2	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG2 bit. 0b = Interrupt disabled 1b = Interrupt enabled
1	ADC12IE1	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG1 bit. 0b = Interrupt disabled 1b = Interrupt enabled
0	ADC12IE0	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG0 bit. 0b = Interrupt disabled 1b = Interrupt enabled

34.3.10 ADC12IER1 Register (offset = 14h) [reset = 0000h]

ADC12_B Interrupt Enable 1 Register

Figure 34-23. ADC12IER1 Register

15	14	13	12	11	10	9	8
ADC12IE31	ADC12IE30	ADC12IE29	ADC12IE28	ADC12IE27	ADC12IE26	ADC12IE25	ADC12IE24
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12IE23	ADC12IE22	ADC12IE21	ADC12IE20	ADC12IE19	ADC12IE18	ADC12IE17	ADC12IE16
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 34-13. ADC12IER1 Register Description

Bit	Field	Type	Reset	Description
15	ADC12IE31	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG31 bit. 0b = Interrupt disabled 1b = Interrupt enabled
14	ADC12IE30	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG30 bit. 0b = Interrupt disabled 1b = Interrupt enabled
13	ADC12IE29	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG29 bit. 0b = Interrupt disabled 1b = Interrupt enabled
12	ADC12IE28	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG28 bit. 0b = Interrupt disabled 1b = Interrupt enabled
11	ADC12IE27	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG27 bit. 0b = Interrupt disabled 1b = Interrupt enabled
10	ADC12IE26	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG26 bit. 0b = Interrupt disabled 1b = Interrupt enabled
9	ADC12IE25	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG25 bit. 0b = Interrupt disabled 1b = Interrupt enabled
8	ADC12IE24	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG24 bit. 0b = Interrupt disabled 1b = Interrupt enabled
7	ADC12IE23	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG23 bit. 0b = Interrupt disabled 1b = Interrupt enabled
6	ADC12IE22	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG22 bit. 0b = Interrupt disabled 1b = Interrupt enabled
5	ADC12IE21	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG21 bit. 0b = Interrupt disabled 1b = Interrupt enabled
4	ADC12IE20	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG20 bit. 0b = Interrupt disabled 1b = Interrupt enabled
3	ADC12IE19	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG19 bit. 0b = Interrupt disabled 1b = Interrupt enabled

Table 34-13. ADC12IER1 Register Description (continued)

Bit	Field	Type	Reset	Description
2	ADC12IE18	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG18 bit. 0b = Interrupt disabled 1b = Interrupt enabled
1	ADC12IE17	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG17 bit. 0b = Interrupt disabled 1b = Interrupt enabled
0	ADC12IE16	RW	0h	Interrupt enable. Enables or disables the interrupt request for ADC12IFG16 bit. 0b = Interrupt disabled 1b = Interrupt enabled

34.3.11 ADC12IER2 Register (offset = 16h) [reset = 0000h]

ADC12_B Interrupt Enable 2 Register

Figure 34-24. ADC12IER2 Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved	ADC12RDYIE	ADC12TOVIE	ADC12OVIE	ADC12HIIE	ADC12LOIE	ADC12INIE	Reserved
r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0

Table 34-14. ADC12IER2 Register Description

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0h	Reserved. Always reads as 0.
6	ADC12RDYIE	RW	0h	ADC12_B local reference buffer ready interrupt enable. The GIE bit must also be set to enable the interrupt. 0b = Interrupt disabled 1b = Interrupt enabled
5	ADC12TOVIE	RW	0h	ADC12_B conversion-time-overflow interrupt enable. The GIE bit must also be set to enable the interrupt. 0b = Interrupt disabled 1b = Interrupt enabled
4	ADC12OVIE	RW	0h	ADC12MEMx overflow-interrupt enable. The GIE bit must also be set to enable the interrupt. 0b = Interrupt disabled 1b = Interrupt enabled
3	ADC12HIIE	RW	0h	Interrupt enable for the exceeding the upper limit interrupt of the window comparator for ADC12MEMx result register. The GIE bit must also be set to enable the interrupt. 0b = Interrupt disabled 1b = Interrupt enabled
2	ADC12LOIE	RW	0h	Interrupt enable for the falling short of the lower limit interrupt of the window comparator for the ADC12MEMx result register. The GIE bit must also be set to enable the interrupt. 0b = Interrupt disabled 1b = Interrupt enabled
1	ADC12INIE	RW	0h	Interrupt enable for the ADC12MEMx result register being greater than the ADC12LO threshold and below the ADC12HI threshold. The GIE bit must also be set to enable the interrupt. 0b = Interrupt disabled 1b = Interrupt enabled
0	Reserved	R	0h	Reserved. Always reads as 0.

34.3.12 ADC12IFGR0 Register (offset = 0Ch) [reset = 0000h]

ADC12_B Interrupt Flag 0 Register

Figure 34-25. ADC12IFGR0 Register

15	14	13	12	11	10	9	8
ADC12IFG15	ADC12IFG14	ADC12IFG13	ADC12IFG12	ADC12IFG11	ADC12IFG10	ADC12IFG9	ADC12IFG8
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12IFG7	ADC12IFG6	ADC12IFG5	ADC12IFG4	ADC12IFG3	ADC12IFG2	ADC12IFG1	ADC12IFG0
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 34-15. ADC12IFGR0 Register Description

Bit	Field	Type	Reset	Description
15	ADC12IFG15	RW	0h	ADC12MEM15 interrupt flag. This bit is set when ADC12MEM15 is loaded with a conversion result. The ADC12IFG15 bit is reset if ADC12MEM15 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
14	ADC12IFG14	RW	0h	ADC12MEM14 interrupt flag. This bit is set when ADC12MEM14 is loaded with a conversion result. The ADC12IFG14 bit is reset if ADC12MEM14 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
13	ADC12IFG13	RW	0h	ADC12MEM13 interrupt flag. This bit is set when ADC12MEM13 is loaded with a conversion result. The ADC12IFG13 bit is reset if ADC12MEM13 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
12	ADC12IFG12	RW	0h	ADC12MEM12 interrupt flag. This bit is set when ADC12MEM12 is loaded with a conversion result. The ADC12IFG12 bit is reset if ADC12MEM12 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
11	ADC12IFG11	RW	0h	ADC12MEM11 interrupt flag. This bit is set when ADC12MEM11 is loaded with a conversion result. The ADC12IFG11 bit is reset if ADC12MEM11 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
10	ADC12IFG10	RW	0h	ADC12MEM10 interrupt flag. This bit is set when ADC12MEM10 is loaded with a conversion result. The ADC12IFG10 bit is reset if ADC12MEM10 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
9	ADC12IFG9	RW	0h	ADC12MEM9 interrupt flag. This bit is set when ADC12MEM9 is loaded with a conversion result. The ADC12IFG9 bit is reset if ADC12MEM9 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
8	ADC12IFG8	RW	0h	ADC12MEM8 interrupt flag. This bit is set when ADC12MEM8 is loaded with a conversion result. The ADC12IFG8 bit is reset if ADC12MEM8 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending

Table 34-15. ADC12IFGR0 Register Description (continued)

Bit	Field	Type	Reset	Description
7	ADC12IFG7	RW	0h	ADC12MEM7 interrupt flag. This bit is set when ADC12MEM7 is loaded with a conversion result. The ADC12IFG7 bit is reset if ADC12MEM7 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
6	ADC12IFG6	RW	0h	ADC12MEM6 interrupt flag. This bit is set when ADC12MEM6 is loaded with a conversion result. The ADC12IFG6 bit is reset if ADC12MEM6 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
5	ADC12IFG5	RW	0h	ADC12MEM5 interrupt flag. This bit is set when ADC12MEM5 is loaded with a conversion result. The ADC12IFG5 bit is reset if ADC12MEM5 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
4	ADC12IFG4	RW	0h	ADC12MEM4 interrupt flag. This bit is set when ADC12MEM4 is loaded with a conversion result. The ADC12IFG4 bit is reset if ADC12MEM4 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
3	ADC12IFG3	RW	0h	ADC12MEM3 interrupt flag. This bit is set when ADC12MEM3 is loaded with a conversion result. The ADC12IFG3 bit is reset if ADC12MEM3 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
2	ADC12IFG2	RW	0h	ADC12MEM2 interrupt flag. This bit is set when ADC12MEM2 is loaded with a conversion result. The ADC12IFG2 bit is reset if ADC12MEM2 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
1	ADC12IFG1	RW	0h	ADC12MEM1 interrupt flag. This bit is set when ADC12MEM1 is loaded with a conversion result. The ADC12IFG1 bit is reset if ADC12MEM1 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
0	ADC12IFG0	RW	0h	ADC12MEM0 interrupt flag. This bit is set when ADC12MEM0 is loaded with a conversion result. The ADC12IFG0 bit is reset if ADC12MEM0 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending

34.3.13 ADC12IFGR1 Register (offset = 0Eh) [reset = 0000h]

ADC12_B Interrupt Flag 1 Register

Figure 34-26. ADC12IFGR1 Register

15	14	13	12	11	10	9	8
ADC12IFG31	ADC12IFG30	ADC12IFG29	ADC12IFG28	ADC12IFG27	ADC12IFG26	ADC12IFG25	ADC12IFG24
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12IFG23	ADC12IFG22	ADC12IFG21	ADC12IFG20	ADC12IFG19	ADC12IFG18	ADC12IFG17	ADC12IFG16
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 34-16. ADC12IFGR1 Register Description

Bit	Field	Type	Reset	Description
15	ADC12IFG31	RW	0h	ADC12MEM31 interrupt flag. This bit is set when ADC12MEM31 is loaded with a conversion result. The ADC12IFG31 bit is reset if ADC12MEM31 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
14	ADC12IFG30	RW	0h	ADC12MEM30 interrupt flag. This bit is set when ADC12MEM30 is loaded with a conversion result. The ADC12IFG30 bit is reset if ADC12MEM30 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
13	ADC12IFG29	RW	0h	ADC12MEM29 interrupt flag. This bit is set when ADC12MEM29 is loaded with a conversion result. The ADC12IFG29 bit is reset if ADC12MEM29 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
12	ADC12IFG28	RW	0h	ADC12MEM28 interrupt flag. This bit is set when ADC12MEM28 is loaded with a conversion result. The ADC12IFG28 bit is reset if ADC12MEM28 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
11	ADC12IFG27	RW	0h	ADC12MEM27 interrupt flag. This bit is set when ADC12MEM27 is loaded with a conversion result. The ADC12IFG27 bit is reset if ADC12MEM27 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
10	ADC12IFG26	RW	0h	ADC12MEM26 interrupt flag. This bit is set when ADC12MEM26 is loaded with a conversion result. The ADC12IFG26 bit is reset if ADC12MEM26 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
9	ADC12IFG25	RW	0h	ADC12MEM25 interrupt flag. This bit is set when ADC12MEM25 is loaded with a conversion result. The ADC12IFG25 bit is reset if ADC12MEM25 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
8	ADC12IFG24	RW	0h	ADC12MEM24 interrupt flag. This bit is set when ADC12MEM24 is loaded with a conversion result. The ADC12IFG24 bit is reset if ADC12MEM24 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending

Table 34-16. ADC12IFGR1 Register Description (continued)

Bit	Field	Type	Reset	Description
7	ADC12IFG23	RW	0h	ADC12MEM23 interrupt flag. This bit is set when ADC12MEM23 is loaded with a conversion result. The ADC12IFG23 bit is reset if ADC12MEM23 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
6	ADC12IFG22	RW	0h	ADC12MEM22 interrupt flag. This bit is set when ADC12MEM22 is loaded with a conversion result. The ADC12IFG22 bit is reset if ADC12MEM22 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
5	ADC12IFG21	RW	0h	ADC12MEM21 interrupt flag. This bit is set when ADC12MEM21 is loaded with a conversion result. The ADC12IFG21 bit is reset if ADC12MEM21 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
4	ADC12IFG20	RW	0h	ADC12MEM20 interrupt flag. This bit is set when ADC12MEM20 is loaded with a conversion result. The ADC12IFG20 bit is reset if ADC12MEM20 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
3	ADC12IFG19	RW	0h	ADC12MEM19 interrupt flag. This bit is set when ADC12MEM19 is loaded with a conversion result. The ADC12IFG19 bit is reset if ADC12MEM19 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
2	ADC12IFG18	RW	0h	ADC12MEM18 interrupt flag. This bit is set when ADC12MEM18 is loaded with a conversion result. The ADC12IFG18 bit is reset if ADC12MEM18 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
1	ADC12IFG17	RW	0h	ADC12MEM17 interrupt flag. This bit is set when ADC12MEM17 is loaded with a conversion result. The ADC12IFG17 bit is reset if ADC12MEM17 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending
0	ADC12IFG16	RW	0h	ADC12MEM16 interrupt flag. This bit is set when ADC12MEM16 is loaded with a conversion result. The ADC12IFG16 bit is reset if ADC12MEM16 is accessed, or it can be reset with software. 0b = No interrupt pending 1b = Interrupt pending

34.3.14 ADC12IFGR2 Register (offset = 10h) [reset = 0000h]

ADC12_B Interrupt Flag 2 Register

Figure 34-27. ADC12IFGR2 Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved	ADC12RDYIFG	ADC12TOVIFG	ADC12OVIFG	ADC12HIIFG	ADC12LOIFG	ADC12INIFG	Reserved
r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0

Table 34-17. ADC12IFGR2 Register Description

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0h	Reserved. Always reads as 0.
6	ADC12RDYIFG	RW	0h	ADC12_B local reference buffer ready interrupt flag. The flag does not occur if the sample trigger has not been asserted. 0b = No interrupt pending 1b = Interrupt pending
5	ADC12TOVIFG	RW	0h	ADC12_B conversion-time-overflow interrupt flag. 0b = No interrupt pending 1b = Interrupt pending
4	ADC12OVIFG	RW	0h	ADC12MEMx overflow-interrupt flag. 0b = No interrupt pending 1b = Interrupt pending
3	ADC12HIIFG	RW	0h	Interrupt flag for exceeding the upper limit interrupt of the window comparator for ADC12MEMx result register. 0b = No interrupt pending 1b = Interrupt pending
2	ADC12LOIFG	RW	0h	Interrupt flag for falling short of the lower limit interrupt of the window comparator for the ADC12MEMx result register. 0b = No interrupt pending 1b = Interrupt pending
1	ADC12INIFG	RW	0h	Interrupt flag for the ADC12MEMx result register being greater than the ADC12LO threshold and below the ADC12HI threshold interrupt. 0b = No interrupt pending 1b = Interrupt pending
0	Reserved	R	0h	Reserved. Always reads as 0.

34.3.15 ADC12IV Register (offset = 18h) [reset = 0000h]

ADC12_B Interrupt Vector

Figure 34-28. ADC12IV Register

15	14	13	12	11	10	9	8
ADC12IVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
ADC12IVx							
r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0

Table 34-18. ADC12IV Register Description

Bit	Field	Type	Reset	Description
15-0	ADC12IVx	RW	0h	ADC12_B interrupt vector value. Writing to this register clears all pending interrupt flags. 000h = Interrupt Source: No interrupt pending, Interrupt Flag: None 002h = Interrupt Source: ADC12MEMx overflow, Interrupt Flag: ADC12OVIFG, Interrupt Priority: Highest 004h = Interrupt Source: Conversion time overflow, Interrupt Flag: ADC12TOVIFG 006h = Interrupt Source: ADC12 window high interrupt flag, Interrupt Flag: ADC12HIIFG 008h = Interrupt Source: ADC12 window low interrupt flag, Interrupt Flag: ADC12LOIFG 00Ah = Interrupt Source: ADC12 in-window interrupt flag, Interrupt Flag: ADC12INIFG 00Ch = Interrupt Source: ADC12MEM0 interrupt flag, Interrupt Flag: ADC12IFG0 00Eh = Interrupt Source: ADC12MEM1 interrupt flag, Interrupt Flag: ADC12IFG1 010h = Interrupt Source: ADC12MEM2 interrupt flag, Interrupt Flag: ADC12IFG2 012h = Interrupt Source: ADC12MEM3 interrupt flag, Interrupt Flag: ADC12IFG3 014h = Interrupt Source: ADC12MEM4 interrupt flag, Interrupt Flag: ADC12IFG4 016h = Interrupt Source: ADC12MEM5 interrupt flag, Interrupt Flag: ADC12IFG5 018h = Interrupt Source: ADC12MEM6 interrupt flag, Interrupt Flag: ADC12IFG6 01Ah = Interrupt Source: ADC12MEM7 interrupt flag, Interrupt Flag: ADC12IFG7 01Ch = Interrupt Source: ADC12MEM8 interrupt flag, Interrupt Flag: ADC12IFG8 01Eh = Interrupt Source: ADC12MEM9 interrupt flag, Interrupt Flag: ADC12IFG9 020h = Interrupt Source: ADC12MEM10 interrupt flag, Interrupt Flag: ADC12IFG10 022h = Interrupt Source: ADC12MEM11 interrupt flag, Interrupt Flag: ADC12IFG11 024h = Interrupt Source: ADC12MEM12 interrupt flag, Interrupt Flag: ADC12IFG12 026h = Interrupt Source: ADC12MEM13 interrupt flag, Interrupt Flag: ADC12IFG13 028h = Interrupt Source: ADC12MEM14 interrupt flag, Interrupt Flag: ADC12IFG14 02Ah = Interrupt Source: ADC12MEM15 interrupt flag, Interrupt Flag: ADC12IFG15 02Ch = Interrupt Source: ADC12MEM16 interrupt flag, Interrupt Flag: ADC12IFG16 02Eh = Interrupt Source: ADC12MEM17 interrupt flag, Interrupt Flag: ADC12IFG17 030h = Interrupt Source: ADC12MEM18 interrupt flag, Interrupt Flag: ADC12IFG18

Table 34-18. ADC12IV Register Description (continued)

Bit	Field	Type	Reset	Description
				032h = Interrupt Source: ADC12MEM19 interrupt flag, Interrupt Flag: ADC12IFG19 034h = Interrupt Source: ADC12MEM20 interrupt flag, Interrupt Flag: ADC12IFG20 036h = Interrupt Source: ADC12MEM21 interrupt flag, Interrupt Flag: ADC12IFG21 038h = Interrupt Source: ADC12MEM22 interrupt flag, Interrupt Flag: ADC12IFG22 03Ah = Interrupt Source: ADC12MEM23 interrupt flag, Interrupt Flag: ADC12IFG23 03Ch = Interrupt Source: ADC12MEM24 interrupt flag, Interrupt Flag: ADC12IFG24 03Eh = Interrupt Source: ADC12MEM25 interrupt flag, Interrupt Flag: ADC12IFG25 040h = Interrupt Source: ADC12MEM26 interrupt flag, Interrupt Flag: ADC12IFG26 042h = Interrupt Source: ADC12MEM27 interrupt flag, Interrupt Flag: ADC12IFG27 044h = Interrupt Source: ADC12MEM28 interrupt flag, Interrupt Flag: ADC12IFG28 046h = Interrupt Source: ADC12MEM29 interrupt flag, Interrupt Flag: ADC12IFG29 048h = Interrupt Source: ADC12MEM30 interrupt flag, Interrupt Flag: ADC12IFG30 04Ah = Interrupt Source: ADC12MEM31 interrupt flag, Interrupt Flag: ADC12IFG31 04Ch = Interrupt Source: ADC12RDYIFG interrupt flag, Interrupt Flag: ADC12RDYIFG

Comparator E (COMP_E) Module

Comparator_E is an analog voltage comparator. This chapter describes the Comparator_E. Comparator_E supports general comparator functionality for up to 16 channels.

Topic	Page
35.1 COMP_E Introduction.....	916
35.2 COMP_E Operation	917
35.3 COMP_E Registers	923

35.1 COMP_E Introduction

The COMP_E module supports precision slope analog-to-digital conversions, supply voltage supervision, and monitoring of external analog signals.

Features of COMP_E include:

- Inverting and noninverting terminal input multiplexer
- Software-selectable RC filter for the comparator output
- Output provided to Timer_A capture input
- Software control of the port input buffer
- Interrupt capability
- Selectable reference voltage generator and voltage hysteresis generator
- Reference voltage input from shared reference
- Ultra-low-power comparator mode
- Interrupt driven measurement system for low-power operation support

Figure 35-1 shows the Comparator_E block diagram.

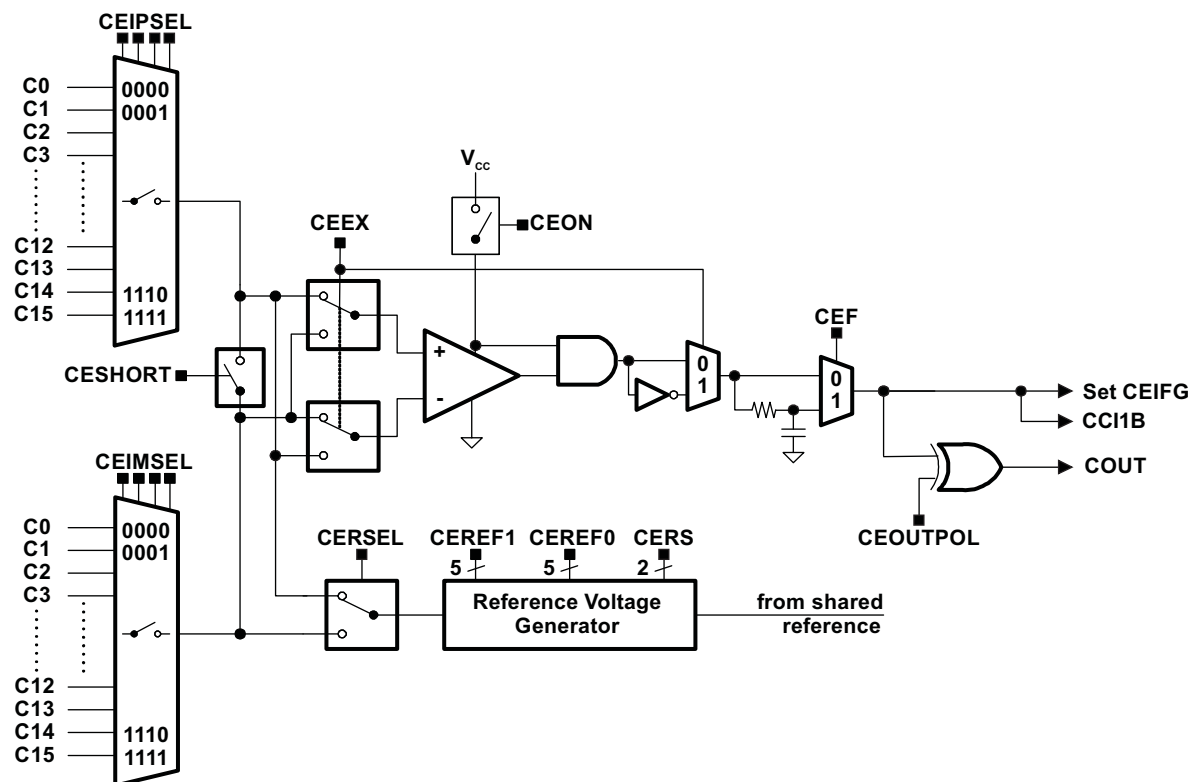


Figure 35-1. Comparator_E Block Diagram

35.2 COMP_E Operation

The COMP_E module is configured by user software. The setup and operation of COMP_E is discussed in the following sections.

35.2.1 Comparator

The comparator compares the analog voltages at the positive (+) and negative (–) input terminals. If the + terminal is more positive than the – terminal, the comparator output CEOUT is high. The comparator can be switched on or off using control bit CEON. The comparator should be switched off when not in use to reduce current consumption. When the comparator is off, CEOUT is low when CEOUTPOL bit is set to 0, and CEOUT is high when CEOUTPOL bit is set to 1.

To optimize current consumption for the application, the lowest power mode that meets the comparator speed requirements (see the device-specific data sheet for the comparator propagation delay and response time) should be selected with the CEPWRMD bits. The CEPWRMD bits default to 0x0, which is the highest power and fastest speed. CEPWRMD = 0x2 is the lowest power and slowest speed option.

35.2.2 Analog Input Switches

The analog input switches connect or disconnect the two comparator input terminals to associated port pins using the CEIPSELx and CEIMSELx bits. The comparator terminal inputs can be controlled individually. The CEIPSELx and CEIMSELx bits allow:

- Application of an external signal to the V+ and V– terminals of the comparator
- Application of an external current source (for example, a resistor) to the V+ or V– terminal of the comparator
- Mapping of both terminals of the internal multiplexer to the outside

Internally, the input switch is constructed as a T-switch to suppress distortion in the signal path.

NOTE: Comparator Input Connection

When the comparator is on, the input terminals must be connected to a signal, power, or ground. Otherwise, floating levels may cause unexpected interrupts and increased current consumption.

The CEEX bit controls the input multiplexer, permuting the input signals of the comparator's V+ and V– terminals. Additionally, when the comparator terminals are permuted, the output signal from the comparator is also inverted. This allows the user to determine or compensate for the comparator input offset voltage.

35.2.3 Port Logic

The Px.y pins that are associated with a comparator channel are enabled by the CEIPSELx or CEIMSELx bits to disable the digital components while the terminals are used as comparator inputs. Only one of the comparator input pins is selected as input to the comparator by the input multiplexer at a time.

35.2.4 Input Short Switch

The CESHORT bit shorts the Comparator_E inputs. This can be used to build a simple sample-and-hold for the comparator as shown in [Figure 35-2](#).

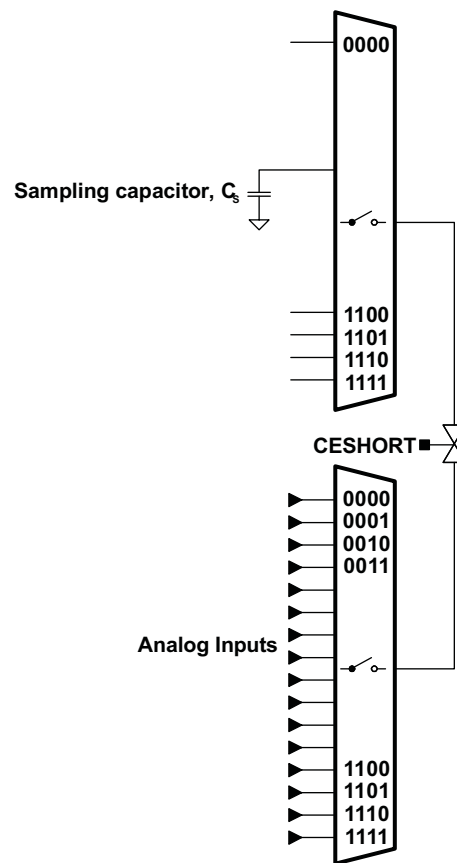


Figure 35-2. Comparator_E Sample-And-Hold

The required sampling time is proportional to the size of the sampling capacitor R_s , the resistance of the input switches in series with the short switch (R_i), and the resistance of the external source (R_s). The sampling capacitor C_s should be greater than 100 pF. The time constant, τ , to charge the sampling capacitor C_s can be calculated with the following equation:

$$\tau = (R_i + R_s) \times C_s$$

Depending on the required accuracy, 3 to 10 τ should be used as a sampling time. With 3 τ , the sampling capacitor is charged to approximately 95% of the input signal's voltage level; with 5 τ , it is charged to more than 99%; and with 10 τ , the sampled voltage is sufficient for 12-bit accuracy.

35.2.5 Output Filter

The output of the comparator can be used with or without internal filtering. When control bit CEF is set, the output is filtered with an on-chip RC filter. The delay of the filter can be adjusted in four steps with the CEFDLY bit.

All comparator outputs oscillate if the voltage difference across the input terminals is small (see [Figure 35-3](#)). Internal and external parasitic effects and cross coupling on and between signal lines, power supply lines, and other parts of the system are responsible for this behavior. The comparator output oscillation reduces the accuracy and resolution of the comparison result. Enable the output filter to reduce errors associated with comparator oscillation.

The voltage reference generator is used to generate VREF, which can be applied to either comparator input terminal. The CEREF1x (VREF1) and CEREF0x (VREF0) bits control the output of the voltage generator. The CERSEL bit selects the comparator terminal to which VREF is applied. If external signals are applied to both comparator input terminals, the internal reference generator should be turned off to reduce current consumption. The voltage reference generator can generate a fraction of the device V_{CC} or of the voltage reference of the integrated precision voltage reference source. Vref1 is used while CEOUT is 1, and Vref0 is used while CEOUT is 0. This allows the generation of a hysteresis without using external components.

35.2.7 Port Disable Register (CEPD)

The comparator input and output functions are multiplexed with I/O port pins, which are digital CMOS gates. When analog signals are applied to digital CMOS gates, parasitic current can flow from V_{CC} to GND. This parasitic current occurs if the input voltage is near the transition level of the gate. Disabling the port pin buffer eliminates the parasitic current flow and therefore reduces overall current consumption.

The CEPDx bits, when set, disable the corresponding Px.y input buffer (see Figure 35-5). When current consumption is critical, any Px.y pin connected to analog signals should be disabled with the associated CEPDx bits.

Selecting an input pin to the comparator multiplexer with the CEIPSEL or CEIMSEL bits automatically disables the input buffer for that pin, regardless of the state of the associated CEPDx bit.

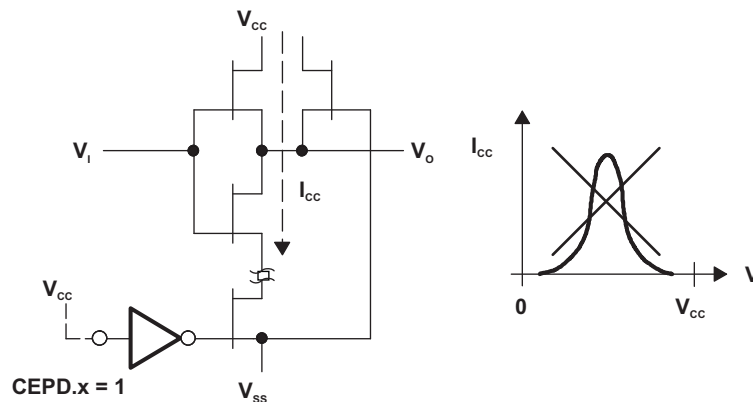


Figure 35-5. Transfer Characteristic and Power Dissipation in a CMOS Inverter and Buffer

35.2.8 Comparator_E Interrupts

One interrupt flag and one interrupt vector are associated with the Comparator_E.

The interrupt flag CEIFG is set on either the rising or falling edge of the comparator output, selected by the CEIES bit. If both the CEIE and the GIE bits are set, then the CEIFG interrupt flag generates an interrupt request.

35.2.9 Comparator_E Used to Measure Resistive Elements

The Comparator_E can be optimized to precisely measure resistive elements using single slope analog-to-digital conversion. For example, temperature can be converted into digital data using a thermistor, by comparing the thermistor's capacitor discharge time to that of a reference resistor as shown in Figure 35-6. A reference resistor Rref is compared to Rmeas.

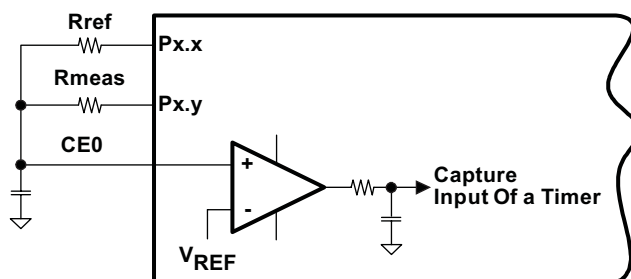


Figure 35-6. Temperature Measurement System

The resources used to calculate the temperature sensed by Rmeas are:

- Two digital I/O pins charge and discharge the capacitor.
- I/O is set to output high (V_{CC}) to charge capacitor, reset to discharge.
- I/O is switched to high-impedance input with CEPDx set when not in use.
- One output charges and discharges the capacitor through Rref.
- One output discharges capacitor through Rmeas.
- The + terminal is connected to the positive terminal of the capacitor.
- The – terminal is connected to a reference level, for example $0.25 \times V_{CC}$.
- The output filter should be used to minimize switching noise.
- CEOUT is used to gate a timer capturing capacitor discharge time.

More than one resistive element can be measured. Additional elements are connected to CE0 with available I/O pins and switched to high impedance when not being measured.

The thermistor measurement is based on a ratiometric conversion principle. The ratio of two capacitor discharge times is calculated as shown in Figure 35-7.

