

# PAC5210

## *Power Application Controller™*

Multi-Mode Power Manager™

Configurable Analog Front End™

Application Specific Power Drivers™

ARM® Cortex™-M0 Controller Core



## TABLE OF CONTENTS

1. Styles and Formatting Conventions.....	31
1.1. Overview.....	31
1.2. Number Representation.....	31
1.3. Formatting Styles.....	31
2. Memory and Register Map.....	32
2.1. Memory Map.....	32
2.2. Register Map.....	33
3. System Clock Control.....	45
3.1. Register.....	45
3.1.1. Register Map.....	45
3.1.2. SCCTL.....	45
3.1.3. PLLCTL.....	46
3.1.4. ROSCCTL.....	46
3.1.5. XTALCTL.....	46
3.2. Details of Operation.....	47
3.2.1. Block Diagram.....	47
3.2.2. Configuration.....	47
3.2.3. ROSC.....	48
3.2.4. CLKREF.....	48
3.2.5. XTAL.....	48
3.2.6. EXTCLK.....	48
3.2.7. PLL.....	48
3.2.8. FRCLK.....	49
3.2.9. FCLK.....	49
3.2.10. HCLK.....	49
3.2.11. ACLK.....	49
3.2.12. Clock Gating.....	49
4. Watchdog Timer.....	50
4.1. Register.....	50
4.1.1. Register Map.....	50
4.1.2. WDTCTL.....	50
4.1.3. WDTCDV.....	51
4.1.4. WDTCTR.....	51
4.2. Details of Operation.....	52
4.2.1. Block Diagram.....	52
4.2.2. Configuration.....	52
4.2.3. Watchdog Timer.....	52
4.2.4. Access WDT Registers.....	52
4.2.5. WDT Clock Setting.....	52
4.2.6. General Purpose Timer Mode.....	52
4.2.7. Watchdog Timer Mode.....	53
5. General Purpose Timer.....	54
5.1. Register.....	54
5.1.1. Register Map.....	54
5.1.2. GPTCTL.....	54
5.1.3. GPTCDV.....	55
5.1.4. GPTCTR.....	55
5.2. Details of Operation.....	56
5.2.1. Block Diagram.....	56
5.2.2. Configuration.....	56
5.2.3. General Purpose Timer.....	56

5.2.4. Access GPT Registers.....	56
5.2.5. GPT Clock Setting.....	56
5.2.6. General Purpose Timer Mode.....	56
6. GPIO Port A.....	57
6.1. Register.....	57
6.1.1. Register Map.....	57
6.1.2. GPIOAO.....	57
6.1.3. GPIOAOE.....	58
6.1.4. GPIOADS.....	58
6.1.5. GPIOARUP.....	59
6.1.6. GPIOARDN.....	60
6.1.7. GPIOAI.....	60
6.1.8. GPIOAPSEL.....	61
6.1.9. GPIOAINTP.....	62
6.1.10. GPIOAINTEN.....	62
6.1.11. GPIOAINT.....	63
6.1.12. GPIOAINTM.....	64
6.2. Details of Operation.....	65
6.2.1. Block Diagram.....	65
6.2.2. Configuration.....	65
6.2.3. GPIO A Block.....	65
6.2.4. Input.....	65
6.2.5. Output and Output Enable.....	65
6.2.6. Output Drive Strength.....	66
6.2.7. Weak Pull Up and Pull Down.....	66
6.2.8. Peripheral Select.....	66
6.2.9. Interrupt.....	66
7. GPIO Port B.....	67
7.1. Register.....	67
7.1.1. Register Map.....	67
7.1.2. GPIOBO.....	67
7.1.3. GPIOBOE.....	67
7.1.4. GPIOBDS.....	68
7.1.5. GPIOBRUP.....	68
7.1.6. GPIOBRDN.....	69
7.1.7. GPIOBI.....	69
7.1.8. GPIOBPSEL.....	69
7.1.9. GPIOBINTP.....	70
7.1.10. GPIOBINTEN.....	71
7.1.11. GPIOBINT.....	71
7.1.12. GPIOBINTM.....	72
7.2. Details of Operation.....	73
7.2.1. Block Diagram.....	73
7.2.2. Configuration.....	73
7.2.3. GPIO B Block.....	73
7.2.4. Input.....	73
7.2.5. Output and Output Enable.....	73
7.2.6. Output Drive Strength.....	73
7.2.7. Weak Pull Up and Pull Down.....	74
7.2.8. Peripheral Select.....	74
7.2.9. Interrupt.....	74
8. GPIO Port C.....	75
8.1. Register.....	75
8.1.1. Register Map.....	75

8.1.2. GPIOCO.....	75
8.1.3. GPIOCOE.....	76
8.1.4. GPIOCI.....	76
8.1.5. GPIOCIE.....	77
8.1.6. GPIOCINTP.....	78
8.1.7. GPIOCINTEN.....	78
8.1.8. GPIOCINT.....	79
8.1.9. GPIOCINTM.....	79
8.2. Details of Operation.....	81
8.2.1. Block Diagram.....	81
8.2.2. Configuration.....	81
8.2.3. GPIO C Block.....	81
8.2.4. Analog Input.....	81
8.2.5. Output and Output Enable.....	81
8.2.6. Interrupt.....	81
9. GPIO Port D.....	83
9.1. Register.....	83
9.1.1. Register Map.....	83
9.1.2. GPIODO.....	83
9.1.3. GPIODOE.....	84
9.1.4. GPIODDS.....	84
9.1.5. GPIODRUP.....	85
9.1.6. GPIODRDN.....	86
9.1.7. GPIODI.....	86
9.1.8. GPIODPSEL.....	87
9.1.9. GPIODINTP.....	88
9.1.10. GPIODINTEN.....	88
9.1.11. GPIODINT.....	89
9.1.12. GPIODINTM.....	89
9.2. Details of Operation.....	91
9.2.1. Block Diagram.....	91
9.2.2. Configuration.....	91
9.2.3. GPIO D Block.....	91
9.2.4. Input.....	91
9.2.5. Output and Output Enable.....	91
9.2.6. Output Drive Strength.....	92
9.2.7. Weak Pull Up and Pull Down.....	92
9.2.8. Peripheral Select.....	92
9.2.9. Interrupt.....	92
10. GPIO Port E.....	93
10.1. Register.....	93
10.1.1. Register Map.....	93
10.1.2. GPIOEO.....	93
10.1.3. GPIOEOE.....	94
10.1.4. GPIOEDS.....	94
10.1.5. GPIOERUP.....	95
10.1.6. GPIOERDN.....	96
10.1.7. GPIOEI.....	96
10.1.8. GPIOEPSEL.....	97
10.1.9. GPIOEINTP.....	98
10.1.10. GPIOEINTEN.....	98
10.1.11. GPIOEINT.....	99
10.1.12. GPIOEINTM.....	99
10.2. Details of Operation.....	101

10.2.1. Block Diagram.....	101
10.2.2. Configuration.....	101
10.2.3. GPIO E Block.....	101
10.2.4. Input.....	101
10.2.5. Output and Output Enable.....	101
10.2.6. Output Drive Strength.....	101
10.2.7. Weak Pull Up and Pull Down.....	102
10.2.8. Peripheral Select.....	102
10.2.9. Interrupt.....	102
11. Timer A.....	103
11.1. Register.....	103
11.1.1. Register Map.....	103
11.1.2. TACTL.....	104
11.1.3. TAPRD.....	105
11.1.4. TACTR.....	105
11.1.5. TACC0CTL.....	105
11.1.6. TACC0CTR.....	105
11.1.7. TACC1CTL.....	106
11.1.8. TACC1CTR.....	106
11.1.9. TACC2CTL.....	106
11.1.10. TACC2CTR.....	107
11.1.11. TACC3CTL.....	107
11.1.12. TACC3CTR.....	107
11.1.13. TACC4CTL.....	107
11.1.14. TACC4CTR.....	108
11.1.15. TACC5CTL.....	108
11.1.16. TACC5CTR.....	108
11.1.17. TACC6CTL.....	109
11.1.18. TACC6CTR.....	109
11.1.19. TACC7CTL.....	109
11.1.20. TACC7CTR.....	110
11.1.21. DTGA0CTL.....	110
11.1.22. DTGA0LED.....	110
11.1.23. DTGA0TED.....	110
11.1.24. DTGA1CTL.....	111
11.1.25. DTGA1LED.....	111
11.1.26. DTGA1TED.....	111
11.1.27. DTGA2CTL.....	111
11.1.28. DTGA2LED.....	112
11.1.29. DTGA2TED.....	112
11.1.30. DTGA3CTL.....	112
11.1.31. DTGA3LED.....	113
11.1.32. DTGA3TED.....	113
11.2. Details of Operation.....	114
11.2.1. Block Diagram.....	114
11.2.2. Configuration.....	114
11.2.3. Timer A Block.....	114
11.2.4. Timer.....	114
11.2.5. Register update.....	115
11.2.6. Timer Modes.....	115
11.2.7. Single Shot Mode.....	115
11.2.8. Input Clock And Pre-Scaler.....	115
11.2.9. Timer Synchronization.....	115
11.2.10. PWM/Compare Units.....	116

11.2.10.1. PWM Mode.....	116
11.2.10.2. Capture Mode.....	117
11.2.11. Timer and PWM/Capture Interrupt.....	117
11.2.12. Dead-Time Generator.....	118
11.2.12.1. Dead Time Input Clock Selection.....	118
11.2.12.2. Dead Time Range.....	118
11.2.12.3. Bypass Mode.....	118
11.2.12.4. Inverting PWM Signal.....	118
11.2.12.5. Dead Time Insertion.....	119
11.2.12.6. Dead Time Insertion with On Time Preservation.....	119
11.2.13. PWM Output and Capture Input Pin Selection.....	120
12. Timer B.....	121
12.1. Register.....	121
12.1.1. Register Map.....	121
12.1.2. TBCTL.....	121
12.1.3. TBPRD.....	122
12.1.4. TBCTR.....	122
12.1.5. TBCC0CTL.....	122
12.1.6. TBCC0CTR.....	123
12.1.7. TBCC1CTL.....	123
12.1.8. TBCC1CTR.....	123
12.1.9. TBCC2CTL.....	124
12.1.10. TBCC2CTR.....	124
12.1.11. TBCC3CTL.....	124
12.1.12. TBCC3CTR.....	125
12.1.13. DTGB0CTL.....	125
12.1.14. DTGB0LED.....	125
12.1.15. DTGB0TED.....	126
12.2. Details of Operation.....	127
12.2.1. Block Diagram.....	127
12.2.2. Configuration.....	127
12.2.3. Timer B Block.....	127
12.2.4. Timer.....	127
12.2.5. Register update.....	128
12.2.6. Timer Modes.....	128
12.2.7. Single Shot Mode.....	128
12.2.8. Input Clock And Pre-Scaler.....	128
12.2.9. Timer Synchronization.....	128
12.2.10. PWM/Compare Units.....	129
12.2.10.1. PWM Mode.....	129
12.2.10.2. Capture Mode.....	130
12.2.11. Timer and PWM/Capture Interrupt.....	130
12.2.12. Dead-Time Generator.....	131
12.2.12.1. Dead Time Input Clock Selection.....	131
12.2.12.2. Dead Time Range.....	131
12.2.12.3. Bypass Mode.....	131
12.2.12.4. Inverting PWM Signal.....	131
12.2.12.5. Dead Time Insertion.....	132
12.2.12.6. Dead Time Insertion with On Time Preservation.....	132
12.2.13. PWM Output and Capture Input Pin Selection.....	133
13. Timer C.....	134
13.1. Register.....	134
13.1.1. Register Map.....	134
13.1.2. TCCTL.....	134

13.1.3. TCPRD.....	135
13.1.4. TCCTR.....	135
13.1.5. TCCC0CTL.....	135
13.1.6. TCCC0CTR.....	136
13.1.7. TCCC1CTL.....	136
13.1.8. TCCC1CTR.....	136
13.1.9. DTGC0CTL.....	137
13.1.10. DTGC0LED.....	137
13.1.11. DTGC0TED.....	137
13.2. Details of Operation.....	138
13.2.1. Block Diagram.....	138
13.2.2. Configuration.....	138
13.2.3. Timer C Block.....	138
13.2.4. Timer.....	138
13.2.5. Register update.....	139
13.2.6. Timer Modes.....	139
13.2.7. Single Shot Mode.....	139
13.2.8. Input clock and Pre-scaler.....	139
13.2.9. Timer synchronization.....	139
13.2.10. PWM/Compare Units.....	140
13.2.10.1. PWM Mode.....	140
13.2.10.2. Capture Mode.....	141
13.2.11. Timer and PWM/Capture Interrupt.....	141
13.2.12. Dead-Time Generator.....	142
13.2.12.1. Dead Time Input Clock Selection.....	142
13.2.12.2. Dead Time Range.....	142
13.2.12.3. Bypass Mode.....	142
13.2.12.4. Inverting PWM Signal.....	142
13.2.12.5. Dead Time Insertion.....	143
13.2.12.6. Dead Time Insertion With On Time Preservation.....	143
13.2.13. PWM Output and Capture Input Pin Selection.....	144
14. Timer D.....	145
14.1. Register.....	145
14.1.1. Register Map.....	145
14.1.2. TDCTL.....	145
14.1.3. TDPRD.....	146
14.1.4. TDCTR.....	146
14.1.5. TDCC0CTL.....	146
14.1.6. TDCC0CTR.....	147
14.1.7. TDCC1CTL.....	147
14.1.8. TDCC1CTR.....	147
14.1.9. DTGD0CTL.....	148
14.1.10. DTGD0LED.....	148
14.1.11. DTGD0TED.....	148
14.2. Details of Operation.....	149
14.2.1. Block Diagram.....	149
14.2.2. Configuration.....	149
14.2.3. Timer D Block.....	149
14.2.4. Timer.....	149
14.2.5. Register Update.....	150
14.2.6. Timer Modes.....	150
14.2.7. Single Shot Mode.....	150
14.2.8. Input Clock And Pre-Scaler.....	150
14.2.9. Timer Synchronization.....	150

14.2.10. PWM/Compare Units.....	151
14.2.10.1. PWM Mode.....	151
14.2.10.2. Capture Mode.....	152
14.2.11. Timer and PWM/Capture Interrupt.....	152
14.2.12. Dead-Time Generator.....	152
14.2.12.1. Dead Time Input Clock Selection.....	152
14.2.12.2. Dead Time Range.....	152
14.2.12.3. Bypass Mode.....	153
14.2.12.4. Inverting PWM Signal.....	153
14.2.12.5. Dead Time Insertion.....	153
14.2.12.6. Dead Time Insertion with On Time Preservation.....	154
14.2.13. PWM Output and Capture Input Pin Selection.....	154
15. FLASH Memory Controller.....	155
15.1. Register.....	155
15.1.1. Register Map.....	155
15.1.2. FLASHLOCK.....	155
15.1.3. FLASHSTATUS.....	156
15.1.4. FLASHPAGE.....	156
15.1.5. FLASHPERASE.....	157
15.1.6. SWDACCESS.....	157
15.1.7. FLASHWSTATE.....	157
15.1.8. FLASHBWRITE.....	157
15.1.9. FLASHBWDATA.....	158
15.2. Details of Operation.....	159
15.2.1. Block Diagram.....	159
15.2.2. Configuration.....	159
15.2.3. FLASH Memory.....	159
15.2.4. Writing to FLASH Controller Registers.....	159
15.2.5. FLASH Wait State.....	159
15.2.6. FLASH Page Erase.....	160
15.2.7. Write to FLASH.....	160
15.2.8. Buffered Write to FLASH.....	160
15.2.9. SWD Debug Access Disable.....	161
16. ADC and AUTO SEQUENCER.....	162
16.1. Register.....	162
16.1.1. Register Map.....	162
16.1.2. EMUXCTL.....	163
16.1.3. EMUXDATA.....	164
16.1.4. ADCCTL.....	164
16.1.5. ADCCR.....	165
16.1.6. ADCINT.....	165
16.1.7. AS0CTL.....	166
16.1.8. AS0S0.....	167
16.1.9. AS0R0.....	167
16.1.10. AS0S1.....	168
16.1.11. AS0R1.....	168
16.1.12. AS0S2.....	168
16.1.13. AS0R2.....	169
16.1.14. AS0S3.....	169
16.1.15. AS0R3.....	169
16.1.16. AS0S4.....	170
16.1.17. AS0R4.....	170
16.1.18. AS0S5.....	170
16.1.19. AS0R5.....	171



16.1.20. AS0S6.....	171
16.1.21. AS0R6.....	171
16.1.22. AS0S7.....	172
16.1.23. AS0R7.....	172
16.1.24. AS1CTL.....	172
16.1.25. AS1S0.....	173
16.1.26. AS1R0.....	174
16.1.27. AS1S1.....	174
16.1.28. AS1R1.....	174
16.1.29. AS1S2.....	174
16.1.30. AS1R2.....	175
16.1.31. AS1S3.....	175
16.1.32. AS1R3.....	176
16.1.33. AS1S4.....	176
16.1.34. AS1R4.....	176
16.1.35. AS1S5.....	176
16.1.36. AS1R5.....	177
16.1.37. AS1S6.....	177
16.1.38. AS1R6.....	178
16.1.39. AS1S7.....	178
16.1.40. AS1R7.....	178
16.2. Details of Operation.....	178
16.2.1. Block Diagram.....	178
16.3. Details of Operation.....	179
16.3.1. Basic Configuration.....	179
16.3.2. ADC, Autosequencer and EMUX.....	179
16.3.3. Clock Setting.....	180
16.3.4. ADC.....	180
16.3.5. EMUX.....	180
16.3.6. Auto Sequencer ASC0, ASC1.....	181
16.3.6.1. Auto Sequencer Modes.....	181
16.3.6.2. Sequencer trigger.....	182
16.3.6.3. ASC Samples.....	182
16.3.6.4. ASC0, ASC1 Priority and Collision.....	183
17. I2C.....	185
17.1. Register.....	185
17.1.1. Register Map.....	185
17.1.2. I2CCFG.....	185
17.1.3. I2CINT.....	185
17.1.4. I2CINTEN.....	187
17.1.5. I2CMACTL.....	187
17.1.6. I2CMARXDATA.....	188
17.1.7. I2CMATXDATA.....	188
17.1.8. I2CBAUD.....	188
17.1.9. I2CSL RXDATA.....	188
17.1.10. I2CSL TXDATA.....	189
17.1.11. I2CSLADDR.....	189
17.2. Details of Operation.....	190
17.2.1. Block Diagram.....	190
17.2.2. Configuration.....	190
17.2.3. I2C.....	190
17.2.4. I2C Clock setting.....	190
17.2.5. I2C Addressing.....	191
18. UART.....	192

18.1. Register.....	192
18.1.1. Register Map.....	192
18.1.2. UARTRTX/UARTDL_L.....	192
18.1.3. UARTIEN/UARTDL_H.....	193
18.1.4. UARTII.....	193
18.1.5. UARTLC.....	194
18.1.6. UARTLS.....	194
18.1.7. UARFCTL.....	195
18.1.8. UARTIEN.....	195
18.1.9. UARTDL_L.....	195
18.1.10. UARTDL_H.....	196
18.1.11. UARTFD_F.....	196
18.1.12. UARTFSTAT.....	196
18.2. Details of Operation.....	197
18.2.1. Block Diagram.....	197
18.2.2. Configuration.....	197
18.2.3. UART.....	197
18.2.4. UART Clock Rate Setting.....	197
18.2.5. Data settings.....	199
18.2.6. FIFO Settings.....	199
18.2.7. Error Checking on Received Data.....	199
19. SOC Bus bridge.....	200
19.1. Register.....	200
19.1.1. Register Map.....	200
19.1.2. SOCBCTL.....	200
19.1.3. SOCBCFG.....	200
19.1.4. SOCBCLKDIV.....	201
19.1.5. SOCBSTAT.....	201
19.1.6. SOCBD.....	202
19.1.7. SOCBINT_EN.....	203
19.2. Details of Operation.....	204
19.2.1. Block Diagram.....	204
19.2.2. Configuration.....	204
19.2.3. SOC Bridge.....	204
19.2.4. SOC Bridge Clock Rate Setting.....	204
19.2.5. Enable and Setup of SOC Bridge.....	205
19.2.6. SOC Interrupt.....	205
19.2.7. SOC Bridge Protocol.....	205
19.2.8. Reading from SOC Bridge.....	205
19.2.9. Writing to SOC Bridge.....	205
20. SPI.....	206
20.1. Register.....	206
20.1.1. Register Map.....	206
20.1.2. SPICTL.....	206
20.1.3. SPICFG.....	207
20.1.4. SPICLKDIV.....	208
20.1.5. SPISTAT.....	208
20.1.6. SPICSSTR.....	210
20.1.7. SPID.....	210
20.1.8. SPIINT_EN.....	211
20.2. Details of Operation.....	212
20.2.1. Block Diagram.....	212
20.2.2. Configuration.....	212
20.2.3. SPI.....	212

20.2.4. SPI Clock Rate Setting.....	212
20.2.5. Master Slave Mode.....	213
20.2.6. Clock Phase, Polarity.....	213
20.2.7. SPI Early Data Transmit.....	213
20.2.8. Data Format.....	214
20.2.9. Chip Select Settings.....	215
20.2.10. Auto Retransmit Data Word.....	215
20.2.11. Loop Back Mode.....	215
20.2.12. SPI Interrupt.....	216
20.2.13. SPI Enable.....	216
21. Multi-mode Power Manager.....	217
21.1. Register.....	217
21.1.1. Register Map.....	217
21.1.2. SOC.SYSSTAT.....	217
21.1.3. SOC.PWRCTL.....	218
21.1.4. SOC.PWRSTAT.....	218
21.1.5. SOC.PWRSTATCFG.....	219
21.1.6. SOC.SCFG.....	219
21.1.7. SOC.PWRCFG.....	220
22. Configurable Analog FRONT END.....	221
22.1. Register.....	221
22.1.1. Register Map.....	221
22.1.2. SOC.AIO0CFG.....	221
22.1.3. SOC.AIO1CFG.....	222
22.1.4. SOC.AIO2CFG.....	223
22.1.5. SOC.AIO3CFG.....	224
22.1.6. SOC.AIO4CFG.....	225
22.1.7. SOC.AIO5CFG.....	226
22.1.8. SOC.AIO6CFG.....	227
22.1.9. SOC.AIO7CFG.....	229
22.1.10. SOC.AIO8CFG.....	230
22.1.11. SOC.AIO9CFG.....	232
22.1.12. SOC.SMCFG.....	233
22.1.13. SOC.HPDAC.....	234
22.1.14. SOC.LPDAC0.....	234
22.1.15. SOC.LPDAC1.....	234
22.1.16. SOC.ADCSCANCFG.....	234
22.1.17. SOC.ADMUX.....	235
22.1.18. SOC.PROTINTEN.....	235
22.1.19. SOC.PROTINT.....	236
22.1.20. SOC.AIOOUT0.....	236
22.1.21. SOC.AIOOUT1.....	237
22.1.22. SOC.AIOIN0.....	237
22.1.23. SOC.AIOIN1.....	238
22.1.24. SOC.AIOINTEN.....	238
22.1.25. SOC.AIOINT.....	238
22.1.26. SOC.SMCTL.....	239
22.2. Details of Operation.....	240
22.2.1. Block Diagram.....	240
22.2.2. Configuration.....	240
22.2.3. Configurable Analog Front End.....	240
22.2.3.1. Configurable Analog Front End Enable.....	240
22.3. AIO1, AIO0.....	241
22.3.1. Block Diagram.....	241

22.3.2. AIO1, AIO0.....	241
22.3.3. AIO1, AIO0 digital I/O Mode.....	241
22.3.3.1. AIO0 IO.....	242
22.3.3.2. AIO1 IO.....	242
22.3.3.3. AIO0 Polarity.....	242
22.3.3.4. AIO1 Polarity.....	242
22.3.4. AIO1, AIO0 differential Amplifier Mode.....	242
22.3.4.1. AIO1, AIO0 Differential Amplifier Gain.....	242
22.3.4.2. AIO1, AIO0 Differential Amplifier Reference.....	242
22.3.4.3. AIO1, AIO0 Differential Amplifier Calibration.....	242
22.3.5. AIO1, AIO0 Protection.....	242
22.3.5.1. HP10 Comparator.....	243
22.3.5.2. LP10 Comparator.....	243
22.4. AIO3, AIO2.....	244
22.4.1. Block Diagram.....	244
22.4.2. AIO3, AIO2.....	244
22.4.3. AIO3, AIO2 digital I/O Mode.....	244
22.4.3.1. AIO2 IO.....	245
22.4.3.2. AIO3 IO.....	245
22.4.3.3. AIO2 Polarity.....	245
22.4.3.4. AIO3 Polarity.....	245
22.4.4. AIO3, AIO2 differential Amplifier Mode DAO32.....	245
22.4.4.1. DAO32 Differential Amplifier Gain.....	245
22.4.4.2. DAO32 Differential Amplifier Reference.....	245
22.4.4.3. DAO32 Differential Amplifier Calibration.....	245
22.4.5. AIO3, AIO2 Protection.....	245
22.4.5.1. HP32 Comparator.....	246
22.4.5.2. LP32 Comparator.....	246
22.5. AIO5, AIO4.....	247
22.5.1. Block Diagram.....	247
22.5.2. AIO5, AIO4.....	247
22.5.3. AIO5, AIO4 digital I/O Mode.....	247
22.5.3.1. AIO4 IO.....	248
22.5.3.2. AIO5 IO.....	248
22.5.3.3. AIO4 Polarity.....	248
22.5.3.4. AIO5 Polarity.....	248
22.5.4. AIO5, AIO4 differential Amplifier Mode DAO54.....	248
22.5.4.1. DAO54 Differential Amplifier Gain.....	248
22.5.4.2. DAO54 Differential Amplifier Reference.....	248
22.5.4.3. DAO54 Differential Amplifier Calibration.....	248
22.5.5. AIO5, AIO4 Protection.....	248
22.5.5.1. HP54 Comparator.....	249
22.5.5.2. LP54 Comparator.....	249
22.6. AIO6.....	250
22.6.1. Block Diagram.....	250
22.6.2. AIO6.....	250
22.6.3. AIO6 digital I/O Mode.....	251
22.6.3.1. AIO6 IO.....	251
22.6.3.2. AIO6 IO Interrupt.....	251
22.6.3.3. AIO6 Polarity.....	251
22.6.4. AIO6 Single Ended Amplifier Mode.....	251
22.6.4.1. AIO6 Amplifier Gain.....	251
22.6.4.2. AIO6 Analog MUX.....	251
22.6.5. AIO6 Comparator Mode.....	251

22.6.5.1. AIO6 Comparator Hysteresis.....	252
22.6.5.2. AIO6 Comparator setting.....	252
22.6.5.3. AIO6 Comparator Polarity.....	252
22.6.5.4. AIO6 Comparator Output MUX.....	252
22.6.6. AIO6 Special Mode.....	252
22.6.6.1. AIO6 Special Mode MUX.....	252
22.6.6.2. AIO6 Special Mode ADMUX.....	252
22.6.6.3. AIO6 Special Mode OFFSET SWAP.....	252
22.6.7. AIO6 Push Button Mode.....	252
22.6.7.1. AIO6 Push Button Wake Up.....	253
22.6.7.2. AIO6 Push Button Power Down.....	253
22.6.7.3. AIO6 Push Button Hard Reset.....	253
22.7. AIO7.....	254
22.7.1. Block Diagram.....	254
22.7.2. AIO7.....	254
22.7.3. AIO7 digital I/O Mode.....	254
22.7.3.1. AIO7 IO.....	255
22.7.3.2. AIO7 IO Interrupt.....	255
22.7.3.3. AIO7 Polarity.....	255
22.7.4. AIO7 Single Ended Amplifier Mode.....	255
22.7.4.1. AIO7 Amplifier Gain.....	255
22.7.4.2. AIO7 Analog MUX.....	255
22.7.5. AIO7 Comparator Mode.....	255
22.7.5.1. AIO7 Comparator Hysteresis.....	255
22.7.5.2. AIO7 Comparator Reference.....	256
22.7.5.3. AIO7 Comparator Polarity.....	256
22.7.5.4. AIO7 Comparator Output.....	256
22.7.6. AIO7 Special Mode.....	256
22.7.6.1. AIO7 Comparator Hysteresis.....	256
22.7.6.2. AIO7 Comparator Reference.....	256
22.7.6.3. AIO7 Comparator Reference Star Point.....	256
22.7.6.4. AIO7 Voltage Reading.....	256
22.7.6.5. AIO7 Comparator Output.....	256
22.7.6.6. AIO7 POS S/H Bypass.....	257
22.7.6.7. AIO7 nIRQ2/POS Selector.....	257
22.8. AIO8.....	258
22.8.1. Block Diagram.....	258
22.8.2. AIO8.....	258
22.8.3. AIO8 digital I/O Mode.....	259
22.8.3.1. AIO8 IO.....	259
22.8.3.2. AIO8 IO Interrupt.....	259
22.8.3.3. AIO8 Polarity.....	259
22.8.4. AIO8 Single Ended Amplifier Mode.....	259
22.8.4.1. AIO8 Amplifier Gain.....	259
22.8.4.2. AIO8 Analog MUX.....	259
22.8.5. AIO8 Comparator Mode.....	259
22.8.5.1. AIO8 Comparator Hysteresis.....	259
22.8.5.2. AIO8 Comparator Reference.....	260
22.8.5.3. AIO8 Comparator Polarity.....	260
22.8.5.4. AIO8 Comparator Output.....	260
22.8.6. AIO8 Special Mode.....	260
22.8.6.1. AIO8 Comparator Hysteresis.....	260
22.8.6.2. AIO8 Comparator Reference.....	260
22.8.6.3. AIO8 Comparator Reference Star Point.....	260

22.8.6.4. AIO8 Voltage Reading.....	260
22.8.6.5. AIO8 Comparator Output.....	261
22.8.6.6. AIO8 POS S/H Bypass.....	261
22.8.6.7. AIO8 nIRQ2/POS Selector.....	261
22.9. AIO9.....	262
22.9.1. Block Diagram.....	262
22.9.2. AIO9.....	262
22.9.3. AIO9 digital I/O Mode.....	263
22.9.3.1. AIO9 IO.....	263
22.9.3.2. AIO9 IO Interrupt.....	263
22.9.3.3. AIO9 Polarity.....	263
22.9.4. AIO9 Single Ended Amplifier Mode.....	263
22.9.4.1. AIO9 Amplifier Gain.....	263
22.9.4.2. AIO9 Analog MUX.....	263
22.9.5. AIO9 Comparator Mode.....	263
22.9.5.1. AIO9 Comparator Hysteresis.....	263
22.9.5.2. AIO9 Comparator Reference.....	264
22.9.5.3. AIO9 Comparator Polarity.....	264
22.9.5.4. AIO9 Comparator Output.....	264
22.9.6. AIO9 Special Mode.....	264
22.9.6.1. AIO9 Comparator Hysteresis.....	264
22.9.6.2. AIO9 Comparator Reference.....	264
22.9.6.3. AIO9 Comparator Reference Star Point.....	264
22.9.6.4. AIO9 Voltage Reading.....	264
22.9.6.5. AIO9 Comparator Output.....	265
22.9.6.6. AIO9 POS S/H Bypass.....	265
22.9.6.7. AIO9 nIRQ2/POS Selector.....	265
22.10. EMUX and ADMUX.....	266
22.10.1. Block Diagram.....	266
22.10.2. EMUX.....	266
22.10.2.1. EMUX Enable.....	266
22.10.2.2. EMUXD Packet Structure.....	266
22.10.2.3. EMUX Timing.....	267
22.10.3. ADMUX.....	267
22.10.3.1. ADMUX Control.....	267
22.10.3.2. ADMUX Buffer.....	268
22.10.3.3. ADMUX.....	268
23. Application Specific Power Driver.....	269
23.1. Register.....	269
23.1.1. Register Map.....	269
23.1.2. SOC.PRCTL.....	269
23.1.3. SOC.OMOUT.....	269
23.1.4. SOC.DRVCTL.....	270
23.1.5. SOC.PROTCTL.....	270
23.2. Details of Operation.....	271
23.2.1. Block Diagram.....	271
23.2.2. Configuration.....	271
23.2.3. Application Specific Power Driver.....	271
23.2.3.1. Application Specific Power Driver Enable.....	271
23.3. OM0.....	272
23.3.1. Block Diagram.....	272
23.3.2. OM0.....	272
23.3.2.1. OM0 Open Drain Output.....	272
23.4. OM2.....	273

23.4.1. Block Diagram.....	273
23.4.2. OM2.....	273
23.4.2.1. OM2 Open Drain Output.....	273
23.5. OM4.....	274
23.5.1. Block Diagram.....	274
23.5.2. OM4.....	274
23.5.2.1. OM4 Open Drain Output.....	274
23.6. ENHS1 Protection.....	275
23.6.1. Block Diagram.....	275
23.6.2. ENHS1 Protection.....	275
23.6.2.1. Protection Enable.....	275
23.6.2.2. Input Signal Selection.....	275
23.7. ENHS2 Protection.....	276
23.7.1. Block Diagram.....	276
23.7.2. ENHS2 Protection.....	276
23.7.2.1. Protection Enable.....	276
24. Arm Cortex-M0 Reference.....	277
24.1. Introduction.....	277
24.1.1. Overview.....	277
24.1.2. About the Cortex-M0 processor and core peripherals.....	277
24.1.2.1. System-level interface.....	278
24.1.2.2. Integrated configurable debug.....	278
24.1.2.3. Cortex-M0 processor features summary.....	278
24.1.2.4. Cortex-M0 core peripherals.....	278
24.1.2.4.1. NVIC.....	278
24.1.2.4.2. System Control Block.....	279
24.1.2.4.3. System timer.....	279
24.2. The Cortex-M0 Processor.....	279
24.2.1. Programmers Model.....	279
24.2.1.1. Processor modes.....	279
24.2.1.2. Stacks.....	279
24.2.1.3. Core Registers.....	280
24.2.1.3.1. General-purpose registers.....	280
24.2.1.3.2. Stack Pointer.....	281
24.2.1.3.3. Link Register.....	281
24.2.1.3.4. Program Counter.....	281
24.2.1.3.5. Program Status Register.....	281
24.2.1.3.6. Application Program Status Register.....	282
24.2.1.3.7. Interrupt Program Status Register.....	282
24.2.1.3.8. Execution Program Status Register.....	283
24.2.1.3.9. Interruptible-restartable instructions.....	283
24.2.1.3.10. Exception mask register.....	284
24.2.1.3.11. Priority Mask Register.....	284
24.2.1.3.12. Control Register.....	284
24.2.1.4. Exceptions and interrupts.....	285
24.2.1.5. Data Types.....	285
24.2.1.6. The Cortex Microcontroller Software Interface Standard.....	286
24.2.2. Memory model.....	286
24.2.2.1. Memory regions, types and attributes.....	287
24.2.2.1.1. Normal.....	287
24.2.2.1.2. Device.....	288
24.2.2.1.3. Strongly-ordered.....	288
24.2.2.1.4. Execute Never (XN).....	288
24.2.2.2. Memory system ordering of memory accesses.....	288



24.2.2.3. Behavior of memory accesses.....	289
24.2.2.4. Software ordering of memory accesses.....	289
24.2.2.4.1. DMB.....	289
24.2.2.4.2. DSB.....	289
24.2.2.4.3. ISB.....	290
24.2.2.4.4. Vector table.....	290
24.2.2.4.5. Self-modifying code.....	290
24.2.2.4.6. Memory map switching.....	290
24.2.2.5. Memory endianness.....	290
24.2.2.5.1. Little-endian format.....	290
24.2.3. Exception model.....	291
24.2.3.1. Exception states.....	291
24.2.3.1.1. Inactive.....	291
24.2.3.1.2. Pending.....	291
24.2.3.1.3. Active.....	291
24.2.3.1.4. Active and pending.....	291
24.2.3.2. Exception types.....	291
24.2.3.2.1. Reset.....	292
24.2.3.2.2. NMI.....	292
24.2.3.2.3. HardFault.....	292
24.2.3.2.4. SVCall.....	292
24.2.3.2.5. PendSV.....	292
24.2.3.2.6. SysTick.....	292
24.2.3.2.7. Interrupt (IRQ).....	292
24.2.3.3. Exception handlers.....	293
24.2.3.3.1. Interrupt Service Routines (ISRs).....	293
24.2.3.3.2. Fault handler.....	293
24.2.3.3.3. System handlers.....	293
24.2.3.4. Vector table.....	293
24.2.3.5. Exception priorities.....	294
24.2.3.6. Exception entry and return.....	295
24.2.3.6.1. Preemption.....	295
24.2.3.6.2. Return.....	295
24.2.3.6.3. Tail-chaining.....	295
24.2.3.6.4. Late-arriving.....	295
24.2.3.6.5. Exception entry.....	296
24.2.3.6.6. Exception return.....	296
24.2.4. Fault handling.....	297
24.2.4.1. Lockup.....	297
24.2.5. Power management.....	298
24.2.5.1. Entering sleep mode.....	298
24.2.5.1.1. Wait for interrupt.....	298
24.2.5.1.2. Wait for event.....	298
24.2.5.1.3. Sleep-on-exit.....	299
24.2.5.2. Wakeup from sleep mode.....	299
24.2.5.2.1. Wakeup from WFI or sleep-on-exit.....	299
24.2.5.2.2. Wakeup from WFE.....	299
24.2.5.3. The Wakeup Interrupt Controller.....	299
24.2.5.4. The external event input.....	300
24.2.5.5. Power management programming hints.....	300
24.3. The Cortex-M0 Instruction Set.....	300
24.3.1. Instruction set summary.....	300
24.3.2. Intrinsic Functions.....	302
24.3.3. About the Instruction Descriptions.....	303