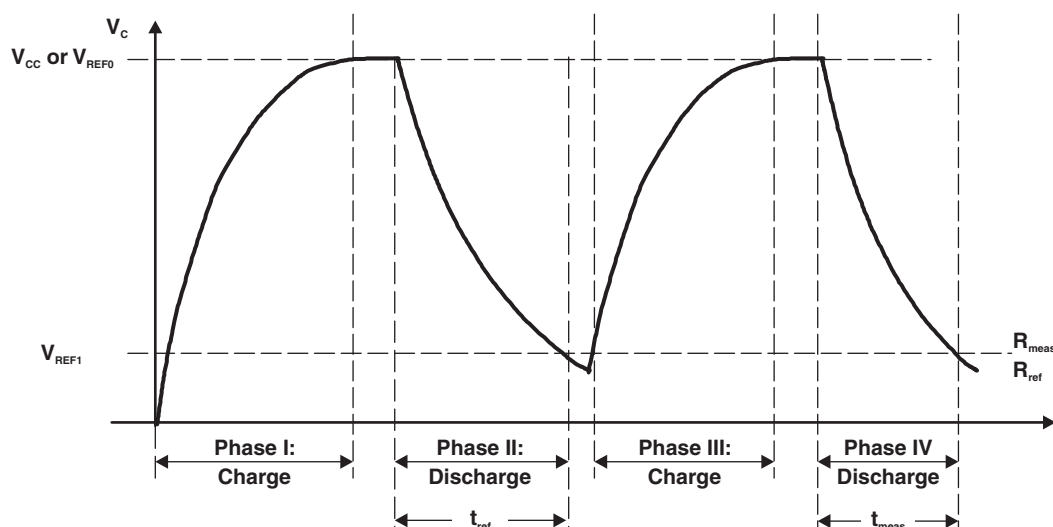


Figure 35-7. Timing for Temperature Measurement Systems

The V_{CC} voltage and the capacitor value should remain constant during the conversion but are not critical, because they cancel in the ratio:

$$\frac{N_{\text{meas}}}{N_{\text{ref}}} = \frac{-R_{\text{meas}} \times C \times \ln \frac{V_{\text{ref1}}}{V_{\text{CC}}}}{-R_{\text{ref}} \times C \times \ln \frac{V_{\text{ref1}}}{V_{\text{CC}}}}$$

$$\frac{N_{\text{meas}}}{N_{\text{ref}}} = \frac{R_{\text{meas}}}{R_{\text{ref}}}$$

$$R_{\text{meas}} = R_{\text{ref}} \times \frac{N_{\text{meas}}}{N_{\text{ref}}}$$

35.3 COMP_E Registers

The Comparator_E registers are listed in [Table 35-1](#). The base address of the Comparator_E module can be found in each device-specific data sheet.

Table 35-1. COMP_E Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	CECTL0	Comparator_E control register 0	Read/write	Word	0000h	Section 35.3.1
02h	CECTL1	Comparator_E control register 1	Read/write	Word	0000h	Section 35.3.2
04h	CECTL2	Comparator_E control register 2	Read/write	Word	0000h	Section 35.3.3
06h	CECTL3	Comparator_E control register 3	Read/write	Word	0000h	Section 35.3.4
0Ch	CEINT	Comparator_E interrupt register	Read/write	Word	0000h	Section 35.3.5
0Eh	CEIV	Comparator_E interrupt vector word	Read	Word	0000h	Section 35.3.6

35.3.1 CECTL0 Register (offset = 00h) [reset = 0000h]

Comparator_E Control Register 0

Figure 35-8. CECTL0 Register

15	14	13	12	11	10	9	8
CEIMEN	Reserved			CEIMSEL			
rw-0	r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CEIPEN	Reserved			CEIPSEL			
rw-0	r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0

Table 35-2. CECTL0 Register Description

Bit	Field	Type	Reset	Description
15	CEIMEN	RW	0h	Channel input enable for the – terminal of the comparator. 0b = Selected analog input channel for V– terminal is disabled. 1b = Selected analog input channel for V– terminal is enabled. The internal reference voltage is disabled for this channel.
14-12	Reserved	R	0h	Reserved. Reads as 0.
11-8	CEIMSEL	RW	0h	Channel input selected for the – terminal of the comparator if CEIMEN is set to 1.
7	CEIPEN	RW	0h	Channel input enable for the V+ terminal of the comparator. 0b = Selected analog input channel for V+ terminal is disabled. 1b = Selected analog input channel for V+ terminal is enabled. The internal reference voltage is disabled for this channel.
6-4	Reserved	R	0h	Reserved. Reads as 0.
3-0	CEIPSEL	RW	0h	Channel input selected for the V+ terminal of the comparator if CEIPEN is set to 1.

35.3.2 CECTL1 Register (offset = 02h) [reset = 0000h]

Comparator_E Control Register 1

Figure 35-9. CECTL1 Register

15	14	13	12	11	10	9	8
Reserved			CEMRVS	CEMRVL	CEON	CEPWRMD	
r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CEFDLY		CEEX	CESHORT	CEIES	CEF	CEOUTPOL	CEOUT
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r-0

Table 35-3. CECTL1 Register Description

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved. Reads as 0.
12	CEMRVS	RW	0h	This bit defines if the comparator output selects between VREF0 or VREF1 if CERS = 00, 01, or 10. 0b = Comparator output state selects between VREF0 or VREF1. 1b = CEMRVL selects between VREF0 or VREF1.
11	CEMRVL	RW	0h	This bit is valid if CEMRVS is set to 1. 0b = VREF0 is selected if CERS = 00, 01, or 10 1b = VREF1 is selected if CERS = 00, 01, or 10
10	CEON	RW	0h	On. This bit turns the comparator on. When the comparator is turned off the Comparator_E consumes no power. 0b = Off 1b = On
9-8	CEPWRMD	RW	0h	Power mode 00b = High-speed mode 01b = Normal mode 10b = Ultra-low power mode 11b = Reserved
7-6	CEFDLY	RW	0h	Filter delay. The filter delay can be selected in four steps. See the device-specific data sheet for details. 00b = Typical filter delay of approximately 450 ns 01b = Typical filter delay of approximately 900 ns 10b = Typical filter delay of approximately 1800 ns 11b = Typical filter delay of approximately 3600 ns
5	CEEX	RW	0h	Exchange. This bit permutes the comparator 0 inputs and inverts the comparator 0 output. 0b = Exchange off 1b = Exchange on
4	CESHORT	RW	0h	Input short. This bit shorts the + and – input terminals. 0b = Inputs not shorted 1b = Inputs shorted
3	CEIES	RW	0h	Interrupt edge select for CEIIFG and CEIFG. Changing CEIES might set CEIFG. 0b = Rising edge for CEIFG, falling edge for CEIIFG 1b = Falling edge for CEIFG, rising edge for CEIIFG
2	CEF	RW	0h	Output filter. Available if CEPWRMD = 00 or 01. 0b = Comparator_E output is not filtered 1b = Comparator_E output is filtered
1	CEOUTPOL	RW	0h	Output polarity. This bit defines the CEOUT polarity. 0b = Noninverted 1b = Inverted
0	CEOUT	R	0h	Output value. This bit reflects the value of the Comparator_E output. Writing this bit has no effect on the comparator output.

35.3.3 CECTL2 Register (offset = 04h) [reset = 0000h]

Comparator_E Control Register 2

Figure 35-10. CECTL2 Register

15	14	13	12	11	10	9	8
CEREFACC	CEREFL		CEREF1				
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CERS		CERSEL	CEREF0				
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 35-4. CECTL2 Register Description

Bit	Field	Type	Reset	Description
15	CEREFACC	RW	0h	Reference accuracy. A reference voltage is requested only if CEREFL > 0. 0b = Static mode 1b = Clocked (low power, low accuracy) mode
14-13	CEREFL	RW	0h	Reference voltage level 00b = Reference amplifier is disabled. No reference voltage is requested. 01b = 1.2 V is selected as shared reference voltage input 10b = 2.0 V is selected as shared reference voltage input 11b = 2.5 V is selected as shared reference voltage input
12-8	CEREF1	RW	0h	Reference resistor tap 1. This register defines the tap of the resistor string while CEOUT = 1.
7-6	CERS	RW	0h	Reference source. This bit define if the reference voltage is derived from VCC or from the precise shared reference. 00b = No current is drawn by the reference circuitry. 01b = VCC applied to the resistor ladder 10b = Shared reference voltage applied to the resistor ladder. 11b = Shared reference voltage supplied to VCCREF. Resistor ladder is off.
5	CERSEL	RW	0h	Reference select. This bit selects to which terminal the VCCREF is applied. When CEEX = 0: 0b = When CEEX = 0: VREF is applied to the V+ terminal; When CEEX = 1: VREF is applied to the V– terminal 1b = When CEEX = 0: VREF is applied to the V– terminal; When CEEX = 1: VREF is applied to the V+ terminal
4-0	CEREF0	RW	0h	Reference resistor tap 0. This register defines the tap of the resistor string while CEOUT = 0.

35.3.4 CECTL3 Register (offset = 06h) [reset = 0000h]

Comparator_E Control Register 3

Figure 35-11. CECTL3 Register

15	14	13	12	11	10	9	8
CEPD15	CEPD14	CEPD13	CEPD12	CEPD11	CEPD10	CEPD9	CEPD8
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
CEPD7	CEPD6	CEPD5	CEPD4	CEPD3	CEPD2	CEPD1	CEPD0
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 35-5. CECTL3 Register Description

Bit	Field	Type	Reset	Description
15	CEPD15	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD15 disables the port of the comparator channel 15. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
14	CEPD14	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD14 disables the port of the comparator channel 14. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
13	CEPD13	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD13 disables the port of the comparator channel 13. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
12	CEPD12	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD12 disables the port of the comparator channel 12. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
11	CEPD11	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD11 disables the port of the comparator channel 11. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
10	CEPD10	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD10 disables the port of the comparator channel 10. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
9	CEPD9	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD9 disables the port of the comparator channel 9. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
8	CEPD8	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD8 disables the port of the comparator channel 8. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
7	CEPD7	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD7 disables the port of the comparator channel 7. 0b = The input buffer is enabled. 1b = The input buffer is disabled.

Table 35-5. CECTL3 Register Description (continued)

Bit	Field	Type	Reset	Description
6	CEPD6	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD6 disables the port of the comparator channel 6. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
5	CEPD5	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD5 disables the port of the comparator channel 5. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
4	CEPD4	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD4 disables the port of the comparator channel 4. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
3	CEPD3	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD3 disables the port of the comparator channel 3. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
2	CEPD2	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD2 disables the port of the comparator channel 2. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
1	CEPD1	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD1 disables the port of the comparator channel 1. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
0	CEPD0	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD0 disables the port of the comparator channel 0. 0b = The input buffer is enabled. 1b = The input buffer is disabled.

35.3.5 CEINT Register (offset = 0Ch) [reset = 0000h]

Comparator_E Interrupt Control Register

Figure 35-12. CEINT Register

15	14	13	12	11	10	9	8
Reserved			CERDYIE	Reserved		CEIIE	CEIE
r-0	r-0	r-0	rw-0	r-0	r-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Reserved			CERDYIFG	Reserved		CEIIFG	CEIFG
r-0	r-0	r-0	rw-0	r-0	r-0	rw-0	rw-0

Table 35-6. CEINT Register Description

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved. Reads as 0.
12	CERDYIE	RW	0h	Comparator_E ready interrupt enable. 0b = Interrupt is disabled 1b = Interrupt is enabled
11-10	Reserved	R	0h	Reserved. Reads as 0.
9	CEIIE	RW	0h	Comparator_E output interrupt enable inverted polarity 0b = Interrupt is disabled 1b = Interrupt is enabled
8	CEIE	RW	0h	Comparator_E output interrupt enable 0b = Interrupt is disabled 1b = Interrupt is enabled
7-5	Reserved	R	0h	Reserved. Reads as 0.
4	CERDYIFG	RW	0h	Comparator_E ready interrupt flag. This bit is set if the Comparator_E reference sources are settled and the Comparator_E module is operational. This bit has to be cleared by software. 0b = No interrupt pending. 1b = Output interrupt pending.
3-2	Reserved	R	0h	Reserved. Reads as 0.
1	CEIIFG	RW	0h	Comparator_E output inverted interrupt flag. The bit CEIES defines the transition of the output setting this bit. 0b = No interrupt pending. 1b = Output interrupt pending.
0	CEIFG	RW	0h	Comparator_E output interrupt flag. The bit CEIES defines the transition of the output setting this bit. 0b = No interrupt pending. 1b = Output interrupt pending.

35.3.6 CEIV Register (offset = 0Eh) [reset = 0000h]

Comparator_E Interrupt Vector Word Register

Figure 35-13. CEIV Register

15	14	13	12	11	10	9	8
CEIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
CEIV							
r0	r0	r0	r0	r0	r-0	r-0	r0

Table 35-7. CEIV Register Description

Bit	Field	Type	Reset	Description
15-0	CEIV	R	0h	Comparator_E interrupt vector word register. The interrupt vector register reflects only interrupt flags whose interrupt enable bit are set. Reading the CEIV register clears the pending interrupt flag with the highest priority. 00h = No interrupt pending 02h = Interrupt Source: CEOUT interrupt; Interrupt Flag: CEIFG; Interrupt Priority: Highest 04h = Interrupt Source: CEOUT interrupt inverted polarity; Interrupt Flag: CEIIFG 06h = Reserved 08h = Reserved 0Ah = Interrupt Source: Comparator ready interrupt; Interrupt Flag: CERDYIFG; Interrupt Priority: Lowest

LCD_C Controller

The LCD_C controller drives static and 2-mux to 8-mux LCDs. This chapter describes the LCD_C controller. The differences between LCD_B and LCD_C are listed in [Table 36-1](#).

Topic	Page
36.1 LCD_C Introduction	932
36.2 LCD_C Operation.....	934
36.3 LCD_C Registers	950

36.1 LCD_C Introduction

The LCD_C controller directly drives LCD displays by automatically creating the ac segment and common voltage signals. The LCD_C controller can support static and 2-mux to 8-mux LCD glasses.

The LCD_C controller features are:

- Display memory
- Automatic signal generation
- Configurable frame frequency
- Blinking of individual segments with separate blinking memory for static, and 2- to 4-mux LCDs
- Blinking of complete display for 5- to 8-mux LCDs
- Regulated charge pump up to 3.44 V (typical)
- Contrast control by software
- Support for the following types of LCDs
 - Static
 - 2-mux, 1/2 bias or 1/3 bias
 - 3-mux, 1/2 bias or 1/3 bias
 - 4-mux, 1/2 bias or 1/3 bias
 - 5-mux, 1/3 bias
 - 6-mux, 1/3 bias
 - 7-mux, 1/3 bias
 - 8-mux, 1/3 bias

The differences between LCD_B and LCD_C are listed in [Table 36-1](#).

Table 36-1. Differences Between LCD_B and LCD_C

Feature	LCD_B	LCD_C
Supported types of LCDs	Static, 2-, 3-, 4-mux	Static, 2-, 3-, 4-, 5-, 6-, 7, 8-mux
Maximum VLCDx settings	001111b	001111b
Maximum LCD voltage ($V_{LCD,typ}$)	3.44 V	3.44 V
Supported biasing schemes for 5-mux to 8-mux	5- to 8-mux not supported	1/3 biasing

[Figure 36-1](#) shows the LCD controller block diagram.

NOTE: Maximum LCD Segment Control

The maximum number of segment lines and memory registers available differs with device. See the device-specific data sheet for available segment pins and the maximum number of segments supported.

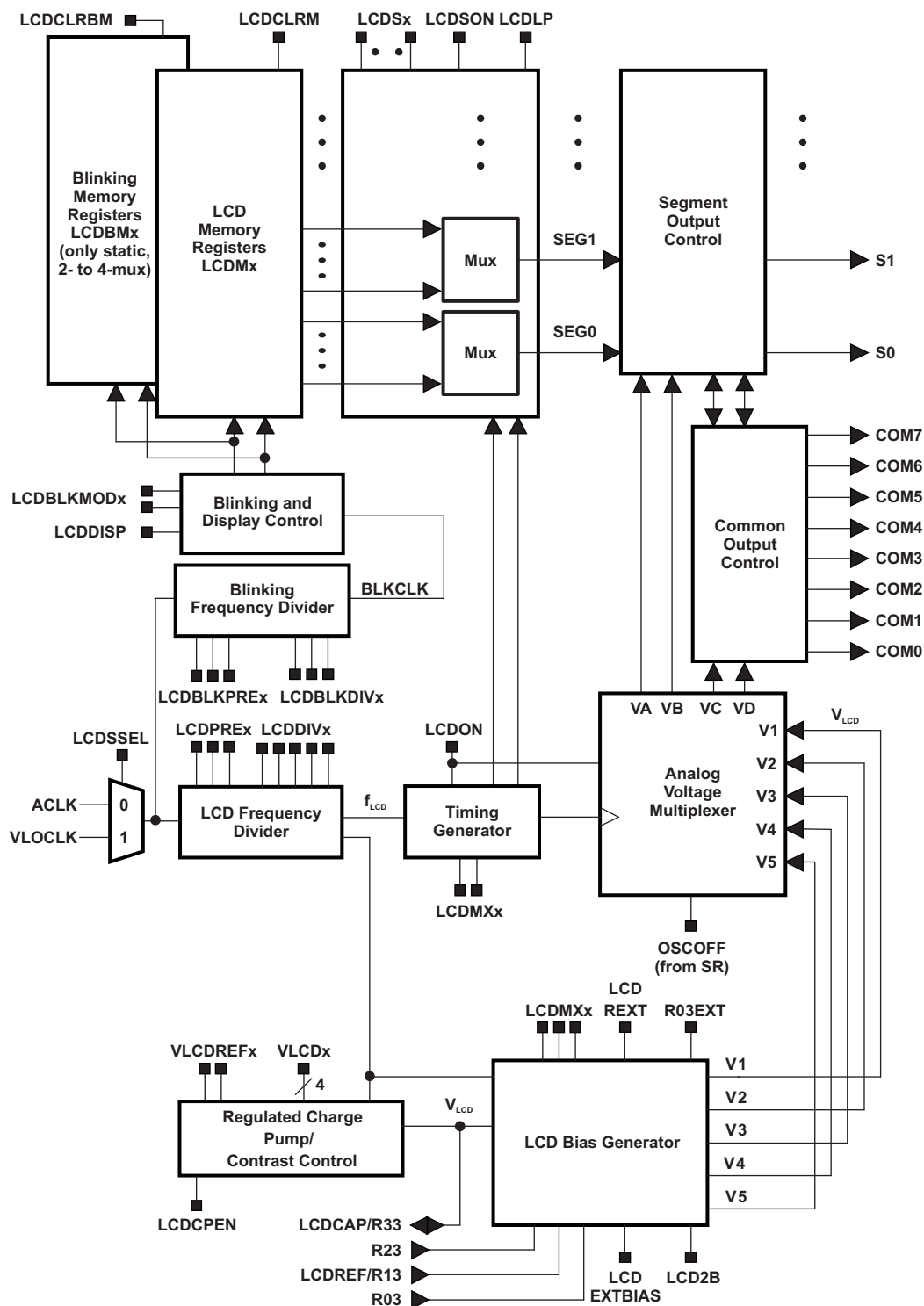


Figure 36-1. LCD Controller Block Diagram

36.2 LCD_C Operation

The LCD controller is configured with user software. The setup and operation of the LCD controller is discussed in the following sections.

36.2.1 LCD Memory

The LCD memory organization differs slightly depending on the mode.

Each memory bit corresponds to one LCD segment or is not used, depending on the mode. To turn on an LCD segment, its corresponding memory bit is set.

The memory can also be accessed word-wise using the even addresses starting at LCDM1, LCDM3, ...

Setting the bit LCDCLRM clears all LCD memory registers at the next frame boundary. It is reset automatically after the registers are cleared.

36.2.1.1 Static and 2-Mux to 4-Mux Mode

For static and 2-mux to 4-mux modes, one byte of the LCD memory contains the information for two segment lines. [Figure 36-2](#) shows an example LCD memory map for these modes with 160 segments.

Associated Common Pins		3	2	1	0	3	2	1	0	Associated Segment Pins	
Register		7							0	n	
LCDM20	--	--	--	--	--	--	--	--	--	38	39, 38
LCDM19	--	--	--	--	--	--	--	--	--	36	37, 36
LCDM18	--	--	--	--	--	--	--	--	--	34	35, 34
LCDM17	--	--	--	--	--	--	--	--	--	32	33, 32
LCDM16	--	--	--	--	--	--	--	--	--	30	31, 30
LCDM15	--	--	--	--	--	--	--	--	--	28	29, 28
LCDM14	--	--	--	--	--	--	--	--	--	26	27, 26
LCDM13	--	--	--	--	--	--	--	--	--	24	25, 24
LCDM12	--	--	--	--	--	--	--	--	--	22	23, 22
LCDM11	--	--	--	--	--	--	--	--	--	20	21, 20
LCDM10	--	--	--	--	--	--	--	--	--	18	19, 18
LCDM9	--	--	--	--	--	--	--	--	--	16	17, 16
LCDM8	--	--	--	--	--	--	--	--	--	14	15, 14
LCDM7	--	--	--	--	--	--	--	--	--	12	13, 12
LCDM6	--	--	--	--	--	--	--	--	--	10	1, 10
LCDM5	--	--	--	--	--	--	--	--	--	8	9, 8
LCDM4	--	--	--	--	--	--	--	--	--	6	7, 6
LCDM3	--	--	--	--	--	--	--	--	--	4	5, 4
LCDM2	--	--	--	--	--	--	--	--	--	2	3, 2
LCDM1	--	--	--	--	--	--	--	--	--	0	1, 0



Sn+1



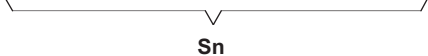
Sn

Figure 36-2. LCD Memory for Static and 2-Mux to 4-Mux Mode - Example for 160 Segments

36.2.1.2 5-Mux to 8-Mux Mode

For 5-mux to 8-mux modes, one byte of the LCD memory contains the information for one segment line. [Figure 36-3](#) shows an example LCD memory map for these modes with 160 segments.

Associated Common Pins	7	6	5	4	3	2	1	0		Associated Segment Pins
Register	7							0	n	
LCDM20	--	--	--	--	--	--	--	--	19	19
LCDM19	--	--	--	--	--	--	--	--	18	18
LCDM18	--	--	--	--	--	--	--	--	17	17
LCDM17	--	--	--	--	--	--	--	--	16	16
LCDM16	--	--	--	--	--	--	--	--	15	15
LCDM15	--	--	--	--	--	--	--	--	14	14
LCDM14	--	--	--	--	--	--	--	--	13	13
LCDM13	--	--	--	--	--	--	--	--	12	12
LCDM12	--	--	--	--	--	--	--	--	11	11
LCDM11	--	--	--	--	--	--	--	--	10	10
LCDM10	--	--	--	--	--	--	--	--	9	9
LCDM9	--	--	--	--	--	--	--	--	8	8
LCDM8	--	--	--	--	--	--	--	--	7	7
LCDM7	--	--	--	--	--	--	--	--	6	6
LCDM6	--	--	--	--	--	--	--	--	5	5
LCDM5	--	--	--	--	--	--	--	--	4	4
LCDM4	--	--	--	--	--	--	--	--	3	3
LCDM3	--	--	--	--	--	--	--	--	2	2
LCDM2	--	--	--	--	--	--	--	--	1	1
LCDM1	--	--	--	--	--	--	--	--	0	0



Sn

Figure 36-3. LCD Memory for 5-Mux to 8-Mux Mode - Example for 160 Segments

36.2.2 LCD Timing Generation

The LCD_C controller uses the f_{LCD} signal from the integrated clock divider to generate the timing for common and segment lines. With the LCDSEL bit, ACLK with a frequency between 30 kHz and 40 kHz or VLOCLK can be selected as clock source into the divider. The f_{LCD} frequency is selected with the LCDPREx and LCDDIVx bits. The resulting f_{LCD} frequency is calculated by:

$$f_{LCD} = \frac{f_{ACLK/VLOCLK}}{(LCDDIVx + 1) \times 2^{LCDPRE}}$$

The proper f_{LCD} frequency depends on the LCD's requirement for framing frequency and the LCD multiplex rate. It is calculated by:

$$f_{LCD} = 2 \times \text{mux} \times f_{Frame}$$

For example, to calculate f_{LCD} for a 3-mux LCD with a frame frequency of 30 Hz to 100 Hz:

$$f_{Frame} \text{ (from LCD data sheet)} = 30 \text{ Hz to } 100 \text{ Hz}$$

$$f_{LCD} = 2 \times 3 \times f_{Frame}$$

$$f_{LCD}(\text{min}) = 180 \text{ Hz}$$

$$f_{LCD}(\text{max}) = 600 \text{ Hz}$$

With $f_{ACLK/VLOCLK} = 32768 \text{ Hz}$, LCDPREx = 011, and LCDDIVx = 10101:

$$f_{LCD} = 32768 \text{ Hz} / ((21+1) \times 2^3) = 32768 \text{ Hz} / 176 = 186 \text{ Hz}$$

With LCDPREx = 001 and LCDDIVx = 11011:

$$f_{\text{LCD}} = 32768 \text{ Hz} / ((27+1) \times 2^1) = 32768 \text{ Hz} / 56 = 585 \text{ Hz}$$

The lowest frequency has the lowest current consumption. The highest frequency has the least flicker.

36.2.3 Blanking the LCD

The LCD controller allows blanking the complete LCD. The LCDSON bit is combined with a logical AND with each segment's memory bit. When LCDSON = 1, each segment is on or off according to its bit value. When LCDSON = 0, each LCD segment is off.

36.2.4 LCD Blinking

The LCD controller also supports blinking. In static and 2-mux to 4-mux mode, the blinking mode LCDBLKMODx = 01 allows blinking of individual segments; with LCDBLKMODx = 10 all segments are blinking; and with LCDBLKMODx = 00 blinking is disabled. In 5-mux mode and above, only blinking mode LCDBLKMODx = 10 that allows blinking of all segments is available; if another mode is selected, blinking is disabled.

36.2.4.1 Blinking Memory

In static and 2-mux to 4-mux mode, a separate blinking memory is implemented to select the blinking segments. To enable individual segments for blinking, the corresponding bit in the blinking memory LCDBMx registers must be set. The memory uses the same structure as the LCD memory (see [Figure 36-2](#)). Each memory bit corresponds to one LCD segment or is not used, depending on the multiplexing mode LCDMXx. To enable blinking for a LCD segment, its corresponding memory bit is set.

The blinking memory can also be accessed word-wise using the even addresses starting at LCDBM1, LCDBM3, ...

Setting the bit LCDCLRBM clears all blinking memory registers at the next frame boundary. It is automatically reset after the registers are cleared.

36.2.4.2 Blinking Frequency

The blinking frequency f_{BLINK} is selected with the LCDBLKPREx and LCDBLKDIVx bits. The same clock is used as selected for the LCD frequency f_{LCD} . The resulting f_{BLINK} frequency is calculated by:

$$f_{\text{Blink}} = \frac{f_{\text{ACLK/VLO}}}{(\text{LCDBLKDIVx} + 1) \times 2^{9+\text{LCDBLKPREx}}}$$

The divider generating the blinking frequency f_{BLINK} is reset when LCDBLKMODx = 00. After a blinking mode LCDBLKMODx = 01 or 10 is selected, the enabled segments or all segments go blank at the next frame boundary and stay off for half of a BLKCLK period. Then they go active at the next frame boundary and stay on for another half BLKCLK period before they go blank again at a frame boundary.

NOTE: Blinking Frequency Restrictions

The blinking frequency must be smaller than the frame frequency f_{Frame} .

The blinking frequency should only be changed when LCDBLKMODx = 00.

36.2.4.3 Dual Display Memory

In static and 2-mux to 4-mux mode, the blinking memory can also be used as a secondary display memory when no blinking mode LCDBLKMODx = 01 or 10 is selected. The memory to be displayed can be selected either manually using the LCDDISP bit or automatically with LCDBLKMODx = 11.

With LCDDISP = 0, the LCD memory is selected, and with LCDDISP = 1 the blinking memory is selected as display memory. Switching between the memories is synchronized to the frame boundaries.

With LCDBLKMODx = 11 the LCD controller switches automatically between the memories using the divider to generate the blinking frequency. After LCDBLKMODx = 11 is selected, the memory to be displayed for the first half a BLKCLK period is the LCD memory. In the second half, the blinking memory is used as display memory. Switching between the memories is synchronized to the frame boundaries.

36.2.5 LCD Voltage And Bias Generation

The LCD_C module allows selectable sources for the peak output waveform voltage, V1, as well as the fractional LCD biasing voltages V2 to V5. V_{LCD} may be sourced from V_{CC} , an internal charge pump, or externally.

All internal voltage generation is disabled if the selected clock source (ACLK or VLOCLK) is turned off (OSCOFF = 1) or the LCD_C module is disabled (LCDON = 0).

36.2.5.1 LCD Voltage Selection

V_{LCD} is sourced from V_{CC} when VLCDEXT = 0, VLCDx = 0, and VREFx = 0. V_{LCD} is sourced from the internal charge pump when VLCDEXT = 0, VLDCPEN = 1, and VLCDx > 0. The charge pump is always sourced from DV_{CC} . The VLCDx bits provide a software selectable LCD voltage from 2.6 V to 3.44 V (typical) independent of DV_{CC} . See the device-specific data sheet for specifications.

When the internal charge pump is used, a 4.7- μ F or larger capacitor must be connected between the LCDCAP pin and ground. If no capacitor is connected and the charge pump is enabled, the LCDNOCAPIFG interrupt flag is set, and the charge pump is disabled to prevent damage to the device.

To reduce system noise the charge pump can be temporarily disabled. It is disabled when LCDCPEN = 0 and re-enabled when LCDCPEN is changed back to 1. If the charge pump is temporarily disabled the voltage stored on the external capacitor is used for the LCD voltages until the charge pump is re-enabled.

Some devices can also automatically disable the charge pump during certain periods of time (for example during an ADC conversion). To enable this feature set the corresponding LCDCPDISx bits in the LCDBCPCTL register. For details see the device-specific data sheet (if the feature is not listed in the data sheet it is not supported by the respective device).

NOTE: Capacitor Required For Internal Charge Pump

A 4.7- μ F or larger capacitor must be connected from the LCDCAP pin to ground when the internal charge pump is enabled. If no capacitor is connected, the LCDNOCAPIFG interrupt flag is set and the charge pump is disabled.

The internal charge pump may use an external reference voltage when VLCDREFx = 01 (and LCDREXT = 0 and LCDEXTBIAS = 0). In this case, the charge pump voltage is set to a multiply of the external reference voltage according to the VLCDx bits setting.

When VLCDEXT = 1, V_{LCD} is sourced externally from the LCDCAP, pin and the internal charge pump is disabled.

36.2.5.2 LCD Bias Generation

The fractional LCD biasing voltages, V2 to V5 can be generated internally or externally, independent of the source for V_{LCD} .

The bias generation block diagram for LCD_C static and 2-mux to 8-mux modes is shown in [Figure 36-4](#).

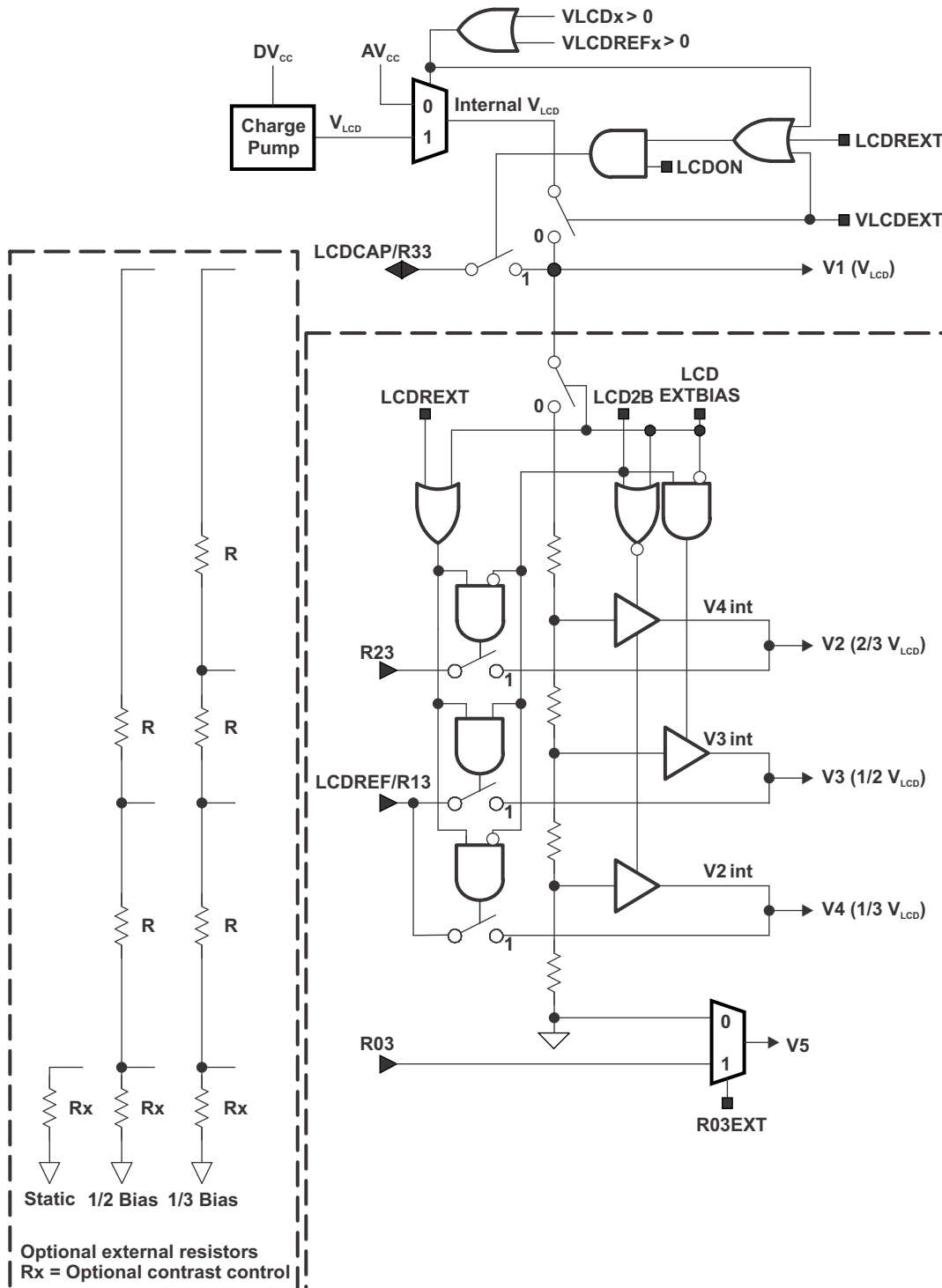


Figure 36-4. Bias Generation

The internally generated bias voltages V2 to V4 are switched to external pins with LCDREXT = 1.