24.3.3.1. Operands.....	303
24.3.3.2. Restrictions when using PC or SP.....	304
24.3.3.3. Shift Operations.....	304
24.3.3.3.1. ASR.....	304
24.3.3.3.2. LSR.....	305
24.3.3.3.3. LSL.....	305
24.3.3.3.4. ROR.....	306
24.3.3.4. Address alignment.....	306
24.3.3.5. PC-relative expressions.....	306
24.3.3.6. Conditional execution.....	307
24.3.3.6.1. The condition flags.....	307
24.3.3.6.2. Condition code suffixes.....	308
24.3.4. Memory access instructions.....	308
24.3.4.1. ADR.....	309
24.3.4.1.1. Syntax.....	309
24.3.4.1.2. Operation.....	309
24.3.4.1.3. Restrictions.....	309
24.3.4.1.4. Condition flags.....	309
24.3.4.1.5. Examples.....	309
24.3.4.2. LDR and STR, immediate offset.....	310
24.3.4.2.1. Syntax.....	310
24.3.4.2.2. Operation.....	310
24.3.4.2.3. Restrictions.....	310
24.3.4.2.4. Condition flags.....	311
24.3.4.2.5. Examples.....	311
24.3.4.3. LDR and STR, register offset.....	311
24.3.4.3.1. Syntax.....	311
24.3.4.3.2. Operation.....	311
24.3.4.3.3. Restrictions.....	311
24.3.4.3.4. Condition flags.....	312
24.3.4.3.5. Examples.....	312
24.3.4.4. LDR, PC-relative.....	312
24.3.4.4.1. Syntax.....	312
24.3.4.4.2. Operation.....	312
24.3.4.4.3. Restrictions.....	312
24.3.4.4.4. Condition flags.....	312
24.3.4.4.5. Examples.....	313
24.3.4.5. LDM and STM.....	313
24.3.4.5.1. Syntax.....	313
24.3.4.5.2. Operation.....	313
24.3.4.5.3. Restrictions.....	314
24.3.4.5.4. Condition flags.....	314
24.3.4.5.5. Examples.....	314
24.3.4.5.6. Incorrect examples.....	314
24.3.4.6. PUSH and POP.....	314
24.3.4.6.1. Syntax.....	314
24.3.4.6.2. Operation.....	314
24.3.4.6.3. Restrictions.....	315
24.3.4.6.4. Condition flags.....	315
24.3.4.6.5. Examples.....	315
24.3.5. General data processing instructions.....	315
24.3.5.1. ADC, ADD, RSB, SBC, and SUB.....	316
24.3.5.1.1. Syntax.....	316
24.3.5.1.2. Operation.....	317

24.3.5.1.3. Restrictions.....	317
24.3.5.1.4. Examples.....	318
24.3.5.2. AND, ORR, EOR, and BIC.....	318
24.3.5.2.1. Syntax.....	319
24.3.5.2.2. Operation.....	319
24.3.5.2.3. Restrictions.....	319
24.3.5.2.4. Condition flags.....	319
24.3.5.2.5. Examples.....	319
24.3.5.3. ASR, LSL, LSR, and ROR.....	320
24.3.5.3.1. Syntax.....	320
24.3.5.3.2. Operation.....	320
24.3.5.3.3. Restrictions.....	321
24.3.5.3.4. Condition flags.....	321
24.3.5.3.5. Examples.....	321
24.3.5.4. CMP and CMN.....	321
24.3.5.4.1. Syntax.....	321
24.3.5.4.2. Operation.....	321
24.3.5.4.3. Restrictions.....	322
24.3.5.4.4. Condition flags.....	322
24.3.5.4.5. Examples.....	322
24.3.5.5. MOV and MVN.....	322
24.3.5.5.1. Syntax.....	322
24.3.5.5.2. Operation.....	323
24.3.5.5.3. Restrictions.....	323
24.3.5.5.4. Condition flags.....	323
24.3.5.5.5. Example.....	323
24.3.5.6. MULS.....	323
24.3.5.6.1. Syntax.....	324
24.3.5.6.2. Operation.....	324
24.3.5.6.3. Restrictions.....	324
24.3.5.6.4. Condition flags.....	324
24.3.5.6.5. Examples.....	324
24.3.5.7. REV, REV16, and REVSH.....	324
24.3.5.7.1. Syntax.....	324
24.3.5.7.2. Operation.....	325
24.3.5.7.3. Restrictions.....	325
24.3.5.7.4. Condition flags.....	325
24.3.5.7.5. Examples.....	325
24.3.5.8. SXT and UXT.....	325
24.3.5.8.1. Syntax.....	325
24.3.5.8.2. Operation.....	326
24.3.5.8.3. Restrictions.....	326
24.3.5.8.4. Condition flags.....	326
24.3.5.8.5. Examples.....	326
24.3.5.9. TST.....	326
24.3.5.9.1. Syntax.....	326
24.3.5.9.2. Operation.....	327
24.3.5.9.3. Restrictions.....	327
24.3.5.9.4. Condition flags.....	327
24.3.5.9.5. Examples.....	327
24.3.6. Branch and control instructions.....	327
24.3.6.1. B, BL, BX, and BLX.....	328
24.3.6.1.1. Syntax.....	328
24.3.6.1.2. Operation.....	328

24.3.6.1.3. Restrictions.....	328
24.3.7. Miscellaneous instructions.....	329
24.3.7.1. BKPT.....	330
24.3.7.1.1. Syntax.....	330
24.3.7.1.2. Operation.....	330
24.3.7.1.3. Restrictions.....	330
24.3.7.1.4. Condition flags.....	330
24.3.7.1.5. Examples.....	330
24.3.7.2. CPS.....	330
24.3.7.2.1. Syntax.....	330
24.3.7.2.2. Operation.....	331
24.3.7.2.3. Restrictions.....	331
24.3.7.2.4. Condition flags.....	331
24.3.7.2.5. Examples.....	331
24.3.7.3. DMB.....	331
24.3.7.3.1. Syntax.....	331
24.3.7.3.2. Operation.....	331
24.3.7.3.3. Restrictions.....	331
24.3.7.3.4. Condition flags.....	332
24.3.7.3.5. Examples.....	332
24.3.7.4. DSB.....	332
24.3.7.4.1. Syntax.....	332
24.3.7.4.2. Operation.....	332
24.3.7.4.3. Restrictions.....	332
24.3.7.4.4. Condition flags.....	332
24.3.7.4.5. Examples.....	332
24.3.7.5. ISB.....	332
24.3.7.5.1. Syntax.....	333
24.3.7.5.2. Operation.....	333
24.3.7.5.3. Restrictions.....	333
24.3.7.5.4. Condition flags.....	333
24.3.7.5.5. Examples.....	333
24.3.7.6. MRS.....	333
24.3.7.6.1. Syntax.....	333
24.3.7.6.2. Operation.....	333
24.3.7.6.3. Restrictions.....	334
24.3.7.6.4. Condition flags.....	334
24.3.7.6.5. Examples.....	334
24.3.7.7. MSR.....	334
24.3.7.7.1. Syntax.....	334
24.3.7.7.2. Operation.....	334
24.3.7.7.3. Restrictions.....	334
24.3.7.7.4. Condition flags.....	334
24.3.7.7.5. Examples.....	335
24.3.7.8. NOP.....	335
24.3.7.8.1. Syntax.....	335
24.3.7.8.2. Operation.....	335
24.3.7.8.3. Restrictions.....	335
24.3.7.8.4. Condition flags.....	335
24.3.7.8.5. Examples.....	335
24.3.7.9. SEV.....	335
24.3.7.9.1. Syntax.....	335
24.3.7.9.2. Operation.....	336
24.3.7.9.3. Restrictions.....	336

24.3.7.9.4. Condition flags.....	336
24.3.7.9.5. Examples.....	336
24.3.7.10. SVC.....	336
24.3.7.10.1. Syntax.....	336
24.3.7.10.2. Operation.....	336
24.3.7.10.3. Restrictions.....	336
24.3.7.10.4. Condition flags.....	337
24.3.7.10.5. Examples.....	337
24.3.7.11. WFE.....	337
24.3.7.11.1. Syntax.....	337
24.3.7.11.2. Operation.....	337
24.3.7.11.3. Restrictions.....	337
24.3.7.11.4. Condition flags.....	338
24.3.7.11.5. Examples.....	338
24.3.7.12. WFI.....	338
24.3.7.12.1. Syntax.....	338
24.3.7.12.2. Operation.....	338
24.3.7.12.3. Restrictions.....	338
24.3.7.12.4. Condition flags.....	338
24.3.7.12.5. Examples.....	338
24.4. Cortex-M0 Peripherals.....	339
24.4.1. About the Cortex-M0 peripherals.....	339
24.4.2. Nested Vectored Interrupt Controller.....	339
24.4.2.1. Accessing the Cortex-M0 NVIC registers using CMSIS.....	340
24.4.2.2. Interrupt Set-enable Register.....	340
24.4.2.3. Interrupt Clear-enable Register.....	341
24.4.2.4. Interrupt Set-pending Register.....	341
24.4.2.5. Interrupt Clear-pending Register.....	342
24.4.2.6. Interrupt Priority Registers.....	342
24.4.2.7. Level-sensitive and pulse interrupts.....	343
24.4.2.7.1. Hardware and software control of interrupts.....	344
24.4.2.8. NVIC usage hints and tips.....	344
24.4.2.8.1. NVIC programming hints.....	344
24.4.3. System Control Block.....	345
24.4.3.1. The CMSIS mapping of the Cortex-M0 SCB registers.....	345
24.4.3.2. CPUID Register.....	346
24.4.3.3. Interrupt Control and State Register.....	346
24.4.3.4. Application Interrupt and Reset Control Register.....	348
24.4.3.5. System Control Register.....	349
24.4.3.6. Configuration and Control Register.....	350
24.4.3.7. System Handler Priority Registers.....	350
24.4.3.7.1. System Handler Priority Register 2.....	351
24.4.3.7.2. System Handler Priority Register 3.....	351
24.4.3.8. SCB usage hints and tips.....	352
24.4.4. System timer, SysTick.....	352
24.4.4.1. SysTick Control and Status Register.....	352
24.4.4.2. SysTick Reload Value Register.....	353
24.4.4.2.1. Calculating the RELOAD value.....	353
24.4.4.3. SysTick Current Value Register.....	354
24.4.4.4. SysTick Calibration Value Register.....	354
24.4.4.5. SysTick usage hints and tips.....	355
25. Legal Information.....	356

## LIST OF TABLES

Table 2-1. Embedded FLASH Register Map.....	33
Table 2-2. ROM Register Map.....	34
Table 2-3. System Clock Control Register Map.....	34
Table 2-4. FLASH Memory Controller Register Map.....	34
Table 2-5. Watchdog Timer Register Map.....	35
Table 2-6. General Purpose Timer Register Map.....	35
Table 2-7. GPIO Port A Register Map.....	35
Table 2-8. GPIO Port B Register Map.....	36
Table 2-9. GPIO Port AB Register Map.....	37
Table 2-10. GPIO Port C Register Map.....	37
Table 2-11. GPIO Port D Register Map.....	37
Table 2-12. GPIO Port CD Register Map.....	38
Table 2-13. GPIO Port E Register Map.....	38
Table 2-14. Timer A Register Map.....	39
Table 2-15. Timer B Register Map.....	40
Table 2-16. Timer C Register Map.....	40
Table 2-17. Timer D Register Map.....	41
Table 2-18. EMUX Register Map.....	41
Table 2-19. ADC Register Map.....	41
Table 2-20. ADC Auto-Sampling Sequencer 0 Register Map.....	41
Table 2-21. ADC Auto-Sampling Sequencer 1 Register Map.....	42
Table 2-22. I2C Register Map.....	42
Table 2-23. UART Register Map.....	43
Table 2-24. SOC Bus Bridge Register Map.....	43
Table 2-25. SPI Register Map.....	44
Table 2-26. Multi-Mode Power Manager Register Map.....	44
Table 3-1. System Clock Control Register Map.....	45
Table 4-1. Watchdog Timer Register Map.....	50
Table 5-1. General Purpose Timer Register Map.....	54
Table 6-1. GPIO Port A Register Map.....	57
Table 7-1. GPIO Port B Register Map.....	67
Table 8-1. GPIO Port C Register Map.....	75
Table 9-1. GPIO Port D Register Map.....	83
Table 10-1. GPIO Port E Register Map.....	93
Table 11-1. Timer A Register Map.....	103
Table 11-2. Timer A Signal to Pin Mapping.....	120
Table 12-1. Timer B Register Map.....	121
Table 12-2. Timer B Signal to Pin Mapping.....	133
Table 13-1. Timer C Register Map.....	134
Table 13-2. Timer C Signal to Pin Mapping.....	144
Table 14-1. Timer D Register Map.....	145
Table 14-2. Timer D Signal to Pin Mapping.....	154
Table 15-1. FLASH Memory Controller Register Map.....	155
Table 16-1. Register Map – EMUX.....	162
Table 16-2. Register Map – ADC.....	162
Table 16-3. Register Map – ADC Auto Sequencer 0.....	162
Table 16-4. Register Map – ADC Auto Sequencer 1.....	163
Table 17-1. I2C Register Map.....	185
Table 18-1. UART Register Map.....	192
Table 19-1. SOC Bus Bridge Register Map.....	200
Table 20-1. SPI Register Map.....	206

Table 21-1. Multi-Mode Power Manager Register Map.....	217
Table 22-1. Configurable Analog Front End Register Map.....	221
Table 23-1. Application Specific Power Driver Register Map.....	269
Table 24-1. Summary of processor mode and stack use options.....	279
Table 24-2. Core register set summary.....	280
Table 24-3. Core register set summary.....	282
Table 24-4. APSR bit assignments.....	282
Table 24-5. IPSR bit assignments.....	282
Table 24-6. EPSR bit assignments.....	283
Table 24-7. PRIMASK register bit assignments.....	284
Table 24-8. CONTROL register bit assignments.....	285
Table 24-9. Memory Access Behavior.....	289
Table 24-10. Properties of the different exception types.....	292
Table 24-11. Execution return behavior.....	297
Table 24-12. Cortex-M0 instructions.....	301
Table 24-13. CMSIS intrinsic functions to generate some Cortex-M0 instructions.....	302
Table 24-14. CMSIS intrinsic functions to access special registers.....	303
Table 24-15. Condition code suffixes.....	308
Table 24-16. Memory access instructions.....	308
Table 24-17. Data processing instructions.....	315
Table 24-18. ADC, ADD, RSB, SBC, and SUB operand restrictions.....	317
Table 24-19. Branch and Control instructions.....	327
Table 24-20. Branch ranges.....	328
Table 24-21. Miscellaneous instructions.....	329
Table 24-22. Core peripheral register regions.....	339
Table 24-23. NVIC register summary.....	339
Table 24-24. CMSIS access NVIC functions.....	340
Table 24-25. ISER bit assignments.....	340
Table 24-26. ICER bit assignments.....	341
Table 24-27. ISPR bit assignments.....	342
Table 24-28. ICPR bit assignments.....	342
Table 24-29. IPR bit assignments.....	343
Table 24-30. CMSIS access NVIC functions.....	345
Table 24-31. Summary of the SCB register.....	345
Table 24-32. CPUID register bit assignments.....	346
Table 24-33. ICSR register bit assignments.....	347
Table 24-34. AIRCR register bit assignments.....	348
Table 24-35. SCR register bit assignments.....	349
Table 24-36. CCR register bit assignments.....	350
Table 24-37. System fault handler priority fields.....	351
Table 24-38. SHPR2 register bit assignments.....	351
Table 24-39. SHPR3 register bit assignments.....	351
Table 24-40. System timer register summary.....	352
Table 24-41. SYST_CSR register bit assignments.....	353
Table 24-42. SYST_RVR register bit assignments.....	353
Table 24-43. SYST_CVR register bit assignments.....	354
Table 24-44. SYST_CALIB register bit assignments.....	354



## LIST OF REGISTERS

Register 3-1. SCCTL (System Clock Control, 0x4000 0000).....	45
Register 3-2. PLLCTL (PLL Control, 0x4000 0004).....	46
Register 3-3. ROSCTL (Ring Oscillator Control, 0x4000 0008).....	46
Register 3-4. XTALCTL (Crystal Driver Control, 0x4000 000C).....	46
Table 3-5. PLL output frequency settings using 4MHz ROSC as input.....	49
Register 4-1. WDTCTL (Watchdog Timer Control, 0x4003 0000).....	50
Register 4-2. WDTCDV (Watchdog Timer Count-Down Value, 0x4003 0004).....	51
Register 4-3. WDTCTR (Watchdog Timer Counter, 0x4003 0008).....	51
Register 5-1. GPTCTL (Real Time Clock Control, 0x4004 0000).....	54
Register 5-2. GPTCDV (Real Time Clock Count-Down Value, 0x4004 0004).....	55
Register 5-3. GPTCTR (Real Time Clock Counter, 0x4004 0008).....	55
Register 6-1. GPIOAO (GPIO Port A Output, 0x4007 0000).....	57
Register 6-2. GPIOAOE (GPIO Port A Output Enable, 0x4007 0004).....	58
Register 6-3. GPIOADS (GPIO Port A Output Drive Strength, 0x4007 0008).....	58
Register 6-4. GPIOARUP (GPIO Port A Weak Pull Up, 0x4007 000C).....	59
Register 6-5. GPIOARDN (GPIO Port A Weak Pull Down, 0x4007 0010).....	60
Register 6-6. GPIOAI (GPIO Port A Input, 0x4007 0014).....	60
Register 6-7. GPIOAPSEL (GPIO Port A Peripheral Select, 0x4007 001C).....	61
Register 6-8. GPIOAINTP (GPIO Port A Interrupt Polarity, 0x4007 0020).....	62
Register 6-9. GPIOAINTEN (GPIO Port A Interrupt Enable, 0x4007 0024).....	62
Register 6-10. GPIOAINT (GPIO Port A Interrupt, 0x4007 0028).....	63
Register 6-11. GPIOAINTM (GPIO Port A Interrupt Mask, 0x4007 002C).....	64
Register 7-1. GPIOBO (GPIO Port B Output, 0x4007 0040).....	67
Register 7-2. GIOBOE (GPIO Port B Output Enable, 0x4007 0044).....	67
Register 7-3. GPIOBDS (GPIO Port B Output Drive Strength, 0x4007 0048).....	68
Register 7-4. GPIOBRUP (GPIO Port B Weak Pull Up, 0x4007 004C).....	68
Register 7-5. GPIOBRDN (GPIO Port B Weak Pull Down, 0x4007 0050).....	69
Register 7-6. GPIOBI (GPIO Port B Input, 0x4007 0054).....	69
Register 7-7. GPIOBPSEL (GPIO Port B Peripheral Select, 0x4007 005C).....	69
Register 7-8. GPGPIOBINTP (GPIO Port B Interrupt Polarity, 0x4007 0060).....	70
Register 7-9. GPIOBINTEN (GPIO Port B Interrupt Enable, 0x4007 0064).....	71
Register 7-10. GPIOBINT (GPIO Port B Interrupt, 0x4007 0068).....	71
Register 7-11. GPIOBINTM (GPIO Port B Interrupt Mask, 0x4007 006C).....	72
Register 8-1. GPIOCO (GPIO Port C Output, 0x4008 0000).....	75
Register 8-2. GPIOCOE (GPIO Port C Output Enable, 0x4008 0004).....	76
Register 8-3. GPIOCI (GPIO Port C Input, 0x4008 0018).....	76
Register 8-4. GPIOCIE (GPIO Port C Input Enable, 0x4008 0014).....	77
Register 8-5. GPIOCINTP (GPIO Port C Interrupt Polarity, 0x4008 0020).....	78
Register 8-6. GPIOCINTEN (GPIO Port C Interrupt Enable, 0x4008 0024).....	78
Register 8-7. GPIOCINT (GPIO Port C Interrupt, 0x4008 0028).....	79
Register 8-8. GPIOCINTM (GPIO Port C Interrupt Mask, 0x4008 002C).....	79
Register 9-1. GPIODO (GPIO Port D Output, 0x4008 0040).....	83
Register 9-2. GPIODOE (GPIO Port D Output Enable, 0x4008 0044).....	84
Register 9-3. GPIODDS (GPIO Port D Output Drive Strength, 0x4008 0048).....	84
Register 9-4. GPIODRUP (GPIO Port D Weak Pull Up, 0x4008 004C).....	85
Register 9-5. GPIODRDN (GPIO Port D Weak Pull Down, 0x4008 0050).....	86
Register 9-6. GPIODI (GPIO Port D Input, 0x4008 0054).....	86
Register 9-7. GPIODPSEL (GPIO Port D Peripheral Select, 0x4008 005C).....	87
Register 9-8. GPIODINTP (GPIO Port D Interrupt Polarity, 0x4008 0060).....	88
Register 9-9. GPIODINTEN (GPIO Port D Interrupt Enable, 0x4008 0064).....	88
Register 9-10. GPIODINT (GPIO Port D Interrupt, 0x4008 0068).....	89

Register 9-11. GPIODINTM (GPIO Port D Interrupt Mask, 0x4008 006C).....	89
Register 10-1. GPIOEO (GPIO Port E Output, 0x4009 0000).....	93
Register 10-2. GPIOEOE (GPIO Port E Output Enable, 0x4009 0004).....	94
Register 10-3. GPIOEDS (GPIO Port E Output Drive Strength, 0x4009 0008).....	94
Register 10-4. GPIOERUP (GPIO Port E Weak Pull Up, 0x4009 000C).....	95
Register 10-5. GPIOERDN (GPIO Port E Weak Pull Down, 0x4009 0010).....	96
Register 10-6. GPIOEI (GPIO Port E Input, 0x4009 0014).....	96
Register 10-7. GPIOEPSEL (GPIO Port E Peripheral Select, 0x4009 001C).....	97
Register 10-8. GPIOEINTP (GPIO Port E Interrupt Polarity, 0x4009 0020).....	98
Register 10-9. GPIOEINTEN (GPIO Port E Interrupt Enable, 0x4009 0024).....	98
Register 10-10. GPIOEINT (GPIO Port E Interrupt, 0x4009 0028).....	99
Register 10-11. GPIOEINTM (GPIO Port E Interrupt Mask, 0x4009 002C).....	99
Register 11-1. TACTL (Timer A Control, 0x400D 0000).....	104
Register 11-2. TAPRD (Timer A Period, 0x400D 0004).....	105
Register 11-3. TACTR (Timer A Counter, 0x400D 0008).....	105
Register 11-4. TACC0CTL (Timer A PWMA0 Capture and Compare Control, 0x400D 0040).....	105
Register 11-5. TACC0CTR (Timer A PWMA0 Capture and Compare Counter, 0x400D 0044).....	105
Register 11-6. TACC1CTL (Timer A PWMA1 Capture and Compare Control, 0x400D 0048).....	106
Register 11-7. TACC1CTR (Timer A PWMA1 Capture and Compare Counter, 0x400D 004C).....	106
Register 11-8. TACC2CTL (Timer A PWMA2 Capture and Compare Control, 0x400D 0050).....	106
Register 11-9. TACC2CTR (Timer A PWMA2 Capture and Compare Counter, 0x400D 0054).....	107
Register 11-10. TACC3CTL (Timer A PWMA3 Capture and Compare Control, 0x400D 0058).....	107
Register 11-11. TACC3CTR (Timer A PWMA3 Capture and Compare Counter, 0x400D 005C).....	107
Register 11-12. TACC4CTL (Timer A PWMA4 Capture and Compare Control, 0x400D 0060).....	107
Register 11-13. TACC4CTR (Timer A PWMA4 Capture and Compare Counter, 0x400D 0064).....	108
Register 11-14. TACC5CTL (Timer A PWMA5 Capture and Compare Control, 0x400D 0068).....	108
Register 11-15. TACC5CTR (Timer A PWMA5 Capture and Compare Counter, 0x400D 006C).....	108
Register 11-16. TACC6CTL (Timer A PWMA6 Capture and Compare Control, 0x400D 0070).....	109
Register 11-17. TACC6CTR (Timer A PWMA6 Capture and Compare Counter, 0x400D 0074).....	109
Register 11-18. TACC7CTL (Timer A PWMA7 Capture and Compare Control, 0x400D 0078).....	109
Register 11-19. TACC7CTR (Timer A PWMA7 Capture and Compare Counter, 0x400D 007C).....	110
Register 11-20. DTGA0CTL (Timer A Dead Time Generator 0 Control, 0x400D 00A0).....	110
Register 11-21. DTGA0LED (Timer A Dead Time Generator 0 Leading Edge Delay, 0x400D 00A4).....	110
Register 11-22. DTGA0TED (Timer A Dead Time Generator 0 Trailing Edge Delay, 0x400D 00A8).....	110
Register 11-23. DTGA1CTL (Timer A Dead Time Generator 1 Control, 0x400D 00B0).....	111
Register 11-24. DTGA1LED (Timer A Dead Time Generator 1 Leading Edge Delay, 0x400D 00B4).....	111
Register 11-25. DTGA1TED (Timer A Dead Time Generator 1 Trailing Edge Delay, 0x400D 00B8).....	111
Register 11-26. DTGA2CTL (Timer A Dead Time Generator 2 Control, 0x400D 00C0).....	111
Register 11-27. DTGA2LED (Timer A Dead Time Generator 2 Leading Edge Delay, 0x400D 00C4).....	112
Register 11-28. DTGA2TED (Timer A Dead Time Generator 2 Trailing Edge Delay, 0x400D 00C8).....	112
Register 11-29. DTGA3CTL (Timer A Dead Time Generator 3 Control, 0x400D 00D0).....	112
Register 11-30. DTGA3LED (Timer A Dead Time Generator 3 Leading Edge Delay, 0x400D 00D4).....	113
Register 11-31. DTGA3TED (Timer A Dead Time Generator 3 Trailing Edge Delay, 0x400D 00D8).....	113
Register 12-1. TBCTL (Timer B Control, 0x400E 0000).....	121
Register 12-2. TBPRD (Timer B Period, 0x400E 0004).....	122
Register 12-3. TBCTR (Timer B Counter, 0x400E 0008).....	122
Register 12-4. TBCC0CTL (Timer B PWMB0 Capture and Compare Control, 0x400E 0040).....	122
Register 12-5. TBCC0CTR (Timer B PWMB0 Capture and Compare Counter, 0x400E 0044).....	123
Register 12-6. TBCC1CTL (Timer B PWMB1 Capture and Compare Control, 0x400E 0048).....	123
Register 12-7. TBCC1CTR (Timer B PWMB1 Capture and Compare Counter, 0x400E 004C).....	123
Register 12-8. TBCC2CTL (Timer B PWMB2 Capture and Compare Control, 0x400E 0050).....	124
Register 12-9. TBCC2CTR (Timer B PWMB2 Capture and Compare Counter, 0x400E 0054).....	124
Register 12-10. TBCC3CTL (Timer B PWMB3 Capture and Compare Control, 0x400E 0058).....	124
Register 12-11. TBCC3CTR (Timer B PWMB3 Capture and Compare Counter, 0x400E 005C).....	125



Register 12-12. DTGB0CTL (Timer B Dead Time Generator 0 Control, 0x400E 00A0).....	125
Register 12-13. DTGB0LED (Timer B Dead Time Generator 0 Leading Edge Delay, 0x400E 00A4).....	125
Register 12-14. DTGB0TED (Timer B Dead Time Generator 0 Trailing Edge Delay, 0x400E 00A8).....	126
Register 13-1. TCCTL (Timer C Control, 0x400F 0000).....	134
Register 13-2. TCPRD (Timer C Period, 0x400F 0004).....	135
Register 13-3. TCCTR (Timer C Counter, 0x400F 0008).....	135
Register 13-4. TCCC0CTL (Timer C PWMC0 Capture and Compare Control, 0x400F 0040).....	135
Register 13-5. TCCC0CTR (Timer C PWMC0 Capture and Compare Counter, 0x400F 0044).....	136
Register 13-6. TCCC1CTL (Timer C PWMC1 Capture and Compare Control, 0x400F 0048).....	136
Register 13-7. TCCC1CTR (Timer C PWMC1 Capture and Compare Counter, 0x400F 004C).....	136
Register 13-8. DTGC0CTL (Timer C Dead Time Generator 0 Control, 0x400F 00A0).....	137
Register 13-9. DTGC0LED (Timer C Dead Time Generator 0 Leading Edge Delay, 0x400F 00A4).....	137
Register 13-10. DTGC0TED (Timer C Dead Time Generator 0 Trailing Edge Delay, 0x400F 00A8).....	137
Register 14-1. TDCTL (Timer D Control, 0x4010 0000).....	145
Register 14-2. TDPRD (Timer D Period, 0x4010 0004).....	146
Register 14-3. TDCTR (Timer D Counter, 0x4010 0008).....	146
Register 14-4. TDCC0CTL (Timer D PWMD0 Capture and Compare Control, 0x4010 0040).....	146
Register 14-5. TDCC0CTR (Timer D PWMD0 Capture and Compare Counter, 0x4010 0044).....	147
Register 14-6. TDCC1CTL (Timer D PWMD1 Capture and Compare Control, 0x4010 0048).....	147
Register 14-7. TDCC1CTR (Timer D PWMD1 Capture and Compare Counter, 0x4010 004C).....	147
Register 14-8. DTGD0CTL (Timer D Dead Time Generator 0 Control, 0x4010 00A0).....	148
Register 14-9. DTGD0LED (Timer D Dead Time Generator 0 Leading Edge Delay, 0x4010 00A4).....	148
Register 14-10. DTGD0TED (Timer D Dead Time Generator 0 Trailing Edge Delay, 0x4010 00A8).....	148
Register 15-1. FLASHLOCK (FLASH Lock, 0x4002 0000).....	155
Register 15-2. FLASHSTATUS (FLASH Status, 0x4002 0004).....	156
Register 15-3. FLASHPAGE (FLASH Page Selector, 0x4002 0008).....	156
Register 15-4. FLASHPERASE (FLASH Page Erase, 0x4002 0014).....	157
Register 15-5. SWDACCESS (SDW Access Status, 0x4002 0024).....	157
Register 15-6. FLASHWSTATE (FLASH Access Wait State, 0x4002 0028).....	157
Register 15-7. FLASHBWRITE (Buffered FLASH Write, 0x4002 002C).....	157
Register 15-8. FLASHBWDATA (Buffered FLASH Write Data, 0x4002 0030).....	158
Register 16-1. EMUXCTL (ADC external MUX control register 0x4015 0000).....	163
Register 16-2. EMUXDATA (EMUX data register 0x4015 0004).....	164
Register 16-3. ADCCTL (ADC control register 0x4015 0008).....	164
Register 16-4. ADCCR (ADC conversion result register 0x4015 000C).....	165
Register 16-5. ADCINT (ADC Interrupt register 0x4015 0010).....	165
Register 16-6. AS0CTL (Auto Sequencer 0 control register 0x4015 0040).....	166
Register 16-7. AS0S0 (Auto sequencer 0-sample 0 control 0x4015 0044).....	167
Register 16-8. AS0R0 ( Auto sequencer 0-sample 0 result register 0x4015 0048).....	167
Register 16-9. AS0S1 (Auto sequencer 0-sample 1 control 0x4015 004C).....	168
Register 16-10. AS0R1 ( Auto sequencer 0-sample 1 result register 0x4015 0050).....	168
Register 16-11. AS0S2 (Auto sequencer 0-sample 2 control 0x4015 0054).....	168
Register 16-12. AS0R2 ( Auto sequencer 0-sample 2 result register 0x4015 0058).....	169
Register 16-13. AS0S3 (Auto sequencer 0-sample 3 control 0x4015 005C).....	169
Register 16-14. AS0R3 ( Auto sequencer 0-sample 3 result register 0x4015 0060).....	169
Register 16-15. AS0S4 (Auto sequencer 0-sample 4 control 0x4015 0064).....	170
Register 16-16. AS0R4 ( Auto sequencer 0-sample 4 result register 0x4015 0068).....	170
Register 16-17. AS0S5 (Auto sequencer 0-sample 5 control 0x4015 006C).....	170
Register 16-18. AS0R5 ( Auto sequencer 0-sample 5 result register 0x4015 0070).....	171
Register 16-19. AS0S6 (Auto sequencer 0-sample 6 control 0x4015 0074).....	171
Register 16-20. AS0R6 ( Auto sequencer 0-sample 6 result register 0x4015 0078).....	171
Register 16-21. AS0S7 (Auto sequencer 0-sample 7 control 0x4015 007C).....	172
Register 16-22. AS0R7 ( Auto sequencer 0-sample 7 result register 0x4015 0080).....	172
Register 16-23. AS1CTL (Auto Sequencer 1 control register 0x4015 0100).....	172

Register 16-24. AS1S0 (Auto sequencer 1-sample 0 control 0x4015 0104).....	173
Register 16-25. AS1R0 ( Auto sequencer 1-sample 0 result register 0x4015 0108).....	174
Register 16-26. AS1S1 (Auto sequencer 1-sample 1 control 0x4015 010C).....	174
Register 16-27. AS1R1 ( Auto sequencer 1-sample 1 result register 0x4015 0110).....	174
Register 16-28. AS1S2 (Auto sequencer 1-sample 2 control 0x4015 0114).....	174
Register 16-29. AS1R2 ( Auto sequencer 1-sample 2 result register 0x4015 0118).....	175
Register 16-30. AS1S3 (Auto sequencer 1-sample 3 control 0x4015 011C).....	175
Register 16-31. AS1R3 ( Auto sequencer 1-sample 3 result register 0x4015 0120).....	176
Register 16-32. AS1S4 (Auto sequencer 1-sample 4 control 0x4015 0124).....	176
Register 16-33. AS1R4 ( Auto sequencer 1-sample 4 result register 0x4015 0128).....	176
Register 16-34. AS1S5 (Auto sequencer 1-sample 5 control 0x4015 012C).....	176
Register 16-35. AS1R5 ( Auto sequencer 1-sample 5 result register 0x4015 0130).....	177
Register 16-36. AS1S6 (Auto sequencer 1-sample 6 control 0x4015 0134).....	177
Register 16-37. AS1R6 ( Auto sequencer 1-sample 6 result register 0x4015 0138).....	178
Register 16-38. AS1S7 (Auto sequencer 1-sample 7 control 0x4015 013C).....	178
Register 16-39. AS1R7 ( Auto sequencer 1-sample 7 result register 0x4015 0140).....	178
Register 17-1. I2CCFG (I2C Configuration, 0x401B 0000).....	185
Register 17-2. I2CINT (I2C Interrupt, 0x401B 0004).....	185
Register 17-3. I2CINTEN (I2C Interrupt Enable, 0x401B 0008).....	187
Register 17-4. I2CMACTL (I2C Master Access Control, 0x401B 0030).....	187
Register 17-5. I2CMARXDATA (I2C Master Receive Data, 0x401B 0034).....	188
Register 17-6. I2CMATXDATA (I2C Master Transmit Data, 0x401B 0038).....	188
Register 17-7. I2CBAUD (I2C Baud Rate, 0x401B 0040).....	188
Register 17-8. I2CSLRXDATA (I2C Slave Receive Data, 0x401B 0070).....	188
Register 17-9. I2CSLTXDATA (I2C Slave Transmit Data, 0x401B 0074).....	189
Register 17-10. I2CSLADDR (I2C Slave Address, 0x401B 0074).....	189
Table 17-11. I2CBAUD settings for different HCLK.....	190
Register 18-1. UARTRTX (UART Receive/Transmit FIFO, 0x401D 0000).....	192
Register 18-2. UARTIEN (UART Interrupt Enable, 0x401D 0004).....	193
Register 18-3. UARTEI (UART Interrupt Identification, 0x401D 0008).....	193
Register 18-4. UARTLC (UART Line Control, 0x401D 000C).....	194
Register 18-5. UARTLS (UART Line Status, 0x401D 0014).....	194
Register 18-6. UARTECTL (FIFO Control, 0x401D 0020).....	195
Register 18-7. UARTEIEN (UART Interrupt Enable, 0x401D 0024).....	195
Register 18-8. UARTELD_L (UART Divisor Latch Low Byte, 0x401D 0028).....	195
Register 18-9. UARTELD_H (UART Divisor Latch High Byte, 0x401D 002C).....	196
Register 18-10. UARTEFD_F (UART Fractional Divisor Value, 0x401D 0038).....	196
Register 18-11. UARTEFSTAT (UART FIFO Status, 0x401D 0040).....	196
Register 18-12. UART Divisor Settings for 50 MHz HCLK.....	198
Register 19-1. SOCBCTL (SOC Bus Bridge Control, 0x4020 0000).....	200
Register 19-2. SOCBCFG (SOC Bus Bridge Configuration, 0x4020 0004).....	200
Register 19-3. SOCBCLK (SOC Bus Bridge Clock Divider, 0x4020 0008).....	201
Register 19-4. SOCBSTAT (SOC Bus Bridge Status, 0x4020 0014).....	201
Register 19-5. SOCBD (SOC Bus Bridge Data, 0x4020 001C).....	202
Register 19-6. SOCBINT_EN (SOC Bus Bridge Interrupt Enable, 0x4020 0020).....	203
Register 20-1. SPICTL (SPI Control, 0x4021 0000).....	206
Register 20-2. SPICFG (SPI Configuration, 0x4021 0004).....	207
Register 20-3. SPICLKDIV (SPI Clock Divider, 0x4021 0008).....	208
Register 20-4. SPISTAT (SPI Status, 0x4021 0014).....	208
Register 20-5. SPICSSSTR (SPI Chip Select Steering, 0x4021 0018).....	210
Register 20-6. SPID (SPI Data, 0x4021 001C).....	210
Register 20-7. SPIINT_EN (SPI Interrupt Enable, 0x4021 0020).....	211
Register 21-1. SOC.SYSSTAT (Analog System Status, SOC 0x00).....	217
Register 21-2. SOC.PWRCTL (Power Control, SOC 0x10).....	218