To source the bias voltages V2 to V4 externally, LCDEXTBIAS is set. This also disables the internal bias generation. Typically, an equally weighted resistor divider is used with resistors ranging from a few k Ω to 1 M Ω , depending on the size of the display. When using an external resistor divider, the V_{LCD} voltage may be sourced from the internal charge pump when VLCDEXT = 0 taking the maximum charge pump load current into account. V5 can also be sourced externally when R03EXT = 1. In static mode and all mux modes using 1/2 biasing or 1/3 biasing, when R03EXT = 1 V5 can control the contrast of the connected display by changing the voltage at the low end of the external resistor divider Rx as shown in the left part of Figure 36-4.

When using an external resistor divider, R33 may serve as a switched V_{LCD} output when VLCDEXT = 0. This allows the power to the resistor ladder to be turned off, which eliminates current consumption when the LCD is not used. When VLCDEXT = 1, R33 serves as a V_{LCD} input.

The bias generator supports 1/2 biasing when LCD2B = 1 and 1/3 biasing when LCD2B = 0. In static mode, the internal divider is disabled.

Table 36-2. Bias Voltages and external Pins

Mode	Bias Configuration	LCD2B	Voltage Level	Pin	Condition
Static	Static	X	V1 ("1")	R33	if LCDREXT = 1 or LCDEXTBIAS = 1
			V5 ("0")	R03	if R03EXT = 1
2-mux, 3-mux, 4-mux	1/2	1	V1 ("1")	R33	if LCDREXT = 1 or LCDEXTBIAS = 1
			V3 ("1/2")	R13	if LCDREXT = 1 or LCDEXTBIAS = 1
			V5 ("0")	R03	if R03EXT = 1
2-mux, 3-mux, 4-mux	1/3	0	V1 ("1")	R33	if LCDREXT = 1 or LCDEXTBIAS = 1
			V2 ("2/3")	R23	if LCDREXT = 1 or LCDEXTBIAS = 1
			V4 ("1/3")	R13	if LCDREXT = 1 or LCDEXTBIAS = 1
			V5 ("0")	R03	if R03EXT = 1
5-mux to 8-mux	1/3	0	V1 ("1")	R33	if LCDREXT = 1 or LCDEXTBIAS = 1
			V2 ("2/3")	R23	if LCDREXT = 1 or LCDEXTBIAS = 1
			V4 ("1/3")	R13	if LCDREXT = 1 or LCDEXTBIAS = 1
			V5 ("0")	R03	if R03EXT = 1

36.2.5.3 LCD Contrast Control

The peak voltage of the output waveforms together with the selected mode and biasing determine the contrast and the contrast ratio of the LCD. The LCD contrast can be controlled in software by adjusting the LCD voltage generated by the integrated charge pump using the VLCDx settings.

The contrast ratio depends on the used LCD display and the selected biasing scheme. Table 36-3 shows the biasing configurations that apply to the different modes together with the RMS voltages for the segments turned on ($V_{\text{RMS,ON}}$) and turned off ($V_{\text{RMS,OFF}}$) as functions of V_{LCD} . It also shows the resulting contrast ratios between the on and off states.

Table 36-3. LCD Voltage and Biasing Characteristics

Mode	Bias Config	LCDMx	LCD2B	COM Lines	Voltage Levels	$V_{\text{RMS,OFF}}/V_{\text{LCD}}$	$V_{\text{RMS,ON}}/V_{\text{LCD}}$	Contrast Ratio $V_{\text{RMS,ON}}/V_{\text{RMS,OFF}}$
Static	Static	0000	X	1	V1, V5	0	1	1/0
2-mux	1/2 ⁽¹⁾	0001	1	2	V1, V3, V5	0.354	0.791	2.236
	1/3	0001	0	2	V1, V2, V4, V5	0.333	0.745	2.236
3-mux	1/2	0010	1	3	V1, V3, V5	0.408	0.707	1.732
	1/3 ⁽¹⁾	0010	0	3	V1, V2, V4, V5	0.333	0.638	1.915
4-mux	1/2	0011	1	4	V1, V3, V5	0.433	0.661	1.528
	1/3 ⁽¹⁾	0011	0	4	V1, V2, V4, V5	0.333	0.577	1.732
5-mux	1/3 ⁽¹⁾	0100	0	5	V1, V2, V4, V5	0.333	0.537	1.612
6-mux	1/3 ⁽¹⁾	0101	0	6	V1, V2, V4, V5	0.333	0.509	1.528
7-mux	1/3 ⁽¹⁾	0110	0	7	V1, V2, V4, V5	0.333	0.488	1.464
8-mux	1/3 ⁽¹⁾	0111	0	8	V1, V2, V4, V5	0.333	0.471	1.414

⁽¹⁾ Recommended setting to achieve best contrast

A typical approach to determine the required V_{LCD} is by equating $V_{\text{RMS,OFF}}$ with a LCD threshold voltage provided by the LCD manufacturer, for example when the LCD exhibits approximately 10% contrast ($V_{\text{th,10\%}}$): $V_{\text{RMS,OFF}} = V_{\text{th,10\%}}$. Using the values for $V_{\text{RMS,OFF}}/V_{\text{LCD}}$ provided in the table results in $V_{\text{LCD}} = V_{\text{th,10\%}}/(V_{\text{RMS,OFF}}/V_{\text{LCD}})$. In the static mode, a suitable choice is V_{LCD} greater than or equal to three times $V_{\text{th,10\%}}$.

In 3-mux and 4-mux mode, a 1/3 biasing is typically used, but a 1/2 biasing scheme is also possible. The 1/2 bias reduces the contrast ratio, but the advantage is a reduction of the required full-scale LCD voltage V_{LCD} .

36.2.6 LCD Outputs

Some LCD segment, common, and Rxx functions are multiplexed with digital I/O functions. These pins can function either as digital I/O or as LCD functions.

The LCD segment functions, when multiplexed with digital I/O, are selected using the LCDSx bits in the LCDCPCTLx registers. The LCDSx bits select the LCD function for each segment line. When LCDSx = 0, a multiplexed pin is set to digital I/O function. When LCDSx = 1, a multiplexed pin is selected as LCD function.

The pin functions for COMx and Rxx, when multiplexed with digital I/O, are selected as described in the port schematic section of the device-specific data sheet. An some devices the COM1 to COM7 pins are shared with segment lines, refer to the device-specific data sheet. If these pins are required as COM pins due to the selected LCD multiplexing mode, the COM functionality takes precedence over the segment function that can be selected for those pins with the LCDSx bits as for all other segment pins.

See the port schematic section of the device-specific data sheet for details on controlling the pin functionality.

NOTE: LCDSx Bits Do Not Affect Dedicated LCD Segment Pins

The LCDSx bits only affect pins with multiplexed LCD segment functions and digital I/O functions. Dedicated LCD segment pins are not affected by the LCDSx bits.

36.2.7 LCD Interrupts

The LCD_C module has four interrupt sources available, each with independent enables and flags.

The four interrupt flags, namely LCDFRMIFG, LCDBLKOFFIFG, LCDBLKONIFG, and LCDNOCAPIFG, are prioritized and combined to source a single interrupt vector. The interrupt vector register LCDCIV is used to determine which flag requested an interrupt.

The highest priority enabled interrupt generates a number in the LCDCIV register (see register description). This number can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled LCD interrupts do not affect the LCDCIV value.

Any read access of the LCDCIV register automatically resets the highest pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. A write access to the LCDCIV register automatically resets all pending interrupt flags. In addition, all flags can be cleared by software.

The LCDNOCAPIFG indicates that no capacitor is present at the LCDCAP pin when the charge pump is enabled. Setting the LCDNOCAPIE bit enables the interrupt.

The LCDBLKONIFG bit is set on the rising edge of BLKCLK when the LCD switches to blinking status if blinking is enabled with LCDBLKMODx = 01 or 10. The LCDBLKONIFG bit is also set on the edge of BLKCLK when the blinking memory is selected as the display memory with LCDBLKMODx = 11. The bit is automatically cleared when an LCD or blinking memory register is written. Set the LCDBLKONIE bit to 1 to enable the interrupt.

The LCDBLKOFFIFG bit is set at the falling edge of BLKCLK when the LCD switches to nonblinking status if blinking is enabled with LCDBLKMODx = 01 or 10. The LCDBLKOFFIFG bit is also set at the edge of BLKCLK when the LCD memory is selected as the display memory with LCDBLKMODx = 11. The bit is automatically cleared when an LCD or blinking memory register is written. Set the LCDBLKOFFIE bit to 1 to enable the interrupt.

The LCDFRMIFG is set at a frame boundary. It is automatically cleared when a LCD or blinking memory register is written. Setting the LCDFRMIFGIE bit enables the interrupt.

36.2.7.1 LCDCIV Software Example

The following software example shows the recommended use of LCDCIV and the handling overhead. The LCDCIV value is added to the PC to automatically jump to the appropriate routine.

The numbers at the right margin show the necessary CPU cycles for each instruction. The software overhead for different interrupt sources includes interrupt latency and return-from-interrupt cycles, but not the task handling itself.

```
; Interrupt handler for LCD_B interrupt flags.
LCDB_HND          ; Interrupt latency          6
  ADD &LCDBIV,PC   ; Add offset to Jump table  3
  RETI             ; Vector 0: No interrupt     5
  JMP LCDNOCAP_HND ; Vector 2: LCDNOCAPIFG      2
  JMP LCDBLKON_HND ; Vector 4: LCDBLKONIFG      2
  JMP LCDBLKOFF_HND ; Vector 6: LCDBLKOFFIFG    2
LCDFRM_HND        ; Vector 8: LCDFRMIFG
  ... ; Task starts here
  RETI             5
LCDNOCAP_HND ; Vector 2: LCDNOCAPIFG
  ... ; Task starts here
  RETI             5
LCDBLKON_HND ; Vector 4: LCDBLKONIFG
  ... ; Task starts here
  RETI ; Back to main program          5
LCDBLKOFF_HND ; Vector 6: LCDBLKOFFIFG
  ... ; Task starts here
  RETI ; Back to main program          5
```


36.2.8 Static Mode

In static mode, each MSP430 segment pin drives one LCD segment, and one common line (COM0) is used. [Figure 36-5](#) shows some example static waveforms.

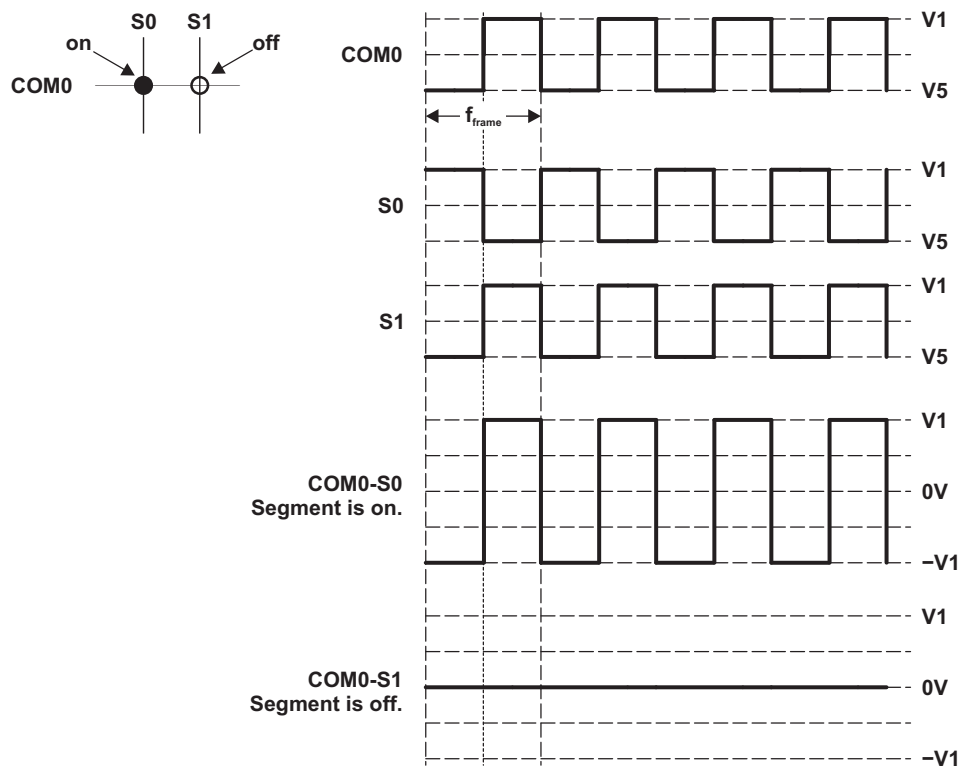


Figure 36-5. Example Static Waveforms

36.2.9 2-Mux Mode

In 2-mux mode, each MSP430 segment pin drives two LCD segments, and two common lines (COM0 and COM1) are used. [Figure 36-6](#) shows some example 2-mux 1/2-bias waveforms.

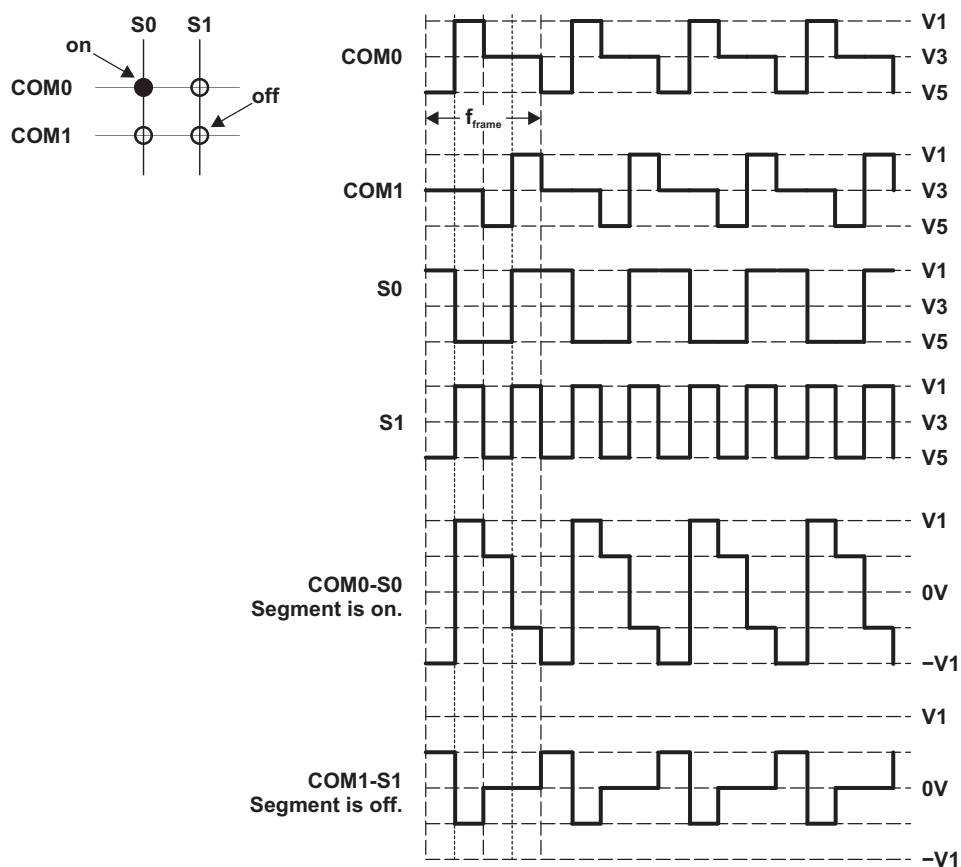


Figure 36-6. Example 2-Mux Waveforms

36.2.10 3-Mux Mode

In 3-mux mode, each MSP430 segment pin drives three LCD segments, and three common lines (COM0, COM1, and COM2) are used. Figure 36-7 shows some example 3-mux 1/3-bias waveforms.

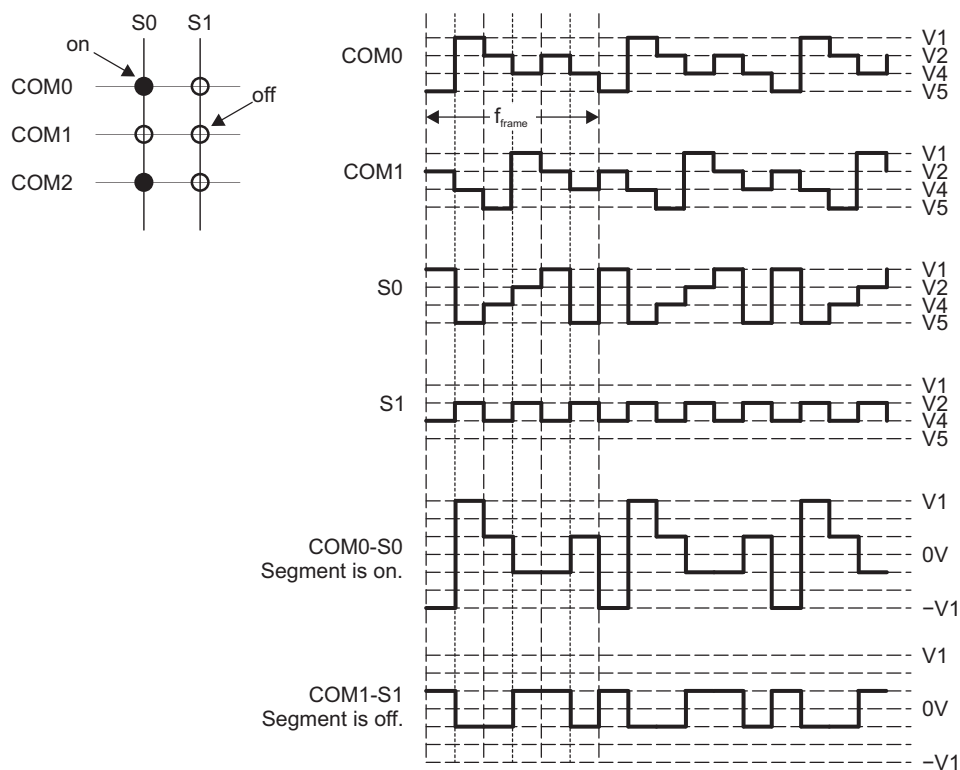


Figure 36-7. Example 3-Mux Waveforms

36.2.11 4-Mux Mode

In 4-mux mode, each MSP430 segment pin drives four LCD segments and four common lines (COM0, COM1, COM2, and COM3) are used. [Figure 36-8](#) shows some example 4-mux 1/3-bias waveforms.

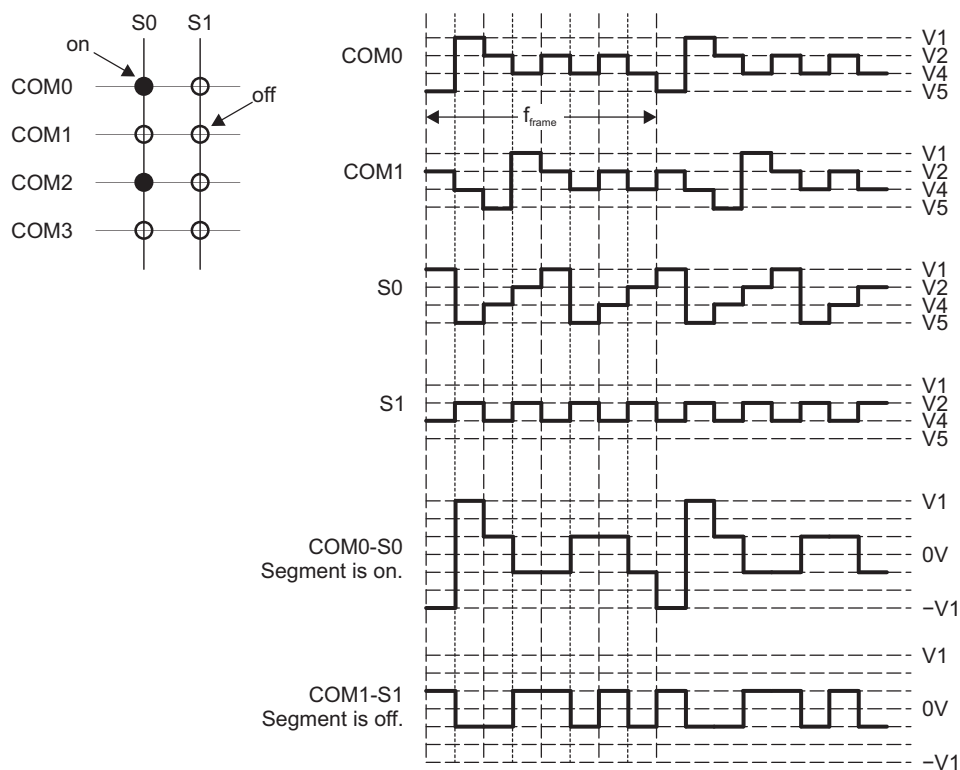


Figure 36-8. Example 4-Mux Waveforms

36.2.12 6-Mux Mode

In 6-mux mode, each MSP430 segment pin drives six LCD segments, and six common lines (COM0, COM1, COM2, COM3, COM4, and COM5) are used. [Figure 36-9](#) shows some example 6-mux 1/3-bias waveforms.

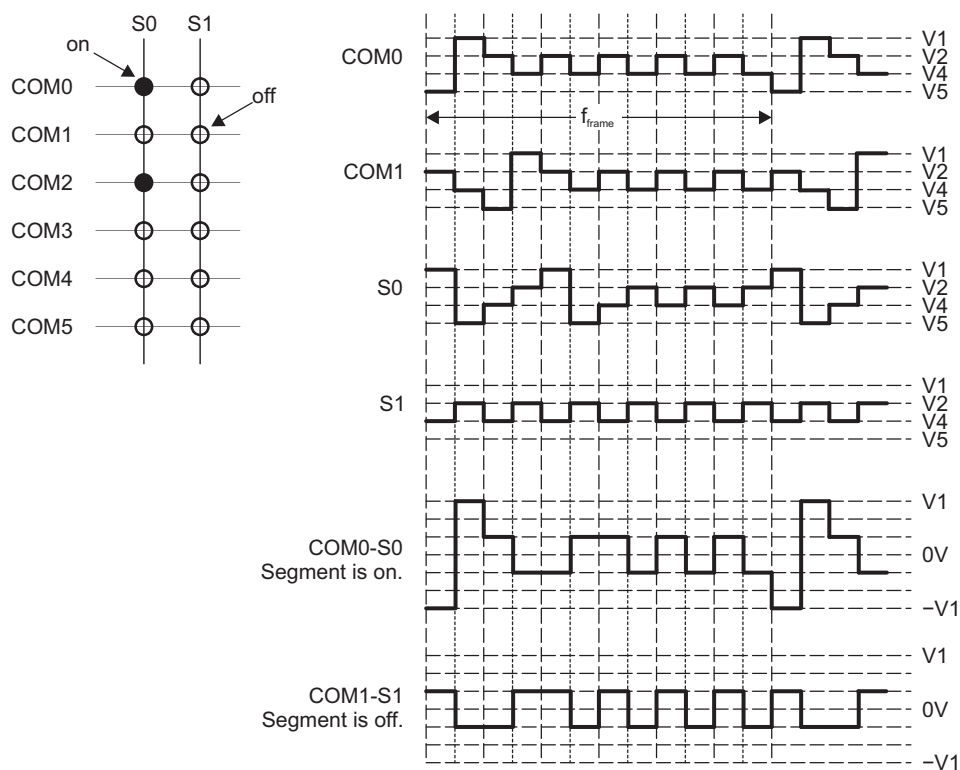


Figure 36-9. Example 6-Mux Waveforms

36.2.13 8-Mux Mode

In 8-mux mode, each MSP430 segment pin drives eight LCD segments, and eight common lines (COM0 through COM7) are used. [Figure 36-10](#) shows some example 8-mux 1/3-bias waveforms.

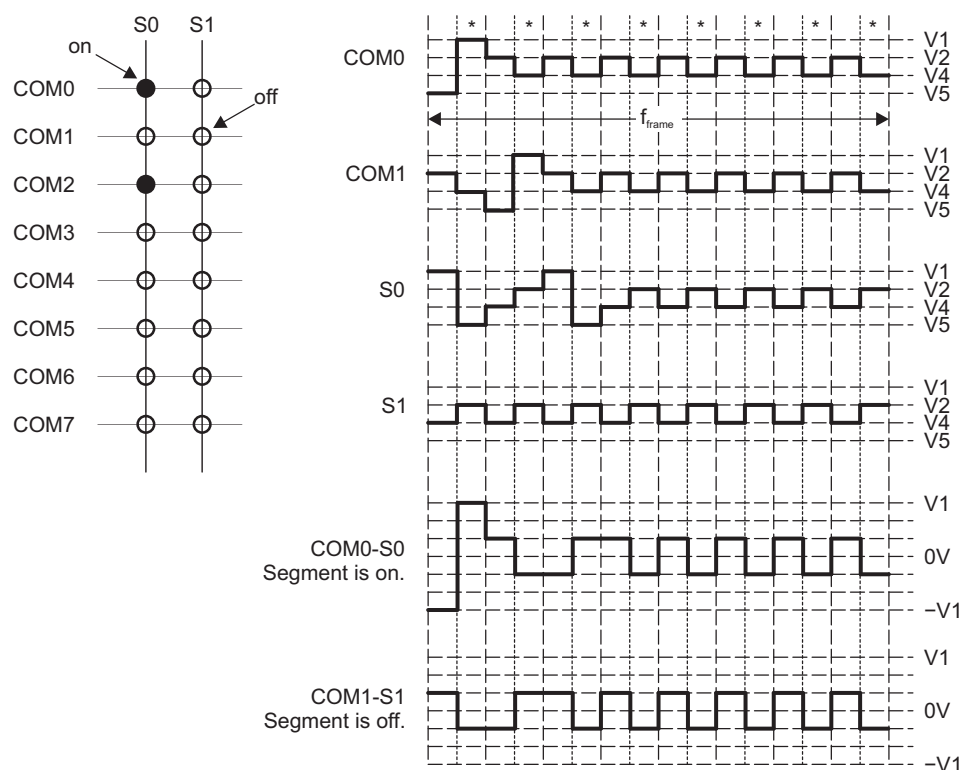


Figure 36-10. Example 8-Mux, 1/3 Bias Waveforms (LCDLP = 0)

Figure 36-11 shows some example 8-mux 1/3-bias waveforms with LCDLP = 1. With LCDLP = 1, the voltage sequence compared to the non-low power waveform is reshuffled; that is, all of the timeslots marked with "*" in Figure 36-10 are grouped together. The same principle applies to all mux modes.

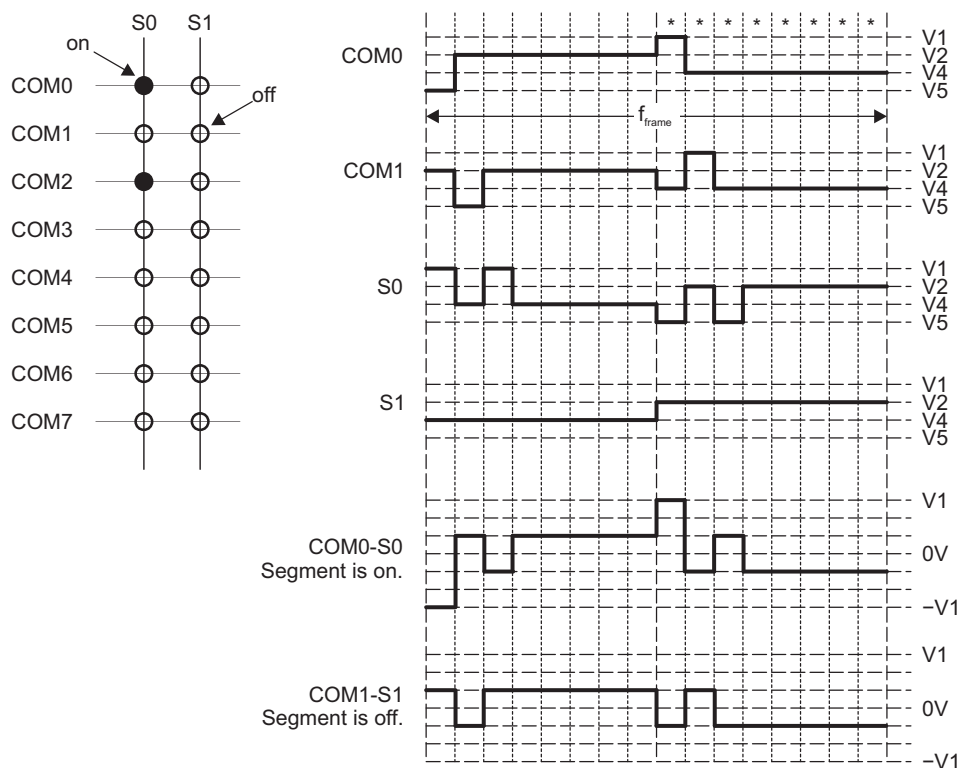


Figure 36-11. Example 8-Mux, 1/3 Bias Low-Power Waveforms (LCDLP = 1)

36.3 LCD_C Registers

The LCD_C registers are listed in [Table 36-4](#) to [Table 36-7](#). The LCD memory and blinking memory registers can also be accessed as word.

The number of available memory registers on a given device depends on the number of available segment pins; see the device-specific data sheet.

Table 36-4. LCD_C Control Registers

Offset	Acronym	Register Name	Type	Reset	Section
000h	LCDCCTL0	LCD_C control 0	Read/write	0000h	Section 36.3.1
002h	LCDCCTL1	LCD_C control 1	Read/write	0000h	Section 36.3.2
004h	LCDCBLKCTL	LCD_C blinking control	Read/write	0000h	Section 36.3.3
006h	LCDCMEMCTL	LCD_C memory control	Read/write	0000h	Section 36.3.4
008h	LCDCVCTL	LCD_C voltage control	Read/write	0000h	Section 36.3.5
00Ah	LCDCPCTL0	LCD_C port control 0	Read/write	0000h	Section 36.3.6
00Ch	LCDCPCTL1	LCD_C port control 1	Read/write	0000h	Section 36.3.7
00Eh	LCDCPCTL2	LCD_C port control 2 (≥ 256 segments)	Read/write	0000h	Section 36.3.8
010h	LCDCPCTL3	LCD_C port control 3 (384 segments)	Read/write	0000h	Section 36.3.9
012h	LCDCCPCTL	LCD_C charge pump control	Read/write	0000h	Section 36.3.10
014h		Reserved			
016h		Reserved			
018h		Reserved			
01Ah		Reserved			
01Ch		Reserved			
01Eh	LCDCIV	LCD_C interrupt vector	Read/write	0000h	Section 36.3.11

Table 36-5. LCD_C Memory Registers for Static and 2-Mux to 4-Mux Modes⁽¹⁾⁽²⁾

Offset	Acronym	Register Name	Type	Reset
020h	LCDM1	LCD memory 1 (S1/S0)	Read/write	Unchanged
021h	LCDM2	LCD memory 2 (S3/S2)	Read/write	Unchanged
022h	LCDM3	LCD memory 3 (S5/S4)	Read/write	Unchanged
023h	LCDM4	LCD memory 4 (S7/S6)	Read/write	Unchanged
024h	LCDM5	LCD memory 5 (S9/S8)	Read/write	Unchanged
025h	LCDM6	LCD memory 6 (S11/S10)	Read/write	Unchanged
026h	LCDM7	LCD memory 7 (S13/S12)	Read/write	Unchanged
027h	LCDM8	LCD memory 8 (S15/S14)	Read/write	Unchanged
028h	LCDM9	LCD memory 9 (S17/S16)	Read/write	Unchanged
029h	LCDM10	LCD memory 10 (S19/S18)	Read/write	Unchanged
02Ah	LCDM11	LCD memory 11 (S21/S20)	Read/write	Unchanged
02Bh	LCDM12	LCD memory 12 (S23/S22)	Read/write	Unchanged
02Ch	LCDM13	LCD memory 13 (S25/S24)	Read/write	Unchanged
02Dh	LCDM14	LCD memory 14 (S27/S26)	Read/write	Unchanged
02Eh	LCDM15	LCD memory 15 (S29/S28)	Read/write	Unchanged
02Fh	LCDM16	LCD memory 16 (S31/S30)	Read/write	Unchanged
030h	LCDM17	LCD memory 17 (S33/S32)	Read/write	Unchanged
031h	LCDM18	LCD memory 18 (S35/S34)	Read/write	Unchanged
032h	LCDM19	LCD memory 19 (S37/S36)	Read/write	Unchanged
033h	LCDM20	LCD memory 20 (S39/S38)	Read/write	Unchanged
034h	LCDM21	LCD memory 21 (S41/S40)	Read/write	Unchanged
035h	LCDM22	LCD memory 22 (S43/S42)	Read/write	Unchanged
036h	LCDM23	LCD memory 23 (S45/S44)	Read/write	Unchanged
037h	LCDM24	LCD memory 24 (S47/S46)	Read/write	Unchanged
038h	LCDM25	LCD memory 25 (S49/S48)	Read/write	Unchanged
039h	LCDM26	LCD memory 26 (S51/S50)	Read/write	Unchanged
03Ah	LCDM27	LCD memory 27 (S53/S52)	Read/write	Unchanged
03Bh	LCDM28	LCD memory 28 (S54)	Read/write	Unchanged
03Ch		Reserved		
03Dh		Reserved		
03Eh		Reserved		
03Fh		Reserved		

⁽¹⁾ The LCD memory registers can also be accessed as word.

⁽²⁾ The number of available memory registers on a given device depends on the number of available segment pins; see the device-specific data sheet.

Table 36-6. LCD Blinking Memory Registers for Static and 2-Mux to 4-Mux Modes⁽¹⁾⁽²⁾

Offset	Acronym	Register Name	Type	Reset
040h	LCDBM1	LCD blinking memory 1	Read/write	Unchanged
041h	LCDBM2	LCD blinking memory 2	Read/write	Unchanged
042h	LCDBM3	LCD blinking memory 3	Read/write	Unchanged
043h	LCDBM4	LCD blinking memory 4	Read/write	Unchanged
044h	LCDBM5	LCD blinking memory 5	Read/write	Unchanged
045h	LCDBM6	LCD blinking memory 6	Read/write	Unchanged
046h	LCDBM7	LCD blinking memory 7	Read/write	Unchanged
047h	LCDBM8	LCD blinking memory 8	Read/write	Unchanged
048h	LCDBM9	LCD blinking memory 9	Read/write	Unchanged
049h	LCDBM10	LCD blinking memory 10	Read/write	Unchanged
04Ah	LCDBM11	LCD blinking memory 11	Read/write	Unchanged
04Bh	LCDBM12	LCD blinking memory 12	Read/write	Unchanged
04Ch	LCDBM13	LCD blinking memory 13	Read/write	Unchanged
04Dh	LCDBM14	LCD blinking memory 14	Read/write	Unchanged
04Eh	LCDBM15	LCD blinking memory 15	Read/write	Unchanged
04Fh	LCDBM16	LCD blinking memory 16	Read/write	Unchanged
050h	LCDBM17	LCD blinking memory 17	Read/write	Unchanged
051h	LCDBM18	LCD blinking memory 18	Read/write	Unchanged
052h	LCDBM19	LCD blinking memory 19	Read/write	Unchanged
053h	LCDBM20	LCD blinking memory 20	Read/write	Unchanged
054h	LCDBM21	LCD blinking memory 21	Read/write	Unchanged
055h	LCDBM22	LCD blinking memory 22	Read/write	Unchanged
056h	LCDBM23	LCD blinking memory 23	Read/write	Unchanged
057h	LCDBM24	LCD blinking memory 24	Read/write	Unchanged
058h	LCDBM25	LCD blinking memory 25	Read/write	Unchanged
059h	LCDBM26	LCD blinking memory 26	Read/write	Unchanged
05Ah	LCDBM27	LCD blinking memory 27	Read/write	Unchanged
05Bh	LCDBM28	LCD blinking memory 28	Read/write	Unchanged
05Ch		Reserved		
05Dh		Reserved		
05Eh		Reserved		
05Fh		Reserved		

⁽¹⁾ The LCD blinking memory registers can also be accessed as word.

⁽²⁾ The number of available memory registers on a given device depends on the number of available segment pins; see the device-specific data sheet.

Table 36-7. LCD Memory Registers for 5-Mux to 8-Mux⁽¹⁾⁽²⁾

Offset	Acronym	Register Name	Type	Reset
020h	LCDM1	LCD memory 1 (S0)	Read/write	Unchanged
021h	LCDM2	LCD memory 2 (S1)	Read/write	Unchanged
022h	LCDM3	LCD memory 3 (S2)	Read/write	Unchanged
023h	LCDM4	LCD memory 4 (S3)	Read/write	Unchanged
024h	LCDM5	LCD memory 5 (S4)	Read/write	Unchanged
025h	LCDM6	LCD memory 6 (S5)	Read/write	Unchanged
026h	LCDM7	LCD memory 7 (S6)	Read/write	Unchanged
027h	LCDM8	LCD memory 8 (S7)	Read/write	Unchanged
028h	LCDM9	LCD memory 9 (S8)	Read/write	Unchanged
029h	LCDM10	LCD memory 10 (S9)	Read/write	Unchanged
02Ah	LCDM11	LCD memory 11 (S10)	Read/write	Unchanged
02Bh	LCDM12	LCD memory 12 (S11)	Read/write	Unchanged
02Ch	LCDM13	LCD memory 13 (S12)	Read/write	Unchanged
02Dh	LCDM14	LCD memory 14 (S13)	Read/write	Unchanged
02Eh	LCDM15	LCD memory 15 (S14)	Read/write	Unchanged
02Fh	LCDM16	LCD memory 16 (S15)	Read/write	Unchanged
030h	LCDM17	LCD memory 17 (S16)	Read/write	Unchanged
031h	LCDM18	LCD memory 18 (S17)	Read/write	Unchanged
032h	LCDM19	LCD memory 19 (S18)	Read/write	Unchanged
033h	LCDM20	LCD memory 20 (S19)	Read/write	Unchanged
034h	LCDM21	LCD memory 21 (S20)	Read/write	Unchanged
035h	LCDM22	LCD memory 22 (S21)	Read/write	Unchanged
036h	LCDM23	LCD memory 23 (S22)	Read/write	Unchanged
037h	LCDM24	LCD memory 24 (S23)	Read/write	Unchanged
038h	LCDM25	LCD memory 25 (S24)	Read/write	Unchanged
039h	LCDM26	LCD memory 26 (S25)	Read/write	Unchanged
03Ah	LCDM27	LCD memory 27 (S26)	Read/write	Unchanged
03Bh	LCDM28	LCD memory 28 (S27)	Read/write	Unchanged
03Ch	LCDM29	LCD memory 29 (S28)	Read/write	Unchanged
03Dh	LCDM30	LCD memory 30 (S29)	Read/write	Unchanged
03Eh	LCDM31	LCD memory 31 (S30)	Read/write	Unchanged
03Fh	LCDM32	LCD memory 32 (S31)	Read/write	Unchanged
040h	LCDM33	LCD memory 33 (S32)	Read/write	Unchanged
041h	LCDM34	LCD memory 34 (S33)	Read/write	Unchanged
042h	LCDM35	LCD memory 35 (S34)	Read/write	Unchanged
043h	LCDM36	LCD memory 36 (S35)	Read/write	Unchanged
044h	LCDM37	LCD memory 37 (S36)	Read/write	Unchanged
045h	LCDM38	LCD memory 38 (S37)	Read/write	Unchanged
046h	LCDM39	LCD memory 39 (S38)	Read/write	Unchanged
047h	LCDM40	LCD memory 40 (S39)	Read/write	Unchanged
048h	LCDM41	LCD memory 41 (S40)	Read/write	Unchanged
049h	LCDM42	LCD memory 42 (S41)	Read/write	Unchanged
04Ah	LCDM43	LCD memory 43 (S42)	Read/write	Unchanged
04Bh	LCDM44	LCD memory 44 (S43)	Read/write	Unchanged
04Ch	LCDM45	LCD memory 45 (S44)	Read/write	Unchanged

⁽¹⁾ The LCD memory registers can also be accessed as word.

⁽²⁾ The number of available memory registers on a given device depends on the number of available segment pins; see the device-specific data sheet.

Table 36-7. LCD Memory Registers for 5-Mux to 8-Mux⁽¹⁾⁽²⁾ (continued)

Offset	Acronym	Register Name	Type	Reset
04Dh	LCDM46	LCD memory 46 (S45)	Read/write	Unchanged
04Eh	LCDM47	LCD memory 47 (S46)	Read/write	Unchanged
04Fh	LCDM48	LCD memory 48 (S47)	Read/write	Unchanged
050h	LCDM49	LCD memory 49 (S48)	Read/write	Unchanged
051h	LCDM50	LCD memory 50 (S49)	Read/write	Unchanged
052h	LCDM51	LCD memory 51 (S50)	Read/write	Unchanged
053h	LCDM52	LCD memory 52 (S51)	Read/write	Unchanged
054h		Reserved		
055h		Reserved		
056h		Reserved		
057h		Reserved		
058h		Reserved		
059h		Reserved		
05Ah		Reserved		
05Bh		Reserved		
05Ch		Reserved		
05Dh		Reserved		
05Eh		Reserved		
05Fh		Reserved		

36.3.1 LCDCCTL0 Register

LCD_C Control Register 0

NOTE: Settings for LCDDIVx, LCDPREx, LCDSSEL, LCDLP and LCDMXx should be changed only while LCDON = 0.

Figure 36-12. LCDCCTL0 Register

15	14	13	12	11	10	9	8
LCDDIVx					LCDPREx		
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
LCDSSSEL	Reserved	LCDMXx			LCDSON	LCDLP	LCDON
rw-0	r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 36-8. LCDCCTL0 Register Description

Bit	Field	Type	Reset	Description
15-11	LCDDIVx	RW	0h	LCD frequency divider. Together with LCDPREx the LCD frequency f_{LCD} is calculated as $f_{LCD} = f_{ACLK/VLO} / ((LCDDIVx + 1) \times 2^{LCDPREx})$. 00000b = Divide by 1 00001b = Divide by 2 ⋮ 11110b = Divide by 31 11111b = Divide by 32
10-8	LCDPREx	RW	0h	LCD frequency pre-scaler. Together with LCDDIVx the LCD frequency f_{LCD} is calculated as $f_{LCD} = f_{ACLK/VLO} / ((LCDDIVx + 1) \times 2^{LCDPREx})$. 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 = Reserved (defaults to divide by 32) 111b = Reserved (defaults to divide by 32)
7	LCDSSSEL	RW	0h	Clock source select for LCD and blinking frequency 0b = ACLK (30 kHz to 40 kHz) 1b = VLOCLK
6	Reserved	R	0h	Reserved
5-3	LCDMXx	RW	0h	LCD mux rate. These bits select the LCD mode. 000b = Static 001b = 2-mux 010b = 3-mux 011b = 4-mux 100b = 5-mux 101b = 6-mux 110b = 7-mux 111b = 8-mux
2	LCDSON	RW	0h	LCD segments on. This bit supports flashing LCD applications by turning off all segment lines, while leaving the LCD timing generator and R33 enabled. 0b = All LCD segments are off 1b = All LCD segments are enabled and on or off according to their corresponding memory location
1	LCDLP	RW	0h	LCD low-power waveform 0b = Standard LCD waveforms on segment and common lines selected 1b = Low-power LCD waveforms on segment and common lines selected

Table 36-8. LCDCCTL0 Register Description (continued)

Bit	Field	Type	Reset	Description
0	LCDON	RW	0h	LCD on. This bit turns the LCD_C module on or off. 0b = LCD_C module off 1b = LCD_C module on

36.3.2 LCDCTL1 Register

LCD_C Control Register 1

Figure 36-13. LCDCTL1 Register

15	14	13	12	11	10	9	8
Reserved				LCDNOCAPIE	LCDBLKONIE	LCDBLKOFFIE	LCDFRMIE
r0	r0	r0	r0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Reserved				LCDNOCAPIFG	LCDBLKONIFG	LCDBLKOFFIFG	LCDFRMIFG
r0	r0	r0	r0	rw-0	rw-0	rw-0	rw-0

Table 36-9. LCDCTL1 Register Description

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11	LCDNOCAPIE	RW	0h	No capacitance connected interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
10	LCDBLKONIE	RW	0h	LCD blinking interrupt enable, segments switched on 0b = Interrupt disabled 1b = Interrupt enabled
9	LCDBLKOFFIE	RW	0h	LCD blinking interrupt enable, segments switched off 0b = Interrupt disabled 1b = Interrupt enabled
8	LCDFRMIE	RW	0h	LCD frame interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
7-4	Reserved	R	0h	Reserved
3	LCDNOCAPIFG	RW	0h	No capacitance connected interrupt flag. Set when charge pump is enabled but no capacitance is connected to LCDCAP pin. 0b = No interrupt pending 1b = Interrupt pending
2	LCDBLKONIFG	RW	0h	LCD blinking interrupt flag, segments switched on. Automatically cleared when data is written into a memory register. 0b = No interrupt pending 1b = Interrupt pending
1	LCDBLKOFFIFG	RW	0h	LCD blinking interrupt flag, segments switched off. Automatically cleared when data is written into a memory register. 0b = No interrupt pending 1b = Interrupt pending
0	LCDFRMIFG	RW	0h	LCD frame interrupt flag. Automatically cleared when data is written into a memory register. 0b = No interrupt pending 1b = Interrupt pending

36.3.3 LCDCBLKCTL Register

LCD_C Blink Control Register

NOTE: Settings for LCDBLKDIVx and LCDBLKPREx should only be changed while LCDBLKMODx = 00.

Figure 36-14. LCDCBLKCTL Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
LCDBLKDIVx			LCDBLKPREx			LCDBLKMODx	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 36-10. LCDCBLKCTL Register Description

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-5	LCDBLKDIVx	RW	0h	Clock divider for blinking frequency. Together with LCDBLKPREx, the blinking frequency f_{BLINK} is calculated as $f_{\text{BLINK}} = f_{\text{ACLK/VLO}} / ((\text{LCDBLKDIVx} + 1) \times 2^{9+\text{LCDBLKPREx}})$. NOTE: Should only be changed while LCDBLKMODx = 00. 000b = Divide by 1 001b = Divide by 2 010b = Divide by 3 011b = Divide by 4 100b = Divide by 5 101b = Divide by 6 110b = Divide by 7 111b = Divide by 8
4-2	LCDBLKPREx	RW	0h	Clock pre-scaler for blinking frequency. Together with LCDBLKDIVx, the blinking frequency f_{BLINK} is calculated as $f_{\text{BLINK}} = f_{\text{ACLK/VLO}} / ((\text{LCDBLKDIVx} + 1) \times 2^{9+\text{LCDBLKPREx}})$. NOTE: Should only be changed while LCDBLKMODx = 00. 000b = Divide by 512 001b = Divide by 1024 010b = Divide by 2048 011b = Divide by 4096 100b = Divide by 8192 101b = Divide by 16384 110b = Divide by 32768 111b = Divide by 65536
1-0	LCDBLKMODx	RW	0h	Blinking mode 00b = Blinking disabled 01b = Blinking of individual segments as enabled in blinking memory register LCDBMx. In mux mode >5 blinking is disabled. 10b = Blinking of all segments 11b = Switching between display contents as stored in LCDMx and LCDBMx memory registers. In mux mode >5 blinking is disabled.

36.3.4 LCDCMEMCTL Register

LCD_C Memory Control Register

Figure 36-15. LCDCMEMCTL Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved					LCDCLRBM	LCDCLRM	LCDDISP
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0

Table 36-11. LCDCMEMCTL Register Description

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0h	Reserved
2	LCDCLRBM	RW	0h	Clear LCD blinking memory Clears all blinking memory registers LCDBMx. The bit is automatically reset when the blinking memory is cleared. Setting this bit has in 5-mux mode and above has no effect. It's immediately reset again. 0b = Contents of blinking memory registers LCDBMx remain unchanged 1b = Clear content of all blinking memory registers LCDBMx
1	LCDCLRM	RW	0h	Clear LCD memory Clears all LCD memory registers LCDMx. The bit is automatically reset when the LCD memory is cleared. 0b = Contents of LCD memory registers LCDMx remain unchanged 1b = Clear content of all LCD memory registers LCDMx
0	LCDDISP	RW	0h	Select LCD memory registers for display The bit is cleared in LCDBLKMODx = 01 and LCDBLKMODx = 10 or if a mux mode ≥5 is selected and cannot be changed by software. When LCDBLKMODx = 11, this bit reflects the currently displayed memory but cannot be changed by software. When returning to LCDBLKMODx = 00 the bit is cleared. 0b = Display content of LCD memory registers LCDMx 1b = Display content of LCD blinking memory registers LCDBMx

36.3.5 LCDCVCTL Register

LCD_C Voltage Control Register

NOTE: Settings for LCDREXT, R03EXT, LCDEXTBIAS, VLCDEXT, VLCDREFx, and LCD2B should be changed only while LCDON = 0.

Figure 36-16. LCDCVCTL Register

15	14	13	12	11	10	9	8
Reserved			VLCDx				Reserved
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r0
7	6	5	4	3	2	1	0
LCDREXT	R03EXT	LCDEXTBIAS	VLCDEXT	LCDCPEN	VLCDREFx		LCD2B
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 36-12. LCDCVCTL Register Description

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved
12-9	VLCDx	RW	0h	Charge pump voltage select. LCDCPEN must be 1 for the charge pump to be enabled. VCC is used for V_{LCD} when VLCDx = 0000 and VLCDREFx = 00 and VLCDEXT = 0. 0000b = Charge pump disabled 0001b = If VLCDREFx = 00 or 10: $V_{LCD} = 2.60\text{ V}$; If VLCDREFx = 01 or 11: $V_{LCD} = 2.17 \times V_{REF}$ 0010b to 1110b = If VLCDREFx = 00 or 10: $V_{LCD} = 2.60\text{ V} + (VLCDx - 1) \times 0.06\text{ V}$; If VLCDREFx = 01 or 11: $V_{LCD} = 2.17 \times V_{REF} + (VLCDx - 1) \times 0.05 \times V_{REF}$ 1111b = If VLCDREFx = 00 or 10: $V_{LCD} = 2.60\text{ V} + (15 - 1) \times 0.06\text{ V} = 3.44\text{ V}$; If VLCDREFx = 01 or 11: $V_{LCD} = 2.17 \times V_{REF} + (15 - 1) \times 0.05 \times V_{REF} = 2.87 \times V_{REF}$
8	Reserved	R	0h	Reserved
7	LCDREXT	RW	0h	V2 to V4 voltage on external Rx3 pins. This bit selects the external connections for voltages V2 to V4 with internal bias generation (LCDEXTBIAS = 0). The bit is don't care if external biasing is selected (LCDEXTBIAS = 1). NOTE: Should be changed only while LCDON = 0. 0b = Internally generated V2 to V4 are not switched to pins (LCDEXTBIAS = 0) 1b = Internally generated V2 to V4 are switched to pins (LCDEXTBIAS = 0)
6	R03EXT	RW	0h	V5 voltage select. This bit selects the external connection for the lowest LCD voltage. R03EXT is ignored if there is no R03 pin available. NOTE: Should be changed only while LCDON = 0. 0b = V5 is VSS 1b = V5 is sourced from the R03 pin
5	LCDEXTBIAS	RW	0h	V2 to V4 voltage select. This bit selects the generation for voltages V2 to V4. NOTE: Should be changed only while LCDON = 0. 0b = V2 to V4 are generated internally 1b = V2 to V4 are sourced externally and the internal bias generator is switched off
4	VLCDEXT	RW	0h	V_{LCD} source select NOTE: Should be changed only while LCDON = 0. 0b = V_{LCD} is generated internally 1b = V_{LCD} is sourced externally

Table 36-12. LCDCVCTL Register Description (continued)

Bit	Field	Type	Reset	Description
3	LCDCPEN	RW	0h	Charge pump enable 0b = Charge pump disabled 1b = Charge pump enabled when V_{LCD} is generated internally (VLCDEXT = 0) and VLCDx > 0 or VLCDREFx > 0
2-1	VLCDREFx	RW	0h	Charge pump reference select. If LCDEXTBIAS = 1 or LCDREXT = 1, settings 01, 10, and 11 are not supported; the internal reference voltage is used instead. NOTE: Should be changed only while LCDON = 0. 00b = Internal reference voltage 01b = External reference voltage 10b = Internal reference voltage switched to external pin LCDREF/R13 11b = Reserved (defaults to external reference voltage)
0	LCD2B	RW	0h	Bias select. LCD2B is ignored in static mode or mux modes ≥ 5 . NOTE: Should be changed only while LCDON = 0. 0b = 1/3 bias 1b = 1/2 bias

36.3.6 LCDPCTL0 Register

LCD_C Port Control Register 0

NOTE: Settings for LCDSx should be changed only while LCDON = 0.

Figure 36-17. LCDPCTL0 Register

15	14	13	12	11	10	9	8
LCDS15	LCDS14	LCDS13	LCDS12	LCDS11	LCDS10	LCDS9	LCDS8
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
LCDS7	LCDS6	LCDS5	LCDS4	LCDS3	LCDS2	LCDS1	LCDS0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 36-13. LCDPCTL0 Register Description

Bit	Field	Type	Reset	Description
15-0	LCDSx	RW	0h	LCD segment line x enable. This bit affects only pins with multiplexed functions. Dedicated LCD pins are always LCD function. NOTE: Settings for LCDSx should be changed only while LCDON = 0. 0b = Multiplexed pins are port functions 1b = Pins are LCD functions

36.3.7 LCDPCTL1 Register

LCD_C Port Control Register 1

NOTE: Settings for LCDSx should be changed only while LCDON = 0.

Figure 36-18. LCDPCTL1 Register

15	14	13	12	11	10	9	8
LCDS31	LCDS30	LCDS29	LCDS28	LCDS27	LCDS26	LCDS25	LCDS24
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
LCDS23	LCDS22	LCDS21	LCDS20	LCDS19	LCDS18	LCDS17	LCDS16
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 36-14. LCDPCTL1 Register Description

Bit	Field	Type	Reset	Description
15-0	LCDSx	RW	0h	LCD segment line x enable. On devices supporting a maximum of 192 segments LCDS31 is reserved, if COM7 to COM1 are shared with segments. If COM7 to COM1 are not shared with segments LCDS24 to LCDS31 are reserved. This bit affects only pins with multiplexed functions. Dedicated LCD pins are always LCD function. NOTE: Settings for LCDSx should be changed only while LCDON = 0. 0b = Multiplexed pins are port functions 1b = Pins are LCD functions

36.3.8 LCDPCTL2 Register

LCD_C Port Control Register 2 (≥ 256 Segments)

NOTE: Settings for LCDSx should be changed only while LCDON = 0.

Figure 36-19. LCDPCTL2 Register

15	14	13	12	11	10	9	8
LCDS47	LCDS46	LCDS45	LCDS44	LCDS43	LCDS42	LCDS41	LCDS40
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
LCDS39	LCDS38	LCDS37	LCDS36	LCDS35	LCDS34	LCDS33	LCDS32
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 36-15. LCDPCTL2 Register Description

Bit	Field	Type	Reset	Description
15-0	LCDSx	RW	0h	LCD segment line x enable. On devices supporting a maximum of 256 segments LCDS39 to LCDS47 are reserved, if COM7 to COM1 are shared with segments. If COM7 to COM1 are not shared with segments the complete register LCDPCTL2 is not available. On devices supporting a maximum of 320 segments, LCDS47 is reserved if COM7 to COM1 are shared with segments. If COM7 to COM1 are not shared with segments, LCDS40 to LCDS47 are reserved. This bit affects only pins with multiplexed functions. Dedicated LCD pins are always LCD function. NOTE: Settings for LCDSx should be changed only while LCDON = 0. 0b = Multiplexed pins are port functions 1b = Pins are LCD functions

36.3.9 LCDPCTL3 Register

LCD_C Port Control Register 2 (384 Segments, COMs Shared With Segments)

NOTE: Settings for LCDSx should be changed only while LCDON = 0.

Figure 36-20. LCDPCTL3 Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved		LCDS53	LCDS52	LCDS51	LCDS50	LCDS49	LCDS48
r0	r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 36-16. LCDPCTL3 Register Description

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved
5-0	LCDSx	RW	0h	LCD segment line x enable. This bit affects only pins with multiplexed functions. Dedicated LCD pins are always LCD function. NOTE: Settings for LCDSx should be changed only while LCDON = 0. 0b = Multiplexed pins are port functions 1b = Pins are LCD functions

36.3.10 LCDCCPCTL Register

LCD_C Charge Pump Control Register

Figure 36-21. LCDCCPCTL Register

15	14	13	12	11	10	9	8
LDCPCLK SYNC	Reserved						
rw-0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
LCDCPDISx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 36-17. LCDCCPCTL Register Description

Bit	Field	Type	Reset	Description
15	LDCPCLKSYNC	RW	0h	LCD charge pump clock synchronization (device specific). The charge pump clock is synchronized to a device specific clock (device-specific) when the respective clock source is enabled and does not indicate a fault with its fault signal - if available. 0b = Synchronization disabled 1b = Synchronization enabled
14-8	Reserved	R	0h	Reserved
7-0	LDCPDISx	RW	0h	LCD charge pump disable (number of implemented bits and connected function is device-specific) 0b = Connected function cannot disable charge pump 1b = Connected function can disable charge pump

36.3.11 LCDCIV Register

LCD_C Interrupt Vector Register

Figure 36-22. LCDCIV Register

15	14	13	12	11	10	9	8
LCDCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
LCDCIVx							
r0	r0	r0	r0	r0	r0	r0	r0

Table 36-18. LCDCIV Register Description

Bit	Field	Type	Reset	Description
15-0	LCDCIVx	R	0h	LCD_C interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: No capacitor connected; Interrupt Flag: LCDNOCAPIFG; Interrupt Priority: Highest 04h = Interrupt Source: Blink, segments off; Interrupt Flag: LCDBLKOFFIFG 06h = Interrupt Source: Blink, segments on; Interrupt Flag: LCDBLKONIFG 08h = Interrupt Source: Frame interrupt; Interrupt Flag: LCDFRMIFG; Interrupt Priority: Lowest

Extended Scan Interface (ESI)

The Extended Scan Interface (ESI) peripheral automatically scans sensors and measures linear or rotational motion. This document describes the Extended Scan interface.

Topic	Page
37.1 ESI Introduction	966
37.2 ESI Operation.....	967
37.3 ESI Registers	993

37.1 ESI Introduction

The ESI module is used to automatically measure linear or rotational motion with the lowest possible power consumption. The ESI consists of following blocks: the analog front end (AFE1 and AFE2), the preprocessing unit (PPU), the processing state machine (PSM) with its associated RAM, the timing state machine (TSM), and the Timer_A Output Stage. The analog front end stimulates the sensors, senses the signal levels, and converts them into their digital representation. The digital representations of a measurement sequence are stored in the preprocessing unit. The stored digital signals are passed into the processing state machine. The processing state machine is used to analyze and count rotation or motion. The timing state machine controls the analog front end, the preprocessing unit, and the processing state machine. The Timer_A Output Stage generates signals that are fed into Timer_A capture inputs; this allows for time measurements (for example, LC-sensor envelope test).

The ESI features include:

- Support for different types of sensors
 - LC sensors
 - Resistive sensors such as Hall-effect or giant magnetoresistive (GMR) sensors
 - Optical sensors
 - And more
- Measurement of sensor signal envelope
- Measurement of sensor signal oscillation amplitude
- Direct analog input for analog-to-digital conversion
- Direct digital input for digital sensors such as optical decoders
- Support for quadrature decoding

Figure 37-1 shows the ESI module block diagram.

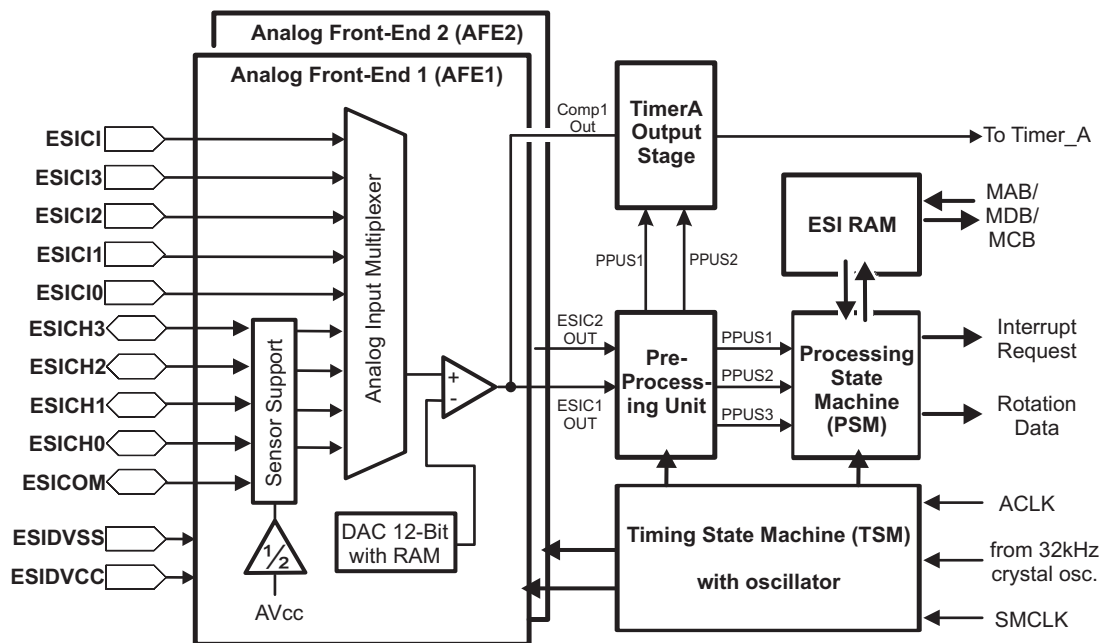


Figure 37-1. ESI Block Diagram

37.2 ESI Operation

The ESI is configured with user software. The setup and operation of the ESI is described in the following sections.

37.2.1 ESI Analog Front End

There are two analog front ends available in the ESI. The analog front end AFE1 provides sensor excitation and sample-and-hold circuit for measurements. The analog front end is automatically controlled by the timing state machine (TSM) according to the information in the timing state machine table. [Figure 37-2](#) shows the analog front end block diagram.

NOTE: Timing State Machine Signals

Throughout this chapter, signals from the ESITSMx registers (x = 0 to 31) are noted in the signal name with (tsm). For example, the signal ESIEX(tsm) comes from the actual active ESITSMx register.