Register 21-3. SOC.PWRSTAT (Power Manager Status, Persistent In Hibernate mode, SOC 0x11).....	218
Register 21-4. SOC.PWRSTATCFG (Power Status Configuration, SOC 0x12).....	219
Register 21-5. SOC.SCFG (Power Switcher Configuration, SOC 0x14).....	219
Register 21-6. SOC.PWRCFG (Power Status Configuration, SOC 0x15).....	220
Register 22-1. SOC.AIO0CFG (AIO0 Configuration, SOC 0x20).....	221
Register 22-2. SOC.AIO1CFG (AIO1 Configuration, SOC 0x21).....	222
Register 22-3. SOC.AIO2CFG (AIO2 Configuration, SOC 0x22).....	223
Register 22-4. SOC.AIO3CFG (AIO3 Configuration, SOC 0x23).....	224
Register 22-5. SOC.AIO4CFG (AIO4 Configuration, SOC 0x24).....	225
Register 22-6. SOC.AIO5CFG (AIO5 Configuration, SOC 0x25).....	226
Register 22-7. SOC.AIO6CFG (AIO6 Configuration, SOC 0x26).....	227
Register 22-8. SOC.AIO7CFG (AIO7 Configuration, SOC 0x27).....	229
Register 22-9. SOC.AIO8CFG (AIO8 Configuration, SOC 0x28).....	230
Register 22-10. SOC.AIO9CFG (AIO9 Configuration, SOC 0x29).....	232
Register 22-11. SOC.SMCFG (Signal Manager Configuration, SOC 0x2A).....	233
Register 22-12. SOC.HPDAC (HPDAC Setting, SOC 0x2B).....	234
Register 22-13. SOC.LPDAC0 (LPDAC Setting [9:2], SOC 0x2C).....	234
Register 22-14. SOC.LPDAC1 (LPDAC Setting [1:0], SOC 0x2C).....	234
Register 22-15. SOC.ADCSCANCFG (ADCSCAN Configuration, SOC 0x2E).....	234
Register 22-16. SOC.ADMUX (AD Mux Selector, SOC 0x2F).....	235
Register 22-17. SOC.PROTINTEN (Protection Interrupt Enable, SOC 0x30).....	235
Register 22-18. SOC.PROTINT (Protection Interrupt, SOC 0x31).....	236
Register 22-19. SOC.AIOOUT0 (AIO Digital Output, SOC 0x32).....	236
Register 22-20. SOC.AIOOUT0 (AIO Digital Output, SOC 0x33).....	237
Register 22-21. SOC.AIOIN0 (AIO Digital Input, SOC 0x34).....	237
Register 22-22. SOC.AIOIN1 (AIO Digital Input 1, SOC 0x35).....	238
Register 22-23. SOC.AIOINTEN (AIO6,7,8,9 Interrupt Enable, SOC 0x36).....	238
Register 22-24. SOC.AIOINT (AIO6,7,8,9 Interrupt, SOC 0x37).....	238
Register 22-25. SOC.SMCTL (Signal Manager Control SOC 0x38).....	239
Register 22-26. SOC.AIOIN0 (AIO Digital Input, SOC 0x34).....	266
Register 23-1. SOC.PRCTL (PR1, PR2 Control, SOC 0x61).....	269
Register 23-2. SOC.OMOUT (OM0/2/4 Output, SOC 0x64).....	269
Register 23-3. SOC.DRVCTL (Application Specific Power Driver Control, SOC 0x66).....	270
Register 23-4. SOC.PROTCTL (PROT Control, SOC 0x67).....	270

## LIST OF FIGURES

Figure 2-1. Memory Map.....	32
Figure 3-1. System Clock Control.....	47
Figure 4-1. WDT.....	52
Figure 5-1. GPT.....	56
Figure 6-1. GPIO Port A.....	65
Figure 7-1. GPIO Port B.....	73
Figure 8-1. GPIO Port C.....	81
Figure 9-1. GPIO Port D.....	91
Figure 10-1. GPIO Port E.....	101
Figure 11-1. Timer A.....	114
Figure 11-2. PWMA[x] and PWMA[x+4] Example Using Timer A Up Mode and Up/Down Mode.....	117
Figure 11-3. CA[x] and CA[x+4] Capture Example.....	117
Figure 11-4. DTGAX Bypass Example.....	118
Figure 11-5. DTGAX Bypass and Inverting LS Example.....	119
Figure 11-6. DTGAX LED and TED Example.....	119
Figure 11-7. DTGAX LED and TED with On Time Preservation Example.....	120
Figure 12-1. Timer B.....	127
Figure 12-2. PWMB0 and PWMB1 Example Using Timer B Up Mode and Up/Down Mode.....	130
Figure 12-3. CB0 and CB1 Capture Example.....	130
Figure 12-4. DTGB0 Bypass Example.....	131
Figure 12-5. DTGB0 Bypass and Inverting LS Example.....	132
Figure 12-6. DTGB0 LED and TED Example.....	132
Figure 12-7. DTGB0 LED and TED with On Time Preservation Example.....	133
Figure 13-1. Timer C.....	138
Figure 13-2. PWMC0 and PWMC1 Example Using Timer C Up Mode and Up/Down Mode.....	141
Figure 13-3. CC0 and CC1 Capture Example.....	141
Figure 13-4. DTGC0 Bypass Example.....	142
Figure 13-5. DTGC0 Bypass and Inverting LS Example.....	143
Figure 13-6. DTGC0 LED and TED Example.....	143
Figure 13-7. DTGC0 LED and TED with On Time Preservation Example.....	144
Figure 14-1. Timer D.....	149
Figure 14-2. PWMD0 and PWMD1 Example Using Timer D Up Mode and Up/Down Mode.....	151
Figure 14-3. CD0 and CD1 Capture Example.....	152
Figure 14-4. DTGD0 Bypass Example.....	153
Figure 14-5. DTGD0 Bypass and Inverting LS Example.....	153
Figure 14-6. DTGD0 LED and TED Example.....	154
Figure 14-7. DTGD0 LED and TED with On Time Preservation Example.....	154
Figure 15-1. FLASH Memory Controller.....	159
Figure 16-1. ADC, EMUX, ASC0, ASC1.....	179
Figure 16-2. ADC Conversion (Single Shot).....	180
Figure 16-3. ADC Conversion (Repeat Mode).....	180
Figure 16-4. ASCx, ADCCTL.ADCMODE = 001b, 010b, 100b, 101b.....	181
Figure 16-5. ASCx, ADCCTL.ADCMODE = 011b, 110b.....	181
Figure 16-6. ASCx, ADCCTL.ADCMODE = 111b.....	182
Figure 16-7. ASxSy Sample with ASxSy.EMUXS = 00b and ASxSy.DELAY = 11b.....	182
Figure 16-8. ASxSy Sample with ASxSy.EMUXS = 01b and ASxSy.DELAY = 11b.....	183
Figure 16-9. ASxSy Sample with ASxSy.EMUXS = 10b and ASxSy.DELAY = 11b.....	183
Figure 16-10. ASCx, 8 samples, No Collision.....	183
Figure 16-11. ASCx 8 samples, Collision.....	184
Figure 16-12. ASC0 8 samples, ASC1 4 samples, Collision.....	184
Figure 17-1. I2C.....	190

Figure 18-1. UART.....	197
Figure 19-1. SOC Bridge.....	204
Figure 19-2. Single Read from SOC Bridge.....	205
Figure 19-3. Single Write to SOC Bridge.....	205
Figure 20-1. SPI.....	212
Figure 20-2. SPI clock polarity and phase.....	213
Figure 20-3. SPIMOSI early transmit in master mode.....	214
Figure 20-4. SPIMISO early transmit in slave mode.....	214
Figure 20-5. SPICSt.....	215
Figure 22-1. Configurable Analog Front End.....	240
Figure 22-2. AIO1, AIO0.....	241
Figure 22-3. AIO3, AIO2.....	244
Figure 22-4. AIO5, AIO4.....	247
Figure 22-5. AIO6.....	250
Figure 22-6. AIO7.....	254
Figure 22-7. AIO8.....	258
Figure 22-8. AIO9.....	262
Figure 22-9. EMUX.....	266
Figure 22-10. EMUX Timing.....	267
Figure 23-1. Application Specific Power Driver.....	271
Figure 23-2. OM0.....	272
Figure 23-3. OM2.....	273
Figure 23-4. OM4.....	274
Figure 23-5. ENHS1 Protection.....	275
Figure 23-6. ENHS2 Protection.....	276
Figure 24-1. Cortex-M0 implementation.....	277
Figure 24-2. Core Registers.....	280
Figure 24-3. PSR.....	281
Figure 24-4. PRIMASK.....	284
Figure 1-6. CONTROL.....	284
Figure 24-5. CONTROL.....	284
Figure 24-6. Memory Map.....	287
Figure 24-7. Memory Ordering Restrictions.....	288
Figure 24-8. Little Endian Format.....	291
Figure 24-9. Vector Table.....	294
Figure 24-10. Exception Entry Stack Contents.....	296
Figure 24-11. ASR #3.....	304
Figure 24-12. LSR #3.....	305
Figure 24-13. LSL #3.....	306
Figure 24-14. ROR #3.....	306
Figure 24-15. ISER.....	340
Figure 24-16. ICER.....	341
Figure 24-17. ISPR.....	341
Figure 24-18. ICPR.....	342
Figure 24-19. IPR.....	343
Figure 24-20. CPUID.....	346
Figure 24-21. ICSR.....	347
Figure 24-22. AIRCR.....	348
Figure 24-23. SCR.....	349
Figure 24-24. CCR.....	350
Figure 24-25. SHPR2.....	351
Figure 24-26. SHPR3.....	351
Figure 24-27. SYST_CSR.....	353
Figure 24-28. SYST_RVR.....	353

Figure 24-29. SYST\_CVR..... 354  
Figure 24-30. SYST\_CALIB..... 354



## 1. STYLES AND FORMATTING CONVENTIONS

### 1.1. Overview

This chapter describes formatting and styles used through the document.

### 1.2. Number Representation

Numbers in a base other than decimal have a prefix or postfix as indicator. All numbers use little endian formatting, most significant bit/digit is to the left. Digits for binary and hexadecimal representation are grouped with a single space every four digits to improve readability. Binary numbers use “b” as postfix, hexadecimal numbers use “0x” as prefix.

For example 1011b binary = 0xB hexadecimal = 11 decimal.

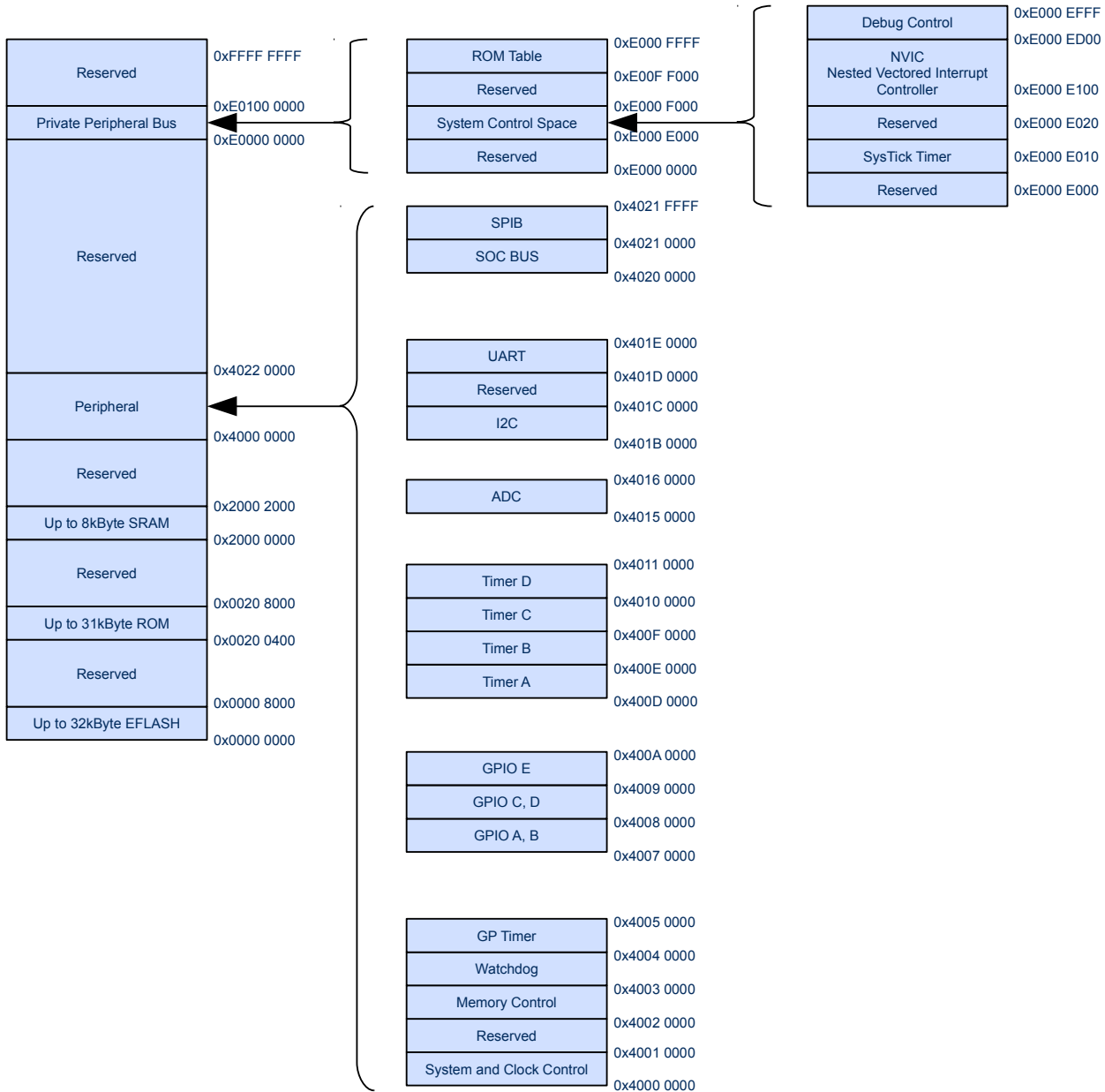
### 1.3. Formatting Styles

TYPE	EXAMPLE	DESCRIPTION
Register Name	<b>RTCCTL</b>	Register names use capital letter and bold formatting.
Register Bit(s)	<b>RTCCTL.RTCCLKDIV</b>	Register bits are always represented with the register name separated with a period
Function selected by Register bit(s)	<b>[RTCCTL.RTCCLKDIV]</b>	Within text blocks, functions selected with a register bit setting are set in brackets. For example <b>[RTCCTL.RTCCLKDIV]</b> means divider settings /2 to /65536.
Pin Function	XIN	Pin functions use capital letters
Formulas	CLK = FCLK / DIV	Formulas use Teletype font.
Links	<a href="#">Number Representation</a>	Clickable Links are underlined and blue
CPU Mnemonic	MRS	CPU Mnemonic use Teletype font.
Operands	{ <i>Rd</i> , } <i>Rn</i> , <i>Rm</i>	Operands use Italic
Code examples	B loopA	Code examples use Teletype font.

## 2. MEMORY AND REGISTER MAP

### 2.1. Memory Map

Figure 2-1. Memory Map





## 2.2. Register Map

**Table 2-1. Embedded FLASH Register Map**

ADDRESS	NAME	DESCRIPTION
<b>Embedded FLASH</b>		
0x0000 0000 – 0x0000 03FF	<b>EFLASHP0</b>	EFLASH page 0
0x0000 0400 – 0x0000 07FF	<b>EFLASHP1</b>	EFLASH page 1
0x0000 0800 – 0x0000 0BFF	<b>EFLASHP2</b>	EFLASH page 2
0x0000 0C00 – 0x0000 0FFF	<b>EFLASHP3</b>	EFLASH page 3
0x0000 1000 – 0x0000 13FF	<b>EFLASHP4</b>	EFLASH page 4
0x0000 1400 – 0x0000 17FF	<b>EFLASHP5</b>	EFLASH page 5
0x0000 1800 – 0x0000 1BFF	<b>EFLASHP6</b>	EFLASH page 6
0x0000 1C00 – 0x0000 1FFF	<b>EFLASHP7</b>	EFLASH page 7
0x0000 2000 – 0x0000 23FF	<b>EFLASHP8</b>	EFLASH page 8
0x0000 2400 – 0x0000 27FF	<b>EFLASHP9</b>	EFLASH page 9
0x0000 2800 – 0x0000 2BFF	<b>EFLASHP10</b>	EFLASH page 10
0x0000 2C00 – 0x0000 2FFF	<b>EFLASHP11</b>	EFLASH page 11
0x0000 3000 – 0x0000 33FF	<b>EFLASHP12</b>	EFLASH page 12
0x0000 3400 – 0x0000 37FF	<b>EFLASHP13</b>	EFLASH page 13
0x0000 3800 – 0x0000 3BFF	<b>EFLASHP14</b>	EFLASH page 14
0x0000 3C00 – 0x0000 3FFF	<b>EFLASHP15</b>	EFLASH page 15
0x0000 4000 – 0x0000 43FF	<b>EFLASHP16</b>	EFLASH page 16
0x0000 4400 – 0x0000 47FF	<b>EFLASHP17</b>	EFLASH page 17
0x0000 4800 – 0x0000 4BFF	<b>EFLASHP18</b>	EFLASH page 18
0x0000 4C00 – 0x0000 4FFF	<b>EFLASHP19</b>	EFLASH page 19
0x0000 5000 – 0x0000 53FF	<b>EFLASHP20</b>	EFLASH page 20
0x0000 5400 – 0x0000 57FF	<b>EFLASHP21</b>	EFLASH page 21
0x0000 5800 – 0x0000 5BFF	<b>EFLASHP22</b>	EFLASH page 22
0x0000 5C00 – 0x0000 5FFF	<b>EFLASHP23</b>	EFLASH page 23
0x0000 6000 – 0x0000 63FF	<b>EFLASHP24</b>	EFLASH page 24
0x0000 6400 – 0x0000 67FF	<b>EFLASHP25</b>	EFLASH page 25
0x0000 6800 – 0x0000 6BFF	<b>EFLASHP26</b>	EFLASH page 26
0x0000 6C00 – 0x0000 6FFF	<b>EFLASHP27</b>	EFLASH page 27
0x0000 7000 – 0x0000 73FF	<b>EFLASHP28</b>	EFLASH page 28
0x0000 7400 – 0x0000 77FF	<b>EFLASHP29</b>	EFLASH page 29
0x0000 7800 – 0x0000 7BFF	<b>EFLASHP30</b>	EFLASH page 30
0x0000 7C00 – 0x0000 7FFF	<b>EFLASHP31</b>	EFLASH page 31

**Table 2-2. ROM Register Map**

ADDRESS	NAME	DESCRIPTION
<b>INFO ROM</b>		
0x0010 0000 – 0x0010 000F	<b>Reserved</b>	Reserved
0x0010 0010	<b>ROSC11</b>	ROSC frequency in Hz at 11b setting
0x0010 0014	<b>ROSC10</b>	ROSC frequency in Hz at 10b setting
0x0010 0018	<b>ROSC01</b>	ROSC frequency in Hz at 01b setting
0x0010 001C	<b>ROSC00</b>	ROSC frequency in Hz at 00b setting
0x0010 0020	<b>VREF</b>	2.5V Voltage reference in mV
0x0010 0022	<b>ADCGAIN</b>	ADC gain in mV
0x0010 0024	<b>ADCOFF</b>	AD offset in mV
0x0010 0026	<b>Reserved</b>	Reserved
0x0010 0028	<b>FTTEMP</b>	Test temperature for internal temp sensor in °C
0x0010 002A	<b>TEMPS</b>	Internal temp sensor reading at FTTEMP temperature in AD count
0x0010 002C	<b>CLKREF</b>	4MHz CLKREF frequency in Hz
0x0010 0030 – 0x0010 003B	<b>Reserved</b>	Reserved
0x0010 003C	<b>PKGDC</b>	Package topside date code
0x0010 0040	<b>DIEREV</b>	Die revision
0x0010 0044	<b>PACIDR</b>	PAC part number and rev
0x0010 0048 – 0x0010 004E	<b>Reserved</b>	Reserved
0x0010 004F	<b>CHKSUM</b>	8-bit CRC checksum of address 0x0010 0000 to 0x0010 004E
0x0010 0050 – 0x0010 00FF	<b>Reserved</b>	Reserved
<b>DATA ROM</b>		
0x0020 0400 – 0x0020 7FFF	<b>ROM</b>	ROM Area

**Table 2-3. System Clock Control Register Map**

ADDRESS	NAME	DESCRIPTION
<b>System Clock Control</b>		
0x4000 0000	<b>SCCTL</b>	System clock control
0x4000 0004	<b>PLLCTL</b>	PLL control
0x4000 0008	<b>ROSCCTL</b>	Ring oscillator control
0x4000 000C	<b>XTALCTL</b>	Crystal driver control

**Table 2-4. FLASH Memory Controller Register Map**

ADDRESS	NAME	DESCRIPTION
<b>FLASH Memory Controller</b>		
0x4002 0000	<b>FLASHLOCK</b>	FLASH lock
0x4002 0004	<b>FLASHSTATUS</b>	FLASH status
0x4002 0008	<b>FLASHPAGE</b>	FLASH page selection
0x4002 000C	<b>Reserved</b>	Reserved
0x4002 0010	<b>Reserved</b>	Reserved
0x4002 0014	<b>FLASHPERASE</b>	FLASH page erase
0x4002 0018	<b>Reserved</b>	Reserved
0x4002 001C	<b>Reserved</b>	Reserved
0x4002 0020	<b>Reserved</b>	Reserved
0x4002 0024	<b>SWDACCESS</b>	SWD access control
0x4002 0028	<b>FLASHWSTATE</b>	FLASH wait state control
0x4002 002C	<b>FLASHBWRITE</b>	FLASH buffered write enable
0x4002 0030	<b>FLASHBWDATA</b>	FLASH buffered write data and address

**Table 2-5. Watchdog Timer Register Map**

ADDRESS	NAME	DESCRIPTION
<b>Watchdog Timer</b>		
0x4003 0000	<b>WDTCTL</b>	Watchdog timer control
0x4003 0004	<b>WDTCDV</b>	Watchdog timer count-down value
0x4003 0008	<b>WDTCTR</b>	Watchdog timer counter

**Table 2-6. General Purpose Timer Register Map**

ADDRESS	NAME	DESCRIPTION
<b>Real Time Clock</b>		
0x4004 0000	<b>GPTCTL</b>	General purpose timer control
0x4004 0004	<b>GPTCDV</b>	General purpose timer count-down value
0x4004 0008	<b>GPTCTR</b>	General purpose timer counter

**Table 2-7. GPIO Port A Register Map**

ADDRESS	NAME	DESCRIPTION
<b>GPIO Port A</b>		
0x4007 0000	<b>GPIOAO</b>	GPIO Port A output
0x4007 0004	<b>GPIOAOE</b>	GPIO Port A output enable
0x4007 0008	<b>GPIOADS</b>	GPIO Port A output drive strength
0x4007 000C	<b>GPIOAORUP</b>	GPIO Port A output weak pull up
0x4007 0010	<b>GPIOAORDN</b>	GPIO Port A output weak pull down
0x4007 0014	<b>GPIOAI</b>	GPIO Port A input
0x4007 0018	<b>Reserved</b>	Reserved
0x4007 001C	<b>GPIOAPSEL</b>	GPIO Port A peripheral select
0x4007 0020	<b>GPIOAINTP</b>	GPIO Port A interrupt polarity select
0x4007 0024	<b>GPIOAINTEN</b>	GPIO Port A interrupt enable select
0x4007 0028	<b>GPIOAINT</b>	GPIO Port A interrupt flag
0x4007 002C	<b>GPIOAINTM</b>	GPIO Port A interrupt mask

**Table 2-8. GPIO Port B Register Map**

ADDRESS	NAME	DESCRIPTION
<b>GPIO Port B</b>		
0x4007 0040	<b>GPIOBO</b>	GPIO Port B output
0x4007 0044	<b>GPIOBOE</b>	GPIO Port B output enable
0x4007 0048	<b>GPIOBODS</b>	GPIO Port B output drive strength
0x4007 004C	<b>GPIOBORUP</b>	GPIO Port B output weak pull up
0x4007 0050	<b>GPIOBORDN</b>	GPIO Port B output weak pull down
0x4007 0054	<b>GPIOBI</b>	GPIO Port B input
0x4007 0058	<b>Reserved</b>	Reserved
0x4007 005C	<b>GPIOBPSEL</b>	GPIO Port B peripheral select
0x4007 0060	<b>GPIOBINTP</b>	GPIO Port B interrupt polarity select
0x4007 0064	<b>GPIOBINTEN</b>	GPIO Port B interrupt enable select
0x4007 0068	<b>GPIOBINT</b>	GPIO Port B interrupt flag
0x4007 006C	<b>GPIOBINTM</b>	GPIO Port B interrupt mask

**Table 2-9. GPIO Port AB Register Map**

ADDRESS	NAME	DESCRIPTION
<b>GPIO Port AB</b>		
0x4007 0080	<b>GPIOABO</b>	GPIO Port AB output
0x4007 0084	<b>GPIOABOE</b>	GPIO Port AB output enable
0x4007 0088	<b>GPIOABODS</b>	GPIO Port AB output drive strength
0x4007 008C	<b>GPIOABORUP</b>	GPIO Port AB output weak pull up
0x4007 0090	<b>GPIOABORDN</b>	GPIO Port AB output weak pull down
0x4007 0094	<b>GPIOABI</b>	GPIO Port AB input
0x4007 0098	<b>Reserved</b>	Reserved
0x4007 009C	<b>GPIOABPSEL</b>	GPIO Port AB peripheral select
0x4007 00A0	<b>GPIOABINTP</b>	GPIO Port AB interrupt polarity select
0x4007 00A4	<b>GPIOABINTEN</b>	GPIO Port AB interrupt enable select
0x4007 00A8	<b>GPIOABINT</b>	GPIO Port AB interrupt flag
0x4007 00AC	<b>GPIOABINTM</b>	GPIO Port AB interrupt mask

**Table 2-10. GPIO Port C Register Map**

ADDRESS	NAME	DESCRIPTION
<b>GPIO Port C</b>		
0x4008 0000	<b>GPIOCO</b>	GPIO Port C output
0x4008 0004	<b>GPIOCOE</b>	GPIO Port C output enable
0x4008 0008	<b>Reserved</b>	Reserved
0x4008 000C	<b>Reserved</b>	Reserved
0x4008 0010	<b>Reserved</b>	Reserved
0x4008 0014	<b>GPIOCI</b>	GPIO Port C input
0x4008 0018	<b>GPIOCIE</b>	GPIO Port C input enable
0x4008 001C	<b>Reserved</b>	Reserved
0x4008 0020	<b>GPIOCINTP</b>	GPIO Port C interrupt polarity select
0x4008 0024	<b>GPIOCINTEN</b>	GPIO Port C interrupt enable select
0x4008 0028	<b>GPIOCINT</b>	GPIO Port C interrupt flag
0x4008 002C	<b>GPIOCINTM</b>	GPIO Port C interrupt mask

**Table 2-11. GPIO Port D Register Map**

ADDRESS	NAME	DESCRIPTION
<b>GPIO Port D</b>		
0x4008 0040	<b>GIODO</b>	GPIO Port D output
0x4008 0044	<b>GIODOE</b>	GPIO Port D output enable
0x4008 0048	<b>GIODODS</b>	GPIO Port D output drive strength
0x4008 004C	<b>GIODORUP</b>	GPIO Port D output weak pull up

ADDRESS	NAME	DESCRIPTION
0x4008 0050	<b>GPIODORDN</b>	GPIO Port D output weak pull down
0x4008 0054	<b>GPIODI</b>	GPIO Port D input
0x4008 0058	<b>Reserved</b>	Reserved
0x4008 005C	<b>GPIODPSEL</b>	GPIO Port D peripheral select
0x4008 0060	<b>GPIODINTP</b>	GPIO Port D interrupt polarity select
0x4008 0064	<b>GPIODINTEN</b>	GPIO Port D interrupt enable select
0x4008 0068	<b>GPIODINT</b>	GPIO Port D interrupt flag
0x4008 006C	<b>GPIODINTM</b>	GPIO Port D interrupt mask

**Table 2-12. GPIO Port CD Register Map**

ADDRESS	NAME	DESCRIPTION
<b>GPIO Port CD</b>		
0x4008 0080	<b>GPIOCDO</b>	GPIO Port CD output
0x4008 0084	<b>GPIOCDOE</b>	GPIO Port CD output enable
0x4008 0088	<b>Reserved</b>	Reserved
0x4008 008C	<b>Reserved</b>	Reserved
0x4008 0090	<b>Reserved</b>	Reserved
0x4008 0094	<b>GPIOCDI</b>	GPIO Port CD input
0x4008 0098	<b>Reserved</b>	Reserved
0x4008 009C	<b>GPIOCDPSEL</b>	GPIO Port CD peripheral select
0x4008 00A0	<b>GPIOCDINTP</b>	GPIO Port CD interrupt polarity select
0x4008 00A4	<b>GPIOCDINTEN</b>	GPIO Port CD interrupt enable select
0x4008 00A8	<b>GPIOCDINT</b>	GPIO Port CD interrupt flag
0x4008 00AC	<b>GPIOCDINTM</b>	GPIO Port CD interrupt mask

**Table 2-13. GPIO Port E Register Map**

ADDRESS	NAME	DESCRIPTION
<b>GPIO Port E</b>		
0x4009 0000	<b>GPIOEO</b>	GPIO Port E output
0x4009 0004	<b>GPIOEOE</b>	GPIO Port E output enable
0x4009 0008	<b>GPIOEODS</b>	GPIO Port E output drive strength
0x4009 000C	<b>GPIOEORUP</b>	GPIO Port E output weak pull up
0x4009 0010	<b>GPIOEORDN</b>	GPIO Port E output weak pull down
0x4009 0014	<b>GPIOEI</b>	GPIO Port E input
0x4009 0018	<b>Reserved</b>	Reserved
0x4009 001C	<b>GPIOEPSSEL</b>	GPIO Port E peripheral select
0x4009 0020	<b>GPIOEINTP</b>	GPIO Port E interrupt polarity select
0x4009 0024	<b>GPIOEINTEN</b>	GPIO Port E interrupt enable select
0x4009 0028	<b>GPIOEINT</b>	GPIO Port E interrupt flag

ADDRESS	NAME	DESCRIPTION
0x4009 002C	<b>GPIOEINTM</b>	GPIO Port E interrupt mask

**Table 2-14. Timer A Register Map**

ADDRESS	NAME	DESCRIPTION
<b>Timer A</b>		
0x400D 0000	<b>TACTL</b>	Timer A control
0x400D 0004	<b>TAPRD</b>	Timer A period
0x400D 0008	<b>TACTR</b>	Timer A counter
<b>Timer A PWMA Capture and Compare</b>		
0x400D 0040	<b>TACCTL0</b>	Timer A capture and compare 0 control
0x400D 0044	<b>TACTR0</b>	Timer A counter 0
0x400D 0048	<b>TACCTL1</b>	Timer A capture and compare 1 control
0x400D 004C	<b>TACTR1</b>	Timer A counter 1
0x400D 0050	<b>TACCTL2</b>	Timer A capture and compare 2 control
0x400D 0054	<b>TACTR2</b>	Timer A counter 2
0x400D 0058	<b>TACCTL3</b>	Timer A capture and compare 3 control
0x400D 005C	<b>TACTR3</b>	Timer A counter 3
0x400D 0060	<b>TACCTL4</b>	Timer A capture and compare 4 control
0x400D 0064	<b>TACTR4</b>	Timer A counter 4
0x400D 0068	<b>TACCTL5</b>	Timer A capture and compare 5 control
0x400D 006C	<b>TACTR5</b>	Timer A counter 5
0x400D 0070	<b>TACCTL6</b>	Timer A capture and compare 6 control
0x400D 0074	<b>TACTR6</b>	Timer A counter 6
0x400D 0078	<b>TACCTL7</b>	Timer A capture and compare 7 control
0x400D 007C	<b>TACTR7</b>	Timer A counter 7
<b>Timer A Dead Time Generator</b>		
0x400D 00A0	<b>DTGA0CTL</b>	Timer A dead time generator 0 control
0x400D 00A4	<b>DTGA0LED</b>	Timer A dead time generator 0 leading edge delay
0x400D 00A8	<b>DTGA0TED</b>	Timer A dead time generator 0 trailing edge delay
0x400D 00B0	<b>DTGA1CTL</b>	Timer A dead time generator 1 control
0x400D 00B4	<b>DTGA1LED</b>	Timer A dead time generator 1 leading edge delay
0x400D 00B8	<b>DTGA1TED</b>	Timer A dead time generator 1 trailing edge delay
0x400D 00C0	<b>DTGA2CTL</b>	Timer A dead time generator 2 control
0x400D 00C4	<b>DTGA2LED</b>	Timer A dead time generator 2 leading edge delay
0x400D 00C8	<b>DTGA2TED</b>	Timer A dead time generator 2 trailing edge delay
0x400D 00D0	<b>DTGA3CTL</b>	Timer A dead time generator 3 control
0x400D 00D4	<b>DTGA3LED</b>	Timer A dead time generator 3 leading edge delay
0x400D 00D8	<b>DTGA3TED</b>	Timer A dead time generator 3 trailing edge delay

**Table 2-15. Timer B Register Map**

ADDRESS	NAME	DESCRIPTION
<b>Timer B</b>		
0x400E 0000	<b>TBCTL</b>	Timer B control
0x400E 0004	<b>TBPRD</b>	Timer B period
0x400E 0008	<b>TBCTR</b>	Timer B counter
<b>Timer B PWMB Capture and Compare</b>		
0x400E 0040	<b>TBCCTL0</b>	Timer B capture and compare 0 control
0x400E 0044	<b>TBCTR0</b>	Timer B counter 0
0x400E 0048	<b>TBCCTL1</b>	Timer B capture and compare 1 control
0x400E 004C	<b>TBCTR1</b>	Timer B counter 1
0x400E 0050	<b>TBCCTL2</b>	Timer B capture and compare 2 control
0x400E 0054	<b>TBCTR2</b>	Timer B counter 2
0x400E 0058	<b>TBCCTL3</b>	Timer B capture and compare 3 control
0x400E 005C	<b>TBCTR3</b>	Timer B counter 3
<b>Timer B Dead Time Generator</b>		
0x400E 00A0	<b>DTGB0CTL</b>	Timer B dead time generator 0 control
0x400E 00A4	<b>DTGB0LED</b>	Timer B dead time generator 0 leading edge delay
0x400E 00A8	<b>DTGB0TED</b>	Timer B dead time generator 0 trailing edge delay

**Table 2-16. Timer C Register Map**

ADDRESS	NAME	DESCRIPTION
<b>Timer C</b>		
0x400F 0000	<b>TCCTL</b>	Timer C control
0x400F 0004	<b>TCPRD</b>	Timer C period
0x400F 0008	<b>TCCTR</b>	Timer C counter
<b>Timer C PWMC Capture and Compare</b>		
0x400F 0040	<b>TCCCTL0</b>	Timer C capture and compare 0 control
0x400F 0044	<b>TCCTR0</b>	Timer C counter 0
0x400F 0048	<b>TCCCTL1</b>	Timer C capture and compare 1 control
0x400F 004C	<b>TCCTR1</b>	Timer C counter 1
<b>Timer C Dead Time Generator</b>		
0x400F 00A0	<b>DTGC0CTL</b>	Timer C dead time generator 0 control
0x400F 00A4	<b>DTGC0LED</b>	Timer C dead time generator 0 leading edge delay
0x400F 00A8	<b>DTGC0TED</b>	Timer C dead time generator 0 trailing edge delay



**Table 2-17. Timer D Register Map**

ADDRESS	NAME	DESCRIPTION
<b>Timer D</b>		
0x4010 0000	<b>TDCTL</b>	Timer D control
0x4010 0004	<b>TDPRD</b>	Timer D period
0x4010 0008	<b>TDCTR</b>	Timer D counter
<b>Timer D PWM Capture and Compare</b>		
0x4010 0040	<b>TDCCTL0</b>	Timer D capture and compare 0 control
0x4010 0044	<b>TDCTR0</b>	Timer D counter 0
0x4010 0048	<b>TDCCTL1</b>	Timer D capture and compare 1 control
0x4010 004C	<b>TDCTR1</b>	Timer D counter 1
<b>Timer D Dead Time Generator</b>		
0x4010 00A0	<b>DTGD0CTL</b>	Timer D dead time generator 0 control
0x4010 00A4	<b>DTGD0LED</b>	Timer D dead time generator 0 leading edge delay
0x4010 00A8	<b>DTGD0TED</b>	Timer D dead time generator 0 trailing edge delay

**Table 2-18. EMUX Register Map**

ADDRESS	NAME	DESCRIPTION
<b>EMUX</b>		
0x4015 0000	<b>EMUXCTL</b>	ADC external MUX control
0x4015 0004	<b>EMUXDATA</b>	ADC external MUX data

**Table 2-19. ADC Register Map**

ADDRESS	NAME	DESCRIPTION
<b>ADC</b>		
0x4015 0008	<b>ADCCTL</b>	ADC control
0x4015 000C	<b>ADCR</b>	ADC conversion result
0x4015 0010	<b>ADCINT</b>	ADC interrupt

**Table 2-20. ADC Auto-Sampling Sequencer 0 Register Map**

ADDRESS	NAME	DESCRIPTION
<b>ADC Auto-Sampling Sequencer 0</b>		
0x4015 0040	<b>ASCTL0</b>	Auto-sampling sequencer 0 control
0x4015 0044	<b>AS0S0</b>	Auto-sampling sequencer 0 sample 0 control
0x4015 0048	<b>AS0R0</b>	Auto-sampling sequencer 0 sample 0 result
0x4015 004C	<b>AS0S1</b>	Auto-sampling sequencer 0 sample 1 control
0x4015 0050	<b>AS0R1</b>	Auto-sampling sequencer 0 sample 1 result
0x4015 0054	<b>AS0S2</b>	Auto-sampling sequencer 0 sample 2 control
0x4015 0058	<b>AS0R2</b>	Auto-sampling sequencer 0 sample 2 result

ADDRESS	NAME	DESCRIPTION
0x4015 005C	<b>AS0S3</b>	Auto-sampling sequencer 0 sample 3 control
0x4015 0060	<b>AS0R3</b>	Auto-sampling sequencer 0 sample 3 result
0x4015 0064	<b>AS0S4</b>	Auto-sampling sequencer 0 sample 4 control
0x4015 0068	<b>AS0R4</b>	Auto-sampling sequencer 0 sample 4 result
0x4015 006C	<b>AS0S5</b>	Auto-sampling sequencer 0 sample 5 control
0x4015 0070	<b>AS0R5</b>	Auto-sampling sequencer 0 sample 5 result
0x4015 0074	<b>AS0S6</b>	Auto-sampling sequencer 0 sample 6 control
0x4015 0078	<b>AS0R6</b>	Auto-sampling sequencer 0 sample 6 result
0x4015 007C	<b>AS0S7</b>	Auto-sampling sequencer 0 sample 7 control
0x4015 0080	<b>AS0R7</b>	Auto-sampling sequencer 0 sample 7 result