Copyright © 2012–2020, Texas Instruments Incorporated

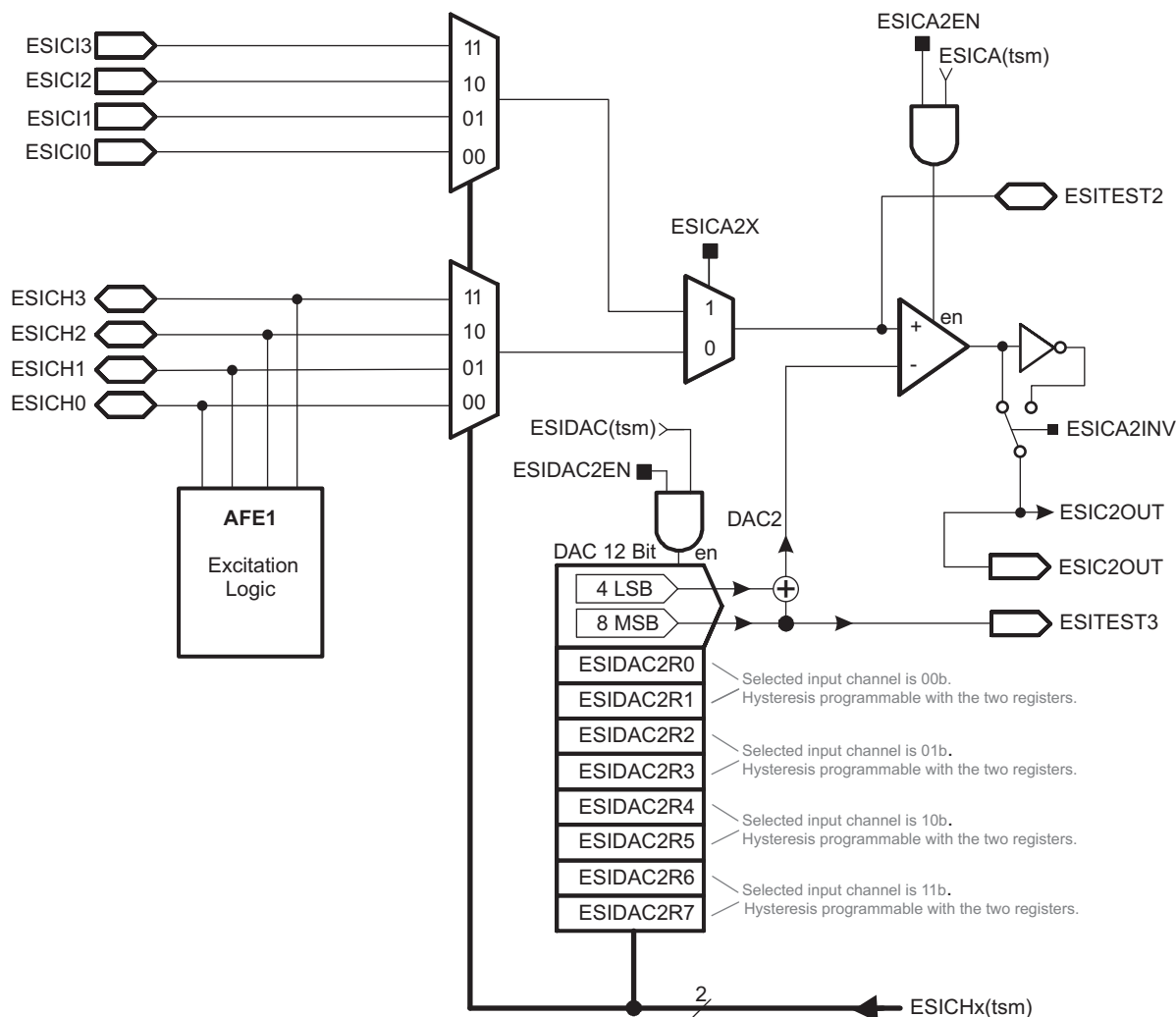


Figure 37-3. ESI Analog Front End AFE2 Block Diagram

37.2.1.1 Excitation

The excitation circuitry is used to excite the LC sensors or to power the resistor dividers. The excitation circuitry is shown in [Figure 37-4](#) for one LC sensor connected. When the ESITEN bit is set and the ESISH bit is cleared, the excitation circuitry is enabled and the sample-and-hold circuitry is disabled.

When the ESIEX(tsm) signal from the timing state machine is high, the ESICHx input of the selected channel is connected to ground (ESIDVSS pin) and the ESICOM input is connected to the mid-voltage generator to excite the sensor. The ESILCEN(tsm) signal must be high for excitation. While one channel is excited and measured, all other channels are automatically disabled. Only the selected channel is excited and measured.

The excitation period should be long enough to overload the LC sensor slightly. After excitation the ESICHx input is released from ground when ESIX(tsm) = 0, and the LC sensor can oscillate freely. The oscillations swing above the positive supply but are clipped by the protection diode to the positive supply voltage plus one diode drop. This gives consistent maximum oscillation amplitude.

At the end of the measurement, the sensor should be damped by setting $ESILCEN(tsm) = 0$ to remove any residual energy before the next measurement.

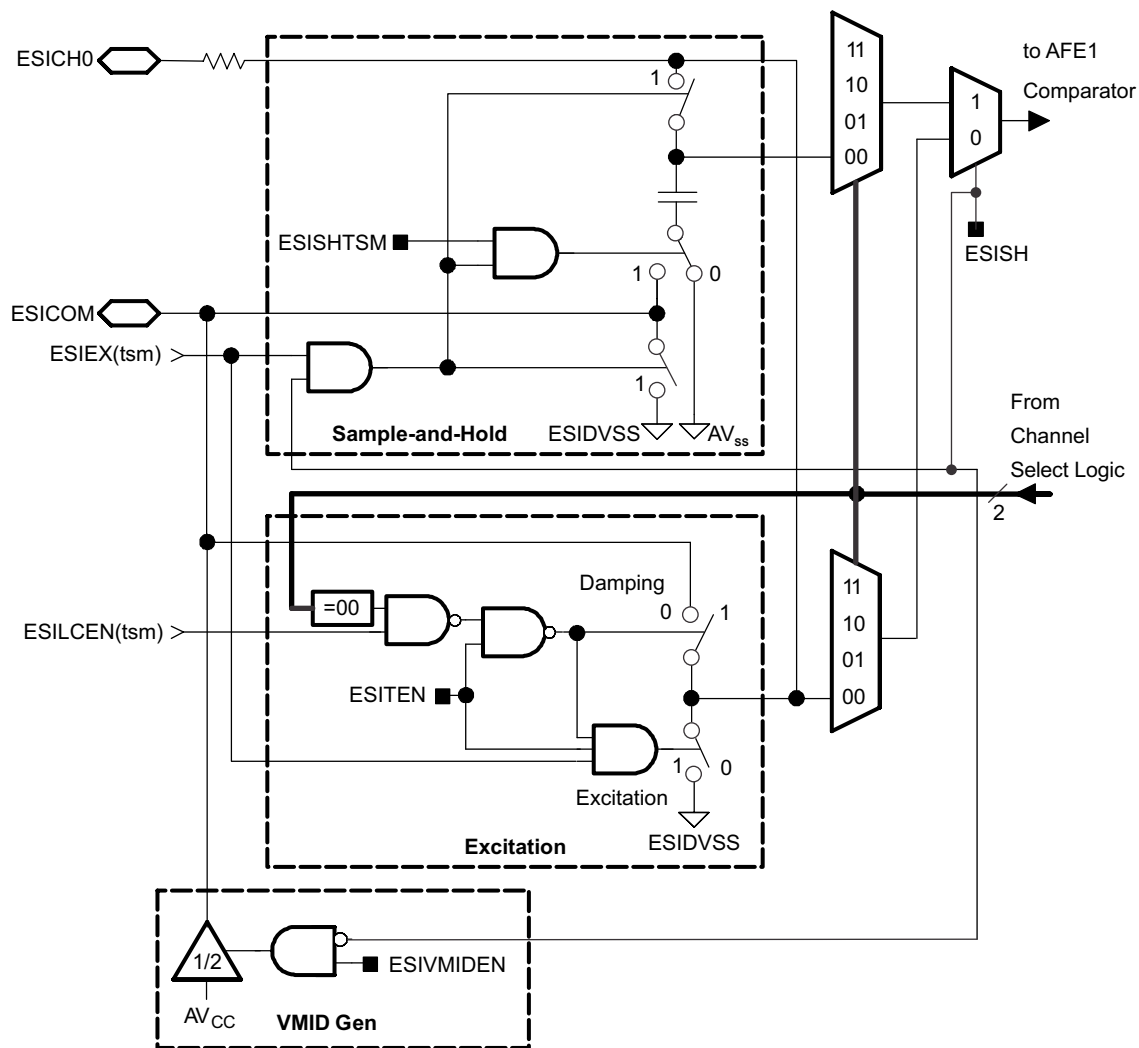


Figure 37-4. Excitation and Sample-And-Hold Circuitry

37.2.1.2 Mid-Voltage Generator

The mid-voltage generator is on when ESIVMIDEN = 1 and allows the LC sensors to oscillate freely. The mid-voltage generator requires a maximum of 6 ms to settle and requires ACLK to be active and operating at 32768 Hz.

37.2.1.3 Sample-And-Hold

Note that the sample-and-hold circuit is only available in the analog front-end AFE1.

The sample-and-hold is used to sample the sensor voltage to be measured. Figure 37-4 shows the sample-and-hold circuitry. When ESISH = 1 and ESITEN = 0, the sample-and-hold circuitry is enabled and the excitation circuitry and mid-voltage generator are disabled. The sample-and-hold is used for resistive dividers or for other analog signals that should be sampled.

Up to four resistor dividers can be connected to ESICHx and ESICOM. AVCC and ESICOM are the common positive and negative potentials for all connected resistor dividers. When ESIEX(tsm) = 1, ESICOM is connected to ESIDVSS and allows current to flow through the dividers. This charges the capacitors of each sample-and-hold circuit to the divider voltages. All resistor divider channels are sampled simultaneously. When ESIEX(tsm) = 0, the sample-and-hold capacitor is disconnected from the resistor divider, and ESICOM is disconnected from ESIDVSS. After sampling, each channel can be measured sequentially using the channel select logic, the comparator, and the DAC.

The selected ESICHx input can be modeled as an RC low-pass filter during the sampling time, t_{sample} , as shown in Figure 37-5. An internal MUX-on input resistance $R_{i(\text{ESICHx})}$ (3 kΩ maximum) in series with capacitor $C_{\text{SHC}(\text{ESICHx})}$ (9 pF maximum) is seen by the resistor-divider. The capacitor voltage V_C must be charged to within one-half LSB of the resistor divider voltage for an accurate 12-bit conversion. See the device-specific data sheet for parameters.

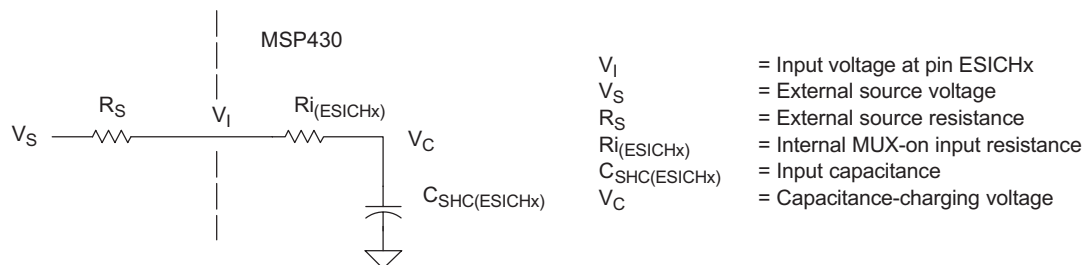


Figure 37-5. Analog Input Equivalent Circuit

The resistance of the source R_S and $R_{i(\text{ESICHx})}$ affect t_{sample} . Equation 18 can be used to calculate the minimum sampling time t_{sample} for a 12-bit conversion:

$$t_{\text{sample}} > (R_S + R_{i(\text{ESICHx})}) \times \ln(2^{13}) \times C_{\text{SHC}(\text{ESICHx})} \quad (18)$$

Substituting the values for $R_{i(\text{ESICHx})}$ and $C_{\text{SHC}(\text{ESICHx})}$ given above, the equation becomes:

$$t_{\text{sample}} > (R_S + 3\text{k}) \times 9.011 \times 9 \text{ pF} \quad (19)$$

For example, if R_S is 10 kΩ, t_{sample} must be greater than 1054 ns.

37.2.1.4 Direct Analog And Digital Inputs

By setting the ESICA1X or ESICA2X bit, external analog or digital signals can be connected directly to the particular comparator through the ESIClx inputs. This allows measurement capabilities for optical encoders and other sensors.

Both analog front-ends have own control bits to select either the sensor input (ESICHx) or the direct input (ESIClx). This allows to use different input settings (selection of ESIClx or ESICHx) for AFE1 and AFE2.

37.2.1.5 Comparator Input Selection And Output Bit Selection

The ESICA1X and ESISH bits within AFE1 select between the ESIClx channels and the ESICHx channels for the comparator input as described in Table 37-1.

The AFE2's ESICA2X bit selects either ESIClx channels (ESICA2X = 1) or the ESICHx channels (ESICA2X = 0) for the analog front-end AFE2.

Table 37-1. ESICAX and ESISH Input Selection

ESICA1X	ESISH	Operation
0	0	ESICHx and excitation circuitry is selected within AFE1
0	1	ESICHx and sample-and-hold circuitry is selected within AFE1
1	X	ESIClx inputs are selected within AFE1

Note that the test insertion feature is only available for AFE1. The TESTDX signal and ESITESTS1(tsm) signal select between the ESIOUTx output bits and the ESITCHOUTx output bits for the comparator output as described in Table 37-2. TESTDX is controlled by the ESITESTD bit.

Table 37-2. Selected Output Bits

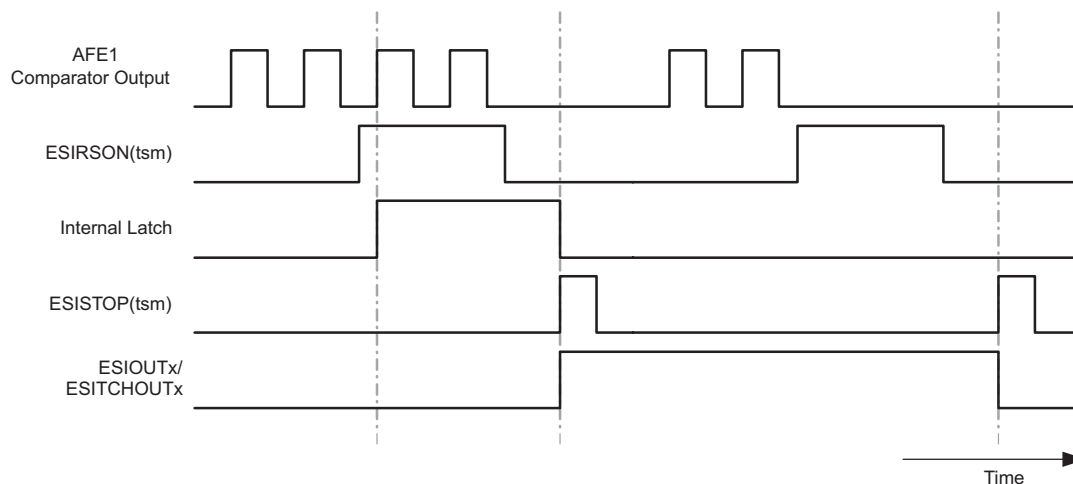
TESTDX	ESICHx(tsm)	ESITESTS1(tsm)	Selected Output Bit
0	00	X	ESIOUT0 and ESIOUT4
0	01	X	ESIOUT1 and ESIOUT5
0	10	X	ESIOUT2 and ESIOUT6
0	11	X	ESIOUT3 and ESIOUT7
1	X	0	ESITCHOUT0
1	X	1	ESITCHOUT1

When TESTDX = 0, the ESICHx(tsm) signals select which ESIClx or ESICHx channel is excited and connected to the comparator. The ESICHx(tsm) signals also select the corresponding output bit for the comparator result.

When TESTDX = 1, channel selection depends on the ESITESTS1(tsm) signal. When TESTDX = 1 and ESITESTS1(tsm) = 0, input channel selection is controlled with the ESITCH0x bits and the output bit is ESITCHOUT0. When TESTDX = 1 and ESITESTS1(tsm) = 1, input channel selection is controlled with the ESITCH1x bits and the output bit is ESITCHOUT1.

When AFE1's ESICA1X = 1, the ESICSEL and ESICl3 bits select between the ESIClx channels and the ESICl input, allowing storage of the comparator output for one input signal into the four output bits ESIOUT0 to ESIOUT3. This can be used to observe the envelope function of sensors.

The output logic is enabled by the ESIRSON(tsm) signal. When a comparator output is high while ESIRSON = 1, an internal latch is set. Otherwise the latch is reset. The latch output is written into the selected output bit with the rising edge of the ESISTOP(tsm) signal as shown in [Figure 37-6](#).


Figure 37-6. Analog Front-End Output Timing

37.2.1.6 Comparator and DAC

The analog input signals are converted into digital signals by the comparator and the programmable 12-bit DAC. The comparator compares the selected analog signal to a reference voltage generated by the DAC. If the voltage is above the reference, the comparator output is high. Otherwise, it is low. The comparator outputs of both analog front-ends can be individually inverted by setting ESICA1INV for AFE1 or ESICA2INV for AFE2. The comparator output is stored in the selected output bit and processed by the processing state machine to detect motion and direction.

The comparator and the DAC in both analog front-ends AFE1 and AFE2 are turned on and off by ESICA(tsm) signal and the ESIDAC(tsm) signal. In case, the AFE1's comparator or DAC are not needed they can be disabled by clearing the ESICA(tsm) and ESIDAC(tsm) control bits within ESITSM0 register. AFE2 is disabled when its comparator and DAC are disabled. This can be done by clearing the ESICA2EN and ESIDAC2EN bits. In case these bits are set the AFE2's comparator and DAC will be controlled by the ESICA(tsm) and ESIDAC(tsm) control bits.

For each input there are two DAC registers to set the reference level as listed in [Table 37-3](#). Together with the last stored output of the comparator, ESIOUTx, the two levels can be used as an analog hysteresis as shown in [Figure 37-7](#). The individual settings for the four inputs can be used to compensate for mismatches between the sensors.

Table 37-3. Selected DAC Registers

Analog Front-End	Selected Output Bit, ESIOUTx	Last Value of ESIOUTx	DAC Register Used
AFE1	ESIOUT0	0	ESIDAC1R0
		1	ESIDAC1R1
	ESIOUT1	0	ESIDAC1R2
		1	ESIDAC1R3
	ESIOUT2	0	ESIDAC1R4
		1	ESIDAC1R5
	ESIOUT3	0	ESIDAC1R6
		1	ESIDAC1R7
AFE2	ESIOUT4	0	ESIDAC2R0
		1	ESIDAC2R1
	ESIOUT5	0	ESIDAC2R2
		1	ESIDAC2R3
	ESIOUT6	0	ESIDAC2R4
		1	ESIDAC2R5
	ESIOUT7	0	ESIDAC2R6
		1	ESIDAC2R7

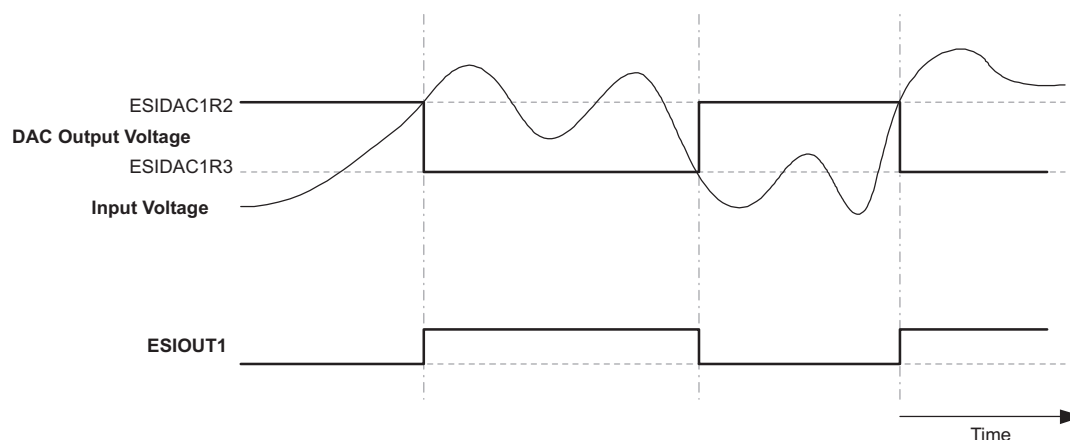


Figure 37-7. Analog Hysteresis With DAC Registers

When TESTDX = 1, the ESIDAC1R6 and ESIDAC1R7 registers are used as the comparator reference as described in [Table 37-4](#). Note that this feature is only available in AFE1.

Table 37-4. DAC Register Select When TESTDX=1

ESITESTS1(tsm)	DAC Register Used
0	ESIDAC1R6
1	ESIDAC1R7

37.2.1.7 Optional Comparator Offset Cancellation

The ESI's comparator has an offset that drifts over temperature and supply voltage. For some applications the specified offset error and offset drift may be acceptable - see device-specific data sheet. If the offset error is not acceptable, adding a comparator autozero cycle within the TSM sequence can minimize the offset error. After the inserted autozero TSM cycle, the comparator operates effectively with a zeroed offset.

Adding an ESITSMx state within the TSM sequence, which has the ESICAAZ bit selected (ESITSM.ESICLKAZSEL=1) and set, performs the comparator autozeroing. As long as the ESICAAZ bit is set the autozeroing is performed. This means, the length of the appropriate ESITSMx state defines the length of autozeroing. The following code excerpt shows how to include an autozeroing cycle within a TSM sequence. The example focus on ESICA and ESICAAZ control bits:

```
...
ESITSM5 = TSM_State5;    // TSM_State5 is a placeholder for any setting;
                                                                    //      Comparator is
disabled (ESICA=0, ESICAAZ=0)
ESITSM6 = ESICA + ESICAAZ + Length + TSM_State6;
                // comparator is switched on and at the same time the autozeroing is
                // performed. Length is defined by ESCLK and ESIREPEATx bits -
                // appropriate setting needed to meet autozeroing timing requirements;
                // see device-specific data sheet (ESICA=1, ESICAAZ=1)
ESITSM7 = ESICA + TSM_State7;
                // normal comparator operation: settle comparator
                // (ESICA=1, ESICAAZ=0)
ESITSM8 = ESICA + TSM_State8;
                // normal comparator operation: processing of comparator output signal
                // (ESICA=1, ESICAAZ=0)
...
```

37.2.2 ESI Timing State Machine

The TSM is a sequential state machine that cycles through the ESITSMx registers and controls the analog front end and sensor excitation automatically with no CPU intervention. The states are defined within a 32 x 16-bit memory, ESITSM0 to ESITSM31. The ESIEN bit enables the TSM. The ESI uses ACLK as its source for the low frequency clock signal ESILFCLK. When ESIEN = 0, the ACLK input divider, the TSM start flip-flop, and the TSM outputs are reset and the internal oscillator is stopped. The TSM block diagram is shown in [Figure 37-8](#).

A TSM sequence begins at ESITSM0 and ends when the TSM encounters a ESITSMx state with a set ESISTOP bit. When a state with a set ESISTOP bit is reached, the state counter is reset to zero and state processing stops. State processing re-starts at ESITSM0 with a software trigger (setting the ESISTART control bit), the next start condition when ESITSMRP = 0, or immediately when ESITSMRP = 1

After generation of the ESISTOP(tsm) pulse, the timing state machine will load and maintain the conditions defined in ESITSM0. For the case an LC sensor is used the ESILCEN(tsm) bit should be reset in ESITSM0 to ensure that all LC oscillators are shorted (damped) while no measurement sequence is in progress.

In case a TSM sequence is started with a software trigger, the ESISTART control bit is automatically cleared as soon as a TSM sequence is completed and the system is again in idle mode (ESITSM0 settings are used in idle mode).

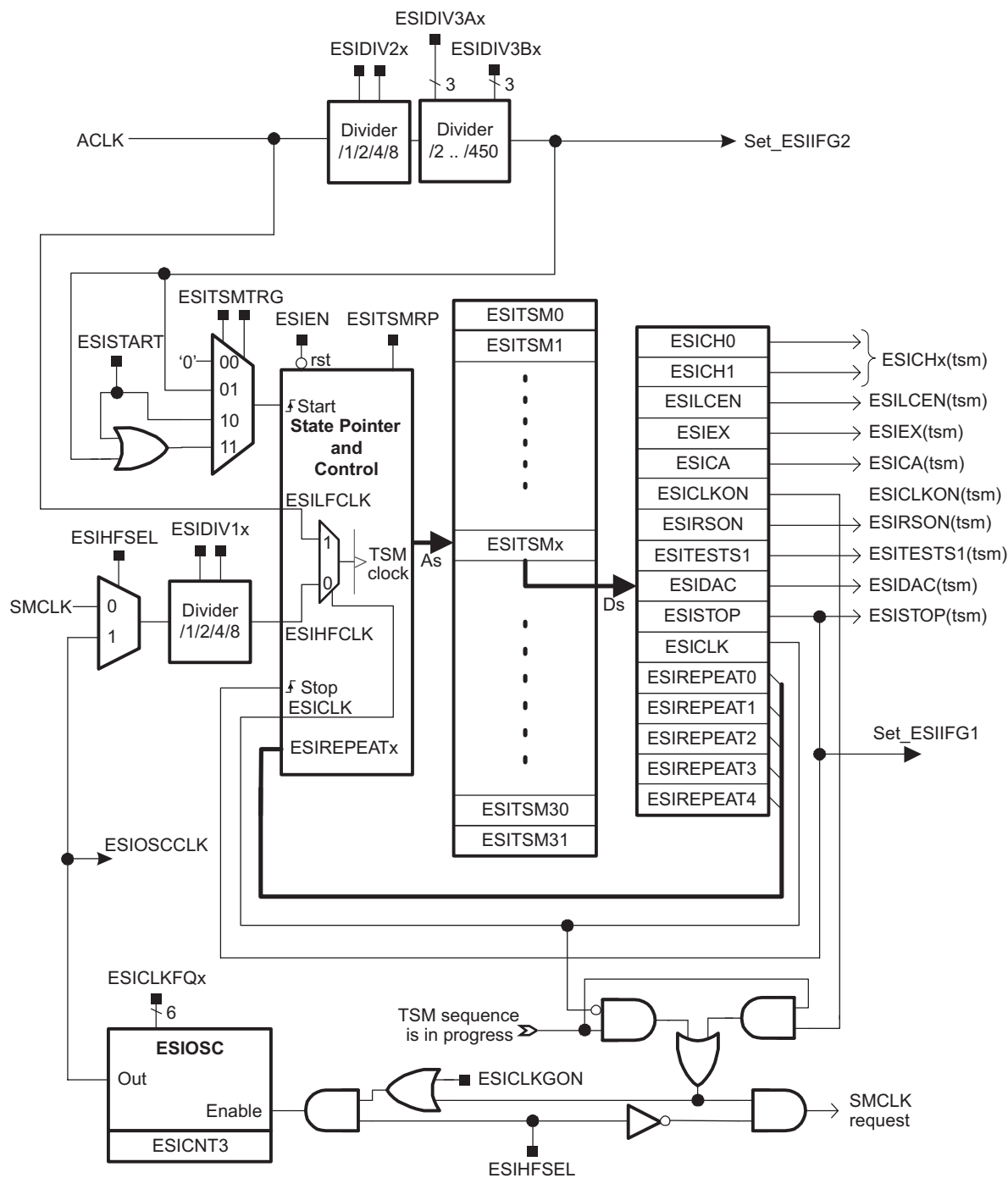


Figure 37-8. Timing State Machine Block Diagram

37.2.2.1 TSM Operation

Starting of a TSM sequence depends on the selected trigger. Possible triggers are the control bit ESISTART, the divided ACLK and an or combination of these two triggers (ESISTART or divided ACLK).

If the divided ACLK is chosen, the TSM automatically starts and re-starts periodically based on a divided ACLK start signal selected with the ESIDIV2x bits, the ESIDIV3Ax and ESIDIV3Bx bits when ESITSMRP = 0. For example, if ESIDIV2x, ESIDIV3Ax, and ESIDIV3Bx are configured to 270 ACLK cycles, then the TSM automatically starts every 270 ACLK cycles. When ESITSMRP = 1 the TSM restarts immediately with the ESITSM0 state at the end of the previous sequence i.e. with the next ACLK cycle after encountering a state with ESISTOP = 1. The ESIIFG2 interrupt flag is set when the TSM starts.

The ESIDIV2x, ESIDIV3Ax, and ESIDIV3Bx bits may be updated anytime during operation. When updated, the current TSM sequence will continue with the old settings until the last state of the sequence completes. The new settings will take affect at the start of the next sequence.

Setting ESISTART bit is another trigger for starting a TSM sequence. The ESISTART bit is set as long as the actual TSM sequence is in progress. As soon as the TSM sequence is completed and ESITSM0 register is again active for idle state configuration the ESISTART bit is automatically cleared. In case ESISTART is the only source for a start trigger (ESITSMTRG = 01) an ACLK synchronization sequence is performed, which may take up to 2.5 ACLK cycles. For all other cases no special synchronization is needed and TSM starts with the appropriate positive ACLK edge.

NOTE: It is important to set the ESISTOP(tsm) bit at least once the control registers ESITSM2 to ESITSM31. The ESISTOP(tsm) control bit ensures that a user-defined TSM sequence is terminated and the TSM progressing is switching into idle mode awaiting the next start trigger.

37.2.2.2 TSM Idle Condition Selectable With ESITSM0

ESITSM0 register is used for two different tasks. First, by definition ESITSM0 register is always the first ESITSMx register within a TSM sequence. The second purpose of ESITSM0 is to define the settings of the analog front ends during idle time; idle time means no TSM sequence is in progress.

When ESITSM0 defines the AFE1 and AFE2 settings in idle time only some of the ESITSMx control bits are functional. These functional bits are ESICLKON, ESIRSON, ESIEX, ESILCEN, and ESICHx. Some bits do not have any effect in idle mode, like ESIDAC, ESICA, ESIREPEATx bits and ESICLK bit; the ESIREPEATx bits and ESICLK bit are only utilized when ESITSM0 is used within a TSM sequence. ESIDAC and ESICA bits should be '0' in ESITSM0 state for reduced current consumption.

Note that changing ESITSM0 register gets effective not before the next TSM sequence is started. This has to be considered especially after powering up the device and doing the first initialization of the ESI.

37.2.2.3 TSM Control of the AFE

The TSM controls the AFE with the ESICHx, ESILCEN, ESIEX, ESICA, ESICLKON, ESIRSON, ESITESTS1, ESIDAC, ESISTOP, and ESICLK bits. When any of these bits are set, their corresponding signal(s), ESICHx(tsm), ESILCEN(tsm), ESIEX(tsm), ESICA(tsm), ESICLKON(tsm), ESIRSON(tsm), ESITESTS1(tsm), ESIDAC(tsm), ESISTOP(tsm), and ESICLK(tsm) are high for the duration of the state. Otherwise, the corresponding signal(s) are low.

37.2.2.4 TSM State Duration

The duration of each state is individually configurable with the ESIREPEATx bits. The duration of each state is ESIREPEATx + 1 times the selected clock source. For example, if a state were defined with ESIREPEATx = 3 and ESICLK = 1, the duration of that state would be 4 x ACLK cycles. Because of clock synchronization, the duration of each state is affected by the clock source for the previous state, as shown in [Table 37-5](#).

Table 37-5. TSM State Duration

ESICLK		State Duration, T
For Previous State	For Current State	
0	0	$T = (\text{ESIREPEATx} + 1) \times 1/f_{\text{ESIHCLK}}$
0	1	$(\text{ESIREPEATx}) \times 1/f_{\text{ACLK}} < T \leq (\text{ESIREPEATx} + 1) \times 1/f_{\text{ACLK}}$
1	0	$(\text{ESIREPEATx} + 1) \times 1/f_{\text{ESIHCLK}} \leq T < (\text{ESIREPEATx} + 3) \times 1/f_{\text{ESIHCLK}}$
1	1	$T = (\text{ESIREPEATx} + 1) \times 1/f_{\text{ACLK}}$

37.2.2.5 TSM State Clock Source Select

The TSM clock source is individually configurable for each state. The TSM can be clocked from ACLK or a high frequency clock selected with the ESICLK bit. When ESICLK = 1, ACLK is used for the state, and when ESICLK = 0, the high frequency clock is used. The high frequency clock can be sourced from SMCLK or the TSM internal oscillator, selected by the ESIHFSEL bit. The high-frequency clock can be divided by 1, 2, 4, or 8 with ESIDIV1x bits.

A set ESICLKON bit is used to turn on the selected high frequency clock source for the duration of the state, when it is not used for the state. If the DCO is selected as the high frequency clock source, it is automatically turned on, regardless of the low-power mode settings of the MSP430.

The TSM internal oscillator should be adjusted to the nominal frequency of 4.8 MHz. To realize this it can be tuned in nominal 3% steps from around 1 MHz to around 8MHz with the ESICLKQx. The frequency and the steps differ from unit to unit. See the device-specific data sheet for parameters.

The TSM internal oscillator frequency can be measured with ACLK. When ESIHFSEL = 1 and ESICLKON = 1 ESICNT3 is reset, and beginning with the next rising edge of ACLK, ESICNT3 counts the clock cycles of the internal oscillator. ESICNT3 counts the internal oscillator cycles for one ACLK period. Reading ESICNT3 while it is counting will result in reading 01h.

The ACLK is automatically turned on for the following cases, regardless of the low-power mode settings of the MSP430:

- ACLK is automatically turned on all the time when the divided ACLK signal (ESIDIV2x, ESIDIV3Ax, and ESIDIV3Bx dividers) is selected as trigger for starting a TSM sequence.
- While a TSM sequence is in progress the ACLK is automatically turned on.

37.2.2.6 TSM Stop Condition

A TSM sequence always starts with ESITSM0, uses some ESITSMx states to do a measurement, and ends at the subsequent ESITSMx state with a set ESISTOP bit (stop state). The duration of this last state (stop state) is always one ESIHFCLK cycle regardless of the ESICLK or ESIREPEATx settings. The ESIIFG1 interrupt flag is set at when the TSM encounters a state with a set ESISTOP bit.

37.2.2.7 TSM Test Cycles

For calibration purposes, to detect sensor drift, or to measure signals other than the sensor signals, a test cycle may be inserted between TSM cycles by setting the ESITESTD bit. The time between the TSM cycles is not altered by the test cycle insertion as shown in [Figure 37-9](#). At the end of the test cycle the ESITESTD bit is automatically cleared. The TESTDX signal is active during the test cycle to control input and output channel selection. TESTDX is generated after the ESITESTD bit is set and the next TSM sequence completes.

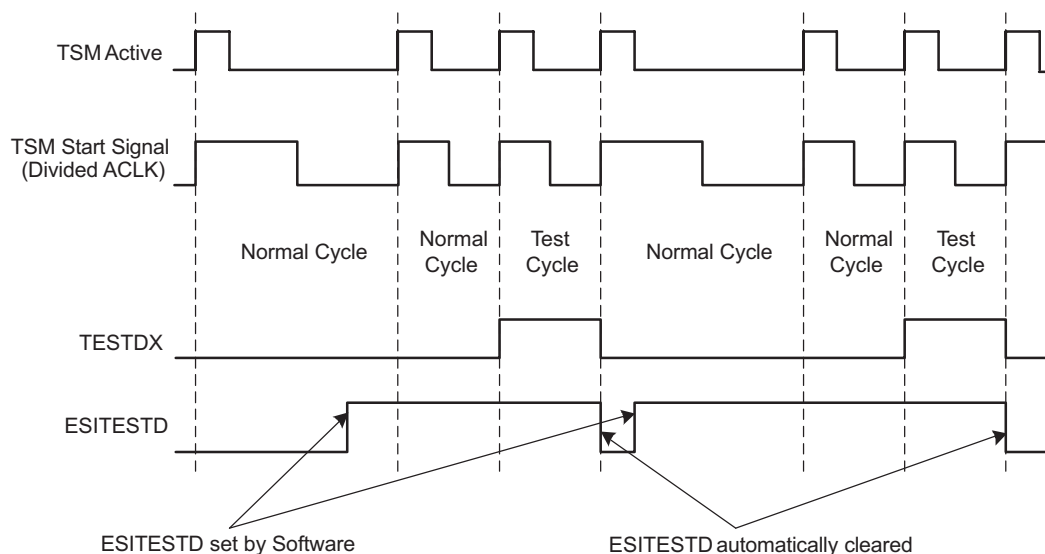


Figure 37-9. Test Cycle Insertion

37.2.2.8 TSM Example

Figure 37-10 shows an example for a TSM sequence. The TSMx register values for the example are shown in Table 37-6. ACLK and ESIHFCLK are not drawn to scale. The TSM sequence starts with ESITSM0 and ends with a set ESISTOP bit in ESITSM9. Only the ESITSM5 to ESITSM9 states are shown.

Table 37-6. TSM Example Register Values

TSMx Register	TSMx Register Contents
ESITSM5	0100Ah
ESITSM6	00402h
ESITSM7	01912h
ESITSM8	00952h
ESITSM9	00200h

The example also shows the affects of the clock synchronization when switching between ESIHFCLK and ACLK. In state ESITSM6, ESICLK is set, whereas in the previous state and the successive state, ESICLK is cleared. The waveform shows the duration of ESITSM6 is less than one ACLK cycle and the duration of state ESITSM7 is up to one ESIHFCLK period longer than configured by the ESIREPEATx bits.

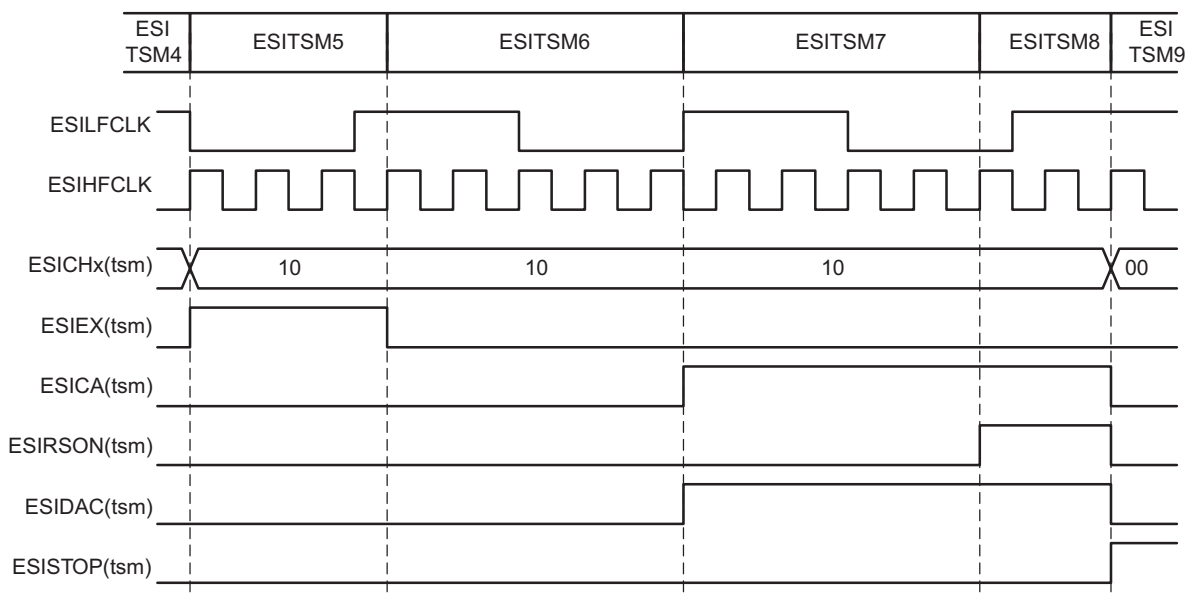


Figure 37-10. Timing State Machine Example

37.2.3 ESI Pre-Processing and State Storage

The Pre-Processing Unit (PPU) stores the measurement results of a TSM sequence. Beside this it also allows to select up to three signals that are processed by the Processing State Machine (PSM).

Up to four regular measurements and two test insertion measurements could sequentially be done within one TSM sequence. When ESIRSON(tsm) is high the comparator output signal is latched in the PPU's State Storage block. The State Storage consist of several latches. The output of these latches can be read from the ESIOUTx and ESITCHOUTx bits located in ESIPPU control register. Each input channel has its own latch. The ESICLKx(tsm), ESITCH0x, or ESITCH1x bits define which of the latches is used.

The block diagram of the Pre-Processing Unit is shown in [Figure 37-11](#).

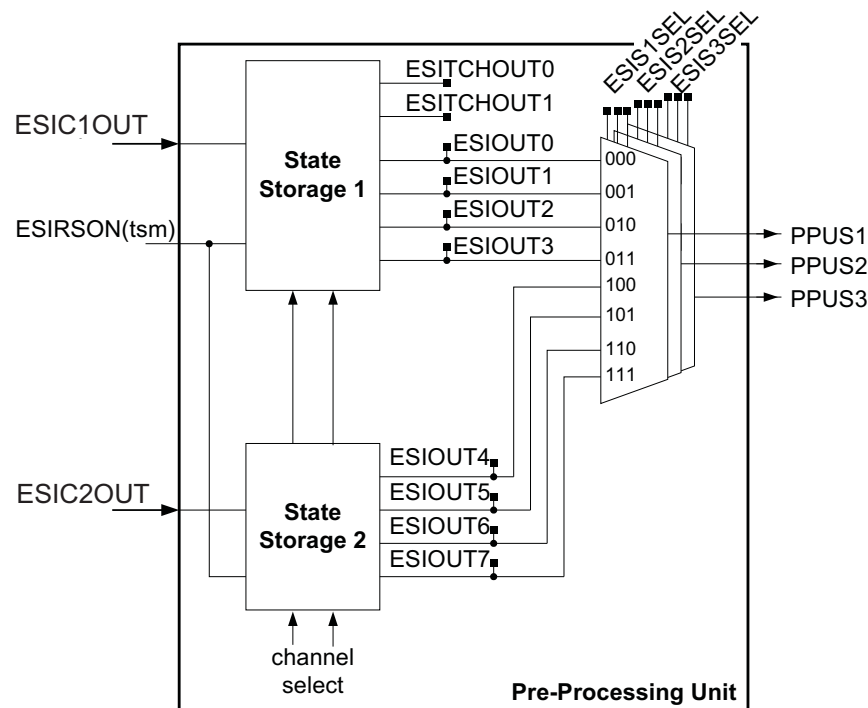


Figure 37-11. Pre-Processing Unit

37.2.4 TimerA Output Stage

The comparator output of the analog front end AFE1, the ESIEX(tsm) signal, and two preprocessing unit outputs PPUS1 and PPUS2 are connected to a Timer_A's capture inputs through the ESI's Timer_A output stage, shown in Figure 37-12. There are two different modes that are selected by the ESICS bit. The Timer_A Output Stage provides the ESIOx signals to one of the device's Timer_A module. See the device-specific data sheet for connection of these signals.

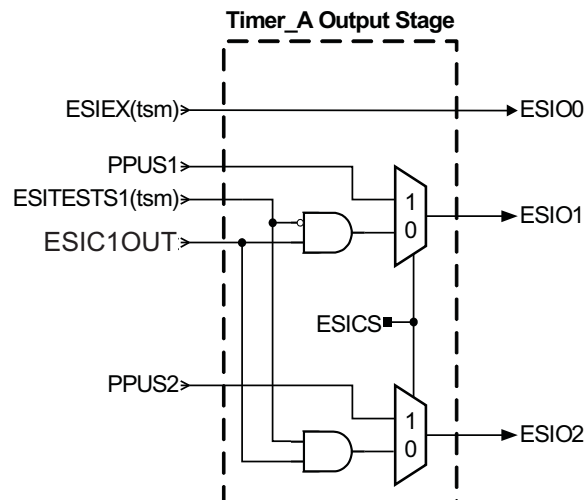


Figure 37-12. Timer_A Output Stage of the Analog Front End

When ESICS = 0, the ESIEX(tsm) signal and the comparator output can be selected as inputs to different Timer_A capture/compare registers. This can be used to measure the time between excitation of a sensor and the last oscillation that passes through the comparator or to perform a slope A/D conversion.

37.2.5 ESI Processing State Machine

Figure 37-13. ESI Processing State Machine Block Diagram

37.2.5.1 PSM Operation

Copyright © 2012–2020, Texas Instruments Incorporated

The current-state and next-state logic are reset while the ESI is disabled. The ESI allow selecting either two signals or three signals for the processing. When the ESI is enabled following scenarios do exist for first processing:

- Two input signals are chosen (ESIV2SEL = 1):
The byte stored at addresses 0 within PSM State Table (ESI RAM) will be loaded first when the ESI is enabled.
- Three input signals are chosen (ESIV2SEL = 0):
The byte stored at addresses 0 within PSM State Table (ESI RAM) will be loaded first when the ESI is enabled.

Signals PPUS1 and PPUS2 form a 2-bit offset (ESIV2SEL = 1) and signals PPUS1, PPUS2, and PPUS3 form a 3-bit offset (ESIV2SEL = 0) added to the base address of the PSM State Table to determine the byte loaded to the PSM current-state output latch. For example, when two input signals are chosen (ESIV2SEL = 1) and PPUS2 = 1, and PPUS1 = 0, the byte loaded by the PSM will be at the address <current address of PSM State Table> + 2. The next byte and further subsequent bytes are determined by the next state calculations and are calculated by the PSM based on the state table contents and the values of signals PPUS1 and PPUS2.

The PSM needs two TSM clock cycles to complete the processing of the measurement results from a single TSM sequence.

37.2.5.2 ESI RAM

The purpose of the ESI RAM is to store the user-defined PSM table. The ESI RAM can be accessed by PSM or CPU. CPU write and read access to ESI RAM is only possible when ESI is disabled (ESIEN = 0). Any CPU write access to ESI RAM is ignored while ESI is active (ESIEN = 1). A CPU read access to ESI RAM is not possible while ESI is active; in this case the CPU would read a 0x00 (byte access) or 0x0000 (word access).

NOTE: The ESI RAM does not support stack usage. This means, stack pointer should not point to ESI RAM addresses.

The ESI RAM start address (base address) can be found in the device-specific data sheet (see *Peripheral File Map* section).

37.2.5.3 Next State Calculation

Either bit 0 (Q0) or signal PPUS3, and bits 3-5 (Q3, Q4, Q5), and, if enabled by ESIQ6EN, bit 6 (Q6) are used together with the signals PPUS1, PPUS2, and optional PPUS3 to calculate the next state. When ESIQ6EN = 1, Q6 is used in the next-state calculation. The next state is:

0	Q6	Q5	Q4	Q3	Q0 or PPUS3	PPUS2	PPUS1
↓	↓	↓	↓	↓	↓	↓	↓
V7	V6	V5	V4	V3	V2	V1	V0

If ESIQ7TRG = 0, the Q7 bit can be used to generate an interrupt (ESIIFG7).

Enabling Q7 as trigger (ESIQ7TRG = 1) causes the following functionality: When Q7 = 0, the PSM state is updated by the falling edge of the ESISTOP(tsm) at the end of a TSM sequence. After updating the current state the PSM moves the corresponding state table entry to the output latch. When Q7 = 1, the next state is calculated immediately without waiting for the next falling edge of ESISTOP(tsm). The state is then updated with the next ESIOSC cycle. The worst-case time between state transitions in this case is 6 ESIOSC cycles.

37.2.5.4 PSM Counters

The PSM has three 16-bit counters ESICNT0, ESICNT1, and ESICNT2. ESICNT0 is updated with Q1, ESICNT1 is updated with Q1 and Q2, and ESICNT2 is updated with Q2. The counters can be read from the ESICNT0, ESICNT1, and ESICNT2 registers. The different counters can individually be reset by setting the ESICNTxRST control bits. When ESIEN = 0, all counters are held in reset.

ESICNT0 increments based on Q1. When ESICNT0EN = 1, ESICNT0 increments on a transition to a state where bit Q1 is set.

ESICNT1 can increment or decrement based on Q1 and Q2. When ESICNT1EN = 1, ESICNT1 decrements on a transition to a state where bit Q2 is set and it increments on a transition to a state where bit Q1 is set. In case both bits Q1 and Q2 are set on a state transition, ESICNT1 does not increment or decrement.

ESICNT2 decrements based on Q2. When ESICNT2EN = 1, ESICNT2 decrements on a transition to a state where bit Q2 is set. On the first count after a reset ESICNT2 will roll over from zero to 65535 (0FFFFh).

When the next state is calculated to be the same state as the current state, the counters ESICNT0, ESICNT1, and ESICNT2 are incremented or decremented according to Q1 and Q2 at the state transition. For example, if the current state is 05h and Q2 is set, and if the next state is calculated to be 05h, the transition from state 05h to 05h will decrement ESICNT2 if ESICNT2EN = 1.

NOTE: A read from any ESICNTx register should occur while PSM counters are not triggered. This can be realized by reading the ESI counters in the ESIIFG1 interrupt service routine and choosing appropriate timing of TSM sequences. Alternatively, the ESI counters may be read multiple times, and a majority vote taken in software to determine the correct reading.

37.2.5.5 Simplest State Machine

Figure 37-14 shows the simplest state machine that can be realized with the PSM. The following code shows the corresponding state table.

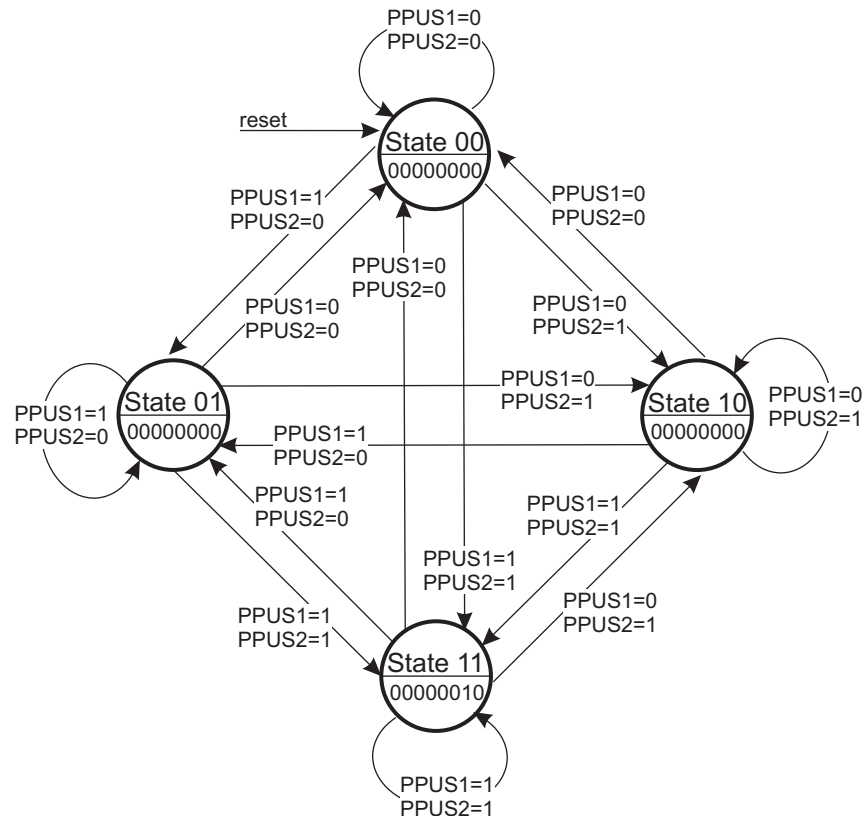


Figure 37-14. Simplest PSM State Diagram (ESIV2SEL=1)

```
; Simplest State Machine Example
SIMPLEST_PSM db 000h ; State 00 (State Table Index 0)
               db 000h ; State 01 (State Table Index 1)
               db 000h ; State 10 (State Table Index 2)
               db 002h ; State 11 (State Table Index 3)
```

If the PSM is in state 01 of the simplest state machine and the PSM has loaded the corresponding byte at index 01h of the state table:

Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
0	0	0	0	0	0	0	0

For this example, PPUS1 and PPUS2 are set at the end of the next TSM sequence. To calculate the next state the bits Q5 - Q3 and Q0 of the state 01 table entry, together with the PPUS1 and PPUS2 signals are combined to form the next state:

V7	V6 (Q6)	V5 (Q5)	V4 (Q4)	V3 (Q3)	V2 (Q0)	V1 (PPUS2)	V0 (PPUS1)
0	0	0	0	0	0	1	1

The state table entry for state 11 is loaded at the next state transition:

V7	V6 (Q6)	V5 (Q5)	V4 (Q4)	V3 (Q3)	V2 (Q0)	V1 (PPUS2)	V0 (PPUS1)
0	0	0	0	0	0	1	0

Q1 is set in state 11, so ESICNT1 will be incremented.

More complex state machines can be built by combining simple state machines to meet the requirements of specific applications.

37.2.6 ESI Debug Register

The Scan IF peripheral has several ESIDEBUGx registers for debugging and development.

- Reading ESIDEBUG1 shows the last address read by the PSM.
- Reading ESIDEBUG2 shows the index of the TSM and the PSM bits Q7 to Q0.
- Reading ESIDEBUG3 shows the TSM output.
- Reading ESIDEBUG4 shows which DAC1 register is selected and its contents.
- Reading ESIDEBUG5 shows which DAC2 register is selected and its contents.

37.2.7 ESI Interrupts

The Extended Scan IF has one interrupt vector for nine interrupt flags listed in [Table 37-7](#). Each interrupt flag has its own interrupt enable bit. When an interrupt is enabled, and the GIE bit is set, the interrupt flag will generate an interrupt. The interrupt flags are not automatically cleared. They must be cleared with software. The interrupt vector register ESIV is used to determine which interrupt flags requested an interrupt.

Table 37-7. ESI Interrupts

Interrupt Flag	Interrupt Condition
ESIIFG0	ESIIFG0 is set by one of the ESIOUT0 to ESIOUT3 outputs selected with the ESIIFGSET1x bits.
ESIIFG1	ESIIFG1 is set by the rising edge of the ESISTOP(tsm) signal.
ESIIFG2	ESIIFG2 is set at the start of a TSM sequence.
ESIIFG3	ESIIFG3 is set at different count intervals of the ESICNT1 counter, selected with the ESITHR1 and ESITHR2 registers.
ESIIFG4	ESIIFG4 is set at different count intervals of the ESICNT2 counter, selected with the ESIS2x bits.
ESIIFG5	ESIIFG5 is set when the PSM transitions to a state with Q6 set.
ESIIFG6	ESIIFG6 is set when the PSM transitions to a state with Q7 set.
ESIIFG7	ESIIFG7 is set at different count intervals of the ESICNT0 counter, selected with the ESIS0x bits.
ESIIFG8	ESIIFG8 is set by one of the ESIOUT4 to ESIOUT7 outputs selected with the ESIIFGSET2x bits.

37.2.7.1 PSM Counter ESICNT0 and ESICNT2 Interrupt Handling

The interrupt logic of the PSM counters ESICNT0 and ESICNT2 is generating an interrupt using either the counter input directly or one out of three defined counter outputs. This means, there are four different settings possible: generating an interrupt on 1, 4, 256, or 65536 count steps.

37.2.7.2 PSM Counter ESICNT1 Interrupt Handling

The PSM ESICNT1 counter interrupt logic generates an interrupt as soon as its counter value is equal to the content of the control registers ESITHR1 or ESITHR2. These two threshold registers can be defined by user; the registers contain a 16-bit value that is compared with the 16-bit ESICNT1 register. The interrupt ESIIFG3 is set as soon as the content of ESICNT1 counter is equal to the content of ESITHR1 or ESITHR2.

37.2.7.3 ESIIV, Interrupt Vector Generator

The ESIIFGx interrupt flags are prioritized and combined to source a single interrupt vector. The interrupt vector register ESIIV is used to determine which flag requested an interrupt.

The highest-priority enabled interrupt generates a number in the ESIIV register (see register description). This number can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled ESI interrupt do not affect the ESIIV value.

A read access of the ESIIV register automatically resets the highest-pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. For example, if the ESICNT1 (ESIIFG3) and ESICNT2 (ESIIFG4) interrupt flags are set when the interrupt service routine accesses the ESIIV register, ESIIFG3 is reset automatically. After the RETI instruction of the interrupt service routine is executed, the ESIIFG4 interrupt flag generates another interrupt.

A write access to the ESIIV register clears all pending ESI interrupt flags.

37.2.8 Overview of ESI Applications

The ESI supports different types of sensors. This chapter introduces only a few of the existing solutions of how to use the ESI.

37.2.8.1 Using the ESI with LC Sensors

Systems with LC sensors use a disk that is partially covered with a damping material to measure rotation. Rotation is measured with LC sensors by exciting the sensors and observing the resulting oscillation. The oscillation is either damped or un-damped by the rotating disk. The oscillation is always decaying because of energy losses but it decays faster when the damping material on the disk is within the field of the LC sensor, as shown in [Figure 37-15](#). The LC oscillations can be measured with the oscillation test or the envelope test.

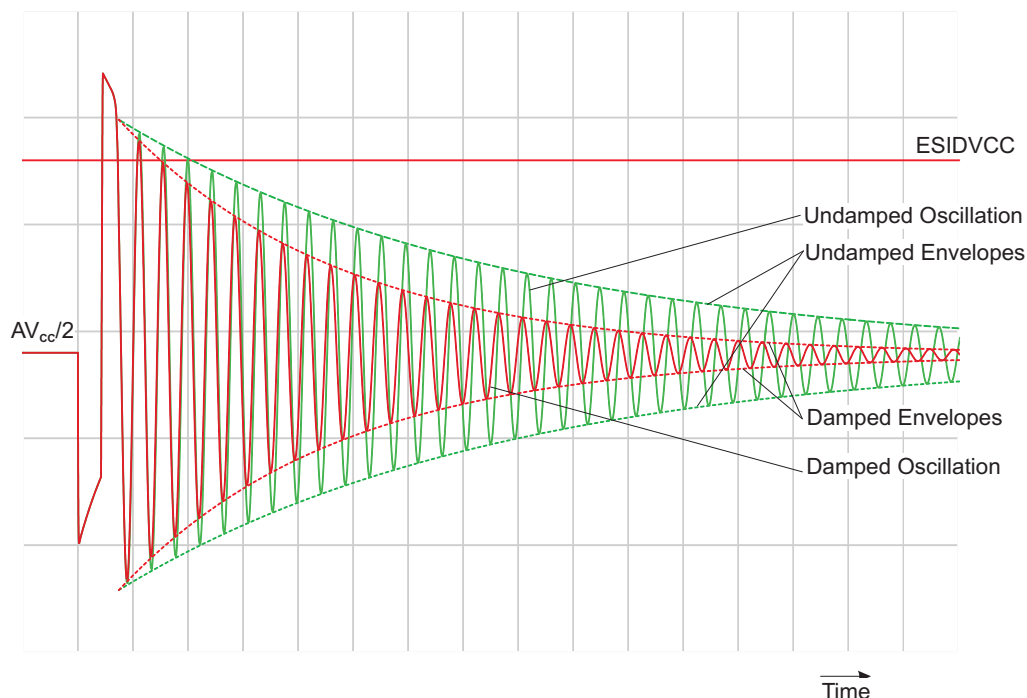


Figure 37-15. LC Sensor Oscillations

37.2.8.1.1 LC-Sensor Oscillation Test

The oscillation test tests if the amplitude of the oscillation after sensor excitation is above a reference level. The DAC is used to set the reference level for the comparator, and the comparator detects if the LC sensor oscillations are above or below the reference level. If the oscillations are above the reference level, the comparator will output a pulse train corresponding to the oscillations and the selected AFE output bit will 1. The measurement timing and reference level depend on the sensors and the system and should be chosen such that the difference between the damped and the undamped amplitude is maximized.

Figure 37-16 shows the connections for the oscillation test.

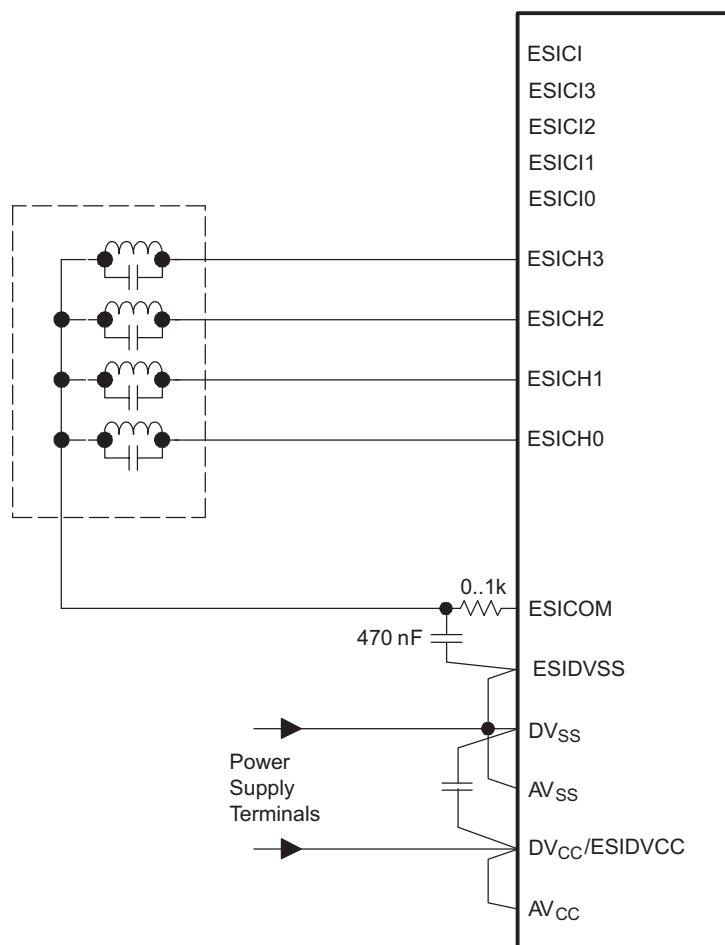


Figure 37-16. Sensor Connections For The Oscillation Test

When the sensors are connected to the individual ESIClx inputs as shown in [Figure 37-17](#), the comparator reference level can be adjusted for each sensor individually. When all sensors are connected to the ESICl input as shown in [Figure 37-18](#), only one comparator reference level is set for all sensors.

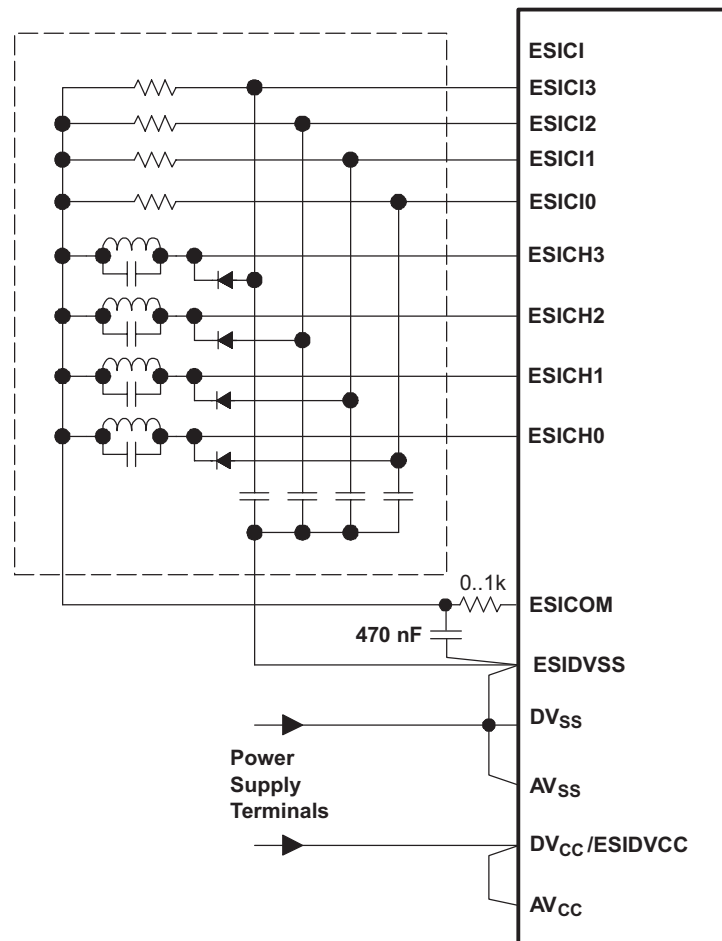


Figure 37-17. LC Sensor Connections For The Envelope Test

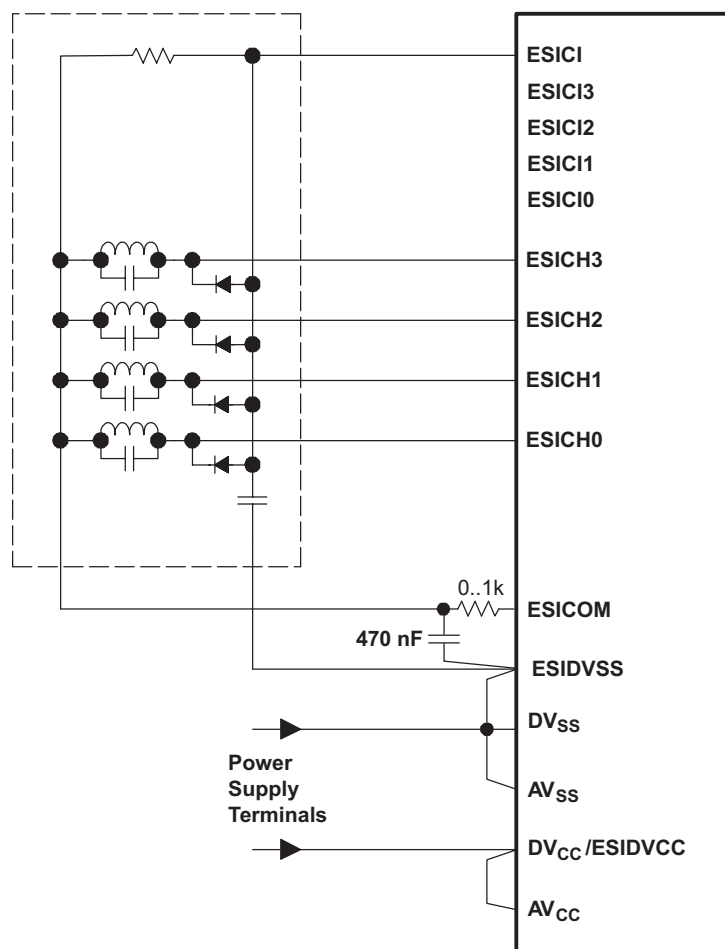


Figure 37-18. LC Sensor Connections For the Envelope Test

37.2.8.2 Using the ESI With Resistive Sensors

Systems with GMRs use magnets on an impeller to measure rotation. The damping material and magnets modify the electrical behavior of the sensor so that rotation and direction can be detected.

Rotation is measured with resistive sensors by connecting the resistor dividers to ground for a short time allowing current flow through the dividers. The resistors are affected by the rotating disc creating different divider voltages. The divider voltages are sampled with the sample-and-hold circuits. After the signals have settled the dividers may be switched off to prevent current flow and reduce power consumption. The DAC is used to set the reference level for the comparator, and the comparator detects if the sampled voltage is above or below the reference level. If the sampled voltage is above the reference level the comparator output is high. [Figure 37-19](#) shows the connection for resistive sensors.