**Table 2-21. ADC Auto-Sampling Sequencer 1 Register Map**

ADDRESS	NAME	DESCRIPTION
<b>ADC Auto-Sampling Sequencer 1</b>		
0x4015 0100	<b>ASCTL1</b>	Auto-sampling sequencer 1 control
0x4015 0104	<b>AS1S0</b>	Auto-sampling sequencer 1 sample 0 control
0x4015 0108	<b>AS1R0</b>	Auto-sampling sequencer 1 sample 0 result
0x4015 010C	<b>AS1S1</b>	Auto-sampling sequencer 1 sample 1 control
0x4015 0110	<b>AS1R1</b>	Auto-sampling sequencer 1 sample 1 result
0x4015 0114	<b>AS1S2</b>	Auto-sampling sequencer 1 sample 2 control
0x4015 0118	<b>AS1R2</b>	Auto-sampling sequencer 1 sample 2 result
0x4015 011C	<b>AS1S3</b>	Auto-sampling sequencer 1 sample 3 control
0x4015 0120	<b>AS1R3</b>	Auto-sampling sequencer 1 sample 3 result
0x4015 0124	<b>AS1S4</b>	Auto-sampling sequencer 1 sample 4 control
0x4015 0128	<b>AS1R4</b>	Auto-sampling sequencer 1 sample 4 result
0x4015 012C	<b>AS1S5</b>	Auto-sampling sequencer 1 sample 5 control
0x4015 0130	<b>AS1R5</b>	Auto-sampling sequencer 1 sample 5 result
0x4015 0134	<b>AS1S6</b>	Auto-sampling sequencer 1 sample 6 control
0x4015 0138	<b>AS1R6</b>	Auto-sampling sequencer 1 sample 6 result
0x4015 013C	<b>AS1S7</b>	Auto-sampling sequencer 1 sample 7 control
0x4015 0140	<b>AS1R7</b>	Auto-sampling sequencer 1 sample 7 result

**Table 2-22. I<sup>2</sup>C Register Map**

ADDRESS	NAME	DESCRIPTION
<b>I<sup>2</sup>C</b>		
0x40B0 0000	<b>I2CCFG</b>	I <sup>2</sup> C configuration
0x40B0 0004	<b>I2CINTSTAT</b>	I <sup>2</sup> C interrupt and status
0x40B0 0008	<b>I2CINTEN</b>	I <sup>2</sup> C interrupt enable
0x40B0 0030	<b>I2CMCTL</b>	I <sup>2</sup> C master access control

ADDRESS	NAME	DESCRIPTION
0x40B0 0034	<b>I2CMRXDATA</b>	I <sup>2</sup> C master receive data
0x40B0 0038	<b>I2CMTXDATA</b>	I <sup>2</sup> C master transmit data
0x40B0 0040	<b>I2CBAUD</b>	I <sup>2</sup> C master baud rate
0x40B0 0070	<b>I2CSRXDATA</b>	I <sup>2</sup> C slave receive data
0x40B0 0074	<b>I2CSTXDATA</b>	I <sup>2</sup> C slave transmit data
0x40B0 0078	<b>I2CSADDR</b>	I <sup>2</sup> C slave address

**Table 2-23. UART Register Map**

ADDRESS	NAME	DESCRIPTION
<b>UART</b>		
0x401D 0000	<b>UARTRX</b>	UART receive/transmit FIFO
0x401D 0004	<b>UARTIEN</b>	UART interrupt enable
0x401D 0008	<b>UARTII</b>	UART interrupt identification
0x401D 000C	<b>UARTLC</b>	UART line control
0x401D 0010	<b>Reserved</b>	Reserved
0x401D 0014	<b>UARTLS</b>	UART line status
0x401D 0018	<b>Reserved</b>	Reserved
0x401D 001C	<b>Reserved</b>	Reserved
0x401D 0020	<b>UARTFCTL</b>	UART FIFO control
0x401D 0024	<b>UARTIEN</b>	UART interrupt enable
0x401D 0028	<b>UARTDL_L</b>	UART divisor latch low byte
0x401D 002C	<b>UARTDL_H</b>	UART divisor latch high byte
0x401D 0038	<b>UARTFD_F</b>	UART fractional divisor value
0x401D 003C	<b>UARTFD_CTL</b>	UART fractional divisor control
0x401D 0040	<b>UARTFSTAT</b>	UART FIFO status

**Table 2-24. SOC Bus Bridge Register Map**

ADDRESS	NAME	DESCRIPTION
<b>SOC Bus Bridge</b>		
0x4020 0000	<b>SOCBCTL</b>	SOC Bus Bridge control
0x4020 0004	<b>SOCBCFG</b>	SOC Bus Bridge configuration
0x4020 0008	<b>SOCBCLKDIV</b>	SOC Bus Bridge clock divider
0x4000 000C	<b>Reserved</b>	Reserved
0x4000 0010	<b>Reserved</b>	Reserved
0x4020 0014	<b>SOCBSTAT</b>	SOC Bus Bridge status
0x4020 0018	<b>Reserved</b>	Reserved
0x4020 001C	<b>SOCBD</b>	SOC Bus Bridge data
0x4020 0020	<b>SOCBINT_EN</b>	SOC Bus Bridge interrupt enable

**Table 2-25. SPI Register Map**

ADDRESS	NAME	DESCRIPTION
<b>SPI</b>		
0x4021 0000	<b>SPICTL</b>	SPI control
0x4021 0004	<b>SPICFG</b>	SPI configuration
0x4021 0008	<b>SPICLKDIV</b>	SPI clock divider
0x4021 000C	<b>Reserved</b>	Reserved
0x4021 0010	<b>Reserved</b>	Reserved
0x4021 0014	<b>SPISTAT</b>	SPI status
0x4021 0018	<b>SPICSSTR</b>	SPI chip select steering
0x4021 001C	<b>SPID</b>	SPI data
0x4021 0020	<b>SPIINT_EN</b>	SPI interrupt enable

**Table 2-26. Multi-Mode Power Manager Register Map**

ADDRESS	NAME	DESCRIPTION
<b>Multi-Mode Power Manager</b>		
0x00	<b>SYSSTAT</b>	System status
0x08	<b>DEVID</b>	Device identification
0x09	<b>VERID</b>	Version identification
0x10	<b>PWRCTL</b>	Power manager control
0x11	<b>PWRSTAT</b>	Power manager status
0x12	<b>PWRSET</b>	Power manager setting
0x13	<b>IMOD</b>	Current modulation
0x14	<b>SSCFG</b>	Switching supply configuration

### 3. SYSTEM CLOCK CONTROL

#### 3.1. Register

##### 3.1.1. Register Map

**Table 3-1. System Clock Control Register Map**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>System Clock Control</b>			
0x4000 0000	<b>SCCTL</b>	System clock control	0x0000 0000
0x4000 0004	<b>PLLCTL</b>	PLL control	0x0000 0000
0x4000 0008	<b>ROSCCTL</b>	Ring oscillator control	0x0000 0007
0x4000 000C	<b>XTALCTL</b>	Crystal driver control	0x0000 0000

##### 3.1.2. SCCTL

**Register 3-1. SCCTL (System Clock Control, 0x4000 0000)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	R	0x0	Reserved
7	<b>FCLK</b>	RW	0x0	FCLK input clock select 1b: PLLOUT clock 0b: FRCLK
6:5	<b>HCLKDIV</b>	RW	0x0	HCLK divider 11b = FCLK / 8 10b = FCLK / 4 01b = FCLK / 2 00b = FCLK / 1
4:2	<b>ACLKDIV</b>	RW	0x0	ACLK divider 111b = FCLK / 128 110b = FCLK / 44 101b = FCLK / 32 100b = FCLK / 16 011b = FCLK / 8 010b = FCLK / 4 001b = FCLK / 2 000b = FCLK / 1
1:0	<b>CLKIN</b>	RW	0x0	FRCLK input clock select 11b = XTAL driver XIN/XOUT 10b = EXTCLK input 01b = CLKREF input (4MHz trimmed RC oscillator) 00b = internal ring oscillator ROSC

### 3.1.3. PLLCTL

**Register 3-2. PLLCTL (PLL Control, 0x4000 0004)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:24	Reserved	R	0x0	Reserved
23:20	Reserved	RW	0x0	Reserved, must be set to 0x0
19:16	PLLOUTDIV	RW	0x0	PLL output divider 1111b: / 15 ... 0001b: / 1 0000b: reserved
15:7	PLLFBDIV	RW	0x0	PLL feedback divider 1 1111 1111b: / 513 ... 0 0000 0001b: / 3 0 0000 0000b: / 2
6:2	PLLINDIV	RW	0x0	PLL input divider 1 1111b: / 33 ... 0 0001b: / 3 0 0000b: / 2
1	Reserved	RW	0x0	Reserved, must be set to 0x0
0	PLEN	RW	0x0	PLL oscillator 1b: enable PLL 0b: disable PLL

### 3.1.4. ROSCCTL

**Register 3-3. ROSCCTL (Ring Oscillator Control, 0x4000 0008)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:3	Reserved	R	0x0	Reserved
2:1	ROSCP	RW	0x3	Ring oscillator frequency setting 11b = 8.3MHz 10b = 10.7MHz 01b = 15.3MHz 00b = 28.7MHz
0	ROSCEN	RW	0x1	Enable Ring oscillator 1b: enable ROSC 0b: disable ROSC

### 3.1.5. XTALCTL

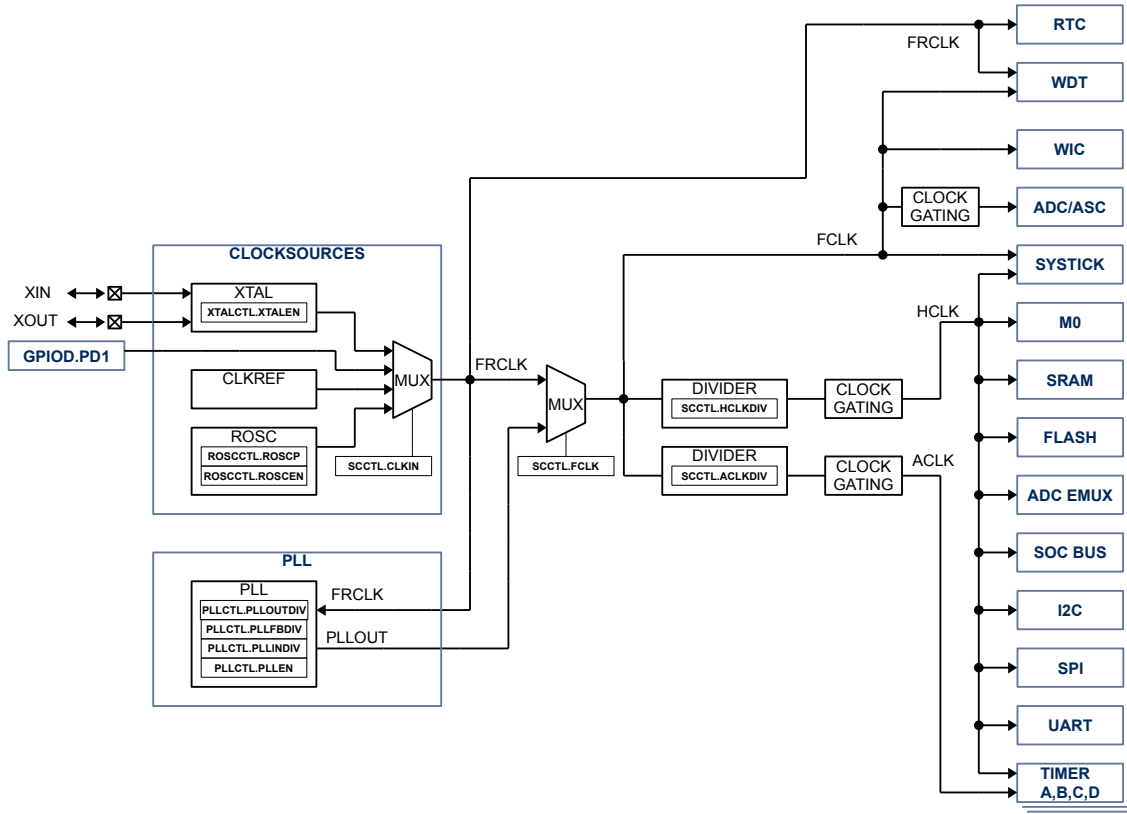
**Register 3-4. XTALCTL (Crystal Driver Control, 0x4000 000C)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:1	Reserved	R	0x0	Reserved
0	XTALEN	RW	0x0	Enable XTAL driver 1b: enable crystal driver 0b: disable crystal driver

## 3.2. Details of Operation

### 3.2.1. Block Diagram

Figure 3-1. System Clock Control



### 3.2.2. Configuration

Following blocks need to be configured for correct use of the system clock control:

- GPIO.D.PD1
- RTC
- WDT
- WIC
- ADC/ASC
- SYSTICK
- SRAM
- FLASH
- ADC EMUX
- SOC BUS
- I2C



- SPI
- UART
- Timer A, B, C, D

### 3.2.3. ROSC

The internal ring oscillator has four frequency settings controllable with **ROSCCTL.ROSCP** from 8.3MHz to 28.7MHz in four steps. The ROSC can also be disabled using **ROSCCTL.ROSCEN**.

### 3.2.4. CLKREF

The CLKREF provides a 1% trimmed 4MHz clock.

### 3.2.5. XTAL

The crystal driver supports a range of crystal frequencies from 2MHz to 10MHz. The crystal driver can also be used to input an external clock using XIN.

The crystal driver can be enabled with **XTALCTL.XTALEN**.

### 3.2.6. EXTCLK

PD1 can be configured as clock input, EXTCLK.

### 3.2.7. PLL

The clock input to the PLL is FRCLK.

The PLL can be enabled with **PLLCTL.PLEN**.

The PLL output clock PLLOUT is following following equation:

$$PLLOUT = \frac{PLLIN * PLLFBDIV}{PLLINDIV * PLLOUTDIV * 2} \quad (1)$$

Where:

PLLOUT: PLL output frequency in MHz

PLLIN: PLL input frequency in MHz (FRCLK)

PLLINDIV: PLL input divider (2 to 33) **PLLCTL.PLLINDIV**

PLLFBDIV: PLL feedback divider (2 to 513) **PLLCTL.PLLFBDIV**

PLLOUTDIV: PLL output divider (1 to 16) **PLLCTL.PLLOUTDIV**

The input clock frequency and input clock divider selection must follow formula below for correct operation of PLL:

$$1\text{MHz} \leq \frac{PLLIN}{PLLINDIV} \leq 25\text{MHz} \quad (2)$$

The output clock frequency and output clock divider selection must following formula below for correct operation of the PLL

$$100\text{MHz} \leq PLLOUT * PLLOUTDIV \leq 250\text{MHz} \\ PLLOUTDIV \geq 1 \quad (3)$$

The table below shows pre-calculated PLL output frequency settings using the 4MHz ROSC as input.

**Table 3-5. PLL output frequency settings using 4MHz ROSC as input**

PLL output	PLLOUTDIV	PLLFBDIV	PLLINDIV
10MHz	0x01 (/[1*2])	0x008 (*10)	0x0 (/2)
16.8MHz	0x05 (/[5*2])	0x052 (*84)	0x0 (/2)
20MHz	0x01 (/[1*2])	0x012 (*20)	0x0 (/2)
25MHz	0x01 (/[1*2])	0x019 (*25)	0x0 (/2)
30MHz	0x01 (/[1*2])	0x01C (*30)	0x0 (/2)
33.333MHz	0x01 (/[1*2])	0x030 (*50)	0x1 (/3)
40MHz	0x01 (/[1*2])	0x026 (*40)	0x0 (/2)
50MHz	0x01 (/[1*2])	0x030 (*50)	0x0 (/2)
60MHz	0x01 (/[1*2])	0x03A (*60)	0x0 (/2)
70MHz	0x01 (/[1*2])	0x044 (*70)	0x0 (/2)
80MHz	0x01 (/[1*2])	0x04E (*80)	0x0 (/2)
90MHz	0x01 (/[1*2])	0x058 (*90)	0x0 (/2)
100MHz	0x01 (/[1*2])	0x062 (*100)	0x0 (/2)

### 3.2.8. FRCLK

The free running clock FRCLK clock source can be selected with **SCCTL.CLKIN**. From XTAL, EXTCLK, CLKREF, or ROSC.

### 3.2.9. FCLK

The fast clock FCLK clock source can be selected with **SCCTL.FCLK** to be either PLLOUT or FRCLK.

### 3.2.10. HCLK

The high speed clock HCLK clock source input is FCLK. The HCLK can be divided down from FCLK using **SCCTL.HCLKDIV** from /1 to /8 in 4 steps.

### 3.2.11. ACLK

The auxiliary clock ACLK clock source input is FCLK. The ACLK can be divided down from FCLK from /1 to /128 in 8 steps.

### 3.2.12. Clock Gating

When the CPU enters deep sleep mode, the HCLK, ACLK and FCLK to the ADC are clock gated and stopped to save power. Only FRCLK and FCLK continue to run to supply WIC, RTC, and WDT.

## 4. WATCHDOG TIMER

### 4.1. Register

#### 4.1.1. Register Map

Table 4-1. Watchdog Timer Register Map

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>Watchdog Timer</b>			
0x4003 0000	<b>WDTCTL</b>	Watchdog timer control	0x6300 0000
0x4003 0004	<b>WDTCDV</b>	Watchdog timer count-down value	0x63FF FFFF
0x4003 0008	<b>WDTCTR</b>	Watchdog timer counter	0x00FF FFFF

#### 4.1.2. WDTCTL

Register 4-1. WDTCTL (Watchdog Timer Control, 0x4003 0000)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:24	<b>KEY</b>	RW	0x63	Key for writing WDTCTL register 0x4A: allow writes to WDTCTL register 0x63: read value of WDTCTL.KEY
23:12	<b>Reserved</b>	R	0x0	Reserved
11	<b>WRBUSY</b>	RW	0x0	WDTCTL write busy 1b = write to WDTCTL still being processed. Any register writes received while this bit is set to a 1b will be dropped and not written to the given register. Any reads will return indeterminate data. 0b = WDT registers not busy
10	<b>WDTCLKSEL</b>	RW	0x0	WDT input clock select 1b: FCLK 0b: FRCLK
9:6	<b>WDTCLKDIV</b>	RW	0x0	WDT clock input divider 1111b: /65536 1110b: /32768 1101b: /16384 1100b: /8192 1011b: /4096 1010b: /2048 1001b: /1024 1000b: /512 0111b: /256 0110b: /128 0101b: /64 0100b: /32 0011b: /16 0010b: /8 0001b: /4 0000b: /2
5	<b>WDTRESETEEN</b>	RW	0x0	Watchdog device RESET enable 1b = WDT trigger device RESET when WDTCTR register counts to 0x0 0b = WDT trigger device RESET disabled
4	<b>WDTINT</b>	RW	0x0	Watchdog interrupt 1b = WDT interrupt 0b = no WDT interrupt

BIT	NAME	ACCESS	RESET	DESCRIPTION
3	<b>WDTINTEN</b>	RW	0x0	Watchdog interrupt enable 1b = enable WDT interrupt 0b = disable WDT interrupt
2:0	<b>WDTCTRRST</b>	RW	0x0	WDTCTR counter reset 101b = write of 101b reset WDTCTR to WDTCDV value

#### 4.1.3. WDTCDV

##### Register 4-2. WDTCDV (Watchdog Timer Count-Down Value, 0x4003 0004)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:24	<b>KEY</b>	RW	0x63	Key for writing WDTCDV register 0x4A: allow writes to WDTCDV register 0x63: read value of WDTCDV.KEY
23:0	<b>RSTVALUE</b>	RW	0xFF FFFF	24-bit WDT count-down value

#### 4.1.4. WDTCTR

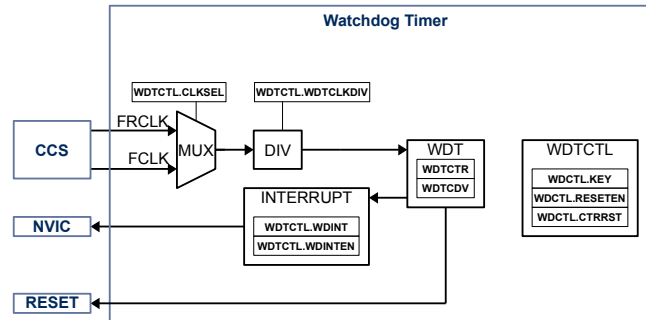
##### Register 4-3. WDTCTR (Watchdog Timer Counter, 0x4003 0008)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:24	<b>Reserved</b>	R	0x0	Reserved
23:0	<b>CTR</b>	RW	0xFF FFFF	Current value of WDT

## 4.2. Details of Operation

### 4.2.1. Block Diagram

Figure 4-1. WDT



### 4.2.2. Configuration

Following blocks need to be configured for correct use of the WDT:

- Clock Control System (CCS)
- Nested Vectored Interrupt Controller (NVIC)

### 4.2.3. Watchdog Timer

The Watchdog timer consist of a 24-bit timer and reset logic. The WDT can be used as general purpose 16bit timer or as watchdog timer that need to be serviced periodically to avoid device reset.

### 4.2.4. Access WDT Registers

The **WDTCTL** and **WDTCDV** registers can only be written to if **WDTCTL.KEY** or **WDTCDV.KEY** are set to 0x4A.

The read back value of **WDTCTL.KEY** or **WDTCDV.KEY** is always 0x63. The watchdog timer has 2 clock domains, HCLK and WDT clock domain set by **WDTCTL.CLKSEL** and **WDTCTL.WDTCLKDIV**. Writing to any WDT registers may take up to 1 clock cycle on the WDT clock domain to finish. Any ongoing writes to WDT registers are shown with **WDTCTL.WRBUSY**. As long as **WDTCTL.WRBUSY** is 1b, any subsequent writes to WDT registers are ignored and reads only provide undetermined data.

### 4.2.5. WDT Clock Setting

The WDT can use 2 clocks FCLK or FRCLK, selectable with **WDTCTL.CLKSEL**. For applications where the WDT need to run in CPU sleep mode, FRCLK should be used. The clock input can be further divided down from /2 to /65536 using **WDTCTL.WDTCLKDIV**.

### 4.2.6. General Purpose Timer Mode

Set **WDTCTL.WDTRESETEN** to 0b to use the WDT as general purpose 24-bit timer. Set the desired count value in WDT clocks in **WDTCDV.RSTVALUE**, set **WDTCTL.WDTCTRRST** to 101b to load **WDTCTR.CTR** with **WDTCDV.RSTVALUE**. To start the GPT timer set **GPTCTL.INTEN**. When **WDTCTR.CTR** reaches 0x0, the timer

automatic reloads **WDTCTR.CTR** with **WDTCDV.RSTVALUE** and continues countdown. The WDT is stopped when **WDTCTL.INTEN** is cleared.

#### 4.2.7. Watchdog Timer Mode

Set **WDTCTL.WDTRESETEEN** to 1b to use the WDT as 24-bit watchdog timer with device reset capability. Set **WDTCTL.WDTINTEN** to 1b to enable interrupt when WDT counts to 0x0. Set the desired count value in WDT clocks in **WDTCDV.RSTVALUE**. To start the WDT count down, set **WDTCTL.WDTCTRRST** to 101b. The WDT will copy the **WDTCTL.RSTVALUE** to **WDTCTR.CTR** and start counting down. When **WDTCTR.CTR** reaches 0x0, the WDT will automatic copy **WDTCTL.RSTVALUE** to **WDTCTR.CTR**, restart the counter and set the interrupt flag if enabled is set. During the second count down, set **WDTCTL.WDTCTRRST** to 101b to restart the 1<sup>st</sup> WDT countdown and avoid device reset. If the **WDTCTR.CTR** reaches 0x0 during the second count down without being reloaded, the WDT will toggle device reset.

## 5. GENERAL PURPOSE TIMER

### 5.1. Register

#### 5.1.1. Register Map

**Table 5-1. General Purpose Timer Register Map**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>Real Time Clock</b>			
0x4004 0000	<b>GPTCTL</b>	General purpose timer control	0x6300 0000
0x4004 0004	<b>GPTCDV</b>	General purpose timer count-down value	0x63FF FFFF
0x4004 0008	<b>GPTCTR</b>	General purpose timer counter	0x00FF FFFF

#### 5.1.2. GPTCTL

**Register 5-1. GPTCTL (Real Time Clock Control, 0x4004 0000)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:24	<b>KEY</b>	RW	0x63	Key for writing <b>GPTCTL</b> register 0x4A: allow writes to <b>GPTCTL</b> register 0x63: read value of <b>GPTCTL.KEY</b>
23:12	<b>Reserved</b>	R	0x0	Reserved
11	<b>WRBUSY</b>	RW	0x0	<b>GPTCTL</b> write busy 1b = write to <b>GPTCTL</b> still being processed. Any register writes received while this bit is set to a 1b will be dropped and not written to the given register. Any reads will return indeterminate data. 0b = GPT registers not busy
10	<b>GPTCLKSEL</b>	RW	0x0	GPT input clock select 1b: FCLK 0b: FRCLK
9:6	<b>GPTCLKDIV</b>	RW	0x0	GPT clock input divider 1111b: /65536 1110b: /32768 1101b: /16384 1100b: /8192 1011b: /4096 1010b: /2048 1001b: /1024 1000b: /512 0111b: /256 0110b: /128 0101b: /64 0100b: /32 0011b: /16 0010b: /8 0001b: /4 0000b: /2
5	<b>Reserved</b>	R	0x0	Reserved
4	<b>GPTINT</b>	RW	0x0	Real time clock interrupt 1b = GPT interrupt 0b = no GPT interrupt



BIT	NAME	ACCESS	RESET	DESCRIPTION
3	<b>GPTINTEN</b>	RW	0x0	Real time clock interrupt enable 1b = enable GPT interrupt 0b = disable GPT interrupt
2:0	<b>GPTCTRRST</b>	RW	0x0	<b>GPTCTR</b> counter reset 101b = write of 101b reset <b>GPTCTR</b> to <b>GPTCDV</b> value

### 5.1.3. GPTCDV

#### Register 5-2. GPTCDV (Real Time Clock Count-Down Value, 0x4004 0004)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:24	<b>KEY</b>	RW	0x63	Key for writing <b>GPTCTL</b> register 0x4A: allow writes to <b>GPTCTL</b> register 0x63: read value of <b>GPTCTL.KEY</b>
23:0	<b>RSTVALUE</b>	RW	0xFF FFFF	24bit GPT count-down value

### 5.1.4. GPTCTR

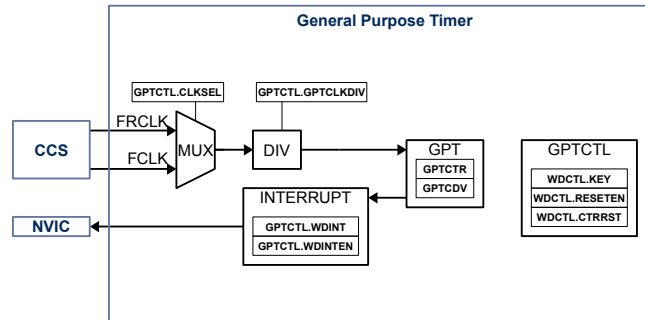
#### Register 5-3. GPTCTR (Real Time Clock Counter, 0x4004 0008)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:24	<b>Reserved</b>	R	0x0	Reserved
23:0	<b>CTR</b>	RW	0xFF FFFF	Current value of GPT

## 5.2. Details of Operation

### 5.2.1. Block Diagram

Figure 5-1. GPT



### 5.2.2. Configuration

Following blocks need to be configured for correct use of the GPT:

- Clock Control System (CCS)
- Nested Vectored Interrupt Controller (NVIC)

### 5.2.3. General Purpose Timer

The General purpose timer consist of a 24-bit timer, can also run in device sleep mode if FRLCK is used.

### 5.2.4. Access GPT Registers

The **GPTCTL** and **GPTCDV** registers can only be written to if **GPTCTL.KEY** or **GPTCDV.KEY** are set to 0x4A.

The read back value of **GPTCTL.KEY** or **GPTCDV.KEY** is always 0x63. The general purpose timer has 2 clock domains, HCLK and GPT clock domain set by **GPTCTL.CLKSEL** and **GPTCTL.GPTCLKDIV**. Writing to any GPT registers may take up to 1 clock cycle on the GPT clock domain to finish. Any ongoing writes to GPT registers are shown with **GPTCTL.WRBUSY**. As long as **GPTCTL.WRBUSY** is 1b, any subsequent writes to GPT registers are ignored and reads only provide undetermined data.

### 5.2.5. GPT Clock Setting

The GPT can use 2 clocks HCLK or FRCLK, selectable with **GPTCTL.CLKSEL**. For applications where the GPT need to run in CPU sleep mode, FRCLK should be used. The clock input can be further divided down from /2 to /65536 using **GPTCTL.GPTCLKDIV**.

### 5.2.6. General Purpose Timer Mode

Set **GPTCTL.GPTRESETEN** to 0b to use the GPT as general purpose 24-bit timer. Set the desired count value in GPT clocks in **GPTCDV.RSTVALUE**, set **GPTCTL.GPTCTRRST** to 101b to load **GPTCTR.CTR** with **GPTCDV.RSTVALUE**. To start the GPT timer set **GPTCTL.INTEN**. When **GPTCTR.CTR** reaches 0x0, the timer automatic reloads **GPTCTR.CTR** with **GPTCDV.RSTVALUE** and continues countdown. The GPT is stopped when **GPTCTL.INTEN** is cleared.

## 6. GPIO PORT A

### 6.1. Register

#### 6.1.1. Register Map

**Table 6-1. GPIO Port A Register Map**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>GPIO Port A</b>			
0x4007 0000	<b>GPIOAO</b>	GPIO Port A output	0x0000 0000
0x4007 0004	<b>GPIOAOE</b>	GPIO Port A output enable	0x0000 0000
0x4007 0008	<b>GPIOADS</b>	GPIO Port A output drive strength	0x0000 0000
0x4007 000C	<b>GPIOAORUP</b>	GPIO Port A output weak pull up	0x0000 0000
0x4007 0010	<b>GPIOAORDN</b>	GPIO Port A output weak pull down	0x0000 0000
0x4007 0014	<b>GPIOAI</b>	GPIO Port A input	0x0000 0000
0x4007 0018	<b>Reserved</b>	Reserved	0x0000 0000
0x4007 001C	<b>GPIOAPSEL</b>	GPIO Port A peripheral select	0x0000 0000
0x4007 0020	<b>GPIOAINTP</b>	GPIO Port A interrupt polarity select	0x0000 0000
0x4007 0024	<b>GPIOAINTEN</b>	GPIO Port A interrupt enable select	0x0000 0000
0x4007 0028	<b>GPIOAINT</b>	GPIO Port A interrupt flag	0x0000 0000
0x4007 002C	<b>GPIOAINTM</b>	GPIO Port A interrupt mask	0x0000 0000

#### 6.1.2. GPIOAO

**Register 6-1. GPIOAO (GPIO Port A Output, 0x4007 0000)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PA7O</b>	RW	0x0	Port A output 7 1b: set output high if <b>GPIOAOE.PAOE7</b> = 1b 0b: set output low if <b>GPIOAOE.PAOE7</b> = 1b
6	<b>PA6O</b>	RW	0x0	Port A output 6 1b: set output high if <b>GPIOAOE.PAOE6</b> = 1b 0b: set output low if <b>GPIOAOE.PAOE6</b> = 1b
5	<b>PA5O</b>	RW	0x0	Port A output 5 1b: set output high if <b>GPIOAOE.PAOE5</b> = 1b 0b: set output low if <b>GPIOAOE.PAOE5</b> = 1b
4	<b>PA4O</b>	RW	0x0	Port A output 4 1b: set output high if <b>GPIOAOE.PAOE4</b> = 1b 0b: set output low if <b>GPIOAOE.PAOE4</b> = 1b
3	<b>PA3O</b>	RW	0x0	Port A output 3 1b: set output high if <b>GPIOAOE.PAOE3</b> = 1b 0b: set output low if <b>GPIOAOE.PAOE3</b> = 1b
2	<b>PA2O</b>	RW	0x0	Port A output 2 1b: set output high if <b>GPIOAOE.PAOE2</b> = 1b 0b: set output low if <b>GPIOAOE.PAOE2</b> = 1b

BIT	NAME	ACCESS	RESET	DESCRIPTION
1	PA1O	RW	0x0	Port A output 1 1b: set output high if GPIOAOE.PAOE1 = 1b 0b: set output low if GPIOAOE.PAOE1 = 1b
0	PA0O	RW	0x0	Port A output 0 1b: set output high if GPIOAOE.PAOE0 = 1b 0b: set output low if GPIOAOE.PAOE0 = 1b

### 6.1.3. GPIOAOE

#### Register 6-2. GPIOAOE (GPIO Port A Output Enable, 0x4007 0004)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PA7OE	RW	0x0	Port A output 7 enable 1b: output state set by GPIOAO.PA07 0b: output disabled, high-impedance state
6	PA6OE	RW	0x0	Port A output 6 enable 1b: output state set by GPIOAO.PA06 0b: output disabled, high-impedance state
5	PA5OE	RW	0x0	Port A output 5 enable 1b: output state set by GPIOAO.PA05 0b: output disabled, high-impedance state
4	PA4OE	RW	0x0	Port A output 4 enable 1b: output state set by GPIOAO.PA04 0b: output disabled, high-impedance state
3	PA3OE	RW	0x0	Port A output 3 enable 1b: output state set by GPIOAO.PA03 0b: output disabled, high-impedance state
2	PA2OE	RW	0x0	Port A output 2 enable 1b: output state set by GPIOAO.PA02 0b: output disabled, high-impedance state
1	PA1OE	RW	0x0	Port A output 1 enable 1b: output state set by GPIOAO.PA01 0b: output disabled, high-impedance state
0	PA0OE	RW	0x0	Port A output 0 enable 1b: output state set by GPIOAO.PA00 0b: output disabled, high-impedance state

### 6.1.4. GPIOADS

#### Register 6-3. GPIOADS (GPIO Port A Output Drive Strength, 0x4007 0008)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PA7ODS	RW	0x0	Port A output 7 drive strength select 1b: high 0b: low
6	PA6ODS	RW	0x0	Port A output 6 drive strength select 1b: high 0b: low

BIT	NAME	ACCESS	RESET	DESCRIPTION
5	PA5ODS	RW	0x0	Port A output 5 drive strength select 1b: high 0b: low
4	PA4ODS	RW	0x0	Port A output 4 drive strength select 1b: high 0b: low
3	PA3ODS	RW	0x0	Port A output 3 drive strength select 1b: high 0b: low
2	PA2ODS	RW	0x0	Port A output 2 drive strength select 1b: high 0b: low
1	PA1ODS	RW	0x0	Port A output 1 drive strength select 1b: high 0b: low
0	PA0ODS	RW	0x0	Port A output 0 drive strength select 1b: high 0b: low

### 6.1.5. GPIOARUP

#### Register 6-4. GPIOARUP (GPIO Port A Weak Pull Up, 0x4007 000C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PA7RUP	RW	0x0	Port A 7 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
6	PA6RUP	RW	0x0	Port A 6 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
5	PA5RUP	RW	0x0	Port A 5 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
4	PA4RUP	RW	0x0	Port A 4 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
3	PA3RUP	RW	0x0	Port A 3 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
2	PA2RUP	RW	0x0	Port A 2 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
1	PA1RUP	RW	0x0	Port A 1 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
0	PA0RUP	RW	0x0	Port A 0 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO

### 6.1.6. GPIOARDN

**Register 6-5. GPIOARDN (GPIO Port A Weak Pull Down, 0x4007 0010)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PA7RDN	RW	0x0	Port A 7 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
6	PA6RDN	RW	0x0	Port A 6 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
5	PA5RDN	RW	0x0	Port A 5 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
4	PA4RDN	RW	0x0	Port A 4 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
3	PA3RDN	RW	0x0	Port A 3 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
2	PA2RDN	RW	0x0	Port A 2 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
1	PA1RDN	RW	0x0	Port A 1 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
0	PA0RDN	RW	0x0	Port A 0 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS

### 6.1.7. GPIOAI

**Register 6-6. GPIOAI (GPIO Port A Input, 0x4007 0014)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RW	0x0	Reserved
7	PA7I	RW	0x0	Port A 7 input state 1b: input high 0b: input low
6	PA6I	RW	0x0	Port A 6 input state 1b: input high 0b: input low
5	PA5I	RW	0x0	Port A 5 input state 1b: input high 0b: input low
4	PA4I	RW	0x0	Port A 4 input state 1b: input high 0b: input low
3	PA3I	RW	0x0	Port A 3 input state 1b: input high 0b: input low

BIT	NAME	ACCESS	RESET	DESCRIPTION
2	<b>PA2I</b>	RW	0x0	Port A 2 input state 1b: input high 0b: input low
1	<b>PA1I</b>	RW	0x0	Port A 1 input state 1b: input high 0b: input low
0	<b>PA0I</b>	RW	0x0	Port A 0 input state 1b: input high 0b: input low