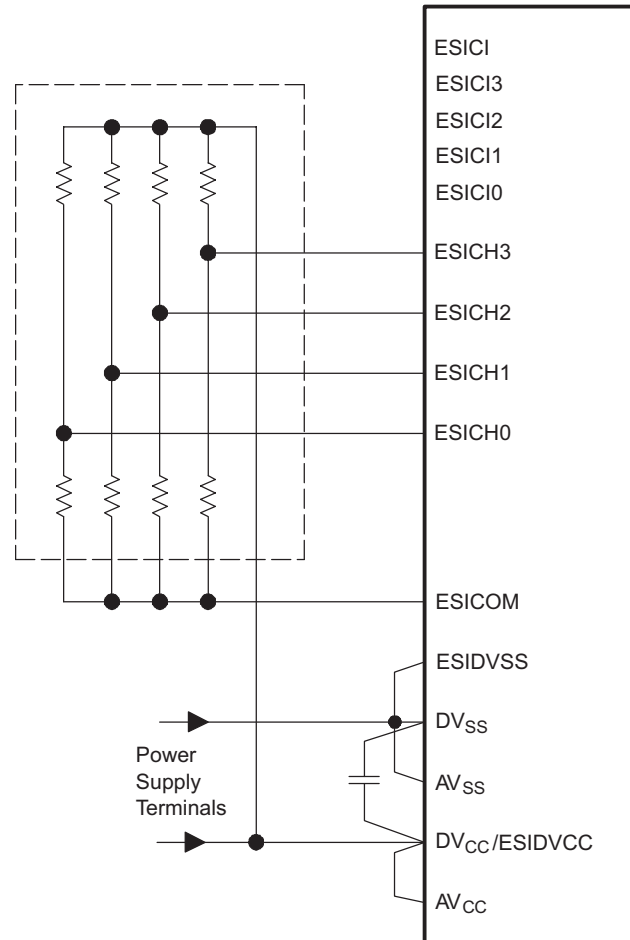


Figure 37-19. Resistive Sensor Connections

37.2.8.3 Quadrature Decoding

The ESI can be used to decode quadrature-encoded signals. Signals that are 90° out of phase with each other are said to be in quadrature. To create the signals, two sensors are positioned depending on the slotting, or coating of the encoder disk. Figure 37-20 shows two examples for the sensor positions and a quadrature-encoded signal waveform.

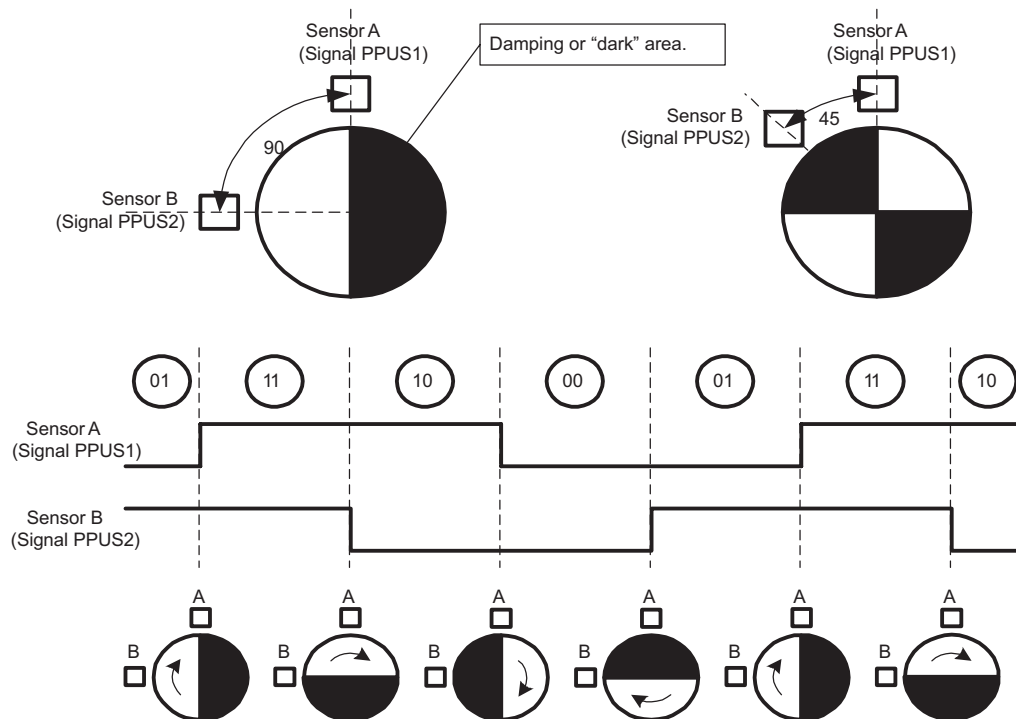


Figure 37-20. Sensor Position and Quadrature Signals (S1=PPUS1, S2=PPUS2)

Quadrature decoding requires knowing the previous quadrature pair S1 (PPUS1) and S2 (PPUS2), as well as the current pair. Comparing these two pairs will tell the direction of the rotation. For example, if the current pair is 00 it can change to 01 or 10, depending on direction. Any other change in the signal pair would represent an error as shown in Figure 37-21.

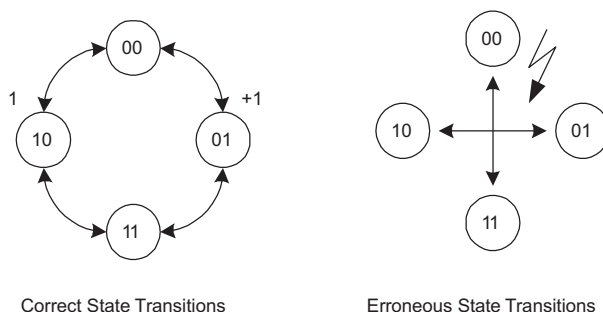


Figure 37-21. Quadrature Decoding State Diagram

To transfer the state encoding into counts it is necessary to decide what fraction of the rotation should be counted and on what state transitions. In this example only full rotations will be counted on the transition from state 00 to 01 or 10 using a 180° disk with the sensors 90° apart. All the possible state transitions can be put into a table and this table can be translated into the corresponding state table entries for the processing state machine as shown in Table 37-8.

Table 37-8. Quadrature Decoding PSM Table

Previous Quadrature Pair	Current Quadrature Pair	Movement	State Table Entry					
			Q6	Q2	Q1	Q3	Q0	Byte Code
			Error	-1	+1	Current Quadrature Pair		
00	00	No Rotation	0	0	0	0	0	000h
00	01	Turns right, +1	0	0	1	0	1	003h
00	10	Turns left, -1	0	1	0	1	0	00Ch
00	11	Error	1	0	0	1	1	049h
01	00	Turns left	0	0	0	0	0	000h
01	01	No rotation	0	0	0	0	1	001h
01	10	Error	1	0	0	1	0	048h
01	11	Turns right	0	0	0	1	1	009h
10	00	Turns right	0	0	0	0	0	000h
10	01	Error	1	0	0	0	1	041h
10	10	No rotation	0	0	0	1	0	008h
10	11	Turns left	0	0	0	1	1	009h
11	00	Error	1	0	0	0	0	040h
11	01	Turns left	0	0	0	0	1	001h
11	10	Turns right	0	0	0	1	0	008h
11	11	No rotation	0	0	0	1	1	009h

37.3 ESI Registers

The Extended Scan Interface registers are listed in [Table 37-9](#).

NOTE: The ESI RAM start address (base address) can be found in the *Peripheral File Map* section of the device-specific data sheet.

Table 37-9. ESI Registers

Offset	Acronym	Register Name	Type	Access	Reset
0h	ESIDEBUG1	ESI debug register 1	Read	Word	Reset with PUC
02h	ESIDEBUG2	ESI debug register 2	Read	Word	Reset with PUC
04h	ESIDEBUG3	ESI debug register 3	Read	Word	Reset with PUC
06h	ESIDEBUG4	ESI debug register 4	Read	Word	Reset with PUC
08h	ESIDEBUG5	ESI debug register 5	Read	Word	Reset with PUC
0Ah	Reserved				
0Ch	Reserved				
0Eh	Reserved				
10h	ESICNT0	ESI PSM counter 0	Read	Word	Reset with PUC
12h	ESICNT1	ESI PSM counter 1	Read	Word	Reset with PUC
14h	ESICNT2	ESI PSM counter 2	Read	Word	Reset with PUC
16h	ESICNT3	ESI oscillator counter register	Read	Word	Reset with PUC
18h	Reserved				
1Ah	ESIIV	ESI interrupt vector	Read	Word	Reset with PUC
1Ch	ESIINT1	ESI interrupt register 1	Read/Write	Word	Reset with PUC
1Eh	ESIINT2	ESI interrupt register 2	Read/Write	Word	Reset with PUC
20h	ESIAFE	ESI AFE control register	Read/Write	Word	Reset with PUC
22h	ESIPPU	ESI PPU control register	Read/Write	Word	Reset with PUC
24h	ESITSM	ESI TSM control register	Read/Write	Word	Reset with PUC
26h	ESIPSM	ESI PSM control register	Read/Write	Word	Reset with PUC
28h	ESIOSC	ESI oscillator control register	Read/Write	Word	Reset with PUC
28h	ESIOSC_L		Read/Write	Byte	Reset with PUC
29h	ESIOSC_H		Read/Write	Byte	Reset with PUC
2Ah	ESICTL	ESI control register	Read/Write	Word	Reset with PUC
2Ch	ESITHR1	ESI PSM Counter Threshold 1 register	Read/Write	Word	Reset with PUC
2Eh	ESITHR2	ESI PSM Counter Threshold 2 register	Read/Write	Word	Reset with PUC
30h	Reserved				
32h	Reserved				
34h	Reserved				
36h	Reserved				
38h	Reserved				
3Ah	Reserved				
3Ch	Reserved				
3Eh	Reserved				
40h to 4Eh	ESIDAC1R0 to ESIDAC1R7	ESI DAC1 register 0 to ESI DAC1 register 7	Read/Write	Word	Unchanged
50h to 5Eh	ESIDAC2R0 to ESIDAC2R7	ESI DAC2 register 0 to ESI DAC2 register 7	Read/Write	Word	Unchanged
60h to 9Eh	ESITSM0 to ESITSM31	ESI TSM 0 to ESI TSM 31	Read/Write	Word	Unchanged

37.3.1 ESIDEBUG1 Register

Extended Scan Interface Debug Register 1

Figure 37-22. ESIDEBUG1 Register

15	14	13	12	11	10	9	8
Unused							
r	r	r	r	r	r	r	r
7	6	5	4	3	2	1	0
Unused		Last_PSM_Address					
r	r	r	r	r	r	r	r

Table 37-10. ESIDEBUG1 Register Description

Bit	Field	Type	Reset	Description
15-7	Unused	R	0h	Unused. These bits are always read as zero.
6-0	Last_PSM_Address	R	0h	ESIDEBUG1 shows the last address read by the PSM.

37.3.2 ESIDEBUG2 Register

Extended Scan Interface Debug Register 2

Figure 37-23. ESIDEBUG2 Register

15	14	13	12	11	10	9	8
Unused				TSM_Index			
r	r	r	r	r	r	r	r
7	6	5	4	3	2	1	0
PSM_Bits							
r	r	r	r	r	r	r	r

Table 37-11. ESIDEBUG2 Register Description

Bit	Field	Type	Reset	Description
15-13	Unused	R	0h	Unused. These bits are always read as zero.
12-8	TSM_Index	R	0h	These bits show the TSM register pointer index.
7-0	PSM_Bits	R	0h	These bits show the PSM bits Q7 to Q0.

37.3.3 ESIDEBUG3 Register

Extended Scan Interface Debug Register 3

Figure 37-24. ESIDEBUG3 Register

15	14	13	12	11	10	9	8
Register_Content							
r	r	r	r	r	r	r	r
7	6	5	4	3	2	1	0
Register_Content							
r	r	r	r	r	r	r	r

Table 37-12. ESIDEBUG3 Register Description

Bit	Field	Type	Reset	Description
15-0	Register_Content	R	0h	Current ESITSMx register content. These bits show the TSM output.

37.3.4 ESIDEBUG4 Register

Extended Scan Interface Debug Register 4

Figure 37-25. ESIDEBUG4 Register

15	14	13	12	11	10	9	8
Unused	DAC1_Register				DAC1_Data		
r	r	r	r	r	r	r	r
7	6	5	4	3	2	1	0
DAC1_Data							
r	r	r	r	r	r	r	r

Table 37-13. ESIDEBUG4 Register Description

Bit	Field	Type	Reset	Description
15	Unused	R	0h	Unused. This bit is always read as zero.
14-12	DAC1_Register	R	0h	These bits show which DAC1 register is currently selected to control the DAC1.
11-0	DAC1_Data	R	0h	These bits show value of the currently selected DAC1 register.

37.3.5 ESIDEBUG5 Register

Extended Scan Interface Debug Register 5

Figure 37-26. ESIDEBUG5 Register

15	14	13	12	11	10	9	8
Unused	DAC2_Register				DAC2_Data		
r	r	r	r	r	r	r	r
7	6	5	4	3	2	1	0
DAC2_Data							
r	r	r	r	r	r	r	r

Table 37-14. ESIDEBUG5 Register Description

Bit	Field	Type	Reset	Description
15	Unused	R	0h	Unused. This bit is always read as zero.
14-12	DAC2_Register	R	0h	These bits show which DAC2 register is currently selected to control the DAC2.
11-0	DAC2_Data	R	0h	These bits show value of the currently selected DAC2 register.

37.3.6 ESICNT0 Register

Extended Scan Interface Counter 0 Register

Figure 37-27. ESICNT0 Register

15	14	13	12	11	10	9	8
ESICNT0x							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
ESICNT0x							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 37-15. ESICNT0 Register Description

Bit	Field	Type	Reset	Description
15-0	ESICNT0x	R	0h	ESICNT0. These bits are the ESICNT0 counter. ESICNT0 is reset when ESIEN = 0 or when ESICNT0RST = 1.

37.3.7 ESICNT1 Register

Extended Scan Interface Counter 1 Register

Figure 37-28. ESICNT1 Register

15	14	13	12	11	10	9	8
ESICNT1x							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
ESICNT1x							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 37-16. ESICNT1 Register Description

Bit	Field	Type	Reset	Description
15-0	ESICNT1x	R	0h	ESICNT1. These bits are the ESICNT1 counter. ESICNT1 is reset when ESIEN = 0 or when ESICNT1RST = 1.

37.3.8 ESICNT2 Register

Extended Scan Interface Counter 2 Register

Figure 37-29. ESICNT2 Register

15	14	13	12	11	10	9	8
ESICNT2x							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
ESICNT2x							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 37-17. ESICNT2 Register Description

Bit	Field	Type	Reset	Description
15-0	ESICNT2x	R	0h	ESICNT2. These bits are the ESICNT2 counter. ESICNT2 is reset when ES/EN = 0 or when ESICNT2RST = 1.

37.3.9 ESICNT3 Register

Extended Scan Interface Oscillator Counter Register

Figure 37-30. ESICNT3 Register

15	14	13	12	11	10	9	8
ESICNT3x							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
ESICNT3x							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

Table 37-18. ESICNT3 Register Description

Bit	Field	Type	Reset	Description
15-0	ESICNT3x	R	0h	Internal oscillator counter. ESICNT3 counts internal oscillator clock cycles during one ACLK period after ES/CLKGON and ES/IFSEL are both set. Setting the control bits ES/IFSEL and ES/CLKGON resets the ESICNT3 counter.

37.3.10 ESIIV Register

Extended Scan Interface Interrupt Vector Register

Figure 37-31. ESIIV Register

15	14	13	12	11	10	9	8
ESIIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
ESIIV							
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Table 37-19. ESIIV Register Description

Bit	Field	Type	Reset	Description
15-0	ESIIV	R	0h	Extended Scan Interface interrupt vector value. The ESIIV register helps to easily find out the source of an Extended Scan Interface interrupt. By adding the ESIIV register content to the program counter (PC) the code execution is continued on one of the following instructions. This allows to realize a jump table that optimizes the detection of interrupt source. Writing to this register clears all pending Extended Scan Interface interrupt flags. 00h = No interrupt pending 02h = Interrupt Source: Rising edge of the ESISTOP(tsm) signal; Interrupt Flag: ESIIFG1; Interrupt Priority: Highest 04h = Interrupt Source: ESIOUT0 to ESIOUT3 conditions selected by ESIIFGSETx bits; Interrupt Flag: ESIIFG0 06h = Interrupt Source: ESIOUT4 to ESIOUT7 conditions selected by ESIIFGSET2x bits; Interrupt Flag: ESIIFG8 08h = Interrupt Source: ESICNT1 counter conditions selected with the ESITHR1 and ESITHR2 registers; Interrupt Flag: ESIIFG3 0Ah = Interrupt Source: PSM transitions to a state with a set Q7 bit; Interrupt Flag: ESIIFG6 0Ch = Interrupt Source: PSM transitions to a state with a set Q6 bit; Interrupt Flag: ESIIFG5 0Eh = Interrupt Source: ESICNT2 counter conditions selected with the ESIIS2x bits; Interrupt Flag: ESIIFG4 10h = Interrupt Source: ESICNT0 counter conditions selected with the ESIIS0x bits; Interrupt Flag: ESIIFG7 12h = Interrupt Source: Start of a TSM sequence; Interrupt Flag: ESIIFG2; Interrupt Priority: Lowest

37.3.11 ESIINT1 Register

Extended Scan Interface Interrupt Register 1

Figure 37-32. ESIINT1 Register

15	14	13	12	11	10	9	8
ESIIFGSET2x			ESIIFGSET1x			Reserved	ESIIE8
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r0	rw-0
7	6	5	4	3	2	1	0
ESIIE7	ESIIE6	ESIIE5	ESIIE4	ESIIE3	ESIIE2	ESIIE1	ESIIE0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 37-20. ESIINT1 Register Description

Bit	Field	Type	Reset	Description
15-13	ESIIFGSET2x	RW	0h	ESIIFG8 interrupt flag source. These bits select when the ESIIFG8 flag is set. 000b = ESIIFG8 is set when ESIOUT4 is set. 001b = ESIIFG8 is set when ESIOUT4 is reset. 010b = ESIIFG8 is set when ESIOUT5 is set. 011b = ESIIFG8 is set when ESIOUT5 is reset. 100b = ESIIFG8 is set when ESIOUT6 is set. 101b = ESIIFG8 is set when ESIOUT6 is reset. 110b = ESIIFG8 is set when ESIOUT7 is set. 111b = ESIIFG8 is set when ESIOUT7 is reset.
12-10	ESIIFGSET1x	RW	0h	ESIIFG0 interrupt flag source. These bits select when the ESIIFG0 flag is set. 000b = ESIIFG0 is set when ESIOUT0 is set. 001b = ESIIFG0 is set when ESIOUT0 is reset. 010b = ESIIFG0 is set when ESIOUT1 is set. 011b = ESIIFG0 is set when ESIOUT1 is reset. 100b = ESIIFG0 is set when ESIOUT2 is set. 101b = ESIIFG0 is set when ESIOUT2 is reset. 110b = ESIIFG0 is set when ESIOUT3 is set. 111b = ESIIFG0 is set when ESIOUT3 is reset.
9	Reserved	R	0h	Reserved. This bit is always read as zero and, when written, does not affect the bit setting.
8	ESIIE8	RW	0h	Interrupt enable. These bits enable or disable the interrupt request for the ESIIFG8 bit. Details about the interrupt functionality can be found in the ESIIFG8 bit descriptions (see control register ESIINT2). 0b = Interrupt disabled 1b = Interrupt enabled
7	ESIIE7	RW	0h	Interrupt enable. These bits enable or disable the interrupt request for the ESIIFG7 bit. Details about the interrupt functionality can be found in the ESIIFG7 bit descriptions (see control register ESIINT2). 0b = Interrupt disabled 1b = Interrupt enabled
6	ESIIE6	RW	0h	Interrupt enable. These bits enable or disable the interrupt request for the ESIIFG6 bit. Details about the interrupt functionality can be found in the ESIIFG6 bit descriptions (see control register ESIINT2). 0b = Interrupt disabled 1b = Interrupt enabled
5	ESIIE5	RW	0h	Interrupt enable. These bits enable or disable the interrupt request for the ESIIFG5 bit. Details about the interrupt functionality can be found in the ESIIFG5 bit descriptions (see control register ESIINT2). 0b = Interrupt disabled 1b = Interrupt enabled

Table 37-20. ESIINT1 Register Description (continued)

Bit	Field	Type	Reset	Description
4	ESIIE4	RW	0h	Interrupt enable. These bits enable or disable the interrupt request for the ESIIFG4 bit. Details about the interrupt functionality can be found in the ESIIFG4 bit descriptions (see control register ESIINT2). 0b = Interrupt disabled 1b = Interrupt enabled
3	ESIIE3	RW	0h	Interrupt enable. These bits enable or disable the interrupt request for the ESIIFG3 bit. Details about the interrupt functionality can be found in the ESIIFG3 bit descriptions (see control register ESIINT2). 0b = Interrupt disabled 1b = Interrupt enabled
2	ESIIE2	RW	0h	Interrupt enable. These bits enable or disable the interrupt request for the ESIIFG2 bit. Details about the interrupt functionality can be found in the ESIIFG2 bit descriptions (see control register ESIINT2). 0b = Interrupt disabled 1b = Interrupt enabled
1	ESIIE1	RW	0h	Interrupt enable. These bits enable or disable the interrupt request for the ESIIFG1 bit. Details about the interrupt functionality can be found in the ESIIFG1 bit descriptions (see control register ESIINT2). 0b = Interrupt disabled 1b = Interrupt enabled
0	ESIIE0	RW	0h	Interrupt enable. These bits enable or disable the interrupt request for the ESIIFG0 bit. Details about the interrupt functionality can be found in the ESIIFG0 bit descriptions (see control register ESIINT2). 0b = Interrupt disabled 1b = Interrupt enabled

37.3.12 ESIINT2 Register

Extended Scan Interface Interrupt Register 2

Figure 37-33. ESIINT2 Register

15	14	13	12	11	10	9	8
Reserved	ESIIS2x		Reserved	ESIIS0x		Reserved	ESIIFG8
r0	rw-0	rw-0	r0	rw-0	rw-0	r0	rw-0
7	6	5	4	3	2	1	0
ESIIFG7	ESIIFG6	ESIIFG5	ESIIFG4	ESIIFG3	ESIIFG2	ESIIFG1	ESIIFG0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 37-21. ESIINT2 Register Description

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved. This bit is always read as zero and, when written, does not affect the bit setting.
14-13	ESIIS2x	RW	0h	ESIIFG4 interrupt flag source 00b = ESIIFG4 is set with each count of ESICNT2. 01b = ESIIFG4 is set if (ESICNT2 modulo 4) = 0. 10b = ESIIFG4 is set if (ESICNT2 modulo 256) = 0. 11b = ESIIFG4 is set when ESICNT2 decrements from 01h to 00h.
12	Reserved	R	0h	Reserved. This bit is always read as zero and, when written, does not affect the bit setting.
11-10	ESIIS0x	RW	0h	ESIIFG7 interrupt flag source 00b = ESIIFG7 is set with each count of ESICNT0. 01b = ESIIFG7 is set if (ESICNT0 modulo 4) = 0. 10b = ESIIFG7 is set if (ESICNT0 modulo 256) = 0. 11b = ESIIFG7 is set when ESICNT0 increments from FFFFh to 00h.
9	Reserved	R	0h	Reserved. This bit is always read as zero and, when written, does not affect the bit setting.
8	ESIIFG8	RW	0h	ESIIFG8 is set by one of the AFE2's ESIOUTx outputs selected with the ESIIFGSET2x bits. 0b = No interrupt pending 1b = Interrupt pending
7	ESIIFG7	RW	0h	ESI interrupt flag 7. ESIIFG7 is set at different count intervals of the ESICNT0 counter, selected with the ESIIS0x bits. ESIIFG6 must be reset with software. 0b = No interrupt pending 1b = Interrupt pending
6	ESIIFG6	RW	0h	ESI interrupt flag 6. This bit is set when the PSM transitions to a state with a set Q7 bit. ESIIFG6 must be reset with software. 0b = No interrupt pending 1b = Interrupt pending
5	ESIIFG5	RW	0h	ESI interrupt flag 5. This bit is set when the PSM transitions to a state with a set Q6 bit. ESIIFG5 must be reset with software. 0b = No interrupt pending 1b = Interrupt pending
4	ESIIFG4	RW	0h	ESI interrupt flag 4. This bit is set by the ESICNT2 counter conditions selected with the ESIIS2x bits. ESIIFG4 must be reset with software. 0b = No interrupt pending 1b = Interrupt pending
3	ESIIFG3	RW	0h	ESI interrupt flag 3. This bit is set by the ESICNT1 counter conditions selected with the ESITHR1 and ESITHR2 registers. ESIIFG3 must be reset with software. 0b = No interrupt pending 1b = Interrupt pending

Table 37-21. ESIINT2 Register Description (continued)

Bit	Field	Type	Reset	Description
2	ESIIFG2	RW	0h	ESI interrupt flag 2. This bit is set at the start of a TSM sequence generated by the divided ACLK. A TSM sequence started with ESISTART bit does not set ESIIFG2. ESIIFG2 must be reset with software. 0b = No interrupt pending 1b = Interrupt pending
1	ESIIFG1	RW	0h	ESI interrupt flag 1. This bit is set by the rising edge of the ESISTOP(tsm) signal. ESIIFG1 must be reset with software. 0b = No interrupt pending 1b = Interrupt pending
0	ESIIFG0	RW	0h	ESI interrupt flag 0. This bit is set by the AFE1's ESIOUTx conditions selected by the ESIIFGSET1x bits. ESIIFG0 must be reset with software. 0b = No interrupt pending 1b = Interrupt pending

37.3.13 ESIAFE Register

Extended Scan Interface Analog Front-End Control Register

Figure 37-34. ESIAFE Register

15	14	13	12	11	10	9	8
Reserved	Reserved	Reserved	Reserved	ESIDAC2EN	ESICA2EN	ESICA2INV	ESICA1INV
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
ESICA2X	ESICA1X	ESICISEL	ESICACI3	ESISHTSM ⁽¹⁾	ESIVMIDEN ⁽²⁾	ESISH	ESITEN
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

⁽¹⁾ The control bit ESIVSS was renamed to ESISHTSM to avoid confusion with supply pin naming.

⁽²⁾ The control bit ESIVCC2 was renamed to ESIVMIDEN to avoid confusion with supply pin naming.

Table 37-22. ESIAFE Register Description

Bit	Field	Type	Reset	Description
15-12	Reserved	RW	0h	Reserved for test purposes. It is strongly recommended to always write these bits as 0.
11	ESIDAC2EN	RW	0h	Enable ESIDAC(tsm) control for DAC in AFE2. 0b = AFE2's DAC is always disabled, independently from ESIDAC(tsm) setting. 1b = AFE2's DAC is controlled by ESIDAC(tsm) bit.
10	ESICA2EN	RW	0h	Enable ESICA(tsm) control for comparator in AFE2. 0b = AFE2's comparator is always disabled, independently from ESICA(tsm) setting. 1b = AFE2's comparator is controlled by ESICA(tsm) bit.
9	ESICA2INV	RW	0h	Invert AFE2's comparator output 0b = Comparator output in AFE2 is not inverted 1b = Comparator output in AFE2 is inverted
8	ESICA1INV	RW	0h	Invert AFE1's comparator output 0b = Comparator output in AFE1 is not inverted 1b = Comparator output in AFE1 is inverted
7	ESICA2X	RW	0h	AFE2's comparator input select. This bit selects groups of signals for the comparator input. 0b = AFE2's comparator input is one of the ESICHx channels, selected with the channel select logic. 1b = AFE2's comparator input is one of the ESIClx channels, selected with the channel select logic and the ESICISEL and ESICACI3 bits.
6	ESICA1X	RW	0h	AFE1's comparator input select. This bit selects groups of signals for the comparator input. 0b = AFE1's comparator input is one of the ESICHx channels, selected with the channel select logic. 1b = AFE1's comparator input is one of the ESIClx channels, selected with the channel select logic and the ESICISEL and ESICACI3 bits.
5	ESICISEL	RW	0h	Comparator input select for AFE1 only. This bit is used with the ESICACI3 bit to select the comparator input when ESICAX = 1. 0b = Comparator input is one of the ESIClx channels, selected with the channel select logic and ESICACI3 bit. 1b = Comparator input is the ESICl channel
4	ESICACI3	RW	0h	Comparator input select for AFE1 only. This bit is selects the comparator input when ESICISEL = 0 and ESICAX = 1. 0b = Comparator input is selected with the channel select logic. 1b = Comparator input is ESICl3.

Table 37-22. ESIAFE Register Description (continued)

Bit	Field	Type	Reset	Description
3	ESISHTSM ⁽¹⁾	RW	0h	Sample-and-hold ESIDVSS select. 0b = The ground connection of the sample capacitor is connected to ESIDVSS, regardless of the TSM control. 1b = The ground connection of the sample capacitor is controlled by the TSM
2	ESIVMIDEN ⁽²⁾	RW	0h	Mid-voltage generator 0b = AVCC/2 generator is off 1b = AVCC/2 generator is on if ESISH = 0
1	ESISH	RW	0h	Sample-and-hold enable 0b = Sample-and-hold is disabled 1b = Sample-and-hold is enabled
0	ESITEN	RW	0h	Excitation enable 0b = Excitation circuitry is disabled 1b = Excitation circuitry is enabled

⁽¹⁾ The control bit ESIVSS was renamed to ESISHTSM to avoid confusion with supply pin naming.

⁽²⁾ The control bit ESIVCC2 was renamed to ESIVMIDEN to avoid confusion with supply pin naming.

37.3.14 ESIPPU Register

Extended Scan Interface Pre-Processing Unit Control Register

Figure 37-35. ESIPPU Register

15	14	13	12	11	10	9	8
Reserved						ESITCHOUT1	ESITCHOUT0
r0	r0	r0	r0	r0	r0	r-(0)	r-(0)
7	6	5	4	3	2	1	0
ESIOUT7	ESIOUT6	ESIOUT5	ESIOUT4	ESIOUT3	ESIOUT2	ESIOUT1	ESIOUT0
r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)

Table 37-23. ESIPPU Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved. These bits are always read as zero and, when written, do not affect the bit setting.
9	ESITCHOUT1	R	0h	Latched AFE1 comparator output for test channel 1
8	ESITCHOUT0	R	0h	Latched AFE1 comparator output for test channel 0
7	ESIOUT7	R	0h	Latched AFE2 comparator output when ESICH3 input is selected
6	ESIOUT6	R	0h	Latched AFE2 comparator output when ESICH2 input is selected
5	ESIOUT5	R	0h	Latched AFE2 comparator output when ESICH1 input is selected
4	ESIOUT4	R	0h	Latched AFE2 comparator output when ESICH0 input is selected
3	ESIOUT3	R	0h	Latched AFE1 comparator output when ESICH3 input is selected
2	ESIOUT2	R	0h	Latched AFE1 comparator output when ESICH2 input is selected
1	ESIOUT1	R	0h	Latched AFE1 comparator output when ESICH1 input is selected
0	ESIOUT0	R	0h	Latched AFE1 comparator output when ESICH0 input is selected

37.3.15 ESITSM Register

Extended Scan Interface Timing State Machine Control Register

Figure 37-36. ESITSM Register

15	14	13	12	11	10	9	8
Reserved	ESICKAZSEL	ESITSMTRGx		ESISTART	ESITSMRP	ESIDIV3Bx	
r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
ESIDIV3Bx	ESIDIV3Ax			ESIDIV2x		ESIDIV1x	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 37-24. ESITSM Register Description

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved. This bit is always read as zero and, when written, does not affect the bit setting.
14	ESICKAZSEL	RW	0h	Control bit functionality selection. This bit allows to define the functionality of bit 5 in register ESITSMx. 0b = ESITSMx.5 bit is used as ESICKON. See ESITSMx control register for further description. 1b = ESITSMx.5 bit is used as ESICAAZ. See ESITSMx control register for further description.
13-12	ESITSMTRGx	RW	0h	TSM start trigger selection. These bits allow to chose the source for the TSM start trigger. 00b = Halt mode. This setting allows to stop the TSM. 01b = TSM start trigger ACLK divider is used. ESIDIV3Ax and ESIDIV3Bx bits select the division rate for the TSM start trigger. 10b = Software trigger for TSM. When ESISTART bit is set by software a TSM start trigger is generated. Note that for this setting an ACLK synchronization sequence is performed that takes up to 2.5 ACLK cycles. 11b = Either the ACLK divider (ESIDIV3Ax and ESIDIV3Bx) or the ESISTART bit is used for TSM start trigger.
11	ESISTART	RW	0h	TSM software start trigger. In case the ESISTART bit is selected for TSM trigger generation this bit allows to generate a TSM start trigger by software. 0b = Idle state 1b = A TSM sequence is started. ESISTART is automatically cleared as soon as the TSM sequence starts.
10	ESITSMRP	RW	0h	TSM repeat mode 0b = Each TSM sequence is triggered by the ACLK divider controlled with the ESIDIV3Ax and ESIDIV3Bx bits or ESISTART control bit depending on ESITSMTRGx setting. 1b = Each TSM sequence is immediately started at the end of the previous sequence.
9-7	ESIDIV3Bx	RW	0h	TSM start trigger ACLK divider. These bits together with the ESIDIV3Ax bits select the division rate for the TSM start trigger. The division rate is shown in Table 37-25 . The division rate can be calculated as: $((ESIDIV3A + 1) \times 2 - 1) \times ((ESIDIV3B + 1) \times 2 - 1) \times 2$
6-4	ESIDIV3Ax	RW	0h	TSM start trigger ACLK divider. These bits together with the ESIDIV3Bx bits select the division rate for the TSM start trigger. The division rate is shown in Table 37-25 . The division rate can be calculated as: $((ESIDIV3A + 1) \times 2 - 1) \times ((ESIDIV3B + 1) \times 2 - 1) \times 2$
3-2	ESIDIV2x	RW	0h	ACLK divider. These bits select the ACLK division. 00b = /1 01b = /2 10b = /4 11b = /8

Table 37-24. ESITSM Register Description (continued)

Bit	Field	Type	Reset	Description
1-0	ESIDIV1x	RW	0h	TSM SMCLK divider. These bits select the SMCLK division for the TSM. 00b = /1 01b = /2 10b = /4 11b = /8

Table 37-25. TSM Start Trigger ACLK Divider

ACLK Divider	ESIDIV3Bx	ESIDIV3Ax		ACLK Divider	ESIDIV3Bx	ESIDIV3Ax
2	000	000		126	011	100
6	000	001		130	010	110
10	000	010		150	010	111
14	000	011		154	011	101
18	000	100		162	100	100
22	000	101		182	011	110
26	000	110		198	100	101
30	000	111		210	011	111
42	001	011		234	100	110
50	010	010		242	101	101
54	001	100		270	100	111
66	001	101		286	101	110
70	010	011		330	101	111
78	001	110		338	110	110
90	001	111		390	110	111
98	011	011		450	111	111
110	010	101				

37.3.16 ESIPSM Register

Extended Scan Interface Processing State Machine Control Register

Figure 37-37. ESIPSM Register

15	14	13	12	11	10	9	8
ESICNT2RST	ESICNT1RST	ESICNT0RST	Reserved			ESITEST4SEL	
rw-0	rw-0	rw-0	r0	r0	r0	rw-0	rw-0
7	6	5	4	3	2	1	0
ESIV2SEL	Reserved	ESICNT2EN	ESICNT1EN	ESICNT0EN	ESIQ7TRG	Reserved	ESIQ6EN
rw-1	r0	rw-0	rw-0	rw-0	rw-0	r0	rw-0

Table 37-26. ESIPSM Register Description

Bit	Field	Type	Reset	Description
15	ESICNT2RST	RW	0h	ESI Counter 2 reset. Setting this bit resets ESICNT2 register. After ESICNT2 register is cleared, the ESICNT2RST bit is automatically reset. This bit is always read as zero.
14	ESICNT1RST	RW	0h	ESI Counter 1 reset. Setting this bit resets ESICNT1 register. After ESICNT1 register is cleared, the ESICNT1RST bit is automatically reset. This bit is always read as zero.
13	ESICNT0RST	RW	0h	ESI Counter 0 reset. Setting this bit resets ESICNT0 register. After ESICNT0 register is cleared, the ESICNT0RST bit is automatically reset. This bit is always read as zero.
12-10	Reserved	R	0h	Reserved. These bits are always read as zero and, when written, do not affect the bit setting.
9-8	ESITEST4SEL	RW	0h	Output signal selection for ESITEST4 pin. 00b = Q2 signal from PSM table 01b = Q1 signal from PSM table 10b = TSM clock signal from Timing State Machine 11b = AFE1's comparator output signal ESIC1OUT
7	ESIV2SEL	RW	1h	Source Selection for V2 bit of Next State Latch 0b = PPUS3 signal is used for V2 bit 1b = Q0 is used for V2 bit
6	Reserved	R	0h	Reserved. This bit is always read as zero and, when written, does not affect the bit setting.
5	ESICNT2EN	RW	0h	ESICNT2 enable (down counter) 0b = ESICNT2 is disabled 1b = ESICNT2 is enabled
4	ESICNT1EN	RW	0h	ESICNT1 enable (up/down counter) 0b = ESICNT1 is disabled 1b = ESICNT1 is enabled
3	ESICNT0EN	RW	0h	ESICNT0 enable (up counter) 0b = ESICNT0 is disabled 1b = ESICNT0 is enabled
2	ESIQ7TRG	RW	0h	Enabling to use Q7 as trigger for a PSM sequence. 0b = Only ESISTOP(tsm) is used as PSM trigger. 1b = ESISTOP(tsm) and Q7 are used as PSM triggers. As soon as a PSM state is reached with Q7 bit set the next state is calculated immediately without waiting for the next falling edge of ESISTOP(tsm).
1	Reserved	R	0h	Reserved. This bit is always read as zero and, when written, does not affect the bit setting.
0	ESIQ6EN	RW	0h	Q6 enable. This bit enables Q6 for the next PSM state calculation. 0b = Q6 is not used to determine the next PSM state 1b = Q6 is used to determine the next PSM state

37.3.17 ESIOSC Register

Extended Scan Interface Oscillator Control Register

Figure 37-38. ESIOSC Register

15	14	13	12	11	10	9	8
Reserved		ESICKLFQx					
r0	r0	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Reserved						ESICKLGON	ESIHFSSEL
r0	r0	r0	r0	r0	r0	rw-0	rw-0

Table 37-27. ESIOSC Register Description

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0h	Reserved. These bits are always read as zero and, when written, do not affect the bit setting.
13-8	ESICKLFQx	RW	20h	Internal oscillator frequency adjust. These bits are used to adjust the internal oscillator frequency. Each increase or decrease of the ESICKLFQx bits increases or decreases the internal oscillator frequency by approximately 3%. 000000b = Minimum frequency ⋮ 100000b = Nominal frequency ⋮ 111111b = Maximum frequency
7-2	Reserved	R	0h	Reserved. These bits are always read as zero and, when written, do not affect the bit setting.
1	ESICKLGON	RW	0h	Internal oscillator control. When ESICKLGON = 1 and ESIHFSEL = 1, the internal oscillator calibration is started. ESICKLGON is not used when ESIHFSEL = 0. 0b = No internal oscillator calibration is started. 1b = The internal oscillator calibration is started when ESIHFSEL = 1.
0	ESIHFSSEL	RW	0h	Internal oscillator enable. This bit selects the high frequency clock source for the TSM. 0b = TSM high frequency clock source is SMCLK. 1b = TSM high frequency clock source is the Extended Scan IF internal oscillator.

37.3.18 ESICTL Register

Extended Scan Interface General Control Register

Figure 37-39. ESICTL Register

15	14	13	12	11	10	9	8
ESIS3SELx			ESIS2SELx			ESIS1SELx	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
ESIS1SELx	ESITCH1x		ESITCH0x		ESICS	ESITESTD	ESIEN
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 37-28. ESICTL Register Description

Bit	Field	Type	Reset	Description
15-13	ESIS3SELx	RW	0h	PPUS3 source select. These bits select the PPUS3 source for the PSM. 000b = ESIOUT0 is the PPUS3 source 001b = ESIOUT1 is the PPUS3 source 010b = ESIOUT2 is the PPUS3 source 011b = ESIOUT3 is the PPUS3 source 100b = ESIOUT4 is the PPUS3 source 101b = ESIOUT5 is the PPUS3 source 110b = ESIOUT6 is the PPUS3 source 111b = ESIOUT7 is the PPUS3 source
12-10	ESIS2SELx	RW	0h	PPUS2 source select. These bits select the PPUS2 source for the PSM. 000b = ESIOUT0 is the PPUS2 source 001b = ESIOUT1 is the PPUS2 source 010b = ESIOUT2 is the PPUS2 source 011b = ESIOUT3 is the PPUS2 source 100b = ESIOUT4 is the PPUS2 source 101b = ESIOUT5 is the PPUS2 source 110b = ESIOUT6 is the PPUS2 source 111b = ESIOUT7 is the PPUS2 source
9-7	ESIS1SELx	RW	0h	PPUS1 source select. These bits select the PPUS1 source for the PSM. 000b = ESIOUT0 is the PPUS1 source 001b = ESIOUT1 is the PPUS1 source 010b = ESIOUT2 is the PPUS1 source 011b = ESIOUT3 is the PPUS1 source 100b = ESIOUT4 is the PPUS1 source 101b = ESIOUT5 is the PPUS1 source 110b = ESIOUT6 is the PPUS1 source 111b = ESIOUT7 is the PPUS1 source
6-5	ESITCH1x	RW	0h	These bits select the comparator input for test channel 1. 00b = Comparator input is ESICH0 when ESICAX = 0; Comparator input is ESICI0 when ESICAX = 1. 01b = Comparator input is ESICH1 when ESICAX = 0; Comparator input is ESICI1 when ESICAX = 1. 10b = Comparator input is ESICH2 when ESICAX = 0; Comparator input is ESICI2 when ESICAX = 1. 11b = Comparator input is ESICH3 when ESICAX = 0; Comparator input is ESICI3 when ESICAX = 1.

Table 37-28. ESICTL Register Description (continued)

Bit	Field	Type	Reset	Description
4-3	ESITCH0	RW	0h	These bits select the comparator input for test channel 0. 00b = Comparator input is ESICH0 when ESICAX = 0; Comparator input is ESICI0 when ESICAX = 1. 01b = Comparator input is ESICH1 when ESICAX = 0; Comparator input is ESICI1 when ESICAX = 1. 10b = Comparator input is ESICH2 when ESICAX = 0; Comparator input is ESICI2 when ESICAX = 1. 11b = Comparator input is ESICH3 when ESICAX = 0; Comparator input is ESICI3 when ESICAX = 1.
2	ESICS	RW	0h	Comparator output or Timer_A input selection 0b = The ESIEX(tsm) signal and the comparator output are connected to the TACCRx inputs. 1b = The ESIEX(tsm) signal and the ESIOUTx outputs are connected to the TACCRx inputs selected with the ESIS1SELx and ESIS2SELx bits (PPUS1 and PPUS2 signals).
1	ESITESTD	RW	0h	Test cycle insertion. Setting this bit inserts a test cycle between TSM cycles. ESITESTD is automatically reset at the end of the test cycle. Note that a test cycle insertion should only be done when divided ACLK is used as start trigger for TSM sequences (ESITSMTRGx = 01 and ESITSMRP=0). 0b = No test cycle inserted 1b = Test cycle inserted between TSM cycles.
0	ESIEN	RW	0h	Extended Scan interface enable. Setting this bit enables the Extended Scan Interface and its components. 0b = Extended Scan Interface disabled 1b = Extended Scan Interface enabled

37.3.19 ESITHR1 Register

ESI PSM Counter Threshold 1 Register

Figure 37-40. ESITHR1 Register

15	14	13	12	11	10	9	8
Threshold1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Threshold1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 37-29. ESITHR1 Register Description

Bit	Field	Type	Reset	Description
15-0	Threshold1	RW	0h	Threshold for ESICNT1 counter. The interrupt flag ESIIFG3 is set when ESICNT1 content and Threshold 1 is equal. (for example, used to detect a certain increase of ESICNT1)

37.3.20 ESITHR2 Register

ESI PSM Counter Threshold 2 Register

Figure 37-41. ESITHR2 Register

15	14	13	12	11	10	9	8
Threshold2							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
7	6	5	4	3	2	1	0
Threshold2							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

Table 37-30. ESITHR2 Register Description

Bit	Field	Type	Reset	Description
15-0	Threshold2	RW	FFFFh	Threshold for ESICNT1 counter. The interrupt flag ESIIFG3 is set when ESICNT1 content and Threshold 2 is equal. (for example, used to detect a certain decrease of ESICNT1)

37.3.21 ESIDAC1Rx Register (x = 0 to 7)

Extended Scan Interface Digital-To-Analog Converter 1 Register x (x = 0 to 7)

Figure 37-42. ESIDAC1Rx Register

15	14	13	12	11	10	9	8
Reserved				DAC_Data			
r0	r0	r0	r0	rw	rw	rw	rw
7	6	5	4	3	2	1	0
DAC_Data							
rw	rw	rw	rw	rw	rw	rw	rw

Table 37-31. ESIDAC1Rx Register Description

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved. These bits are always read as zero and, when written, do not affect the bit setting.
11-0	DAC_Data	RW	0h	12-bit DAC data

37.3.22 ESIDAC2Rx Register (x = 0 to 7)

Extended Scan Interface Digital-To-Analog Converter 2 Register x (x = 0 to 7)

Figure 37-43. ESIDAC2Rx Register

15	14	13	12	11	10	9	8
Reserved				DAC_Data			
r0	r0	r0	r0	rw	rw	rw	rw
7	6	5	4	3	2	1	0
DAC_Data							
rw	rw	rw	rw	rw	rw	rw	rw

Table 37-32. ESIDAC2Rx Register Description

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved. These bits are always read as zero and, when written, do not affect the bit setting.
11-0	DAC_Data	RW	0h	12-bit DAC data

37.3.23 ESITSMx Register (x = 0 to 31)

Extended Scan Interface Timing State Machine Register

NOTE:

A TSM sequence should at least consist of three ESITSMx registers. For example, using ESITSM0 for idle state, ESITSM1 for measurement, and ESITSM2 as stop state; note that usually several ESITSMx registers are needed to perform a measurement.

While a TSM sequence is in progress the access to the ESITSMx registers is blocked. Reading the ESITSMx registers while a TSM sequence is in progress returns always a 0x0000.

Figure 37-44. ESITSMx Register

15	14	13	12	11	10	9	8
ESIREPEATx					ESICLK	ESISTOP	ESIDAC
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
ESITESTS1	ESIRSON	ESICLKON ESICAAZ	ESICA	ESIEX	ESILCEN	ESICHx	
rw	rw	rw	rw	rw	rw	rw	rw

Table 37-33. ESITSMx Register Description

Bit	Field	Type	Reset	Description
15-11	ESIREPEATx	RW	0h	These bits together with the ESICLK bit configure the duration of this state. ESIREPEATx selects the number of clock cycles for this state. The number of clock cycles = ESIREPEATx + 1. Note that all ESIREPEATx bits should be cleared within the ESITSMx state that generates the end of sequence (ESISTOP bit is set).
10	ESICLK	RW	0h	This bit selects the clock source for the TSM. 0b = The TSM clock source is the high frequency source selected by the ESIHFSEL bit. 1b = The TSM clock source is ACLK
9	ESISTOP	RW	0h	This bit indicates the end of the TSM sequence. The duration of this state is always one high-frequency clock period, regardless of the ESICLK and ESIREPEATx settings. 0b = TSM sequence continues with next state 1b = End of TSM sequence
8	ESIDAC	RW	0h	TSM DAC on. This bit turns the AFE1 DAC and optionally also AFE2 DAC on. 0b = AFE1 DAC and AFE2 DAC are off during this state. 1b = AFE1 DAC is on during this state. AFE2 DAC is only on when ESIDAC2EN in ESIAFE control register is set.
7	ESITESTS1	RW	0h	TSM test cycle control. This bit selects for this state which channel-control bits and which DAC registers are used for a test cycle. 0b = The ESITCH0x bits select the channel and ESIDACR6 is used for the DAC 1b = The ESITCH1x bits select the channel and ESIDACR7 is used for the DAC
6	ESIRSON	RW	0h	Internal output latches enabled. This bit enables the internal latches of the AFE output stage. 0b = Output latches disabled 1b = Output latches enabled

Table 37-33. ESITSMx Register Description (continued)

Bit	Field	Type	Reset	Description
5	ESICKON ESICAAZ	RW	0h	This control bit in the ESITSMx control register can either be used as ESICKON bit or ESICAAZ bit. Its functionality is selectable by the control bit CLKCAAZSEL in register TSM. ESITSM.ESICKAZSEL=0 → ESICKON: High-frequency clock on. Setting this bit turns the high-frequency clock source on for this state when ESICLK = 1, even though the high frequency clock is not used for the TSM. When the high-frequency clock is sourced from the DCO, the DCO is forced on for this state, regardless of the MSP430 low-power mode. 0b = High-frequency clock is off for this state when ESICLK = 1 1b = High-frequency clock is on for this state when ESICLK = 1 ESITSM.ESICKAZSEL=1 → ESICAAZ: Comparator Offset cancellation by doing an autozero cycle. 0b = "AZ-compensation Compare phase", Comparator compares (this phase must be preceded by the "AZ-compensation Auto Zero Phase" for each compare). 1b = "AZ-compensation Auto Zero phase", Comparator Offset cancellation sequence is active (autozero). The length for autozero is adjusted by the selected clock (ESICLK) and the programmed repeat cycles (ESIREPEATx). See device-specific data sheet for appropriate timing requirements.
4	ESICA	RW	0h	TSM comparator on. Setting this bit turns the AFE1 comparator and optionally the AFE2 comparator on for this state. 0b = AFE1 comparator and AFE2 comparator are off during this state 1b = AFE1 comparator is on during this state. AFE2 comparator is only on when ESICA2EN in ESIAFE control register is set.
3	ESIEX	RW	0h	Excitation and sample-and-hold. This bit, together with the ESISH and ESITEN bits, enables the excitation transistor or samples the input voltage during this state. ESILCEN must be set to 1 when ESIEX = 1. 0b = Excitation transistor disabled when ESISH = 0 and ESITEN = 1. Sampling disabled when ESISH = 1 and ESITEN = 0. 1b = Excitation transistor enabled when ESISH = 0 and ESITEN = 1. Sampling enabled when ESISH = 1 and ESITEN = 0.
2	ESILCEN	RW	0h	LC enable. Setting this bit turns the damping transistor off, enabling the LC oscillations during this state when ESITEN = 1. 0b = All ESICHx channels are internally damped. No LC oscillations. 1b = The selected ESICHx channel is not internally damped; the LC oscillates. All other unselected ESICHx channels are internally damped (no LC oscillations).
1-0	ESICHx	RW	0h	Input channel select. These bits select the input channel to be measured or excited during this state. 00b = ESICH0 01b = ESICH1 10b = ESICH2 11b = ESICH3

37.3.24 Extended Scan Interface Processing State Machine Table Entry (ESi Memory)

Extended Scan Interface Processing State Machine Table Entry (ESi Memory)

Figure 37-45. Extended Scan Interface Processing State Machine Table Entry Register

7	6	5	4	3	2	1	0
Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0

Table 37-34. Extended Scan Interface Processing State Machine Table Entry Description

Bit	Field	Type	Reset	Description
7	Q7			When Q7 = 1, ESIFG6 will be set. When ESQ6EN = 1 and ESQ7EN = 1 and Q7 = 1, the PSM proceeds to the next state immediately, regardless of the ESISTOP(tsm) signal and Q7 is used in the next-state calculation.
6	Q6			When Q6 = 1, ESIFG5 will be set. When ESQ6EN = 1, Q6 will be used in the next-state calculation.
5	Q5			Bit 5 of the next state
4	Q4			Bit 4 of the next state
3	Q3			Bit 3 of the next state
2	Q2			When Q2 = 1, ESICNT1 decrements if ESICNT1EN = 1 and ESICNT2 decrements if ESICNT2EN = 1.
1	Q1			When Q1 = 1, ESICNT1 increments if ESICNT1EN = 1 and ESICNT0 increments if ESICNT0EN = 1.
0	Q0			When ESIV2SEL=0 the PPUS3 signal is used as bit 2 (V2) for the next state. For the case ESIV2SEL=1 the Q0 bit is used.

Embedded Emulation Module (EEM)

This chapter describes the embedded emulation module (EEM) that is implemented in all devices.

Topic	Page
38.1 Embedded Emulation Module (EEM) Introduction	1018
38.2 EEM Building Blocks	1020
38.3 EEM Configurations	1021

38.1 Embedded Emulation Module (EEM) Introduction

Every MSP430 microcontroller implements an EEM. It is accessed and controlled through either 4-wire JTAG mode or Spy-Bi-Wire mode. Each implementation is device-dependent and is described in [Section 38.3](#), the EEM Configurations section, and the device-specific data sheet.

In general, the following features are available:

- Nonintrusive code execution with real-time breakpoint control
- Single-step, step-into, and step-over functionality
- Full support of all low-power modes
- Support for all system frequencies, for all clock sources
- Up to eight (device-dependent) hardware triggers or breakpoints on memory address bus (MAB) or memory data bus (MDB)
- Up to two (device-dependent) hardware triggers or breakpoints on CPU register write accesses
- MAB, MDB, and CPU register access triggers can be combined to form up to ten (device-dependent) complex triggers or breakpoints
- Up to two (device-dependent) cycle counters
- Trigger sequencing (device-dependent)
- Storage of internal bus and control signals using an integrated trace buffer (device-dependent)
- Clock control for timers, communication peripherals, and other modules on a global device level or on a per-module basis during an emulation stop
- EnergyTrace++ Technology

[Figure 38-1](#) shows a simplified block diagram of the largest currently-available EEM implementation.

For more details on how the features of the EEM can be used together with the IAR Embedded Workbench™ debugger or with Code Composer Studio (CCS), see [Advanced debugging using the enhanced emulation module \(EEM\) with CCS](#). Most other debuggers supporting the MSP430 devices have the same or a similar feature set. For details, see the user's guide of the applicable debugger.

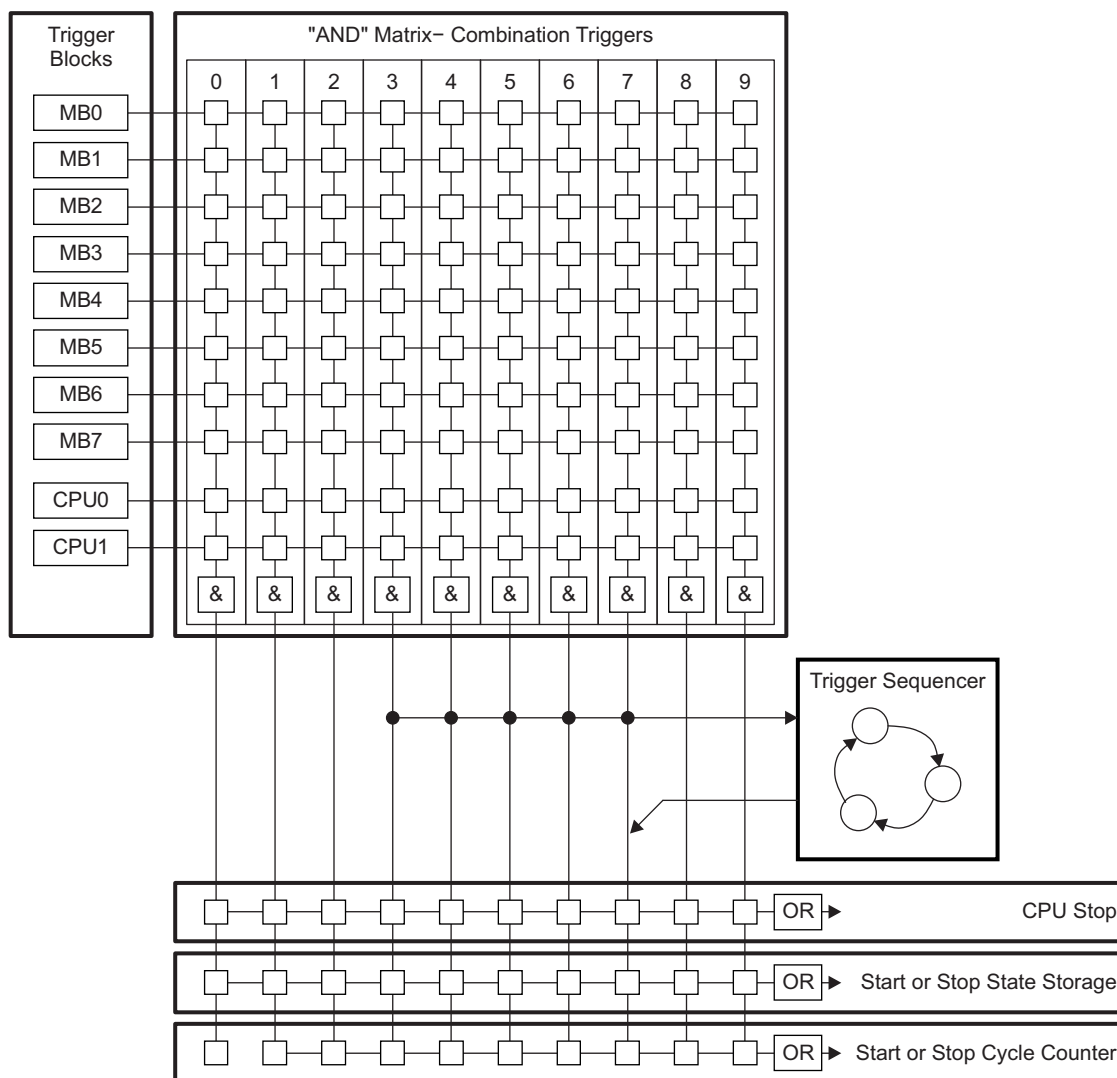


Figure 38-1. Large Implementation of EEM

38.2 EEM Building Blocks

38.2.1 Triggers

The event control in the EEM of the MSP430 system consists of triggers, which are internal signals indicating that a certain event has happened. These triggers may be used as simple breakpoints, but it is also possible to combine two or more triggers to allow detection of complex events and cause various reactions other than stopping the CPU.

In general, the triggers can be used to control the following functional blocks of the EEM:

- Breakpoints (CPU stop)
- State storage
- Sequencer
- Cycle counter

There are two different types of triggers – the memory trigger and the CPU register write trigger.

Each memory trigger block can be independently selected to compare either the MAB or the MDB with a given value. Depending on the implemented EEM, the comparison can be $=$, $\neq$, $\geq$, or $\leq$. The comparison can also be limited to certain bits with the use of a mask. The mask is either bit-wise or byte-wise, depending upon the device. In addition to selecting the bus and the comparison, the condition under which the trigger is active can be selected. The conditions include read access, write access, DMA access, and instruction fetch.

Each CPU register write trigger block can be independently selected to compare what is written into a selected register with a given value. The observed register can be selected for each trigger independently. The comparison can be $=$, $\neq$, $\geq$, or $\leq$. The comparison can also be limited to certain bits with the use of a bit mask.

Both types of triggers can be combined to form more complex triggers. For example, a complex trigger can signal when a particular value is written into a user-specified address.

38.2.2 Trigger Sequencer

The trigger sequencer allows the definition of a certain sequence of trigger signals before an event is accepted for a break or state storage event. Within the trigger sequencer, it is possible to use the following features:

- Four states (State 0 to State 3)
- Two transitions per state to any other state
- Reset trigger that resets the sequencer to State 0.

The trigger sequencer always starts at State 0 and must execute to State 3 to generate an action. If State 1 or State 2 are not required, they can be bypassed.

38.2.3 State Storage (Internal Trace Buffer)

The state storage function uses a built-in buffer to store MAB, MDB, and CPU control signal information (that is, read, write, or instruction fetch) in a nonintrusive manner. The built-in buffer can hold up to eight entries. The flexible configuration allows the user to record the information of interest very efficiently.

38.2.4 Cycle Counter

The cycle counter provides one or two 40-bit counters to measure the cycles used by the CPU to execute certain tasks. On some devices, the cycle counter operation can be controlled using triggers. This allows, for example, conditional profiling, such as profiling a specific section of code.

38.2.5 EnergyTrace++ Technology

The EEM implements circuitry to support EnergyTrace++ technology. The EnergyTrace++ technology allows you to observe information about the internal states of the microcontroller. These states include the CPU Program Counter (PC), the ON or OFF status of the peripherals and the system clocks (regardless of the clock source), and the low-power mode currently in use. These states can always be read by a debug tool, even when the microcontroller sleeps in LPMx.5 modes. See [Code Composer Studio™ IDE for MSP430 MCUs](#) for more information about integration into the IDE. See [MSP430™ advanced power optimizations: ULP advisor software and EnergyTrace™ technology](#) for examples of use.

38.2.6 Clock Control

The EEM provides device-dependent flexible clock control. This is useful in applications where a running clock is needed for peripherals after the CPU is stopped (for example, to allow a UART module to complete its transfer of a character or to allow a timer to continue generating a PWM signal).

The clock control is flexible and supports both modules that need a running clock and modules that must be stopped when the CPU is stopped due to a breakpoint.

38.2.7 Debug Modes

The TEST/SBWTK pin is used to enable the connection of external development tools with the EEM through Spy-Bi-Wire or JTAG debug protocols. The connection is usually enabled when the TEST/SBWTK is high. When the connection is enabled, the device enters a debug mode. In the debug mode, the entry and wakeup times to and from low-power modes may be different compared to normal operation (application mode).

NOTE: Pay careful attention to the real-time behavior when using low-power modes with the device connected to a development tool.

There are two different debug modes available: the default debug mode and a ultra-low power debug mode. See [Code Composer Studio IDE for MSP430 MCUs](#) for more information how to select the mode in the IDE.

Features and restrictions of the default debug mode are:

- It is possible to change breakpoint settings while the program is executed
- LPMx.5 is not supported
- Wakeup from low-power modes are faster than in application mode
- FRAM is forced on. It cannot be switched off using the FRAM Power Control bits

Features and restrictions of the ultra-low power debug mode are:

- It is not possible to change breakpoint settings while the program is executed
- LPMx.5 is supported
- Entry and wakeup times to and from low power modes may be longer than in application mode (for details see below)
- FRAM can be switched off using the FRAM Power Control bits.

In ultra-low power debug mode, the LPM entry and exit may be delayed. Low-power mode entry and wakeup from low-power modes is only possible while the debug protocol is in a certain state. Thus, the delay depends on the speed of the selected debug protocol. With Spy-Bi-Wire the delay is longer than when using JTAG. Also, the reaction on a DMA trigger or on a SMCLK or MCLK request may be delayed.

38.3 EEM Configurations

[Table 38-1](#) gives an overview of the EEM configurations. The implemented configuration is device-dependent, and details can be found in the device-specific data sheet and these documents:

[Advanced debugging using the enhanced emulation module \(EEM\) with CCS](#)
[IAR Embedded Workbench® IDE for MSP430™ MCUs](#)
[Code Composer Studio™ IDE for MSP430™ MCUs](#)

Table 38-1. EEM Configurations

Feature	XS	S	M	L
Memory bus triggers	2 (=, ≠ only)	3	5	8
Memory bus trigger mask for	1) Low byte 2) High byte 3) Four upper addr bits	1) Low byte 2) High byte 3) Four upper addr bits	1) Low byte 2) High byte 3) Four upper addr bits	All 16 or 20 bits
CPU register write triggers	0	1	1	2
Combination triggers	2	4	6	10
Sequencer	No	No	Yes	Yes
State storage	No	No	No	Yes
Cycle counter	1	1	1	2 (including triggered start or stop)

In general, the following features can be found on any device:

- At least two MAB or MDB triggers supporting:
 - Distinction between CPU, DMA, read, and write accesses
 - =, ≠, ≥, or ≤ comparison (in XS, only =, ≠)
- At least two trigger combination registers
- Hardware breakpoints using the CPU stop reaction
- At least one 40-bit cycle counter
- Enhanced clock control with individual control of module clocks
- EnergyTrace++ technology

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from December 16, 2017 to April 21, 2020	Page
• Corrected the result address in the Destination description in Section 4.4.5, Indirect Register Mode	137
• Updated the description in Section 7.6, FRAM ECC	292
• Updated the description in Section 8.3, FRAM ECC	302
• Corrected the name of the DMAxCTL register in the first paragraph in Section 11.2.3, Initiating DMA Transfers	348
• Added P5IV, P6IV, P7IV, P8IV, and P9IV in Table 12-3, Digital I/O Registers	374
• Replaced P1IV, P2IV, P3IV, and P4IV register description sections with Section 12.4.1, PxIV Register	388
• Updated a link and description in Section 17.2.3, Where to Start	449
• Removed WDTCTL_L and WDTCTL_H registers, because any read or write access must use word instructions	640
• Changed the offset of the WDTCTL register in Table 24-1, WDT_A Registers , to match base address in data sheets.	640
• Changed the step "Adjust the frequency", the example that follows this step, and the "Minimum possible calibration" note to clarify how the RTCCAL bit is used in Section 28.2.5, Real-Time Clock Calibration	696
• Added the note that begins "Disable the RTC interrupts when changing the V _{CORE} level..." in Section 29.2.6, Real-Time Clock Interrupts	724
• Added the table note that begins "Some bits in this register are retained..." in Table 29-2, RTC_C Registers	734
• Corrected the description of the LCDBLKONIFG and LCDBLKOFFIFG bits in Section 36.2.7, LCD Interrupts	941
• Corrected the column heading "ESITESTS1(tsm)" in Table 37-2, Selected Output Bits	972

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2020, Texas Instruments Incorporated

Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

⊖ View [MSP430FR6989IPNR](#) on WIN SOURCE

⊖ [Texas Instruments](#) Information

Optimize Your Supply Chain with WIN SOURCE Solutions

- ✓ Global Sourcing Solution
- ✓ Obsolete Management
- ✓ Cost Control Management
- ✓ Shortage Management
- ✓ Alternative Solution
- ✓ Excess Inventory Management