### 6.1.8. GPIOAPSEL

#### Register 6-7. GPIOAPSEL (GPIO Port A Peripheral Select, 0x4007 001C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RW	0x0	Reserved
15:14	<b>PA7PSEL</b>	RW	0x0	Port A 7 peripheral select 11b: PWMC1 / DTGC0HS output or CC1 capture and compare input 10b: PWMA7 / DTGA3HS output or CA7 capture and compare input 01b: PWMA5 / DTGA1HS output or CA5 capture and compare input 00b: I/O mode PA7
13:12	<b>PA6PSEL</b>	RW	0x0	Port A 6 peripheral select 11b: reserved 10b: PWMB0 / DTGB0LS output or CB0 capture and compare input 01b: PWMA4 / DTGA0HS output or CA4 capture and compare input 00b: I/O mode PA6
11:10	<b>PA5PSEL</b>	RW	0x0	Port A 5 peripheral select 11b: reserved 10b: PWMD0 / DTGD0LS output or CD0 capture and compare input / IBCTL5 01b: PWMA6 / DTGA2HS output or CA6 capture and compare input / IBCTL5 00b: I/O mode PA5
9:8	<b>PA4PSEL</b>	RW	0x0	Port A 4 peripheral select 11b: reserved 10b: PWMC0 / DTGC0LS output or CC0 capture and compare input / IBCTL4 01b: PWMA5 / DTGA1HS output or CA5 capture and compare input / IBCTL4 00b: I/O mode PA4
7:6	<b>PA3PSEL</b>	RW	0x0	Port A 3 peripheral select 11b: PWMB0 / DTGB0LS output or CB0 capture and compare input / IBCTL3 10b: PWMA4 / DTGA0HS output or CA4 capture and compare input / IBCTL3kl 01b: PWMA3 / DTGA3LS output or CA3 capture and compare input / IBCTL3 00b: I/O mode PA3
5:4	<b>PA2PSEL</b>	RW	0x0	Port A 2 peripheral select 11b: reserved 10b: reserved 01b: PWMA2 / DTGA2LS output or CA2 capture and compare input / IBCTL2 00b: I/O mode PA2



BIT	NAME	ACCESS	RESET	DESCRIPTION
3:2	<b>PA1PSEL</b>	RW	0x0	Port A 1 peripheral select 11b: reserved 10b: reserved 01b: PWMA1 / DTGA1LS output or CA1 capture and compare input / IBCTL1 00b: I/O mode PA1
1:0	<b>PA0PSEL</b>	RW	0x0	Port A 0 peripheral select 11b: reserved 10b: reserved 01b: PWMA0 / DTGA0LS output or CA0 capture and compare input / IBCTL0 00b: I/O mode PA0

### 6.1.9. GPIOAINTP

#### Register 6-8. GPIOAINTP (GPIO Port A Interrupt Polarity, 0x4007 0020)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PA7INTP</b>	RW	0x0	Port A 7 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
6	<b>PA6INTP</b>	RW	0x0	Port A 6 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
5	<b>PA5INTP</b>	RW	0x0	Port A 5 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
4	<b>PA4INTP</b>	RW	0x0	Port A 4 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
3	<b>PA3INTP</b>	RW	0x0	Port A 3 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
2	<b>PA2INTP</b>	RW	0x0	Port A 2 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
1	<b>PA1INTP</b>	RW	0x0	Port A 1 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
0	<b>PA0INTP</b>	RW	0x0	Port A 0 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition

### 6.1.10. GPIOAINTEN

#### Register 6-9. GPIOAINTEN (GPIO Port A Interrupt Enable, 0x4007 0024)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PA7INTEN</b>	RW	0x0	Port A 7 interrupt enable 1b: enabled interrupt 0b: disable interrupt

BIT	NAME	ACCESS	RESET	DESCRIPTION
6	<b>PA6INTEN</b>	RW	0x0	Port A 6 interrupt enable 1b: enabled interrupt 0b: disable interrupt
5	<b>PA5INTEN</b>	RW	0x0	Port A 5 interrupt enable 1b: enabled interrupt 0b: disable interrupt
4	<b>PA4INTEN</b>	RW	0x0	Port A 4 interrupt enable 1b: enabled interrupt 0b: disable interrupt
3	<b>PA3INTEN</b>	RW	0x0	Port A 3 interrupt enable 1b: enabled interrupt 0b: disable interrupt
2	<b>PA2INTEN</b>	RW	0x0	Port A 2 interrupt enable 1b: enabled interrupt 0b: disable interrupt
1	<b>PA1INTEN</b>	RW	0x0	Port A 1 interrupt enable 1b: enabled interrupt 0b: disable interrupt
0	<b>PA0INTEN</b>	RW	0x0	Port A 0 interrupt enable 1b: enabled interrupt 0b: disable interrupt

### 6.1.11. GPIOAINT

#### Register 6-10. GPIOAINT (GPIO Port A Interrupt, 0x4007 0028)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PA7INT</b>	RW	0x0	Port A 7 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
6	<b>PA6INT</b>	RW	0x0	Port A 6 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
5	<b>PA5INT</b>	RW	0x0	Port A 5 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
4	<b>PA4INT</b>	RW	0x0	Port A 4 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
3	<b>PA3INT</b>	RW	0x0	Port A 3 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
2	<b>PA2INT</b>	RW	0x0	Port A 2 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
1	<b>PA1INT</b>	RW	0x0	Port A 1 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
0	<b>PA0INT</b>	RW	0x0	Port A 0 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending

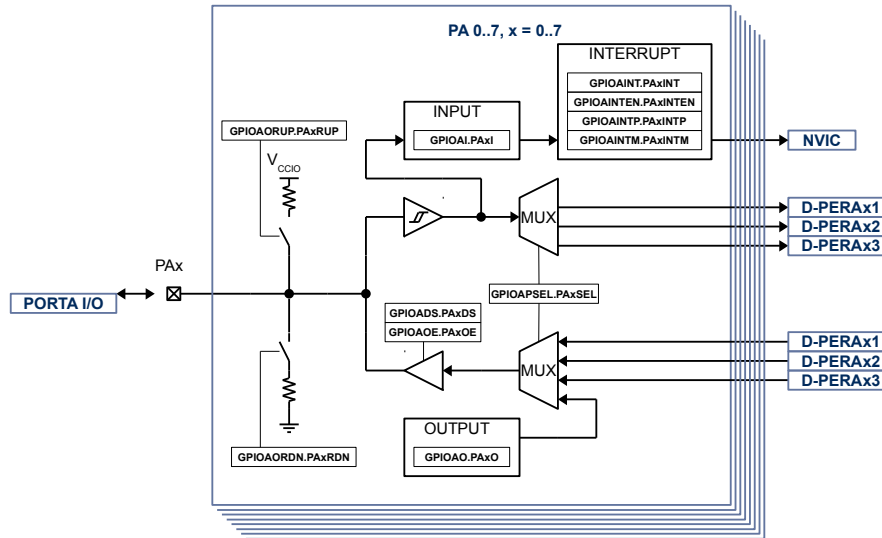
**6.1.12. GPIOAINTM**
**Register 6-11. GPIOAINTM (GPIO Port A Interrupt Mask, 0x4007 002C)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PA7INTM</b>	RW	0x0	Port A 7 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
6	<b>PA6INTM</b>	RW	0x0	Port A 6 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
5	<b>PA5INTM</b>	RW	0x0	Port A 5 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
4	<b>PA4INTM</b>	RW	0x0	Port A 4 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
3	<b>PA3INTM</b>	RW	0x0	Port A 3 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
2	<b>PA2INTM</b>	RW	0x0	Port A 2 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
1	<b>PA1INTM</b>	RW	0x0	Port A 1 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
0	<b>PA0INTM</b>	RW	0x0	Port A 0 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask

## 6.2. Details of Operation

### 6.2.1. Block Diagram

Figure 6-1. GPIO Port A



### 6.2.2. Configuration

Following blocks need to be configured for correct use of the GPIO A:

- Nested Vectored Interrupt Controller (NVIC)
- Gate Driver
- Timer A, PWMA, DTGA
- Timer B, PWMB, DTGB
- Timer C, PWMC, DTGC
- Timer D, PWMD, DTGD
- General Purpose Gate Drivers

### 6.2.3. GPIO A Block

The GPIO A block consists of up to 8 general purpose input output (GPIO). Each GPIO has interrupt capabilities, weak pull-up or pull-down, programmable output drive strength, High-Z output operation. Some of the GPIO can be configured as PWM output, or capture and compare input.

### 6.2.4. Input

The input state of GPIOA can be monitored with **GPIOAI.PAxI**. The input state can be monitored regardless of the peripheral select setting **GPIOAPSEL**.

### 6.2.5. Output and Output Enable

When **GPIOAOE.PAxOE** is enabled, the output state is controlled by **GPIOAO.PAxO**.

When **GPIOAOE.PAxOE** is disabled, the output is in High-Z state.

### 6.2.6. Output Drive Strength

The output drive strength can be adjusted using **GPIOADS** to meet application needs. Set **GPIOADS.PAxODS** to enable high current drive strength, reset to enable low current drive strength.

### 6.2.7. Weak Pull Up and Pull Down

Independent from the output settings, weak pull up can be enabled with **GPIOAORUP** and weak pull down can be enabled with **GPIOARDN**.

#### NOTE:

**GPIOARUP.PAxRUP** or **GPIOARDN.PAxRDN** should never be enabled at the same time for a single GPIO. If switching from weak pull-up to weak pull-down is required, disable weak pull-up first before enable weak pull-down and vice versa.

### 6.2.8. Peripheral Select

Each GPIO is connected to up to 4 digital peripherals, selectable with **GPIOAPSEL**. When a different function than IO is selected the input state can still be read with **GPIOAI** and the pull-up and pull-down is still controllable.

### 6.2.9. Interrupt

The interrupt for each GPIO can be enabled with **GPIOAINTEN**. The interrupt can be configured to be rising signal edge or falling signal edge using **GPIOAINTP**. The state of the interrupt can be read from **GPIOAINT**. The individual interrupt bits can be cleared by writing to 1.

When the GPIO interrupts are enabled for the first time after device start-up, it may be in an uncertain state and generate an interrupt. To avoid this the **GPIOAINTM** mask bit need to be set before enabled interrupt bits.

To allow interrupt to be recognized by the CPU the GPIO interrupt need also be enabled in the NVIC.

## 7. GPIO PORT B

### 7.1. Register

#### 7.1.1. Register Map

**Table 7-1. GPIO Port B Register Map**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>GPIO Port B</b>			
0x4007 0040	<b>GPIOBO</b>	GPIO Port B output	0x0000 0000
0x4007 0044	<b>GPIOBOE</b>	GPIO Port B output enable	0x0000 0000
0x4007 0048	<b>GPIOBODS</b>	GPIO Port B output drive strength	0x0000 0000
0x4007 004C	<b>GPIOBORUP</b>	GPIO Port B output weak pull up	0x0000 0000
0x4007 0050	<b>GPIOBORDN</b>	GPIO Port B output weak pull down	0x0000 0000
0x4007 0054	<b>GPIOBI</b>	GPIO Port B input	0x0000 0000
0x4007 0058	<b>Reserved</b>	Reserved	0x0000 0000
0x4007 005C	<b>GPIOBPSEL</b>	GPIO Port B peripheral select	0x0000 0000
0x4007 0060	<b>GPIOBINTP</b>	GPIO Port B interrupt polarity select	0x0000 0000
0x4007 0064	<b>GPIOBINTEN</b>	GPIO Port B interrupt enable select	0x0000 0000
0x4007 0068	<b>GPIOBINT</b>	GPIO Port B interrupt flag	0x0000 0000
0x4007 006C	<b>GPIOBINTM</b>	GPIO Port B interrupt mask	0x0000 0000

#### 7.1.2. GPIOBO

**Register 7-1. GPIOBO (GPIO Port B Output, 0x4007 0040)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
6	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
5	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
4	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
3	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
2	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
1	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
0	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b

#### 7.1.3. GPIOBOE

**Register 7-2. GIOBOE (GPIO Port B Output Enable, 0x4007 0044)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0	Reserved

BIT	NAME	ACCESS	RESET	DESCRIPTION
7	Reserved	RW	0x0	Reserved, must be written to 0b
6	Reserved	RW	0x0	Reserved, must be written to 0b
5	Reserved	RW	0x0	Reserved, must be written to 0b
4	Reserved	RW	0x0	Reserved, must be written to 0b
3	Reserved	RW	0x0	Reserved, must be written to 0b
2	Reserved	RW	0x0	Reserved, must be written to 0b
1	Reserved	RW	0x0	Reserved, must be written to 0b
0	Reserved	RW	0x0	Reserved, must be written to 0b

#### 7.1.4. GPIOBDS

**Register 7-3. GPIOBDS (GPIO Port B Output Drive Strength, 0x4007 0048)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	Reserved	RW	0x0	Reserved, must be written to 0b
6	Reserved	RW	0x0	Reserved, must be written to 0b
5	Reserved	RW	0x0	Reserved, must be written to 0b
4	Reserved	RW	0x0	Reserved, must be written to 0b
3	Reserved	RW	0x0	Reserved, must be written to 0b
2	Reserved	RW	0x0	Reserved, must be written to 0b
1	Reserved	RW	0x0	Reserved, must be written to 0b
0	Reserved	RW	0x0	Reserved, must be written to 0b

#### 7.1.5. GPIOBRUP

**Register 7-4. GPIOBRUP (GPIO Port B Weak Pull Up, 0x4007 004C)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	Reserved	RW	0x0	Reserved, must be written to 0b
6	Reserved	RW	0x0	Reserved, must be written to 0b
5	Reserved	RW	0x0	Reserved, must be written to 0b
4	Reserved	RW	0x0	Reserved, must be written to 0b
3	Reserved	RW	0x0	Reserved, must be written to 0b
2	Reserved	RW	0x0	Reserved, must be written to 0b
1	Reserved	RW	0x0	Reserved, must be written to 0b
0	Reserved	RW	0x0	Reserved, must be written to 0b



### 7.1.6. GPIOBRDN

**Register 7-5. GPIOBRDN (GPIO Port B Weak Pull Down, 0x4007 0050)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	Reserved	RW	0x0	Reserved, must be written to 0b
6	Reserved	RW	0x0	Reserved, must be written to 0b
5	Reserved	RW	0x0	Reserved, must be written to 0b
4	Reserved	RW	0x0	Reserved, must be written to 0b
3	Reserved	RW	0x0	Reserved, must be written to 0b
2	Reserved	RW	0x0	Reserved, must be written to 0b
1	Reserved	RW	0x0	Reserved, must be written to 0b
0	Reserved	RW	0x0	Reserved, must be written to 0b

### 7.1.7. GPIOBI

**Register 7-6. GPIOBI (GPIO Port B Input, 0x4007 0054)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RW	0x0	Reserved
7	PB7I	R	0x0	Port B 7 input state 1b: input high 0b: input low
6	Reserved	R	0x0	Reserved
5	Reserved	R	0x0	Reserved
4	Reserved	R	0x0	Reserved
3	Reserved	R	0x0	Reserved
2	Reserved	R	0x0	Reserved
1	Reserved	R	0x0	Reserved
0	PB0I	R	0x0	Port B 0 input state 1b: input high 0b: input low

### 7.1.8. GPIOBPSEL

**Register 7-7. GPIOBPSEL (GPIO Port B Peripheral Select, 0x4007 005C)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	Reserved	RW	0x0	Reserved
15:14	PB7PSEL	RW	0x0	Port B 7 peripheral select 11b: reserved 10b: reserved 01b: reserved 00b: nIRQ2 / POS

BIT	NAME	ACCESS	RESET	DESCRIPTION
13:12	<b>PB6PSEL</b>	RW	0x0	Port B 6 peripheral select 11b: reserved 10b: reserved 01b: EMUXDATA 00b: reserved
11:10	<b>PB5PSEL</b>	RW	0x0	Port B 5 peripheral select 11b: reserved 10b: reserved 01b: EMUXCLK 00b: reserved
9:8	<b>PB4PSEL</b>	RW	0x0	Port B 4 peripheral select 11b: reserved 10b: reserved 01b: SOC Bus SCLK 00b: reserved
7:6	<b>PB3PSEL</b>	RW	0x0	Port B 3 peripheral select 11b: reserved 10b: reserved 01b: SOC Bus MOSI 00b: reserved
5:4	<b>PB2PSEL</b>	RW	0x0	Port B 2 peripheral select 11b: reserved 10b: reserved 01b: SOC Bus MISO 00b: reserved
3:2	<b>PB1PSEL</b>	RW	0x0	Port B 1 peripheral select 11b: reserved 10b: reserved 01b: SOC Bus CS 00b: reserved
1:0	<b>PB0PSEL</b>	RW	0x0	Port B 0 peripheral select 11b: reserved 10b: reserved 01b: reserved 00b: nIRQ1

### 7.1.9. GPIOBINTP

#### Register 7-8. GPGPIOBINTP (GPIO Port B Interrupt Polarity, 0x4007 0060)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RW	0x0	Reserved
7	<b>PB7INTP</b>	RW	0x0	Port B 7 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
6	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
5	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
4	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
3	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
2	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
1	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b

BIT	NAME	ACCESS	RESET	DESCRIPTION
0	PB0INTP	RW	0x0	Port B 0 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition

### 7.1.10. GPIOBINTEN

#### Register 7-9. GPIOBINTEN (GPIO Port B Interrupt Enable, 0x4007 0064)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RW	0x0	Reserved
7	PB7INTEN	RW	0x0	Port B 7 interrupt enable 1b: enabled interrupt 0b: disable interrupt
6	Reserved	RW	0x0	Reserved, must be written to 0b
5	Reserved	RW	0x0	Reserved, must be written to 0b
4	Reserved	RW	0x0	Reserved, must be written to 0b
3	Reserved	RW	0x0	Reserved, must be written to 0b
2	Reserved	RW	0x0	Reserved, must be written to 0b
1	Reserved	RW	0x0	Reserved, must be written to 0b
0	PB0INTEN	RW	0x0	Port B 0 interrupt enable 1b: enabled interrupt 0b: disable interrupt

### 7.1.11. GPIOBINT

#### Register 7-10. GPIOBINT (GPIO Port B Interrupt, 0x4007 0068)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RW	0x0	Reserved
7	PB7INT	RW	0x0	Port B 7 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
6	Reserved	RW	0x0	Reserved, must be written to 0b
5	Reserved	RW	0x0	Reserved, must be written to 0b
4	Reserved	RW	0x0	Reserved, must be written to 0b
3	Reserved	RW	0x0	Reserved, must be written to 0b
2	Reserved	RW	0x0	Reserved, must be written to 0b
1	Reserved	RW	0x0	Reserved, must be written to 0b
0	PB0INT	RW	0x0	Port B 0 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending

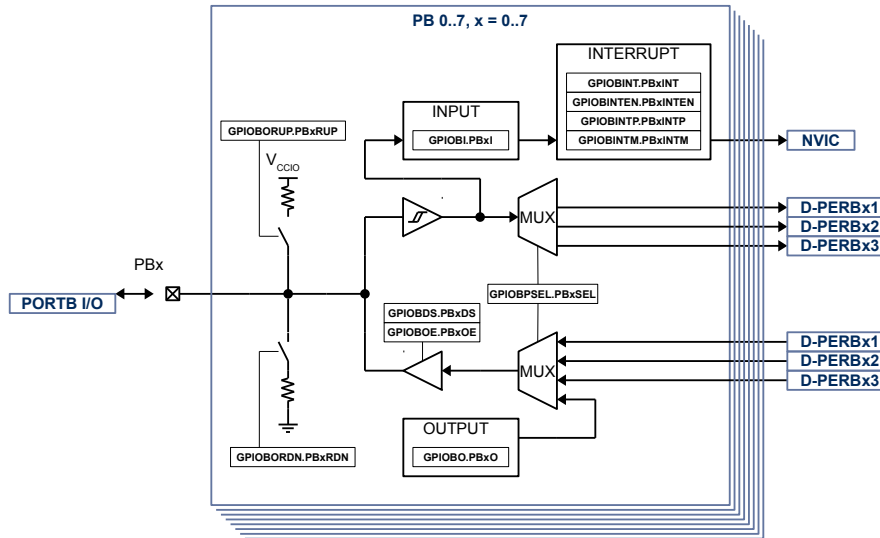
**7.1.12. GPIOBINTM**
**Register 7-11. GPIOBINTM (GPIO Port B Interrupt Mask, 0x4007 006C)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PB7INTM</b>	RW	0x0	Port B 7 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
6	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
5	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
4	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
3	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
2	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
1	<b>Reserved</b>	RW	0x0	Reserved, must be written to 0b
0	<b>PB0INTM</b>	RW	0x0	Port B 0 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask

## 7.2. Details of Operation

### 7.2.1. Block Diagram

Figure 7-1. GPIO Port B



### 7.2.2. Configuration

Following blocks need to be configured for correct use of the GPIO B:

- Nested Vectored Interrupt Controller (NVIC)
- SOC Bus

### 7.2.3. GPIO B Block

The GPIO B block consists of up to 8 general purpose input output (GPIO). Each GPIO has interrupt capabilities, weak pull-up or pull-down, programmable output drive strength, High-Z output operation. Some of the GPIO can be configured as PWM output, or capture and compare input.

### 7.2.4. Input

The input state of GPIOB can be monitored with **GPIOBI.PBxi**. The input state can be monitored regardless of the peripheral select setting **GPIOBPSSEL**.

### 7.2.5. Output and Output Enable

When **GPIOBOE.PBxOE** is enabled, the output state is controlled by **GPIOBO.PBxO**.

When **GPIOBOE.PBxOE** is disabled, the output is in High-Z state.

### 7.2.6. Output Drive Strength

The output drive strength can be adjusted using **GPIOBDS** to meet application needs. Set **GPIOBDS.PBxODS** to enable high current drive strength, reset to enable low current drive strength.

### 7.2.7. Weak Pull Up and Pull Down

Independent from the output settings, weak pull up can be enabled with **GPIOBORUP** and weak pull down can be enabled with **GPIOBRDN**.

#### NOTE:

**GPIOBORUP.PAxRUP** or **GPIOBRDN.PAxRDN** should never be enabled at the same time for a single GPIO. If switching from weak pull-up to weak pull-down is required, disable weak pull-up first before enable weak pull-down and vice versa.

### 7.2.8. Peripheral Select

Each GPIO is connected to up to 4 digital peripherals, selectable with **GPIOBPSEL**. When a different function than IO is selected the input state can still be read with **GPIOBI** and the pull-up and pull-down is still controllable.

### 7.2.9. Interrupt

The interrupt for each GPIO can be enabled with **GPIOBINTEN**. The interrupt can be configured to be rising signal edge or falling signal edge using **GPIOBINTP**. The state of the interrupt can be read from **GPIOBINT**. The individual interrupt bits can be cleared by writing to 1.

When the GPIO interrupts are enabled for the first time after device start-up, it may be in an uncertain state and generate an interrupt. To avoid this the **GPIOBINTM** mask bit need to be set before enabled interrupt bits.

To allow interrupt to be recognized by the CPU the GPIO interrupt need also be enabled in the NVIC.

## 8. GPIO PORT C

### 8.1. Register

#### 8.1.1. Register Map

Table 8-1. GPIO Port C Register Map

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>GPIO Port C</b>			
0x4008 0000	<b>GPIOCO</b>	GPIO Port C output	0x0000 0000
0x4008 0004	<b>GPIOCOE</b>	GPIO Port C output enable	0x0000 0000
0x4008 0008	<b>Reserved</b>	Reserved	0x0000 0000
0x4008 000C	<b>Reserved</b>	Reserved	0x0000 0000
0x4008 0010	<b>Reserved</b>	Reserved	0x0000 0000
0x4008 0014	<b>GPIOCI</b>	GPIO Port C input	0x0000 0000
0x4008 0018	<b>GPIOCIE</b>	GPIO Port C input enable	0x0000 0000
0x4008 001C	<b>Reserved</b>	Reserved	0x0000 0000
0x4008 0020	<b>GPIOCINTP</b>	GPIO Port C interrupt polarity select	0x0000 0000
0x4008 0024	<b>GPIOCINTEN</b>	GPIO Port C interrupt enable select	0x0000 0000
0x4008 0028	<b>GPIOCINT</b>	GPIO Port C interrupt flag	0x0000 0000
0x4008 002C	<b>GPIOCINTM</b>	GPIO Port C interrupt mask	0x0000 0000

#### 8.1.2. GPIOCO

Register 8-1. GPIOCO (GPIO Port C Output, 0x4008 0000)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PC7O</b>	RW	0x0	Port C output 7 1b: set output high if <b>GPIOCOE.PCOE7</b> = 1b 0b: set output low if <b>GPIOCOE.PCOE7</b> = 1b
6	<b>PC6O</b>	RW	0x0	Port C output 6 1b: set output high if <b>GPIOCOE.PCOE6</b> = 1b 0b: set output low if <b>GPIOCOE.PCOE6</b> = 1b
5	<b>PC5O</b>	RW	0x0	Port C output 5 1b: set output high if <b>GPIOCOE.PCOE5</b> = 1b 0b: set output low if <b>GPIOCOE.PCOE5</b> = 1b
4	<b>PC4O</b>	RW	0x0	Port C output 4 1b: set output high if <b>GPIOCOE.PCOE4</b> = 1b 0b: set output low if <b>GPIOCOE.PCOE4</b> = 1b
3	<b>PC3O</b>	RW	0x0	Port C output 3 1b: set output high if <b>GPIOCOE.PCOE3</b> = 1b 0b: set output low if <b>GPIOCOE.PCOE3</b> = 1b
2	<b>PC2O</b>	RW	0x0	Port C output 2 1b: set output high if <b>GPIOCOE.PCOE2</b> = 1b 0b: set output low if <b>GPIOCOE.PCOE2</b> = 1b

BIT	NAME	ACCESS	RESET	DESCRIPTION
1	PC1O	RW	0x0	Port C output 1 1b: set output high if <b>GPIOCOE.PCOE1</b> = 1b 0b: set output low if <b>GPIOCOE.PCOE1</b> = 1b
0	PC0O	RW	0x0	Port C output 0 1b: set output high if <b>GPIOCOE.PCOE0</b> = 1b 0b: set output low if <b>GPIOCOE.PCOE0</b> = 1b

### 8.1.3. GPIOCOE

#### Register 8-2. GPIOCOE (GPIO Port C Output Enable, 0x4008 0004)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0	Reserved
7	PC7OE	RW	0x0	Port C output 7 enable 1b: output state set by <b>GPIOCO.PC07</b> 0b: output disabled, high-impedance state
6	PC6OE	RW	0x0	Port C output 6 enable 1b: output state set by <b>GPIOCO.PC06</b> 0b: output disabled, high-impedance state
5	PC5OE	RW	0x0	Port C output 5 enable 1b: output state set by <b>GPIOCO.PC05</b> 0b: output disabled, high-impedance state
4	PC4OE	RW	0x0	Port C output 4 enable 1b: output state set by <b>GPIOCO.PC04</b> 0b: output disabled, high-impedance state
3	PC3OE	RW	0x0	Port C output 3 enable 1b: output state set by <b>GPIOCO.PC03</b> 0b: output disabled, high-impedance state
2	PC2OE	RW	0x0	Port C output 2 enable 1b: output state set by <b>GPIOCO.PC02</b> 0b: output disabled, high-impedance state
1	PC1OE	RW	0x0	Port C output 1 enable 1b: output state set by <b>GPIOCO.PC01</b> 0b: output disabled, high-impedance state
0	PC0OE	RW	0x0	Port C output 0 enable 1b: output state set by <b>GPIOCO.PC00</b> 0b: output disabled, high-impedance state

### 8.1.4. GPIOCI

#### Register 8-3. GPIOCI (GPIO Port C Input, 0x4008 0018)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PC7I	RW	0x0	Port C 7 input state 1b: input high 0b: input low
6	PC6I	RW	0x0	Port C 6 input state 1b: input high 0b: input low



BIT	NAME	ACCESS	RESET	DESCRIPTION
5	<b>PC5I</b>	RW	0x0	Port C 5 input state 1b: input high 0b: input low
4	<b>PC4I</b>	RW	0x0	Port C 4 input state 1b: input high 0b: input low
3	<b>PC3I</b>	RW	0x0	Port C 3 input state 1b: input high 0b: input low
2	<b>PC2I</b>	RW	0x0	Port C 2 input state 1b: input high 0b: input low
1	<b>PC1I</b>	RW	0x0	Port C 1 input state 1b: input high 0b: input low
0	<b>PC0I</b>	RW	0x0	Port C 0 input state 1b: input high 0b: input low

### 8.1.5. GPIOCIE

#### Register 8-4. GPIOCIE (GPIO Port C Input Enable, 0x4008 0014)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PC7IE</b>	RW	0x0	Port C 7 input enable 1b: input enabled, for I/O operation 0b: input disabled
6	<b>PC6IE</b>	RW	0x0	Port C 6 input enable 1b: input enabled, for I/O operation 0b: input disabled
5	<b>PC5IE</b>	RW	0x0	Port C 5 input enable 1b: input enabled, for I/O operation 0b: input disabled, for AD5 ADC input operation
4	<b>PC4IE</b>	RW	0x0	Port C 4 input enable 1b: input enabled, for I/O operation 0b: input disabled, for AD4 ADC input operation
3	<b>PC3IE</b>	RW	0x0	Port C 3 input enable 1b: input enabled, for I/O operation 0b: input disabled, for AD3 ADC input operation
2	<b>PC2IE</b>	RW	0x0	Port C 2 input enable 1b: input enabled, for I/O operation 0b: input disabled, for AD2 ADC input operation
1	<b>PC1IE</b>	RW	0x0	Port C 1 input enable 1b: input enabled, for I/O operation 0b: input disabled, for AD1 ADC input operation
0	<b>PC0IE</b>	RW	0x0	Port C 0 input enable 1b: input enabled, for I/O operation 0b: input disabled, for AD0 ADC input operation

### 8.1.6. GPIOCINTP

#### Register 8-5. GPIOCINTP (GPIO Port C Interrupt Polarity, 0x4008 0020)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PC7INTP	RW	0x0	Port C 7 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
6	PC6INTP	RW	0x0	Port C 6 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
5	PC5INTP	RW	0x0	Port C 5 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
4	PC4INTP	RW	0x0	Port C 4 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
3	PC3INTP	RW	0x0	Port C 3 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
2	PC2INTP	RW	0x0	Port C 2 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
1	PC1INTP	RW	0x0	Port C 1 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
0	PC0INTP	RW	0x0	Port C 0 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition

### 8.1.7. GPIOCINTEN

#### Register 8-6. GPIOCINTEN (GPIO Port C Interrupt Enable, 0x4008 0024)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PC7INTEN	RW	0x0	Port C 7 interrupt enable 1b: enabled interrupt 0b: disable interrupt
6	PC6INTEN	RW	0x0	Port C 6 interrupt enable 1b: enabled interrupt 0b: disable interrupt
5	PC5INTEN	RW	0x0	Port C 5 interrupt enable 1b: enabled interrupt 0b: disable interrupt
4	PC4INTEN	RW	0x0	Port C 4 interrupt enable 1b: enabled interrupt 0b: disable interrupt
3	PC3INTEN	RW	0x0	Port C 3 interrupt enable 1b: enabled interrupt 0b: disable interrupt
2	PC2INTEN	RW	0x0	Port C 2 interrupt enable 1b: enabled interrupt 0b: disable interrupt

BIT	NAME	ACCESS	RESET	DESCRIPTION
1	<b>PC1INTEN</b>	RW	0x0	Port C 1 interrupt enable 1b: enabled interrupt 0b: disable interrupt
0	<b>PC0INTEN</b>	RW	0x0	Port C 0 interrupt enable 1b: enabled interrupt 0b: disable interrupt

### 8.1.8. GPIOCINT

#### Register 8-7. GPIOCINT (GPIO Port C Interrupt, 0x4008 0028)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PC7INT</b>	RW	0x0	Port C 7 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
6	<b>PC6INT</b>	RW	0x0	Port C 6 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
5	<b>PC5INT</b>	RW	0x0	Port C 5 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
4	<b>PC4INT</b>	RW	0x0	Port C 4 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
3	<b>PC3INT</b>	RW	0x0	Port C 3 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
2	<b>PC2INT</b>	RW	0x0	Port C 2 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
1	<b>PC1INT</b>	RW	0x0	Port C 1 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
0	<b>PC0INT</b>	RW	0x0	Port C 0 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending

### 8.1.9. GPIOCINTM

#### Register 8-8. GPIOCINTM (GPIO Port C Interrupt Mask, 0x4008 002C)

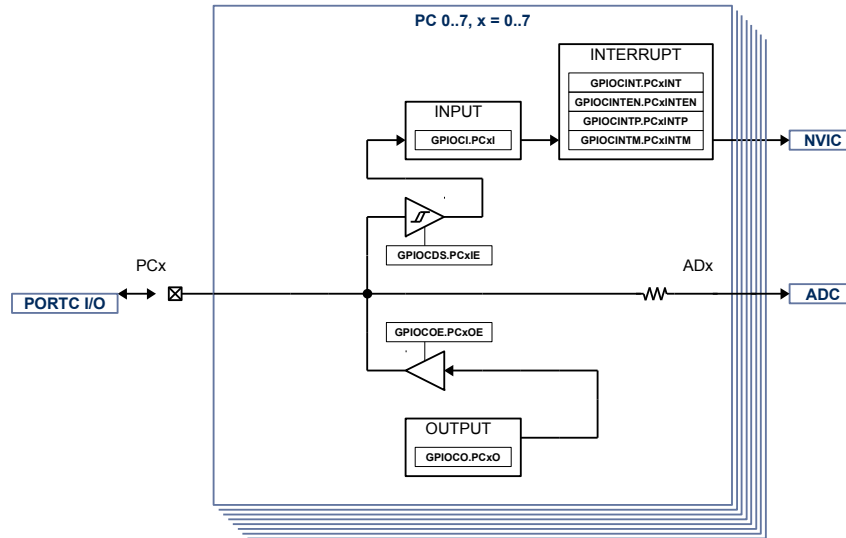
BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PC7INTM</b>	RW	0x0	Port C 7 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
6	<b>PC6INTM</b>	RW	0x0	Port C 6 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask

BIT	NAME	ACCESS	RESET	DESCRIPTION
5	<b>PC5INTM</b>	RW	0x0	Port C 5 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
4	<b>PC4INTM</b>	RW	0x0	Port C 4 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
3	<b>PC3INTM</b>	RW	0x0	Port C 3 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
2	<b>PC2INTM</b>	RW	0x0	Port C 2 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
1	<b>PC1INTM</b>	RW	0x0	Port C 1 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
0	<b>PC0INTM</b>	RW	0x0	Port C 0 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask

## 8.2. Details of Operation

### 8.2.1. Block Diagram

Figure 8-1. GPIO Port C



### 8.2.2. Configuration

Following blocks need to be configured for correct use of the GPIO C:

- Nested Vectored Interrupt Controller (NVIC)
- ADC

### 8.2.3. GPIO C Block

The GPIOC block consists of up to 8 general purpose input output (GPIO). Each GPIO has interrupt capabilities, High-Z output operation, analog ADx input to the ADC Mux. The GPIOC block IO voltage is different from the other GPIO blocks, it is supplied from VCC33.

### 8.2.4. Analog Input

The digital input state of GPIOC can be monitored with **GPIOCI.PCxI** if **GPIOCIE.PCxIE** is set.

Clear **GPIOCIE.PCxIE** and **GPIOCOE.PCxOE** to disable digital input and output to allow use of analog input ADx to the ADC.

### 8.2.5. Output and Output Enable

When **GPIOCOE.PCxOE** is enabled, the output state is controlled by **GPIOCO.PCxO**.

When **GPIOCOE.PCxOE** is disabled, the output is in High-Z state.

### 8.2.6. Interrupt

The interrupt for each GPIO can be enabled with **GPIOCINTEN**. The interrupt can be configured to be rising signal edge or falling signal edge using **GPIOCINTP**. The state of the interrupt can be read from **GPIOCINT**. The individual interrupt bits can be cleared by writing to 1.

When the GPIO interrupts are enabled for the first time after device start-up, it may be in an uncertain state and generate an interrupt. To avoid this the **GPIOCINTM** mask bit need to be set before enabled interrupt bits.

To allow interrupt to be recognized by the CPU the GPIO interrupt need also be enabled in the NVIC.

## 9. GPIO PORT D

### 9.1. Register

#### 9.1.1. Register Map

Table 9-1. GPIO Port D Register Map

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>GPIO Port D</b>			
0x4008 0040	<b>GPIODO</b>	GPIO Port D output	0x0000 0000
0x4008 0044	<b>GPIODOE</b>	GPIO Port D output enable	0x0000 0000
0x4008 0048	<b>GPIODODS</b>	GPIO Port D output drive strength	0x0000 0000
0x4008 004C	<b>GPIODORUP</b>	GPIO Port D output weak pull up	0x0000 0000
0x4008 0050	<b>GPIODORDN</b>	GPIO Port D output weak pull down	0x0000 0000
0x4008 0054	<b>GPIODI</b>	GPIO Port D input	0x0000 0000
0x4008 0058	<b>Reserved</b>	Reserved	0x0000 0000
0x4008 005C	<b>GPIODPSEL</b>	GPIO Port D peripheral select	0x0000 0005
0x4008 0060	<b>GPIODINTP</b>	GPIO Port D interrupt polarity select	0x0000 0000
0x4008 0064	<b>GPIODINTEN</b>	GPIO Port D interrupt enable select	0x0000 0000
0x4008 0068	<b>GPIODINT</b>	GPIO Port D interrupt flag	0x0000 0000
0x4008 006C	<b>GPIODINTM</b>	GPIO Port D interrupt mask	0x0000 0000

#### 9.1.2. GPIODO

Register 9-1. GPIODO (GPIO Port D Output, 0x4008 0040)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PD7O</b>	RW	0x0	Port D output 7 1b: set output high if <b>GPIODOE.PDOE7</b> = 1b 0b: set output low if <b>GPIODOE.PDOE7</b> = 1b
6	<b>PD6O</b>	RW	0x0	Port D output 6 1b: set output high if <b>GPIODOE.PDOE6</b> = 1b 0b: set output low if <b>GPIODOE.PDOE6</b> = 1b
5	<b>PD5O</b>	RW	0x0	Port D output 5 1b: set output high if <b>GPIODOE.PDOE5</b> = 1b 0b: set output low if <b>GPIODOE.PDOE5</b> = 1b
4	<b>PD4O</b>	RW	0x0	Port D output 4 1b: set output high if <b>GPIODOE.PDOE4</b> = 1b 0b: set output low if <b>GPIODOE.PDOE4</b> = 1b
3	<b>PD3O</b>	RW	0x0	Port D output 3 1b: set output high if <b>GPIODOE.PDOE3</b> = 1b 0b: set output low if <b>GPIODOE.PDOE3</b> = 1b
2	<b>PD2O</b>	RW	0x0	Port D output 2 1b: set output high if <b>GPIODOE.PDOE2</b> = 1b 0b: set output low if <b>GPIODOE.PDOE2</b> = 1b

BIT	NAME	ACCESS	RESET	DESCRIPTION
1	PD1O	RW	0x0	Port D output 1 1b: set output high if <b>GPIODOE.PDOE1</b> = 1b 0b: set output low if <b>GPIODOE.PDOE1</b> = 1b
0	PD0O	RW	0x0	Port D output 0 1b: set output high if <b>GPIODOE.PDOE0</b> = 1b 0b: set output low if <b>GPIODOE.PDOE0</b> = 1b

### 9.1.3. GPIODOE

#### Register 9-2. GPIODOE (GPIO Port D Output Enable, 0x4008 0044)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PD7OE	RW	0x0	Port D output 7 enable 1b: output state set by <b>GPIODO.PDO7</b> 0b: output disabled, high-impedance state
6	PD6OE	RW	0x0	Port D output 6 enable 1b: output state set by <b>GPIODO.PDO6</b> 0b: output disabled, high-impedance state
5	PD5OE	RW	0x0	Port D output 5 enable 1b: output state set by <b>GPIODO.PDO5</b> 0b: output disabled, high-impedance state
4	PD4OE	RW	0x0	Port D output 4 enable 1b: output state set by <b>GPIODO.PDO4</b> 0b: output disabled, high-impedance state
3	PD3OE	RW	0x0	Port D output 3 enable 1b: output state set by <b>GPIODO.PDO3</b> 0b: output disabled, high-impedance state
2	PD2OE	RW	0x0	Port D output 2 enable 1b: output state set by <b>GPIODO.PDO2</b> 0b: output disabled, high-impedance state
1	PD1OE	RW	0x0	Port D output 1 enable 1b: output state set by <b>GPIODO.PDO1</b> 0b: output disabled, high-impedance state
0	PD0OE	RW	0x0	Port D output 0 enable 1b: output state set by <b>GPIODO.PDO0</b> 0b: output disabled, high-impedance state

### 9.1.4. GPIODDS

#### Register 9-3. GPIODDS (GPIO Port D Output Drive Strength, 0x4008 0048)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PD7ODS	RW	0x0	Port D output 7 drive strength select 1b: high 0b: low
6	PD6ODS	RW	0x0	Port D output 6 drive strength select 1b: high 0b: low



BIT	NAME	ACCESS	RESET	DESCRIPTION
5	PD5ODS	RW	0x0	Port D output 5 drive strength select 1b: high 0b: low
4	PD4ODS	RW	0x0	Port D output 4 drive strength select 1b: high 0b: low
3	PD3ODS	RW	0x0	Port D output 3 drive strength select 1b: high 0b: low
2	PD2ODS	RW	0x0	Port D output 2 drive strength select 1b: high 0b: low
1	PD1ODS	RW	0x0	Port D output 1 drive strength select 1b: high 0b: low
0	PD0ODS	RW	0x0	Port D output 0 drive strength select 1b: high 0b: low

### 9.1.5. GPIODRUP

#### Register 9-4. GPIODRUP (GPIO Port D Weak Pull Up, 0x4008 004C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PD7UP	RW	0x0	Port D 7 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
6	PD6UP	RW	0x0	Port D 6 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
5	PD5UP	RW	0x0	Port D 5 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
4	PD4UP	RW	0x0	Port D 4 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
3	PD3UP	RW	0x0	Port D 3 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
2	PD2UP	RW	0x0	Port D 2 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
1	PD1UP	RW	0x0	Port D 1 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
0	PD0UP	RW	0x0	Port D 0 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO

### 9.1.6. GPIODRDN

**Register 9-5. GPIODRDN (GPIO Port D Weak Pull Down, 0x4008 0050)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PD7RDN	RW	0x0	Port D 7 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
6	PD6RDN	RW	0x0	Port D 6 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
5	PD5RDN	RW	0x0	Port D 5 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
4	PD4RDN	RW	0x0	Port D 4 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
3	PD3RDN	RW	0x0	Port D 3 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
2	PD2RDN	RW	0x0	Port D 2 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
1	PD1RDN	RW	0x0	Port D 1 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
0	PD0RDN	RW	0x0	Port D 0 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS

### 9.1.7. GPIODI

**Register 9-6. GPIODI (GPIO Port D Input, 0x4008 0054)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RW	0x0	Reserved
7	PD7I	RW	0x0	Port D 7 input state 1b: input high 0b: input low
6	PD6I	RW	0x0	Port D 6 input state 1b: input high 0b: input low
5	PD5I	RW	0x0	Port D 5 input state 1b: input high 0b: input low
4	PD4I	RW	0x0	Port D 4 input state 1b: input high 0b: input low
3	PD3I	RW	0x0	Port D 3 input state 1b: input high 0b: input low
2	PD2I	RW	0x0	Port D 2 input state 1b: input high 0b: input low

BIT	NAME	ACCESS	RESET	DESCRIPTION
1	<b>PD1I</b>	RW	0x0	Port D 1 input state 1b: input high 0b: input low
0	<b>PD0I</b>	RW	0x0	Port D 0 input state 1b: input high 0b: input low

### 9.1.8. GPIODPSEL

#### Register 9-7. GPIODPSEL (GPIO Port D Peripheral Select, 0x4008 005C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:14	<b>PD7PSEL</b>	RW	0x0	Port D 7 peripheral select 11b: reserved 10b: PWMD0 / DTGD0LS output or CD0 capture and compare input 01b: PWMA6 / DTGA2HS output or CA6 capture and compare input 00b: I/O mode PD7
13:12	<b>PD6PSEL</b>	RW	0x0	Port D 6 peripheral select 11b: reserved 10b: PWMB1 / DTGB0HS output or CB1 capture and compare input 01b: PWMA7 / DTGA3HS output or CA7 capture and compare input 00b: I/O mode PD6
11:10	<b>PD5PSEL</b>	RW	0x0	Port D 5 peripheral select 11b: reserved 10b: PWMC1 / DTGC0HS output or CC1 capture and compare input 01b: PWMA5 / DTGA1HS output or CA5 capture and compare input 00b: I/O mode PD5
9:8	<b>PD4PSEL</b>	RW	0x0	Port D 4 peripheral select 11b: reserved 10b: reserved 01b: PWMD1 / DTGD0HS output or CD1 capture and compare input 00b: I/O mode PD4
7:6	<b>PD3PSEL</b>	RW	0x0	Port D 3 peripheral select 11b: PWMB1 / DTGB0HS output or CB1 capture and compare input 10b: PWMA7 / DTGA3HS output or CA7 capture and compare input 01b: PWMA5 / DTGA1HS output or CA5 capture and compare input 00b: I/O mode PD3
5:4	<b>PD2PSEL</b>	RW	0x0	Port D 2 peripheral select 11b: PWMB0 / DTGB0LS output or CB0 capture and compare input 10b: PWMA4 / DTGA0HS output or CA4 capture and compare input 01b: PWMA3 / DTGA3LS output or CA3 capture and compare input 00b: I/O mode PD2
3:2	<b>PD1PSEL</b>	RW	0x1	Port D 1 peripheral select 11b: reserved 10b: EXTCLK input 01b: Serial wire debug SWDCLK 00b: I/O mode PD1
1:0	<b>PD0PSEL</b>	RW	0x1	Port D 0 peripheral select 11b: reserved 10b: reserved 01b: Serial wire debug SWDDATA 00b: I/O mode PD0

### 9.1.9. GPIODINTP

**Register 9-8. GPIODINTP (GPIO Port D Interrupt Polarity, 0x4008 0060)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PD7INTP	RW	0x0	Port D 7 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
6	PD6INTP	RW	0x0	Port D 6 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
5	PD5INTP	RW	0x0	Port D 5 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
4	PD4INTP	RW	0x0	Port D 4 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
3	PD3INTP	RW	0x0	Port D 3 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
2	PD2INTP	RW	0x0	Port D 2 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
1	PD1INTP	RW	0x0	Port D 1 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
0	PD0INTP	RW	0x0	Port D 0 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition

### 9.1.10. GPIODINTEN

**Register 9-9. GPIODINTEN (GPIO Port D Interrupt Enable, 0x4008 0064)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PD7INTEN	RW	0x0	Port D 7 interrupt enable 1b: enabled interrupt 0b: disable interrupt
6	PD6INTEN	RW	0x0	Port D 6 interrupt enable 1b: enabled interrupt 0b: disable interrupt
5	PD5INTEN	RW	0x0	Port D 5 interrupt enable 1b: enabled interrupt 0b: disable interrupt
4	PD4INTEN	RW	0x0	Port D 4 interrupt enable 1b: enabled interrupt 0b: disable interrupt
3	PD3INTEN	RW	0x0	Port D 3 interrupt enable 1b: enabled interrupt 0b: disable interrupt
2	PD2INTEN	RW	0x0	Port D 2 interrupt enable 1b: enabled interrupt 0b: disable interrupt

BIT	NAME	ACCESS	RESET	DESCRIPTION
1	<b>PD1INTEN</b>	RW	0x0	Port D 1 interrupt enable 1b: enabled interrupt 0b: disable interrupt
0	<b>PD0INTEN</b>	RW	0x0	Port D 0 interrupt enable 1b: enabled interrupt 0b: disable interrupt

### 9.1.11. GPIODINT

#### Register 9-10. GPIODINT (GPIO Port D Interrupt, 0x4008 0068)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PD7INT</b>	RW	0x0	Port D 7 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
6	<b>PD6INT</b>	RW	0x0	Port D 6 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
5	<b>PD5INT</b>	RW	0x0	Port D 5 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
4	<b>PD4INT</b>	RW	0x0	Port D 4 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
3	<b>PD3INT</b>	RW	0x0	Port D 3 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
2	<b>PD2INT</b>	RW	0x0	Port D 2 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
1	<b>PD1INT</b>	RW	0x0	Port D 1 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
0	<b>PD0INT</b>	RW	0x0	Port D 0 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending

### 9.1.12. GPIODINTM

#### Register 9-11. GPIODINTM (GPIO Port D Interrupt Mask, 0x4008 006C)

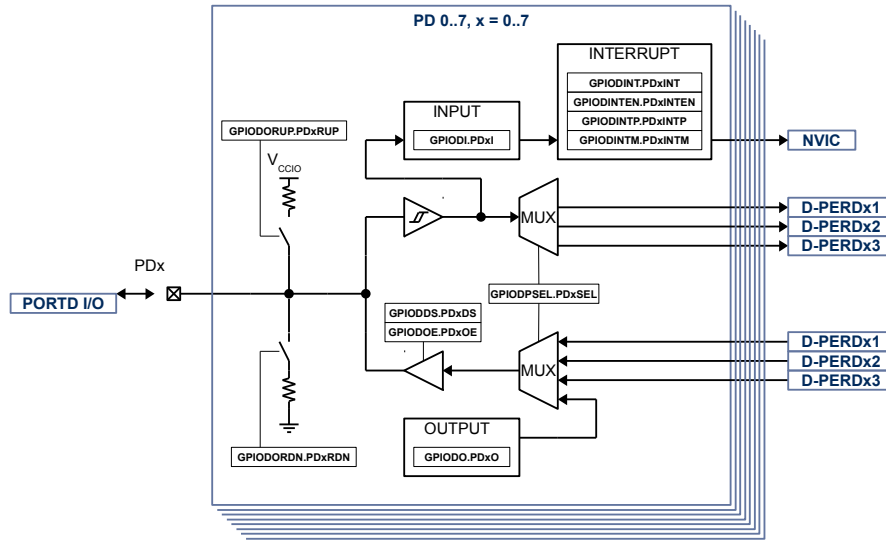
BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PD7INTM</b>	RW	0x0	Port D 7 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
6	<b>PD6INTM</b>	RW	0x0	Port D 6 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask

BIT	NAME	ACCESS	RESET	DESCRIPTION
5	<b>PD5INTM</b>	RW	0x0	Port D 5 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
4	<b>PD4INTM</b>	RW	0x0	Port D 4 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
3	<b>PD3INTM</b>	RW	0x0	Port D 3 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
2	<b>PD2INTM</b>	RW	0x0	Port D 2 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
1	<b>PD1INTM</b>	RW	0x0	Port D 1 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
0	<b>PD0INTM</b>	RW	0x0	Port D 0 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask

## 9.2. Details of Operation

### 9.2.1. Block Diagram

Figure 9-1. GPIO Port D



### 9.2.2. Configuration

Following blocks need to be configured for correct use of the GPIO D:

- Nested Vectored Interrupt Controller (NVIC)
- Gate Driver
- Timer A, PWMA, DTGA
- Timer B, PWMB, DTGB
- Timer C, PWMC, DTGC
- Timer D, PWMD, DTGD
- CCS
- SWD Debugger

### 9.2.3. GPIO D Block

The GPIO D block consists of up to 8 general purpose input output (GPIO). Each GPIO has interrupt capabilities, weak pull-up or pull-down, programmable output drive strength, High-Z output operation. Some of the GPIO can be configured as PWM output, or capture and compare input.

### 9.2.4. Input

The input state of GPIOD can be monitored with **GPIODI.PDxi**. The input state can be monitored regardless of the peripheral select setting **GPIODPSEL**.

### 9.2.5. Output and Output Enable

When **GPIODOE.PDxOE** is enabled, the output state is controlled by **GPIODO.PDxiO**.

When **GPIODOE.PDxOE** is disabled, the output is in High-Z state.

#### 9.2.6. Output Drive Strength

The output drive strength can be adjusted using **GPIODDS** to meet application needs. Set **GPIODDS.PDxODS** to enable high current drive strength, reset to enable low current drive strength.

#### 9.2.7. Weak Pull Up and Pull Down

Independent from the output settings, weak pull up can be enabled with **GPIODORUP** and weak pull down can be enabled with **GPIODRDN**.

#### NOTE:

**GPIODRUP.PDxRUP** or **GPIODRDN.PDxRDN** should never be enabled at the same time for a single GPIO. If switching from weak pull-up to weak pull-down is required, disable weak pull-up first before enable weak pull-down and vice versa.

#### 9.2.8. Peripheral Select

Each GPIO is connected to up to 4 digital peripherals, selectable with **GPIODPSEL**. When a different function than IO is selected the input state can still be read with **GPIODI** and the pull-up and pull-down is still controllable.

#### 9.2.9. Interrupt

The interrupt for each GPIO can be enabled with **GPIODINTEN**. The interrupt can be configured to be rising signal edge or falling signal edge using **GPIODINTP**. The state of the interrupt can be read from **GPIODINT**. The individual interrupt bits can be cleared by writing to 1.

When the GPIO interrupts are enabled for the first time after device start-up, it may be in an uncertain state and generate an interrupt. To avoid this the **GPIODINTM** mask bit need to be set before enabled interrupt bits.

To allow interrupt to be recognized by the CPU the GPIO interrupt need also be enabled in the NVIC.



## 10. GPIO PORT E

### 10.1. Register

#### 10.1.1. Register Map

**Table 10-1. GPIO Port E Register Map**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>GPIO Port E</b>			
0x4009 0000	<b>GPIOEO</b>	GPIO Port E output	0x0000 0000
0x4009 0004	<b>GPIOEOE</b>	GPIO Port E output enable	0x0000 0000
0x4009 0008	<b>GPIOEODS</b>	GPIO Port E output drive strength	0x0000 0000
0x4009 000C	<b>GPIOEORUP</b>	GPIO Port E output weak pull up	0x0000 0000
0x4009 0010	<b>GPIOEORDN</b>	GPIO Port E output weak pull down	0x0000 0000
0x4009 0014	<b>GPIOEI</b>	GPIO Port E input	0x0000 0000
0x4009 0018	<b>Reserved</b>	Reserved	0x0000 0000
0x4009 001C	<b>GPIOEPSSEL</b>	GPIO Port E peripheral select	0x0000 0000
0x4009 0020	<b>GPIOEINTP</b>	GPIO Port E interrupt polarity select	0x0000 0000
0x4009 0024	<b>GPIOEINTEN</b>	GPIO Port E interrupt enable select	0x0000 0000
0x4009 0028	<b>GPIOEINT</b>	GPIO Port E interrupt flag	0x0000 0000
0x4009 002C	<b>GPIOEINTM</b>	GPIO Port E interrupt mask	0x0000 0000

#### 10.1.2. GPIOEO

**Register 10-1. GPIOEO (GPIO Port E Output, 0x4009 0000)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>PE7O</b>	RW	0x0	Port E output 7 1b: set output high if <b>GPIOEOE.PEOE7</b> = 1b 0b: set output low if <b>GPIOEOE.PEOE7</b> = 1b
6	<b>PE6O</b>	RW	0x0	Port E output 6 1b: set output high if <b>GPIOEOE.PEOE6</b> = 1b 0b: set output low if <b>GPIOEOE.PEOE6</b> = 1b
5	<b>PE5O</b>	RW	0x0	Port E output 5 1b: set output high if <b>GPIOEOE.PEOE5</b> = 1b 0b: set output low if <b>GPIOEOE.PEOE5</b> = 1b
4	<b>PE4O</b>	RW	0x0	Port E output 4 1b: set output high if <b>GPIOEOE.PEOE4</b> = 1b 0b: set output low if <b>GPIOEOE.PEOE4</b> = 1b
3	<b>PE3O</b>	RW	0x0	Port E output 3 1b: set output high if <b>GPIOEOE.PEOE3</b> = 1b 0b: set output low if <b>GPIOEOE.PEOE3</b> = 1b
2	<b>PE2O</b>	RW	0x0	Port E output 2 1b: set output high if <b>GPIOEOE.PEOE2</b> = 1b 0b: set output low if <b>GPIOEOE.PEOE2</b> = 1b

BIT	NAME	ACCESS	RESET	DESCRIPTION
1	PE1O	RW	0x0	Port E output 1 1b: set output high if GPIOEOE.PEOE1 = 1b 0b: set output low if GPIOEOE.PEOE1 = 1b
0	PE0O	RW	0x0	Port E output 0 1b: set output high if GPIOEOE.PEOE0 = 1b 0b: set output low if GPIOEOE.PEOE0 = 1b

### 10.1.3. GPIOEOE

#### Register 10-2. GPIOEOE (GPIO Port E Output Enable, 0x4009 0004)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0	Reserved
7	PE7OE	RW	0x0	Port E output 7 enable 1b: output state set by GPIOEO.PE07 0b: output disabled, high-impedance state
6	PE6OE	RW	0x0	Port E output 6 enable 1b: output state set by GPIOEO.PE06 0b: output disabled, high-impedance state
5	PE5OE	RW	0x0	Port E output 5 enable 1b: output state set by GPIOEO.PE05 0b: output disabled, high-impedance state
4	PE4OE	RW	0x0	Port E output 4 enable 1b: output state set by GPIOEO.PE04 0b: output disabled, high-impedance state
3	PE3OE	RW	0x0	Port E output 3 enable 1b: output state set by GPIOEO.PE03 0b: output disabled, high-impedance state
2	PE2OE	RW	0x0	Port E output 2 enable 1b: output state set by GPIOEO.PE02 0b: output disabled, high-impedance state
1	PE1OE	RW	0x0	Port E output 1 enable 1b: output state set by GPIOEO.PE01 0b: output disabled, high-impedance state
0	PE0OE	RW	0x0	Port E output 0 enable 1b: output state set by GPIOEO.PE00 0b: output disabled, high-impedance state

### 10.1.4. GPIOEDS

#### Register 10-3. GPIOEDS (GPIO Port E Output Drive Strength, 0x4009 0008)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PE7ODS	RW	0x0	Port E output 7 drive strength select 1b: high 0b: low
6	PE6ODS	RW	0x0	Port E output 6 drive strength select 1b: high 0b: low

BIT	NAME	ACCESS	RESET	DESCRIPTION
5	PE5ODS	RW	0x0	Port E output 5 drive strength select 1b: high 0b: low
4	PE4ODS	RW	0x0	Port E output 4 drive strength select 1b: high 0b: low
3	PE3ODS	RW	0x0	Port E output 3 drive strength select 1b: high 0b: low
2	PE2ODS	RW	0x0	Port E output 2 drive strength select 1b: high 0b: low
1	PE1ODS	RW	0x0	Port E output 1 drive strength select 1b: high 0b: low
0	PE0ODS	RW	0x0	Port E output 0 drive strength select 1b: high 0b: low

### 10.1.5. GPIOERUP

#### Register 10-4. GPIOERUP (GPIO Port E Weak Pull Up, 0x4009 000C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PE7RUP	RW	0x0	Port E 7 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
6	PE6RUP	RW	0x0	Port E 6 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
5	PE5RUP	RW	0x0	Port E 5 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
4	PE4RUP	RW	0x0	Port E 4 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
3	PE3RUP	RW	0x0	Port E 3 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
2	PE2RUP	RW	0x0	Port E 2 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
1	PE1RUP	RW	0x0	Port E 1 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO
0	PE0RUP	RW	0x0	Port E 0 weak pull up select 1b: enable weak pull-up to VCCIO 0b: disable weak pull-up to VCCIO

### 10.1.6. GPIOERDN

#### Register 10-5. GPIOERDN (GPIO Port E Weak Pull Down, 0x4009 0010)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PE7RDN	RW	0x0	Port E 7 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
6	PE6RDN	RW	0x0	Port E 6 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
5	PE5RDN	RW	0x0	Port E 5 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
4	PE4RDN	RW	0x0	Port E 4 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
3	PE3RDN	RW	0x0	Port E 3 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
2	PE2RDN	RW	0x0	Port E 2 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
1	PE1RDN	RW	0x0	Port E 1 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS
0	PE0RDN	RW	0x0	Port E 0 weak pull down select 1b: enable weak pull-down to VSS 0b: disable weak pull-down to VSS

### 10.1.7. GPIOEI

#### Register 10-6. GPIOEI (GPIO Port E Input, 0x4009 0014)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PE7I	RW	0x0	Port E 7 input state 1b: input high 0b: input low
6	PE6I	RW	0x0	Port E 6 input state 1b: input high 0b: input low
5	PE5I	RW	0x0	Port E 5 input state 1b: input high 0b: input low
4	PE4I	RW	0x0	Port E 4 input state 1b: input high 0b: input low
3	PE3I	RW	0x0	Port E 3 input state 1b: input high 0b: input low
2	PE2I	RW	0x0	Port E 2 input state 1b: input high 0b: input low

BIT	NAME	ACCESS	RESET	DESCRIPTION
1	<b>PE1I</b>	RW	0x0	Port E 1 input state 1b: input high 0b: input low
0	<b>PE0I</b>	RW	0x0	Port E 0 input state 1b: input high 0b: input low

### 10.1.8. GPIOEPSEL

#### Register 10-7. GPIOEPSEL (GPIO Port E Peripheral Select, 0x4009 001C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:14	<b>PE7PSEL</b>	RW	0x0	Port E 7 peripheral select 11b: reserved 10b: reserved 01b: reserved 00b: I/O mode PE7
13:12	<b>PE6PSEL</b>	RW	0x0	Port E 6 peripheral select 11b: reserved 10b: reserved 01b: reserved 00b: I/O mode PE6
11:10	<b>PE5PSEL</b>	RW	0x0	Port E 5 peripheral select 11b: reserved 10b: I2C clock I2CSDA 01b: SPI chip select 2 SPICS2 00b: I/O mode PE5
9:8	<b>PE4PSEL</b>	RW	0x0	Port E 4 peripheral select 11b: reserved 10b: I2C clock I2CSCL 01b: SPI chip select 1 SPICS1 00b: I/O mode PE4
7:6	<b>PE3PSEL</b>	RW	0x0	Port E 3 peripheral select 11b: reserved 10b: Device Reset input nRESET1 01b: SPI chip select 0 SPICS0 00b: I/O mode PE3
5:4	<b>PE2PSEL</b>	RW	0x0	Port E 2 peripheral select 11b: reserved 10b: UART Receive UARTRX 01b: SPI Master in Slave out SPIMISO 00b: I/O mode PE2
3:2	<b>PE1PSEL</b>	RW	0x0	Port E 1 peripheral select 11b: reserved 10b: UART Transmit UARTRX 01b: SPI Master out Slave in SPIMOSI 00b: I/O mode PE1
1:0	<b>PE0PSEL</b>	RW	0x0	Port E 0 peripheral select 11b: reserved 10b: reserved 01b: SPI Clock SPICLK 00b: I/O mode PE0

### 10.1.9. GPIOINTP

#### Register 10-8. GPIOINTP (GPIO Port E Interrupt Polarity, 0x4009 0020)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PE7INTP	RW	0x0	Port E 7 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
6	PE6INTP	RW	0x0	Port E 6 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
5	PE5INTP	RW	0x0	Port E 5 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
4	PE4INTP	RW	0x0	Port E 4 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
3	PE3INTP	RW	0x0	Port E 3 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
2	PE2INTP	RW	0x0	Port E 2 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
1	PE1INTP	RW	0x0	Port E 1 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition
0	PE0INTP	RW	0x0	Port E 0 interrupt polarity select 1b: Rising edge, low to high transition 0b: Falling edge, high to low transition

### 10.1.10. GPIOINTEN

#### Register 10-9. GPIOINTEN (GPIO Port E Interrupt Enable, 0x4009 0024)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PE7INTEN	RW	0x0	Port E 7 interrupt enable 1b: enabled interrupt 0b: disable interrupt
6	PE6INTEN	RW	0x0	Port E 6 interrupt enable 1b: enabled interrupt 0b: disable interrupt
5	PE5INTEN	RW	0x0	Port E 5 interrupt enable 1b: enabled interrupt 0b: disable interrupt
4	PE4INTEN	RW	0x0	Port E 4 interrupt enable 1b: enabled interrupt 0b: disable interrupt
3	PE3INTEN	RW	0x0	Port E 3 interrupt enable 1b: enabled interrupt 0b: disable interrupt
2	PE2INTEN	RW	0x0	Port E 2 interrupt enable 1b: enabled interrupt 0b: disable interrupt

BIT	NAME	ACCESS	RESET	DESCRIPTION
1	PE1INTEN	RW	0x0	Port E 1 interrupt enable 1b: enabled interrupt 0b: disable interrupt
0	PE0INTEN	RW	0x0	Port E 0 interrupt enable 1b: enabled interrupt 0b: disable interrupt

### 10.1.11. GPIOEINT

#### Register 10-10. GPIOEINT (GPIO Port E Interrupt, 0x4009 0028)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PE7INT	RW	0x0	Port E 7 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
6	PE6INT	RW	0x0	Port E 6 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
5	PE5INT	RW	0x0	Port E 5 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
4	PE4INT	RW	0x0	Port E 4 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
3	PE3INT	RW	0x0	Port E 3 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
2	PE2INT	RW	0x0	Port E 2 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
1	PE1INT	RW	0x0	Port E 1 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending
0	PE0INT	RW	0x0	Port E 0 interrupt 1b: interrupt pending, clear with write to 1b 0b: no interrupt pending

### 10.1.12. GPIOEINTM

#### Register 10-11. GPIOEINTM (GPIO Port E Interrupt Mask, 0x4009 002C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	PE7INTM	RW	0x0	Port E 7 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
6	PE6INTM	RW	0x0	Port E 6 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask

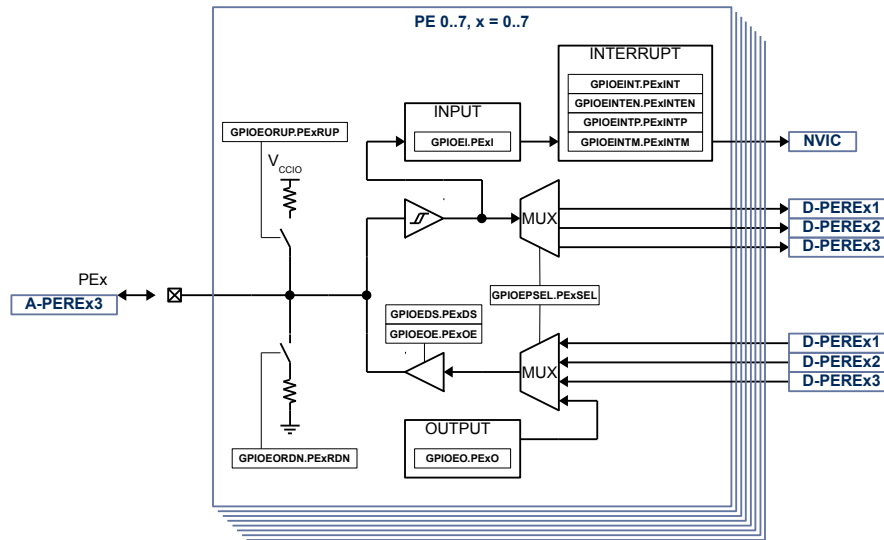
BIT	NAME	ACCESS	RESET	DESCRIPTION
5	<b>PE5INTM</b>	RW	0x0	Port E 5 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
4	<b>PE4INTM</b>	RW	0x0	Port E 4 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
3	<b>PE3INTM</b>	RW	0x0	Port E 3 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
2	<b>PE2INTM</b>	RW	0x0	Port E 2 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
1	<b>PE1INTM</b>	RW	0x0	Port E 1 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask
0	<b>PE0INTM</b>	RW	0x0	Port E 0 interrupt mask 1b: enable interrupt mask 0b: disable interrupt mask



## 10.2. Details of Operation

### 10.2.1. Block Diagram

Figure 10-1. GPIO Port E



### 10.2.2. Configuration

Following blocks need to be configured for correct use of the GPIO E:

- Nested Vectored Interrupt Controller (NVIC)
- SPI
- I2C
- UART

### 10.2.3. GPIO E Block

The GPIO E block consists of up to 8 general purpose input output (GPIO). Each GPIO has interrupt capabilities, weak pull-up or pull-down, programmable output drive strength, High-Z output operation.

### 10.2.4. Input

The input state of GPIOE can be monitored with **GPIOEI.PExI**. The input state can be monitored regardless of the peripheral select setting **GPIOEPSSEL**.

### 10.2.5. Output and Output Enable

When **GPIOEOE.PExOE** is enabled, the output state is controlled by **GPIOEO.PExO**.

When **GPIOEOE.PExOE** is disabled, the output is in High-Z state.

### 10.2.6. Output Drive Strength

The output drive strength can be adjusted using **GPIOEDS** to meet application needs. Set **GPIOEDS.PExODS** to enable high current drive strength, reset to enable low current drive strength.

### 10.2.7. Weak Pull Up and Pull Down

Independent from the output settings, weak pull up can be enabled with **GPIOEORUP** and weak pull down can be enabled with **GPIOERDN**.

#### NOTE:

**GPIOERUP.PExRUP** or **GPIOERDN.PExRDN** should never be enabled at the same time for a single GPIO. If switching from weak pull-up to weak pull-down is required, disable weak pull-up first before enable weak pull-down and vice versa.

### 10.2.8. Peripheral Select

Each GPIO is connected to up to 4 digital peripherals, selectable with **GPIOEPSEL**. When a different function than IO is selected the input state can still be read with **GPIOEI** and the pull-up and pull-down is still controllable.

### 10.2.9. Interrupt

The interrupt for each GPIO can be enabled with **GPIOEINTEN**. The interrupt can be configured to be rising signal edge or falling signal edge using **GPIOEINTP**. The state of the interrupt can be read from **GPIOEINT**. The individual interrupt bits can be cleared by writing to 1.

When the GPIO interrupts are enabled for the first time after device start-up, it may be in an uncertain state and generate an interrupt. To avoid this the **GPIOEINTM** mask bit need to be set before enabled interrupt bits.

To allow interrupt to be recognized by the CPU the GPIO interrupt need also be enabled in the NVIC.

## 11. TIMER A

### 11.1. Register

#### 11.1.1. Register Map

**Table 11-1. Timer A Register Map**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>Timer A</b>			
0x400D 0000	<b>TACTL</b>	Timer A control	0x0000 0000
0x400D 0004	<b>TAPRD</b>	Timer A period	0x0000 0000
0x400D 0008	<b>TACTR</b>	Timer A counter	0x0000 0000
<b>Timer A PWMA Capture and Compare</b>			
0x400D 0040	<b>TACCTL0</b>	Timer A capture and compare 0 control	0x0000 0000
0x400D 0044	<b>TACTR0</b>	Timer A counter 0	0x0000 0000
0x400D 0048	<b>TACCTL1</b>	Timer A capture and compare 1 control	0x0000 0000
0x400D 004C	<b>TACTR1</b>	Timer A counter 1	0x0000 0000
0x400D 0050	<b>TACCTL2</b>	Timer A capture and compare 2 control	0x0000 0000
0x400D 0054	<b>TACTR2</b>	Timer A counter 2	0x0000 0000
0x400D 0058	<b>TACCTL3</b>	Timer A capture and compare 3 control	0x0000 0000
0x400D 005C	<b>TACTR3</b>	Timer A counter 3	0x0000 0000
0x400D 0060	<b>TACCTL4</b>	Timer A capture and compare 4 control	0x0000 0000
0x400D 0064	<b>TACTR4</b>	Timer A counter 4	0x0000 0000
0x400D 0068	<b>TACCTL5</b>	Timer A capture and compare 5 control	0x0000 0000
0x400D 006C	<b>TACTR5</b>	Timer A counter 5	0x0000 0000
0x400D 0070	<b>TACCTL6</b>	Timer A capture and compare 6 control	0x0000 0000
0x400D 0074	<b>TACTR6</b>	Timer A counter 6	0x0000 0000
0x400D 0078	<b>TACCTL7</b>	Timer A capture and compare 7 control	0x0000 0000
0x400D 007C	<b>TACTR7</b>	Timer A counter 7	0x0000 0000
<b>Timer A Dead Time Generator</b>			
0x400D 00A0	<b>DTGA0CTL</b>	Timer A dead time generator 0 control	0x0000 0080
0x400D 00A4	<b>DTGA0LED</b>	Timer A dead time generator 0 leading edge delay	0x0000 0000
0x400D 00A8	<b>DTGA0TED</b>	Timer A dead time generator 0 trailing edge delay	0x0000 0000
0x400D 00B0	<b>DTGA1CTL</b>	Timer A dead time generator 1 control	0x0000 0080
0x400D 00B4	<b>DTGA1LED</b>	Timer A dead time generator 1 leading edge delay	0x0000 0000
0x400D 00B8	<b>DTGA1TED</b>	Timer A dead time generator 1 trailing edge delay	0x0000 0000
0x400D 00C0	<b>DTGA2CTL</b>	Timer A dead time generator 2 control	0x0000 0080
0x400D 00C4	<b>DTGA2LED</b>	Timer A dead time generator 2 leading edge delay	0x0000 0000
0x400D 00C8	<b>DTGA2TED</b>	Timer A dead time generator 2 trailing edge delay	0x0000 0000
0x400D 00D0	<b>DTGA3CTL</b>	Timer A dead time generator 3 control	0x0000 0080
0x400D 00D4	<b>DTGA3LED</b>	Timer A dead time generator 3 leading edge delay	0x0000 0000

ADDRESS	NAME	DESCRIPTION	RESET VALUE
0x400D 00D8	<b>DTGA3TED</b>	Timer A dead time generator 3 trailing edge delay	0x0000 0000

### 11.1.2. TACTL

#### Register 11-1. TACTL (Timer A Control, 0x400D 0000)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:14	<b>Reserved</b>	RO	0x0	Reserved
13	<b>DTGCLK</b>	RW	0x0	DTGAX clock select 1b: DTGAX use clock after <b>TACTL.CLKDIV</b> 0b: DTGAX use clock selected by <b>TACTL.CLK</b>
12	<b>Reserved</b>	RW	0x0	Reserved, Write to 0x0
11:10	<b>MODE</b>	RW	0x0	Timer A Mode 11b: reserved 10b: up / down 01b: up 00b: disabled
9	<b>CLK</b>	RW	0x0	Timer A clock input source 1b: ACLK 0b: HCLK
8:6	<b>CLKDIV</b>	RW	0x0	Timer A input clock divider 111b: / 128 110b: / 64 101b: /32 100b: /16 011b: /8 010b: /4 001b: /2 000b: /1
5	<b>INTEN</b>	RW	0x0	Timer A interrupt enable 1b: enable Timer A interrupt 0b: disable Timer A interrupt
4	<b>INT</b>	RW	0x0	Timer A interrupt 1b: interrupt, clear by write 1b 0b: no interrupt
3	<b>SS</b>	RW	0x0	Timer A single shot 1b: single shot mode 0b: continuous timer mode
2	<b>CLR</b>	RW	0x0	Timer A clear 1b: Clear Timer A, hold Timer A in reset and set SYNC_OUT 0b: Do not clear timer and clear SYNC_OUT
1	<b>Reserved</b>	RO	0x0	Reserved
0	<b>PRDL</b>	RW	0x0	Timer A <b>TAPRD</b> update 1b: Latch new <b>TAPRD</b> value when timer A counting down, <b>TACTR</b> value = 0x1 and <b>TACTL.MODE</b> = 10b 0b: Latch new <b>TAPRD</b> value when timer A counting up and <b>TACTR</b> value = <b>TAPRD</b> – 0x1.

### 11.1.3. TAPRD

#### Register 11-2. TAPRD (Timer A Period, 0x400D 0004)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0	Reserved
15:0	<b>PERIOD</b>	RW	0x0	Timer A period value

### 11.1.4. TACTR

#### Register 11-3. TACTR (Timer A Counter, 0x400D 0008)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CTR</b>	RO	0x0	Current Timer A counter value

### 11.1.5. TACC0CTL

#### Register 11-4. TACC0CTL (Timer A PWMA0 Capture and Compare Control, 0x400D 0040)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMA0 input 0b: Compare mode PWMA0 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMA0 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

### 11.1.6. TACC0CTR

#### Register 11-5. TACC0CTR (Timer A PWMA0 Capture and Compare Counter, 0x400D 0044)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMA0 compare mode or counter value for PWMA0 capture mode

### 11.1.7. TACC1CTL

#### Register 11-6. TACC1CTL (Timer A PWMA1 Capture and Compare Control, 0x400D 0048)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMA1 input 0b: Compare mode PWMA1 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMA1 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

### 11.1.8. TACC1CTR

#### Register 11-7. TACC1CTR (Timer A PWMA1 Capture and Compare Counter, 0x400D 004C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMA1 compare mode or counter value for PWMA1 capture mode

### 11.1.9. TACC2CTL

#### Register 11-8. TACC2CTL (Timer A PWMA2 Capture and Compare Control, 0x400D 0050)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMA2 input 0b: Compare mode PWMA2 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMA2 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

### 11.1.10. TACC2CTR

#### Register 11-9. TACC2CTR (Timer A PWMA2 Capture and Compare Counter, 0x400D 0054)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	Reserved	RO	0x0	Reserved
15:0	CCCTR	RW	0x0	Counter value for PWMA2 compare mode or counter value for PWMA2 capture mode

### 11.1.11. TACC3CTL

#### Register 11-10. TACC3CTL (Timer A PWMA3 Capture and Compare Control, 0x400D 0058)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	Reserved	RO	0x0	Reserved
4	CCMODE	RW	0x0	Capture and compare mode 1b: Capture mode PWMA3 input 0b: Compare mode PWMA3 output
3	CCINTEN	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	CCINT	RW	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	CCEDG	RW	0x0	Capture mode edge detect PWMA3 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

### 11.1.12. TACC3CTR

#### Register 11-11. TACC3CTR (Timer A PWMA3 Capture and Compare Counter, 0x400D 005C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	Reserved	RO	0x0	Reserved
15:0	CCCTR	RW	0x0	Counter value for PWMA3 compare mode or counter value for PWMA3 capture mode

### 11.1.13. TACC4CTL

#### Register 11-12. TACC4CTL (Timer A PWMA4 Capture and Compare Control, 0x400D 0060)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	Reserved	RO	0x0	Reserved
4	CCMODE	RW	0x0	Capture and compare mode 1b: Capture mode PWMA4 input 0b: Compare mode PWMA4 output
3	CCINTEN	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt

BIT	NAME	ACCESS	RESET	DESCRIPTION
2	<b>CCINT</b>	RW	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMA4 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

#### 11.1.14. TACC4CTR

##### Register 11-13. TACC4CTR (Timer A PWMA4 Capture and Compare Counter, 0x400D 0064)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMA4 compare mode or counter value for PWMA4 capture mode

#### 11.1.15. TACC5CTL

##### Register 11-14. TACC5CTL (Timer A PWMA5 Capture and Compare Control, 0x400D 0068)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMA5 input 0b: Compare mode PWMA5 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMA5 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

#### 11.1.16. TACC5CTR

##### Register 11-15. TACC5CTR (Timer A PWMA5 Capture and Compare Counter, 0x400D 006C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMA5 compare mode or counter value for PWMA5 capture mode



### 11.1.17. TACC6CTL

#### Register 11-16. TACC6CTL (Timer A PWMA6 Capture and Compare Control, 0x400D 0070)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMA6 input 0b: Compare mode PWMA6 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMA6 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

### 11.1.18. TACC6CTR

#### Register 11-17. TACC6CTR (Timer A PWMA6 Capture and Compare Counter, 0x400D 0074)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMA6 compare mode or counter value for PWMA6 capture mode

### 11.1.19. TACC7CTL

#### Register 11-18. TACC7CTL (Timer A PWMA7 Capture and Compare Control, 0x400D 0078)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMA7 input 0b: Compare mode PWMA7 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMA7 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

### 11.1.20. TACC7CTR

#### Register 11-19. TACC7CTR (Timer A PWMA7 Capture and Compare Counter, 0x400D 007C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	Reserved	RO	0x0	Reserved
15:0	CCCTR	RW	0x0	Counter value for PWMA7 compare mode or counter value for PWMA7 capture mode

### 11.1.21. DTGA0CTL

#### Register 11-20. DTGA0CTL (Timer A Dead Time Generator 0 Control, 0x400D 00A0)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	BYPASS	RW	0x1	Bypass dead time generator 1b: DTGA0 bypass active, DTGA0LS = PWMA0, DTGA0HS = PWMA4 0b: DTGA0 bypass inactive, dead time inserted to DTGA0LS and DTGA0HS, DTGA0LS = PWMA4, DTGA0HS = PWMA4
6	OTP	RW	0x0	One Time Preservation 1b: DTGA0HS high time is same as PWMA4 hightime and is shifted by <b>DTGA0LED</b> 0b: DTGA0HS high time is reduced by <b>DTGA0LED</b>
5	INVHS	RW	0x0	Invert DTGA0HS output signal 1b: invert DTGA0HS 0b: do not invert DTGA0HS
4	INVLS	RW	0x0	Invert DTGA0LS output signal 1b: invert DTGA0LS 0b: do not invert DTGA0LS
3:0	Reserved	RO	0x0	Reserved

### 11.1.22. DTGA0LED

#### Register 11-21. DTGA0LED (Timer A Dead Time Generator 0 Leading Edge Delay, 0x400D 00A4)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	Reserved	RO	0x0	Reserved
11:0	LED	RW	0x0	Counter value DTGA0 leading edge dead time in clock cycles defined by <b>TACTL.DTGCLK</b>

### 11.1.23. DTGA0TED

#### Register 11-22. DTGA0TED (Timer A Dead Time Generator 0 Trailing Edge Delay, 0x400D 00A8)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	Reserved	RO	0x0	Reserved
11:0	TED	RW	0x0	Counter value DTGA0 trailing edge dead time in clock cycles defined by <b>TACTL.DTGCLK</b>

### 11.1.24. DTGA1CTL

#### Register 11-23. DTGA1CTL (Timer A Dead Time Generator 1 Control, 0x400D 00B0)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	BYPASS	RW	0x1	Bypass dead time generator 1b: DTGA1 bypass active, DTGA1LS = PWMA1, DTGA1HS = PWMA5 0b: DTGA1 bypass inactive, dead time inserted to DTGA1LS and DTGA1HS, DTGA1LS = PWMA5, DTGA1HS = PWMA5
6	OTP	RW	0x0	One Time Preservation 1b: DTGA1HS high time is same as PWMA5 high time and is shifted by <b>DTGA1LED</b> 0b: DTGA1HS high time is reduced by <b>DTGA1LED</b>
5	INVHS	RW	0x0	Invert DTGA1HS output signal 1b: invert DTGA1HS 0b: do not invert DTGA1HS
4	INVLS	RW	0x0	Invert DTGA1LS output signal 1b: invert DTGA1LS 0b: do not invert DTGA1LS
3:0	Reserved	RO	0x0	Reserved

### 11.1.25. DTGA1LED

#### Register 11-24. DTGA1LED (Timer A Dead Time Generator 1 Leading Edge Delay, 0x400D 00B4)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	Reserved	RO	0x0	Reserved
11:0	LED	RW	0x0	Counter value DTGA1 leading edge dead time in clock cycles defined by <b>TACTL.DTGCLK</b>

### 11.1.26. DTGA1TED

#### Register 11-25. DTGA1TED (Timer A Dead Time Generator 1 Trailing Edge Delay, 0x400D 00B8)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	Reserved	RO	0x0	Reserved
11:0	TED	RW	0x0	Counter value DTGA1 trailing edge dead time in clock cycles defined by <b>TACTL.DTGCLK</b>

### 11.1.27. DTGA2CTL

#### Register 11-26. DTGA2CTL (Timer A Dead Time Generator 2 Control, 0x400D 00C0)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	Reserved	RO	0x0	Reserved
7	BYPASS	RW	0x1	Bypass dead time generator 1b: DTGA2 bypass active, DTGA2LS = PWMA2, DTGA2HS = PWMA6 0b: DTGA2 bypass inactive, dead time inserted to DTGA2LS and DTGA2HS, DTGA2LS = PWMA6, DTGA2HS = PWMA6

BIT	NAME	ACCESS	RESET	DESCRIPTION
6	<b>OTP</b>	RW	0x0	One Time Preservation 1b: DTGA2HS high time is same as PWMA6 high time and is shifted by <b>DTGA2LED</b> 0b: DTGA2HS high time is reduced by <b>DTGA2LED</b>
5	<b>INVHS</b>	RW	0x0	Invert DTGA2HS output signal 1b: invert DTGA2HS 0b: do not invert DTGA2HS
4	<b>INVLS</b>	RW	0x0	Invert DTGA2LS output signal 1b: invert DTGA2LS 0b: do not invert DTGA2LS
3:0	<b>Reserved</b>	RO	0x0	Reserved

### 11.1.28. DTGA2LED

#### Register 11-27. DTGA2LED (Timer A Dead Time Generator 2 Leading Edge Delay, 0x400D 00C4)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	<b>Reserved</b>	RO	0x0	Reserved
11:0	<b>LED</b>	RW	0x0	Counter value DTGA2 leading edge dead time in clock cycles defined by <b>TACTL.DTGCLK</b>

### 11.1.29. DTGA2TED

#### Register 11-28. DTGA2TED (Timer A Dead Time Generator 2 Trailing Edge Delay, 0x400D 00C8)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	<b>Reserved</b>	RO	0x0	Reserved
11:0	<b>TED</b>	RW	0x0	Counter value DTGA2 trailing edge dead time in clock cycles defined by <b>TACTL.DTGCLK</b>

### 11.1.30. DTGA3CTL

#### Register 11-29. DTGA3CTL (Timer A Dead Time Generator 3 Control, 0x400D 00D0)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>BYPASS</b>	RW	0x1	Bypass dead time generator 1b: DTGA3 bypass active, DTGA3LS = PWMA3, DTGA3HS = PWMA7 0b: DTGA3 bypass inactive, dead time inserted to DTGA3LS and DTGA3HS, DTGA3LS = PWMA7, DTGA1HS = PWMA7
6	<b>OTP</b>	RW	0x0	One Time Preservation 1b: DTGA3HS high time is same as PWMA7 high time and is shifted by <b>DTGA3LED</b> 0b: DTGA3HS high time is reduced by <b>DTGA3LED</b>
5	<b>INVHS</b>	RW	0x0	Invert DTGA3HS output signal 1b: invert DTGA3HS 0b: do not invert DTGA3HS
4	<b>INVLS</b>	RW	0x0	Invert DTGA3LS output signal 1b: invert DTGA3LS 0b: do not invert DTGA3LS

BIT	NAME	ACCESS	RESET	DESCRIPTION
3:0	<b>Reserved</b>	RO	0x0	Reserved

### 11.1.31. DTGA3LED

#### Register 11-30. DTGA3LED (Timer A Dead Time Generator 3 Leading Edge Delay, 0x400D 00D4)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	<b>Reserved</b>	RO	0x0	Reserved
11:0	<b>LED</b>	RW	0x0	Counter value DTGA3 leading edge dead time in clock cycles defined by <b>TACTL.DTGCLK</b>

### 11.1.32. DTGA3TED

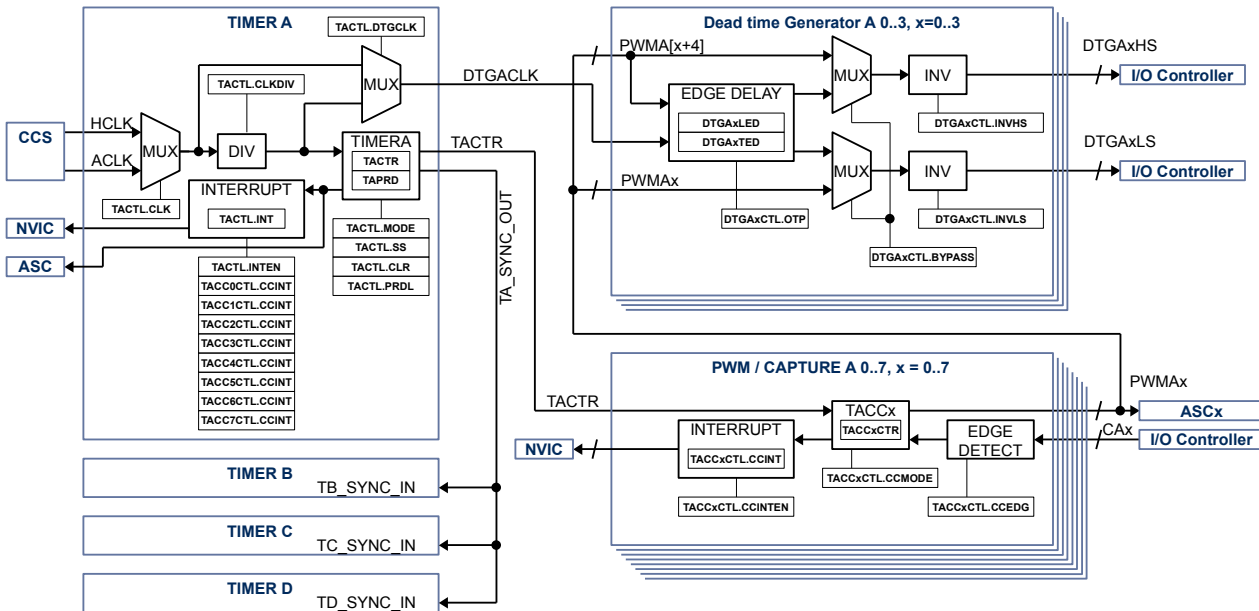
#### Register 11-31. DTGA3TED (Timer A Dead Time Generator 3 Trailing Edge Delay, 0x400D 00D8)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	<b>Reserved</b>	RO	0x0	Reserved
11:0	<b>TED</b>	RW	0x0	Counter value DTGA3 trailing edge dead time in clock cycles defined by <b>TACTL.DTGCLK</b>

## 11.2. Details of Operation

### 11.2.1. Block Diagram

Figure 11-1. Timer A



### 11.2.2. Configuration

Following blocks need to be configured for correct use of the Timer A:

- Clock Control System (CCS)
- Nested Vectored Interrupt Controller (NVIC)
- I/O Controller
- Gate Driver
- Auto sequencing controller (ASC)
- Timer B
- Timer C
- Timer D

### 11.2.3. Timer A Block

The timer A block consist of a 16-bit timer with up mode or up/down mode with 8 PWM/capture units and 4 dead-time generator units.

### 11.2.4. Timer

Once enabled the timer counts up to the Timer A period value **TAPRD**. The **TAPRD** register can be written to

while the timer is running, the new **TAPRD** value will be latched when the counter reaches old **TAPRD** value in up mode. In up/down mode there is the option to latch the new **TAPRD** value when counter counts back to zero. **TACTL.PRDL** configures when the timer will be updated with the new **TAPRD** value in up/down mode.

The current timer value is accessible with the timer A counter value register **TACTR**.

#### 11.2.5. Register update

The **TAPRD**, **TACCxCTR** register can be written to while the timer is running, the new **TAPRD**, **TACCxCTR** value will be latched when the counter reaches old **TAPRD** value in up mode. In up/down mode there is the option to latch the new **TAPRD**, **TACCxCTR** values when counter counts back to zero. **TACTL.PRDL** configures when the timer will be updated with the new **TAPRD**, **TACCxCTR** value in up/down mode.

#### 11.2.6. Timer Modes

The timer supports 3 modes of operation: disabled, up and up/down using **TACTL.MODE**.

By default, the timer mode is disabled. When the timer is disabled, the timer counter does not increment or decrement. If the timer is disabled when previously in up or up/down mode, then the timer counter stops where it is. If the timer is re-enabled by putting it back into up or up/down modes, then the counter continues from the point at which it was disabled. To reset the current counter value **TACTR** to zero use **TACTL.CLR**.

In up mode, the timer starts counting from 0 up to the value of **TAPRD**. When the timer counter reaches the value of **TAPRD**, then the timer counter is reset to a value of 0. This mode is typically used for timed events or edge-aligned PWM output.

In up/down mode, the timer starts counting from 0 up to the value of **TAPRD**, and then back down to a value of 0. This timer mode is typically used for center-aligned PWM output. It can also be used for timed events, and will allow a longer timer range due to the fact it counts up and down

#### 11.2.7. Single Shot Mode

The timer can be configured to run either once or continuously.

When the timer is configured in single shot mode using **TACTL.SS** the timer will only count to **TAPRD** value and stops in up mode. In up/down mode the timer will count to **TAPRD** and back to zero only once.

To start the timer in single-shot mode, **TACTL.SS** must be set. The timer will start when **TACTL.CLR** is set. To re-start a single-shot timer, **TACTL.CLR** must be reset, and then set again.

#### 11.2.8. Input Clock And Pre-Scaler

The timer can be configured to use HCLK or ACLK using **TACTL.CLK**. The input clock for the can be divided further down from /1 to /128 using the **TACTL.CLKDIV**.

#### 11.2.9. Timer Synchronization

The Timer A, B, C, D in the system have the ability to have synchronization between them. Each timer has a synchronization in signal (**SYNC\_IN**) and the synchronization out signal (**SYNC\_OUT**).

Timer A can be synchronized with Timer B, C, and D with timer A as master.

The timer asserts the **SYNC\_OUT** pulse when the **TACTL.CLR** bit is set and de-asserts the **SYNC\_OUT** pulse when the **TACTL.CLR** bit is cleared.

Each timer B, C, or D that need to be synchronized as slave with master timer A need to set the **TxCTL.SYNC** bit. If this bit is not set, then the **sync\_in** signal is ignored and the timer operates independently.

When the **TxCTL.SYNC** bit is set and the **SYNC\_IN** signal is asserted, the timer clears the timer counter. The

timer counter is also cleared anytime the **TxCTL.CLR** bit is set to a 1. When the **TxCTL.SYNC** bit is set and the **SYNC\_IN** signal is de-asserted and the timer mode is either up or up/down, then the timer will start counting. The timer will not start counting when the mode is set to up or up/down unless the **SYNC\_IN** signal is de-asserted when **TxCTL.SYNC** is set.

**NOTE:**

In order for this feature to work correctly, all timers that are synchronized must be set to the same mode (up or up/down), with the same timer pre-scaler, timer clock input and timer period.

To enable synchronized timers, the following steps should be followed:

1. All slave timers B, C, or D are configured with the selected timer input clock, timer pre-scaler, timer period and set the **TxCTL.SYNC** bit. The timer should still be set to disabled at this point.
2. The master timer A is configured with the same timer input clock, timer pre-scaler, timer period and sets the **TxCTL.CLR** bit. This should clear all timer counters of the master and slave timers.
3. The slave timers set the timer mode to the desired state (either up or up/down).
4. The master timer sets the timer mode to either up or up/down and clears the **TACTL.CLR** bit. This should start the master and all slave timers simultaneously based on the selected timer clock input.
5. Once configured as above, all timers can be disabled by the master setting **TACTL.CLR** signal, to assert the **SYNC\_OUT** signal. The timers can be re-enabled by clearing the **TACTL.CLR** bit, which de-asserts the **SYNC\_OUT** signal.

### 11.2.10. PWM/Compare Units

Timer A supports up to 8 PWM/Capture units PWMA0 to PWMA7. Each PWM/Compare unit can be configured independently in PWM mode or capture mode using **TACCxCTL.CCMODE**.

#### 11.2.10.1. PWM Mode

The PWM mode is enabled with setting **TACCxCTL.CCMODE** to 0.

The timer configuration allows either edge-aligned (timer in up mode) or center-aligned (timer in up/down mode) modes of PWM operation.

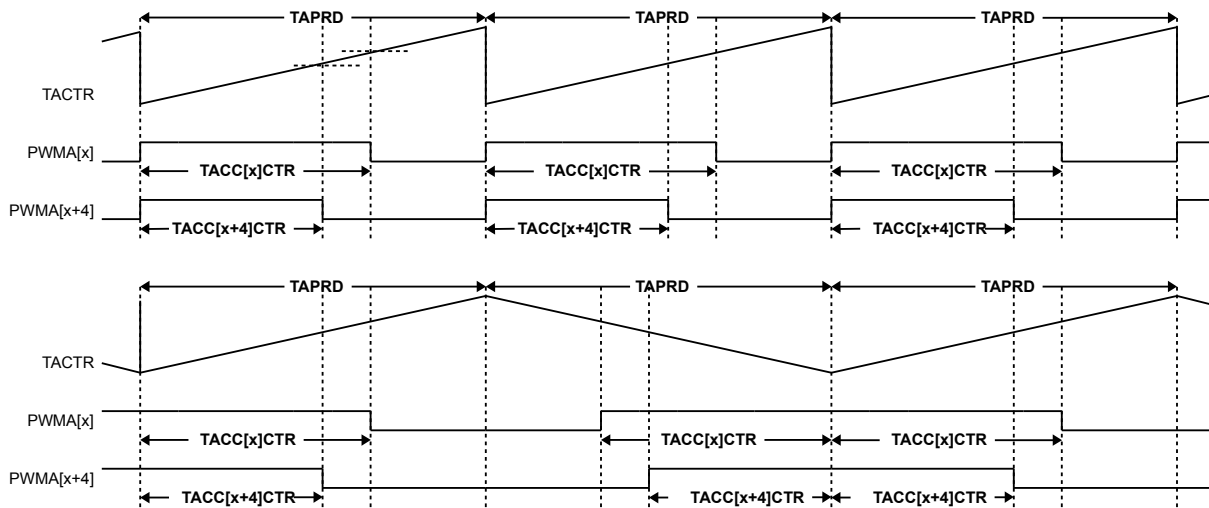
In both edge-aligned and center-aligned modes of operation, the timer block outputs a PWM waveform that starts out high at a **TACTR** value of 0 and then transitions to low when **TACTR** counts up to **TACCxCTR** compare value.

To configure a duty cycle of 0%, the **TACCxCTR** should be set to 0; to configure a duty cycle of 100%, the **TACCxCTR** value should be set to a value greater than or equal to **TAPRD**.

The polarity of the timer PWM outputs are not configurable. Adjustments to the polarity of the PWM outputs may be adjusted in the Dead-Time Generator (DTG) unit connected to the timer peripheral for each output independently.



**Figure 11-2. PWMA[x] and PWMA[x+4] Example Using Timer A Up Mode and Up/Down Mode**

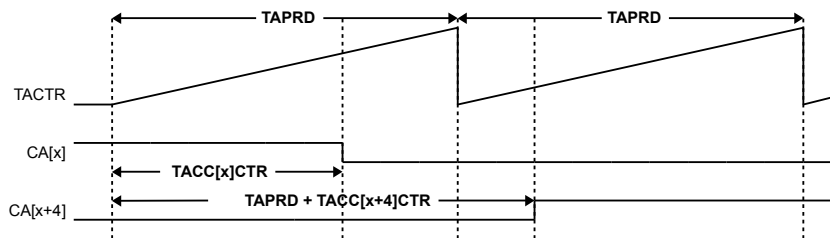


### 11.2.10.2. Capture Mode

The Capture mode is enabled with setting **TACCxCTL.CCMODE** to 1. The trip condition for capture mode can be configured using **TACCxCTL.CCEDGE**, high-to-low signal edge transition, low-to-high signal edge transition or both.

When trip condition is detected the actual **TACTR** value is copied into **TACCxCTR**.

**Figure 11-3. CA[x] and CA[x+4] Capture Example**



### 11.2.11. Timer and PWM/Capture Interrupt

The timer may generate interrupt based on the base timer wrap, or when a capture and compare event occurs.

In the base timer both up and up/down timer modes allow an interrupt to be generated when the count reaches 0. Each time the count reaches zero, the **TACTL.INT** interrupt flag is set. If the interrupt is enabled using the **TACTL.INTEN**, then the Timer IRQ signal will be asserted to the CPU. The interrupt flag may be cleared by writing a 1 to the **TACTL.INT** interrupt flag bit.

In the capture and compare PWM units, each time a compare threshold is reached or each time a capture event is detected the **TACCxCTL.CCINT** bit will be set for that particular timer unit. If the interrupt is enabled via the **TACCxCTL.CCINTEN**, then the Timer IRQ signal will be asserted to the CPU. The interrupt flag may be cleared by writing a 1 to the **TACCxCTL.CCINT** interrupt flag bit.

The timer IRQ signal will be asserted if any of the timer interrupt flags **TACTL.INT** or **TACCxCTL.CCINT** are set. The Timer IRQ signal will be de-asserted when all of the timer interrupt flags are cleared.

### 11.2.12. Dead-Time Generator

The dead-time generator can be configured to introduce dead-time for a complementary PWM output. The Timer A block supports up to 4 dead time generators.

#### 11.2.12.1. Dead Time Input Clock Selection

The clock source for the DTG<sub>x</sub> can be selected using **TACTL.DTGCLK**.

Clear **TACTL.DTGCLK** to 0 to use clock source selected by **TACTL.CLK** directly to use higher resolution for dead time insertion.

Set **TACTL.DTGCLK** to 1 to use divided clock source selected by **TACTL.CLK** and **TACTL.CLKDIV** divider to use the same dead time resolution as Timer A.

#### 11.2.12.2. Dead Time Range

The resolution for leading edge and trailing edge dead time is 12bits. Leading and trailing edge can be set independently for each DTG using **DTGxLED** and **DTGxTED**.

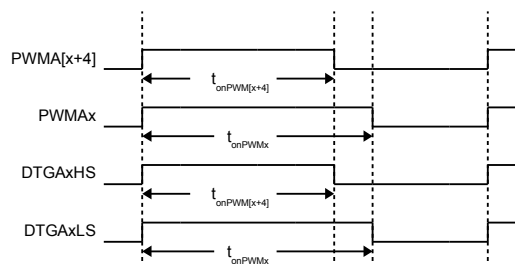
#### 11.2.12.3. Bypass Mode

Set **DTGxCTL.BYPASS** to 0 to enable dead time insertion.

Set **DTGxCTL.BYPASS** to 1 to enable bypass mode, no deadtime is inserted, PWMA<sub>[x+4]</sub> is routed to DTG<sub>x</sub>HS and PWMA<sub>x</sub> is routed to DTG<sub>x</sub>LS.

The DTG<sub>x</sub>HS and DTG<sub>x</sub>LS signals can be inverted in bypass mode.

**Figure 11-4. DTG<sub>x</sub> Bypass Example**



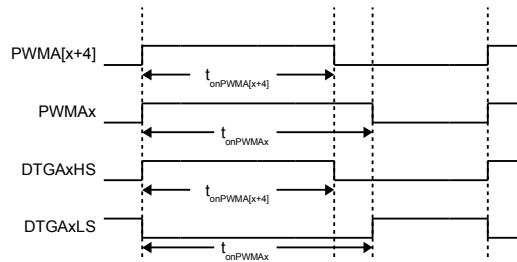
#### 11.2.12.4. Inverting PWM Signal

The DTG output signals DTG<sub>x</sub>HS and DTG<sub>x</sub>LS can be inverted independently.

Set **DTGxCTL.INVHS** to invert DTG<sub>x</sub>HS signal.

Set **DTGxCTL.INVLS** to invert DTG<sub>x</sub>LS signal.

**Figure 11-5. DTGx Bypass and Inverting LS Example**



**11.2.12.5. Dead Time Insertion**

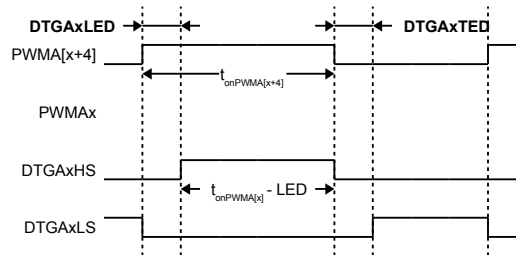
Set **DTGxCTL.BYPASS** to 0 to enable dead time insertion. In dead time insertion mode only PWM[x+4] signal is used to generate DTGxHS and DTGxLS. PWMA[x] signal is ignored and can be used for other purposes.

Set **DTGxLED** for desired leading-edge and **DTGxTED** for desired trailing edge in clock-cycles defined by **TACTL.DTGCLK** clock source

**NOTE:**

In dead time insertion mode the DTGxLS signal is automatically inverted compared to PWM[x+4] signal. Set **DTGxCTL.INVLS** to 0, if this is desired behavior.

**Figure 11-6. DTGx LED and TED Example**



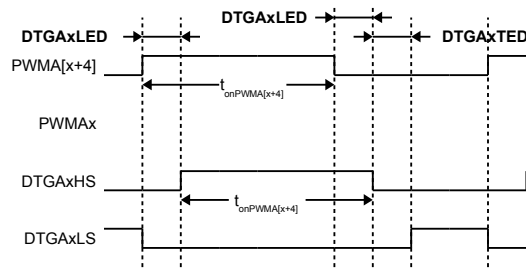
**11.2.12.6. Dead Time Insertion with On Time Preservation**

Set **DTGxCTL.OTP** to 1 to enable on time preservation. In this mode the DTGxHS is same as PWM[x+4] on time.

**NOTE:**

In dead time insertion mode the DTGxLS signal is automatically inverted compared to PWMA[x+4] signal. Set **DTGxCTL.INVLS** to 0, if this is desired behavior.

**Figure 11-7. DTGx LED and TED with On Time Preservation Example**



### 11.2.13. PWM Output and Capture Input Pin Selection

Each of the DTGxHS, DTGxLS outputs, and CAx inputs can be routed to different I/Os, allowing great flexibility in pin assignments.

In capture mode only one I/O should be enabled as input to the capture. If more than one pin input is enabled, the capture might not work properly.

**Note:**

Not all pins are available pending package option, consult data sheet for available pins and signals.

**Table 11-2. Timer A Signal to Pin Mapping**

PWM	CAPTURE	DEADTIME	PINS
PWMA0	CA0	DTGA0LS	PA0
PWMA1	CA1	DTGA1LS	PA1
PWMA2	CA2	DTGA2LS	PA2
PWMA3	CA3	DTGA3LS	PA3, PD2
PWMA4	CA4	DTGA0HS	PA3, PA6, PD2
PWMA5	CA5	DTGA1HS	PA4, PA7, PD3, PD5
PWMA6	CA6	DTGA2HS	PA5, PD7
PWMA7	CA7	DTGA3HS	PA7, PD6, PD3

## 12. TIMER B

### 12.1. Register

#### 12.1.1. Register Map

Table 12-1. Timer B Register Map

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>Timer B</b>			
0x400E 0000	<b>TBCTL</b>	Timer B control	0x0000 0000
0x400E 0004	<b>TBPRD</b>	Timer B period	0x0000 0000
0x400E 0008	<b>TBCTR</b>	Timer B counter	0x0000 0000
<b>Timer B PWMB Capture and Compare</b>			
0x400E 0040	<b>TBCCTL0</b>	Timer B capture and compare 0 control	0x0000 0000
0x400E 0044	<b>TBCTR0</b>	Timer B counter 0	0x0000 0000
0x400E 0048	<b>TBCCTL1</b>	Timer B capture and compare 1 control	0x0000 0000
0x400E 004C	<b>TBCTR1</b>	Timer B counter 1	0x0000 0000
0x400E 0050	<b>TBCCTL2</b>	Timer B capture and compare 2 control	0x0000 0000
0x400E 0054	<b>TBCTR2</b>	Timer B counter 2	0x0000 0000
0x400E 0058	<b>TBCCTL3</b>	Timer B capture and compare 3 control	0x0000 0000
0x400E 005C	<b>TBCTR3</b>	Timer B counter 3	0x0000 0000
<b>Timer B Dead Time Generator</b>			
0x400E 00A0	<b>DTGB0CTL</b>	Timer B dead time generator 0 control	0x0000 0080
0x400E 00A4	<b>DTGB0LED</b>	Timer B dead time generator 0 leading edge delay	0x0000 0000
0x400E 00A8	<b>DTGB0TED</b>	Timer B dead time generator 0 trailing edge delay	0x0000 0000

#### 12.1.2. TBCTL

Register 12-1. TBCTL (Timer B Control, 0x400E 0000)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:14	<b>Reserved</b>	RO	0x0	Reserved
13	<b>DTGCLK</b>	RW	0x0	DTGBx clock select 1b: DTGBx use clock after <b>TBCTL.CLKDIV</b> 0b: DTGBx use clock selected by <b>TBCTL.CLK</b>
12	<b>SYNC</b>	RW	0x0	Timer B synchronization 1b: Synchronize Timer B, enable SYNC_IN 0b: Do not synchronize Timer B, disabled SYNC_IN
11:10	<b>MODE</b>	RW	0x0	Timer B Mode 11b: reserved 10b: up / down 01b: up 00b: disabled
9	<b>CLK</b>	RW	0x0	Timer B clock input source 1b: ACLK 0b: HCLK

BIT	NAME	ACCESS	RESET	DESCRIPTION
8:6	<b>CLKDIV</b>	RW	0x0	Timer B input clock divider 111b: / 128 110b: / 64 101b: /32 100b: /16 011b: /8 010b: /4 001b: /2 000b: /1
5	<b>INTEN</b>	RW	0x0	Timer B interrupt enable 1b: enable Timer B interrupt 0b: disable Timer B interrupt
4	<b>INT</b>	RW1C	0x0	Timer B interrupt 1b: interrupt, clear by write 1b 0b: no interrupt
3	<b>SS</b>	RW	0x0	Timer B single shot 1b: single shot mode 0b: continuous timer mode
2	<b>CLR</b>	RW	0x0	Timer B clear 1b: Clear Timer B, hold Timer B in reset and set SYNC_OUT 0b: Do not clear Timer and clear SYNC_OUT
1	<b>Reserved</b>	RO	0x0	Reserved
0	<b>PRDL</b>	RW	0x0	Timer B <b>TBPRD</b> update 1b: Latch new <b>TBPRD</b> value when Timer B counting down, <b>TBCTR</b> value = 0x1 and <b>TBCTL.MODE</b> = 10b 0b: Latch new <b>TBPRD</b> value when Timer B counting up and <b>TBCTR</b> value = <b>TBPRD</b> – 0x1.

### 12.1.3. TBPRD

**Register 12-2. TBPRD (Timer B Period, 0x400E 0004)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0	Reserved
15:0	<b>PERIOD</b>	RW	0x0	Timer B period value

### 12.1.4. TBCTR

**Register 12-3. TBCTR (Timer B Counter, 0x400E 0008)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CTR</b>	RO	0x0	Current Timer B counter value

### 12.1.5. TBCC0CTL

**Register 12-4. TBCC0CTL (Timer B PWMB0 Capture and Compare Control, 0x400E 0040)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved

BIT	NAME	ACCESS	RESET	DESCRIPTION
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMB0 input 0b: Compare mode PWMB0 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW1C	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMB0 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

### 12.1.6. TBCC0CTR

#### Register 12-5. TBCC0CTR (Timer B PWMB0 Capture and Compare Counter, 0x400E 0044)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMB0 compare mode or counter value for PWMB0 capture mode

### 12.1.7. TBCC1CTL

#### Register 12-6. TBCC1CTL (Timer B PWMB1 Capture and Compare Control, 0x400E 0048)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMB1 input 0b: Compare mode PWMB1 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW1C	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMB1 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

### 12.1.8. TBCC1CTR

#### Register 12-7. TBCC1CTR (Timer B PWMB1 Capture and Compare Counter, 0x400E 004C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved

BIT	NAME	ACCESS	RESET	DESCRIPTION
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMB1 compare mode or counter value for PWMB1 capture mode

### 12.1.9. TBCC2CTL

**Register 12-8. TBCC2CTL (Timer B PWMB2 Capture and Compare Control, 0x400E 0050)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMB2 input 0b: Compare mode PWMB2 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW1C	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMB2 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

### 12.1.10. TBCC2CTR

**Register 12-9. TBCC2CTR (Timer B PWMB2 Capture and Compare Counter, 0x400E 0054)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMB2 compare mode or counter value for PWMB2 capture mode

### 12.1.11. TBCC3CTL

**Register 12-10. TBCC3CTL (Timer B PWMB3 Capture and Compare Control, 0x400E 0058)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMB3 input 0b: Compare mode PWMB3 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW1C	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected



BIT	NAME	ACCESS	RESET	DESCRIPTION
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMB3 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

### 12.1.12. TBCC3CTR

#### Register 12-11. TBCC3CTR (Timer B PWMB3 Capture and Compare Counter, 0x400E 005C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMB3 compare mode or counter value for PWMB3 capture mode

### 12.1.13. DTGB0CTL

#### Register 12-12. DTGB0CTL (Timer B Dead Time Generator 0 Control, 0x400E 00A0)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>BYPASS</b>	RW	0x1	Bypass dead time generator 1b: DTGB0 bypass active, DTGB0LS = PWMB0, DTGB0HS = PWMB1 0b: DTGB0 bypass inactive, dead time inserted to DTGB0LS and DTGB0HS, DTGB0LS = PWMB1, DTGB0HS = PWMB1
6	<b>OTP</b>	RW	0x0	One Time Preservation 1b: DTGB0HS high time is same as PWMB1 high time and is shifted by <b>DTGB0LED</b> 0b: DTGB0HS high time is reduced by <b>DTGB0LED</b>
5	<b>INVHS</b>	RW	0x0	Invert DTGB0HS output signal 1b: invert DTGB0HS 0b: do not invert DTGB0HS
4	<b>INVLS</b>	RW	0x0	Invert DTGB0LS output signal 1b: invert DTGB0LS 0b: do not invert DTGB0LS
3:0	<b>Reserved</b>	RO	0x0	Reserved

### 12.1.14. DTGB0LED

#### Register 12-13. DTGB0LED (Timer B Dead Time Generator 0 Leading Edge Delay, 0x400E 00A4)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	<b>Reserved</b>	RO	0x0	Reserved
11:0	<b>LED</b>	RW	0x0	Counter value DTGB0 leading edge dead time in clock cycles defined by <b>TBCTL.DTGCLK</b>

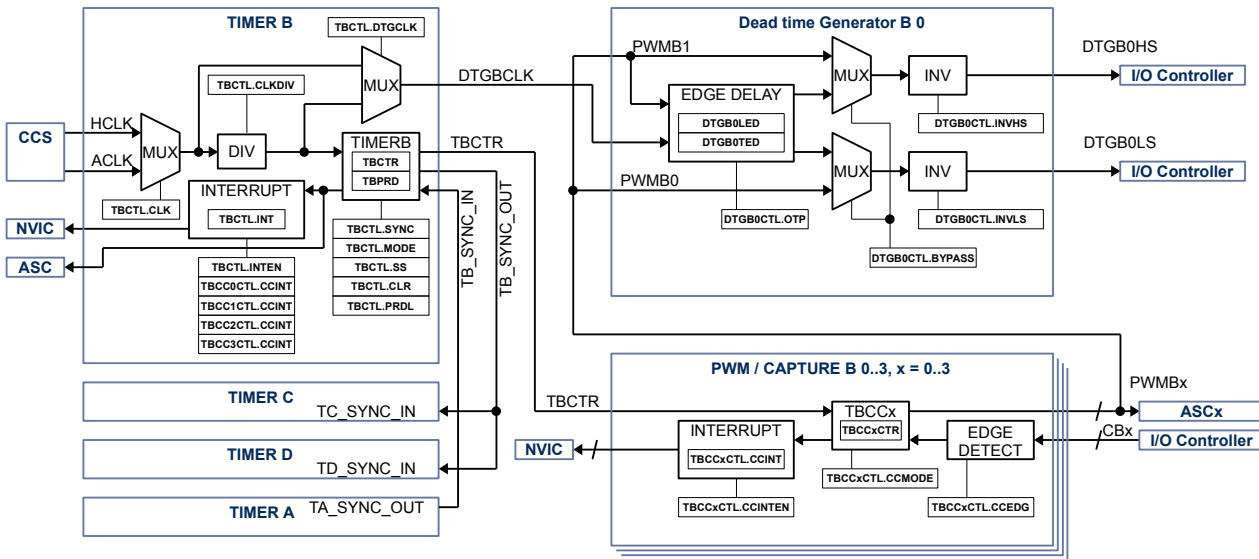
**12.1.15. DTGB0TED****Register 12-14. DTGB0TED (Timer B Dead Time Generator 0 Trailing Edge Delay, 0x400E 00A8)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	<b>Reserved</b>	RO	0x0	Reserved
11:0	<b>TED</b>	RW	0x0	Counter value DTGB0 trailing edge dead time in clock cycles defined by <b>TBCTL.DTGCLK</b>

## 12.2. Details of Operation

### 12.2.1. Block Diagram

Figure 12-1. Timer B



### 12.2.2. Configuration

Following blocks need to be configured for correct use of the Timer B:

- Clock Control System (CCS)
- Nested Vectored Interrupt Controller (NVIC)
- IO Controller
- Gate Driver
- Auto sequencing controller (ASC)
- Timer A
- Timer C
- Timer D
- 

### 12.2.3. Timer B Block

The timer B block consist of a 16-bit timer with up mode or up/down mode with 4 PWM/capture units and 1 dead-time generator unit.

### 12.2.4. Timer

Once enabled the timer counts up to the Timer B period value **TBPRD**. The **TBPRD** register can be written to while the timer is running, the new **TBPRD** value will be latched when the counter reaches old **TBPRD** value in up mode. In up/down mode there is the option to latch the new **TBPRD** value when counter counts back to zero.

**TBCTL.PRDL** configures when the timer will be updated with the new **TBPRD** value in up/down mode.

The current timer value is accessible with the timer B counter value register **TBCTR**.

#### 12.2.5. Register update

The **TBPRD**, **TBCCxCTR** register can be written to while the timer is running, the new **TBPRD**, **TBCCxCTR** value will be latched when the counter reaches old **TBPRD** value in up mode. In up/down mode there is the option to latch the new **TBPRD**, **TBCCxCTR** values when counter counts back to zero. **TBCTL.PRDL** configures when the timer will be updated with the new **TBPRD**, **TBCCxCTR** value in up/down mode.

#### 12.2.6. Timer Modes

The timer supports 3 modes of operation: disabled, up and up/down using **TBCTL.MODE**.

By default, the timer mode is disabled. When the timer is disabled, the timer counter does not increment or decrement. If the timer is disabled when previously in up or up/down mode, then the timer counter stops where it is. If the timer is re-enabled by putting it back into up or up/down modes, then the counter continues from the point at which it was disabled. To reset the current counter value **TBCTR** to zero use **TBCTL.CLR**.

In up mode, the timer starts counting from 0 up to the value of **TBPRD**. When the timer counter reaches the value of **TBPRD**, then the timer counter is reset to a value of 0. This mode is typically used for timed events or edge-aligned PWM output.

In up/down mode, the timer starts counting from 0 up to the value of **TBPRD**, and then back down to a value of 0. This timer mode is typically used for center-aligned PWM output. It can also be used for timed events, and will allow a longer timer range due to the fact it counts up and down.

#### 12.2.7. Single Shot Mode

The timer can be configured to run either once or continuously.

When the timer is configured in single shot mode using **TBCTL.SS** the timer will only count to **TBPRD** value and stops in up mode. In up/down mode the timer will count to **TBPRD** and back to zero only once.

To start the timer in single-shot mode, **TBCTL.SS** must be set. The timer will start when **TBCTL.CLR** is set. To re-start a single-shot timer, **TBCTL.CLR** must be reset, and then set again.

#### 12.2.8. Input Clock And Pre-Scaler

The timer can be configured to use HCLK or ACLK using **TBCTL.CLK**. The input clock for the can be divided further down from /1 to /128 using the **TBCTL.CLKDIV**.

#### 12.2.9. Timer Synchronization

The Timer A, B, C, D in the system have the ability to have synchronization between them. Each timer has a synchronization in signal (**SYNC\_IN**) and the synchronization out signal (**SYNC\_OUT**).

Timer B can be synchronized with Timer C, and D with timer B as master. Timer B can be synchronized with Timer A as slave.

The timer asserts the **SYNC\_OUT** pulse when the **TBCTL.CLR** bit is set and de-asserts the **SYNC\_OUT** pulse when the **TBCTL.CLR** bit is cleared.

Each timer C, or D that need to be synchronized as slave with master timer B need to set the **TxCTL.SYNC** bit. If this bit is not set, then the sync\_in signal is ignored and the timer operates independently.

When the **TxCTL.SYNC** bit is set and the SYNC\_IN signal is asserted, the timer clears the timer counter. The timer counter is also cleared anytime the **TxCTL.CLR** bit is set to a 1. When the **TxCTL.SYNC** bit is set and the SYNC\_IN signal is de-asserted and the timer mode is either up or up/down, then the timer will start counting. The timer will not start counting when the mode is set to up or up/down unless the SYNC\_IN signal is de-asserted when **TxCTL.SYNC** is set.

**NOTE:**

In order for this feature to work correctly, all timers that are synchronized must be set to the same mode (up or up/down), with the same timer pre-scaler, timer clock input and timer period.

To enable synchronized timers, the following steps should be followed:

1. All slave timers B, C, or D are configured with the selected timer input clock, timer pre-scaler, timer period and set the **TxCTL.SYNC** bit. The timer should still be set to disabled at this point.
2. The master timer A, or B is configured with the same timer input clock, timer pre-scaler, timer period and sets the **TxCTL.CLR** bit. This should clear all timer counters of the master and slave timers.
3. The slave timers set the timer mode to the desired state (either up or up/down).
4. The master timer sets the timer mode to either up or up/down and clears the **TxCTL.CLR** bit. This should start the master and all slave timers simultaneously based on the selected timer clock input.
5. Once configured as above, all timers can be disabled by the master setting **TxCTL.CLR** signal, to assert the SYNC\_OUT signal. The timers can be re-enabled by clearing the **TxCTL.CLR** bit, which de-asserts the SYNC\_OUT signal.
- 6.

### 12.2.10. PWM/Compare Units

Timer B supports up to 4 PWM/Capture units PWMB0 to PWMB3. Each PWM/Compare unit can be configured independently in PWM mode or capture mode using **TBCCxCTL.CCMODE**.

#### 12.2.10.1. PWM Mode

The PWM mode is enabled with setting **TBCCxCTL.CCMODE** to 0.

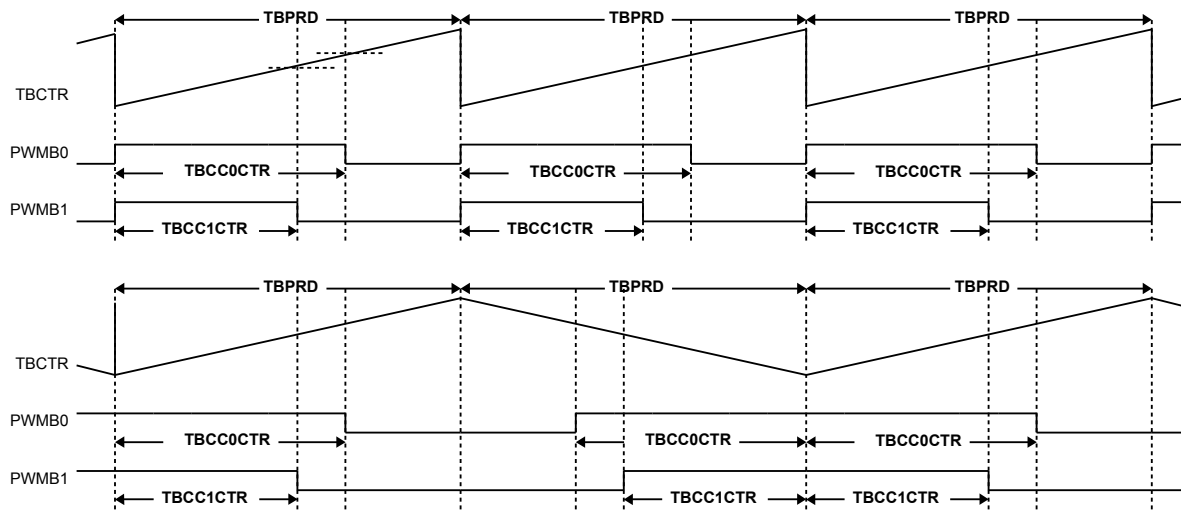
The timer configuration allows either edge-aligned (timer in up mode) or center-aligned (timer in up/down mode) modes of PWM operation.

In both edge-aligned and center-aligned modes of operation, the timer block outputs a PWM waveform that starts out high at a **TBCTR** value of 0 and then transitions to low when **TBCTR** counts up to **TBCCxCTR** compare value.

To configure a duty cycle of 0%, the **TBCCxCTR** should be set to 0; to configure a duty cycle of 100%, the **TBCCxCTR** value should be set to a value greater than or equal to **TBPRD**.

The polarity of the timer PWM outputs are not configurable. Adjustments to the polarity of the PWM outputs may be adjusted in the Dead-Time Generator (DTG) unit connected to the timer peripheral for each output independently.

**Figure 12-2. PWMB0 and PWMB1 Example Using Timer B Up Mode and Up/Down Mode**

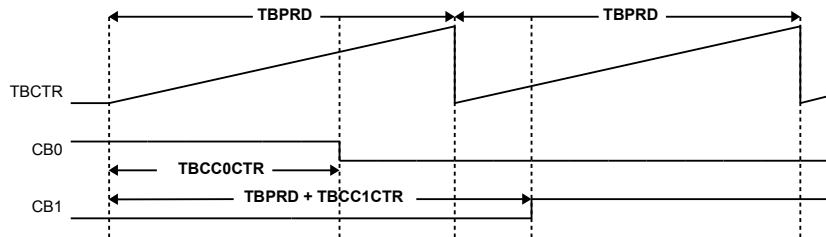


**12.2.10.2. Capture Mode**

The Capture mode is enabled with setting **TBCCxCTL.CCMODE** to 1. The trip condition for capture mode can be configured using **TBCCxCTL.CCEDGE**, high-to-low signal edge transition, low-to-high signal edge transition or both.

When trip condition is detected the actual **TBCTR** value is copied into **TBCCxCTR**.

**Figure 12-3. CB0 and CB1 Capture Example**



**12.2.11. Timer and PWM/Capture Interrupt**

The timer may generate interrupt based on the base timer wrap, or when a capture and compare event occurs.

In the base timer both up and up/down timer modes allow an interrupt to be generated when the count reaches 0. Each time the count reaches zero, the **TBCTL.INT** interrupt flag is set. If the interrupt is enabled using the **TBCTL.INTEN**, then the Timer IRQ signal will be asserted to the CPU. The interrupt flag may be cleared by writing a 1 to the **TBCTL.INT** interrupt flag bit.

In the capture and compare PWM units, each time a compare threshold is reached or each time a capture event is detected the **TBCCxCTL.CCINT** bit will be set for that particular timer unit. If the interrupt is enabled via the **TBCCxCTL.CCINTEN**, then the Timer IRQ signal will be asserted to the CPU. The interrupt flag may be cleared by writing a 1 to the **TBCCxCTL.CCINT** interrupt flag bit.

The timer IRQ signal will be asserted if any of the timer interrupt flags **TBCTL.INT** or **TBCCxCTL.CCINT** are set. The Timer IRQ signal will be de-asserted when all of the timer interrupt flags are cleared.

### 12.2.12. Dead-Time Generator

The dead-time generator can be configured to introduce dead-time for a complementary PWM output. The Timer B block supports up to 1 dead time generator.

#### 12.2.12.1. Dead Time Input Clock Selection

The clock source for the DTGB0 can be selected using **TBCTL.DTGCLK**.

Clear **TBCTL.DTGCLK** to 0 to use clock source selected by **TBCTL.CLK** directly to use higher resolution for dead time insertion.

Set **TBCTL.DTGCLK** to 1 to use divided clock source selected by **TBCTL.CLK** and **TBCTL.CLKDIV** divider to use the same dead time resolution as Timer B.

#### 12.2.12.2. Dead Time Range

The resolution for leading edge and trailing edge dead time is 12bits. Leading and trailing edge can be set independently using **DTGB0LED** and **DTGB0TED**.

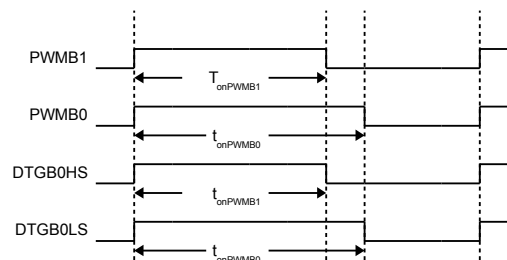
#### 12.2.12.3. Bypass Mode

Set **DTGB0CTL.BYPASS** to 0 to enable dead time insertion.

Set **DTGB0CTL.BYPASS** to 1 to enable bypass mode, no dead time is inserted, PWMB1 is routed to DTGB0HS and PWMB0 is routed to DTGB0LS.

The DTGB0HS and DTGB0LS signals can be inverted in bypass mode.

**Figure 12-4. DTGB0 Bypass Example**



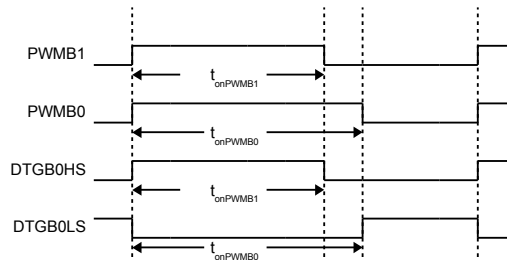
#### 12.2.12.4. Inverting PWM Signal

The DTG output signals DTGB0HS and DTGB0LS can be inverted independently.

Set **DTGB0CTL.INVHS** to invert DTGB0HS signal.

Set **DTGB0CTL.INVLS** to invert DTGB0LS signal.

**Figure 12-5. DTGB0 Bypass and Inverting LS Example**



**12.2.12.5. Dead Time Insertion**

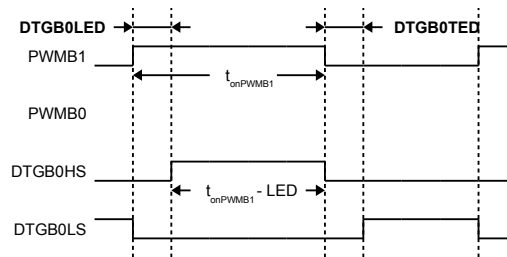
Set **DTGB0CTL.BYPASS** to 0 to enable dead time insertion. In dead time insertion mode only PWMB1 signal is used to generate DTGB0HS and DTGB0LS. PWMB0 signal is ignored and can be used for other purposes.

Set **DTGB0LED** for desired leading-edge and **DTGB0TED** for desired trailing edge in clock-cycles defined by **TBCTL.DTGCLK** clock source

**NOTE:**

In dead time insertion mode the DTGB0LS signal is automatically inverted compared to PWMB1 signal. Set **DTGB0CTL.INVLS** to 0, if this is desired behavior.

**Figure 12-6. DTGB0 LED and TED Example**



**12.2.12.6. Dead Time Insertion with On Time Preservation**

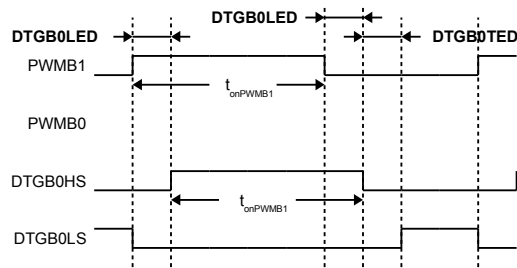
Set **DTGB0CTL.OTP** to 1 to enable on time preservation. In this mode the DTGB0HS is same as PWMB1 on time.

**NOTE:**

In dead time insertion mode the DTGB0LS signal is automatically inverted compared to PWMB1 signal. Set **DTGB0CTL.INVLS** to 0, if this is desired behavior.



**Figure 12-7. DTGB0 LED and TED with On Time Preservation Example**



### 12.2.13. PWM Output and Capture Input Pin Selection

Each of the DTGB0HS, DTGB0LS outputs, and CBx inputs can be routed to different I/Os, allowing great flexibility in pin assignments.

In capture mode only one I/O should be enabled as input to the capture. If more than one pin input is enabled, the capture might not work properly.

**Note:**

Not all pins are available pending package option, consult data sheet for available pins and signals.

**Table 12-2. Timer B Signal to Pin Mapping**

PWM	CAPTURE	DEADTIME	PINS
PWMB0	CB0	DTGB0LS	PA3, PA6, PD2
PWMB1	CB1	DTGB0HS	PD3, PD6
PWMB2	CB2	N/A	N/A
PWMB3	CB3	N/A	N/A

## 13. TIMER C

### 13.1. Register

#### 13.1.1. Register Map

Table 13-1. Timer C Register Map

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>Timer C</b>			
0x400F 0000	<b>TCCTL</b>	Timer C control	0x0000 0000
0x400F 0004	<b>TCPRD</b>	Timer C period	0x0000 0000
0x400F 0008	<b>TCCTR</b>	Timer C counter	0x0000 0000
<b>Timer C PWM Capture and Compare</b>			
0x400F 0040	<b>TCCCTL0</b>	Timer C capture and compare 0 control	0x0000 0000
0x400F 0044	<b>TCCTR0</b>	Timer C counter 0	0x0000 0000
0x400F 0048	<b>TCCCTL1</b>	Timer C capture and compare 1 control	0x0000 0000
0x400F 004C	<b>TCCTR1</b>	Timer C counter 1	0x0000 0000
<b>Timer C Dead Time Generator</b>			
0x400F 00A0	<b>DTGC0CTL</b>	Timer C dead time generator 0 control	0x0000 0080
0x400F 00A4	<b>DTGC0LED</b>	Timer C dead time generator 0 leading edge delay	0x0000 0000
0x400F 00A8	<b>DTGC0TED</b>	Timer C dead time generator 0 trailing edge delay	0x0000 0000

#### 13.1.2. TCCTL

Register 13-1. TCCTL (Timer C Control, 0x400F 0000)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:14	<b>Reserved</b>	RO	0x0	Reserved
13	<b>DTGCLK</b>	RW	0x0	DTGCx clock select 1b: DTGCx use clock after <b>TCCTL.CLKDIV</b> 0b: DTGCx use clock selected by <b>TCCTL.CLK</b>
12	<b>SYNC</b>	RW	0x0	Timer C synchronization 1b: Synchronize Timer C, enable SYNC_IN 0b: Do not synchronize Timer C, disabled SYNC_IN
11:10	<b>MODE</b>	RW	0x0	Timer C Mode 11b: reserved 10b: up / down 01b: up 00b: disabled
9	<b>CLK</b>	RW	0x0	Timer C clock input source 1b: ACLK 0b: HCLK

BIT	NAME	ACCESS	RESET	DESCRIPTION
8:6	<b>CLKDIV</b>	RW	0x0	Timer C input clock divider 111b: /128 110b: /64 101b: /32 100b: /16 011b: /8 010b: /4 001b: /2 000b: /1
5	<b>INTEN</b>	RW	0x0	Timer C interrupt enable 1b: enable Timer C interrupt 0b: disable Timer C interrupt
4	<b>INT</b>	RW1C	0x0	Timer C interrupt 1b: interrupt, clear by write 1b 0b: no interrupt
3	<b>SS</b>	RW	0x0	Timer C single shot 1b: single shot mode 0b: continuous timer mode
2	<b>CLR</b>	RW	0x0	Timer C clear 1b: Clear Timer C, hold Timer C in reset and set SYNC_OUT 0b: Do not clear Timer and clear SYNC_OUT
1	<b>Reserved</b>	RO	0x0	Reserved
0	<b>PRDL</b>	RW	0x0	Timer C <b>TCPRD</b> update 1b: Latch new <b>TCPRD</b> value when Timer C counting down, <b>TCCTR</b> value = 0x1 and <b>TCCTL.MODE</b> = 10b 0b: Latch new <b>TCPRD</b> value when Timer C counting up and <b>TCCTR</b> value = <b>TCPRD</b> – 0x1.

### 13.1.3. TCPRD

**Register 13-2. TCPRD (Timer C Period, 0x400F 0004)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0	Reserved
15:0	<b>PERIOD</b>	RW	0x0	Timer C period value

### 13.1.4. TCCTR

**Register 13-3. TCCTR (Timer C Counter, 0x400F 0008)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CTR</b>	RW	0x0	Current Timer C counter value

### 13.1.5. TCCC0CTL

**Register 13-4. TCCC0CTL (Timer C PWMC0 Capture and Compare Control, 0x400F 0040)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved

BIT	NAME	ACCESS	RESET	DESCRIPTION
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMC0 input 0b: Compare mode PWMC0 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW1C	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMC0 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

### 13.1.6. TCCC0CTR

**Register 13-5. TCCC0CTR (Timer C PWMC0 Capture and Compare Counter, 0x400F 0044)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMC0 compare mode or counter value for PWMC0 capture mode

### 13.1.7. TCCC1CTL

**Register 13-6. TCCC1CTL (Timer C PWMC1 Capture and Compare Control, 0x400F 0048)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMC1 input 0b: Compare mode PWMC1 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW1C	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMC1 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

### 13.1.8. TCCC1CTR

**Register 13-7. TCCC1CTR (Timer C PWMC1 Capture and Compare Counter, 0x400F 004C)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved

BIT	NAME	ACCESS	RESET	DESCRIPTION
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMC1 compare mode or counter value for PWMC1 capture mode

### 13.1.9. DTGC0CTL

**Register 13-8. DTGC0CTL (Timer C Dead Time Generator 0 Control, 0x400F 00A0)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>BYPASS</b>	RW	0x1	Bypass dead time generator 1b: DTGC0 bypass active, DTGC0LS = PWMC0, DTGC0HS = PWMC1 0b: DTGC0 bypass inactive, dead time inserted to DTGC0LS and DTGC0HS, DTGC0LS = PWMC1, DTGC0HS = PWMC1
6	<b>OTP</b>	RW	0x0	One Time Preservation 1b: DTGC0HS high time is same as PWMC1 high time is shifted by <b>DTGC0LED</b> 0b: DTGC0HS high time is reduced by <b>DTGC0LED</b>
5	<b>INVHS</b>	RW	0x0	Invert DTGC0HS output signal 1b: invert DTGC0HS 0b: do not invert DTGC0HS
4	<b>INVLS</b>	RW	0x0	Invert DTGC0LS output signal 1b: invert DTGC0LS 0b: do not invert DTGC0LS
3:0	<b>Reserved</b>	RO	0x0	Reserved

### 13.1.10. DTGC0LED

**Register 13-9. DTGC0LED (Timer C Dead Time Generator 0 Leading Edge Delay, 0x400F 00A4)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	<b>Reserved</b>	RO	0x0	Reserved
11:0	<b>LED</b>	RW	0x0	Counter value DTGC0 leading edge dead time in clock cycles defined by <b>TCCTL.DTGCLK</b>

### 13.1.11. DTGC0TED

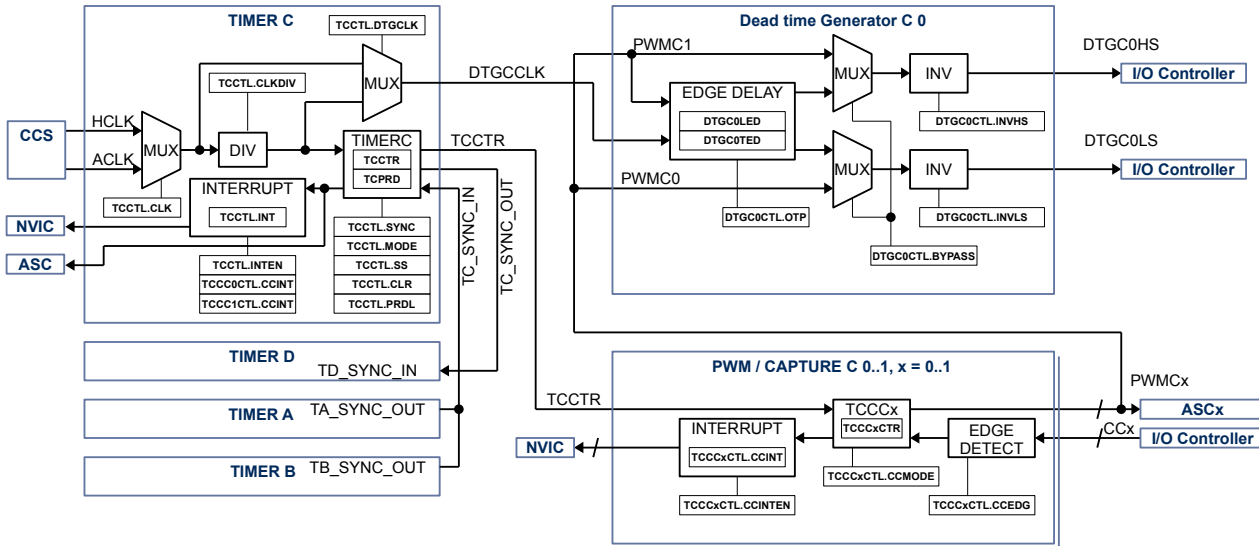
**Register 13-10. DTGC0TED (Timer C Dead Time Generator 0 Trailing Edge Delay, 0x400F 00A8)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	<b>Reserved</b>	RO	0x0	Reserved
11:0	<b>TED</b>	RW	0x0	Counter value DTGC0 trailing edge dead time in clock cycles defined by <b>TCCTL.DTGCLK</b>

## 13.2. Details of Operation

### 13.2.1. Block Diagram

Figure 13-1. Timer C



### 13.2.2. Configuration

Following blocks need to be configured for correct use of the Timer C:

- Clock Control System (CCS)
- Nested Vectored Interrupt Controller (NVIC)
- IO Controller
- Gate Driver
- Auto sequencing controller (ASC)
- Timer A
- Timer B
- Timer D
- 

### 13.2.3. Timer C Block

The timer C block consist of a 16-bit timer with up mode or up/down mode with 2 PWM/capture units and 1 dead-time generator unit.

### 13.2.4. Timer

Once enabled the timer counts up to the Timer C period value **TCPRD**. The **TCPRD** register can be written to while the timer is running, the new **TCPRD** value will be latched when the counter reaches old **TCPRD** value in up mode. In up/down mode there is the option to latch the new **TCPRD** value when counter counts back to zero.

**TCCTL.PRDL** configures when the timer will be updated with the new **TCPRD** value in up/down mode.

The current timer value is accessible with the timer C counter value register **TCCTR**.

### 13.2.5. Register update

The **TCPRD**, **TCCCxCTR** register can be written to while the timer is running, the new **TCPRD**, **TCCCxCTR** value will be latched when the counter reaches old **TCPRD** value in up mode. In up/down mode there is the option to latch the new **TCPRD**, **TCCCxCTR** values when counter counts back to zero. **TCCTL.PRDL** configures when the timer will be updated with the new **TCPRD**, **TCCCxCTR** value in up/down mode.

### 13.2.6. Timer Modes

The timer supports 3 modes of operation: disabled, up and up/down using **TCCTL.MODE**.

By default, the timer mode is disabled. When the timer is disabled, the timer counter does not increment or decrement. If the timer is disabled when previously in up or up/down mode, then the timer counter stops where it is. If the timer is re-enabled by putting it back into up or up/down modes, then the counter continues from the point at which it was disabled. To reset the current counter value **TCCTR** to zero use **TCCTL.CLR**.

In up mode, the timer starts counting from 0 up to the value of **TCPRD**. When the timer counter reaches the value of **TCPRD**, then the timer counter is reset to a value of 0. This mode is typically used for timed events or edge-aligned PWM output.

In up/down mode, the timer starts counting from 0 up to the value of **TCPRD**, and then back down to a value of 0. This timer mode is typically used for center-aligned PWM output. It can also be used for timed events, and will allow a longer timer range due to the fact it counts up and down

### 13.2.7. Single Shot Mode

The timer can be configured to run either once or continuously.

When the timer is configured in single shot mode using **TCCTL.SS** the timer will only count to **TCPRD** value and stops in up mode. In up/down mode the timer will count to **TCPRD** and back to zero only once.

To start the timer in single-shot mode, **TCCTL.SS** must be set. The timer will start when **TCCTL.CLR** is set. To re-start a single-shot timer, **TCCTL.CLR** must be reset, and then set again.

### 13.2.8. Input clock and Pre-scaler

The timer can be configured to use HCLK or ACLK using **TCCTL.CLK**. The input clock for the can be divided further down from /1 to /128 using the **TCCTL.CLKDIV**.

### 13.2.9. Timer synchronization

The Timer A, B, C, D in the system have the ability to have synchronization between them. Each timer has a synchronization in signal (**SYNC\_IN**) and the synchronization out signal (**SYNC\_OUT**).

Timer C can be synchronized with Timer D as master. Timer B can be synchronized with Timer A, and B as slave.

The timer asserts the **SYNC\_OUT** pulse when the **TCCTL.CLR** bit is set and de-asserts the **SYNC\_OUT** pulse when the **TCCTL.CLR** bit is cleared.

If timer D that need to be synchronized as slave with master timer C need to set the **TxCTL.SYNC** bit. If this is bit is not set, then the sync\_in signal is ignored and the timer operates independently.

When the **TxCTL.SYNC** bit is set and the SYNC\_IN signal is asserted, the timer clears the timer counter. The timer counter is also cleared anytime the **TxCTL.CLR** bit is set to a 1. When the **TxCTL.SYNC** bit is set and the SYNC\_IN signal is de-asserted and the timer mode is either up or up/down, then the timer will start counting. The timer will not start counting when the mode is set to up or up/down unless the SYNC\_IN signal is de-asserted when **TxCTL.SYNC** is set.

**NOTE:**

In order for this feature to work correctly, all timers that are synchronized must be set to the same mode (up or up/down), with the same timer pre-scaler, timer clock input and timer period.

To enable synchronized timers, the following steps should be followed:

1. The slave timer C, D is configured with the selected timer input clock, timer pre-scaler, timer period and set the **TxCTL.SYNC** bit. The timer should still be set to disabled at this point.
2. The master timer A, or B is configured with the same timer input clock, timer pre-scaler, timer period and sets the **TxCTL.CLR** bit. This should clear all timer counters of the master and slave timers.
3. The slave timers set the timer mode to the desired state (either up or up/down).
4. The master timer sets the timer mode to either up or up/down and clears the **TxCTL.CLR** bit. This should start the master and all slave timers simultaneously based on the selected timer clock input.
5. Once configured as above, all timers can be disabled by the master setting **TxCTL.CLR** signal, to assert the SYNC\_OUT signal. The timers can be re-enabled by clearing the **TxCTL.CLR** bit, which de-asserts the SYNC\_OUT signal.

### 13.2.10. PWM/Compare Units

Timer C supports up to 2 PWM/Capture units PWMC0 to PWMC1. Each PWM/Compare unit can be configured independently in PWM mode or capture mode using **TCCxCTL.CCMODE**.

#### 13.2.10.1. PWM Mode

The PWM mode is enabled with setting **TCCxCTL.CCMODE** to 0.

The timer configuration allows either edge-aligned (timer in up mode) or center-aligned (timer in up/down mode) modes of PWM operation.

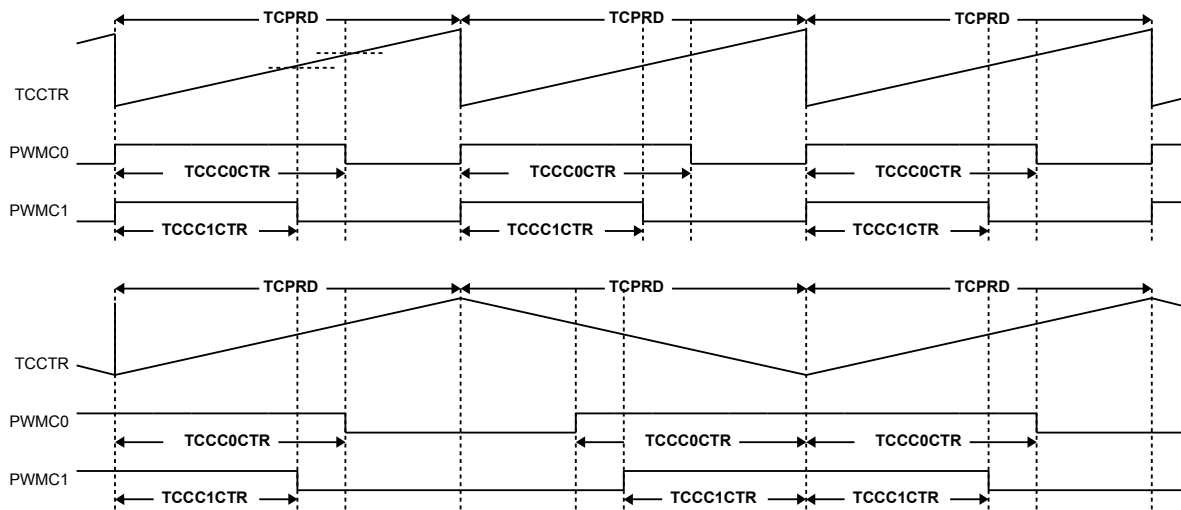
In both edge-aligned and center-aligned modes of operation, the timer block outputs a PWM waveform that starts out high at a **TCCTR** value of 0 and then transitions to low when **TCCTR** counts up to **TCCxCTR** compare value.

To configure a duty cycle of 0%, the **TCCxCTR** should be set to 0; to configure a duty cycle of 100%, the **TCCxCTR** value should be set to a value greater than or equal to **TCPRD**.

The polarity of the timer PWM outputs are not configurable. Adjustments to the polarity of the PWM outputs may be adjusted in the Dead-Time Generator (DTG) unit connected to the timer peripheral for each output independently.



**Figure 13-2. PWMC0 and PWMC1 Example Using Timer C Up Mode and Up/Down Mode**

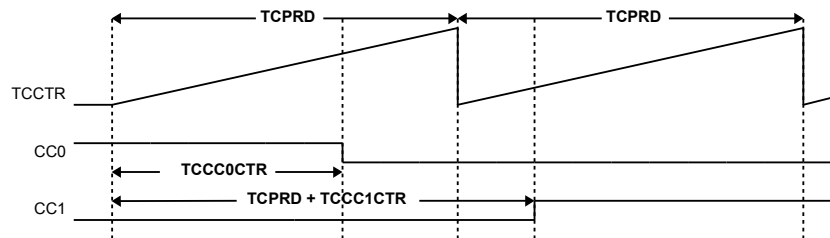


**13.2.10.2. Capture Mode**

The Capture mode is enabled with setting **TCCCxCTL.CCMODE** to 1. The trip condition for capture mode can be configured using **TCCCxCTL.CCEDGE**, high-to-low signal edge transition, low-to-high signal edge transition or both.

When trip condition is detected the actual **TCCTR** value is copied into **TCCCxCTR**.

**Figure 13-3. CC0 and CC1 Capture Example**



**13.2.11. Timer and PWM/Capture Interrupt**

The timer may generate interrupt based on the base timer wrap, or when a capture and compare event occurs.

In the base timer both up and up/down timer modes allow an interrupt to be generated when the count reaches 0. Each time the count reaches zero, the **TCCTL.INT** interrupt flag is set. If the interrupt is enabled using the **TCCTL.INTEN**, then the Timer IRQ signal will be asserted to the CPU. The interrupt flag may be cleared by writing a 1 to the **TCCTL.INT** interrupt flag bit.

In the capture and compare PWM units, each time a compare threshold is reached or each time a capture event is detected the **TCCCxCTL.CCINT** bit will be set for that particular timer unit. If the interrupt is enabled via the **TCCCxCTL.CCINTEN**, then the Timer IRQ signal will be asserted to the CPU. The interrupt flag may be cleared by writing a 1 to the **TCCCxCTL.CCINT** interrupt flag bit.

The timer IRQ signal will be asserted if any of the timer interrupt flags **TCCTL.INT** or **TCCCxCTL.CCINT** are set. The Timer IRQ signal will be de-asserted when all of the timer interrupt flags are cleared.

### 13.2.12. Dead-Time Generator

The dead-time generator can be configured to introduce dead-time for a complementary PWM output. The Timer C block supports up to 1 dead time generator.

#### 13.2.12.1. Dead Time Input Clock Selection

The clock source for the DTGC0 can be selected using **TCCTL.DTGCLK**.

Clear **TCCTL.DTGCLK** to 0 to use clock source selected by **TCCTL.CLK** directly to use higher resolution for dead time insertion.

Set **TCCTL.DTGCLK** to 1 to use divided clock source selected by **TCCTL.CLK** and **TCCTL.CLKDIV** divider to use the same dead time resolution as Timer C.

#### 13.2.12.2. Dead Time Range

The resolution for leading edge and trailing edge dead time is 12bits. Leading and trailing edge can be set independently using **DTGC0LED** and **DTGC0TED**.

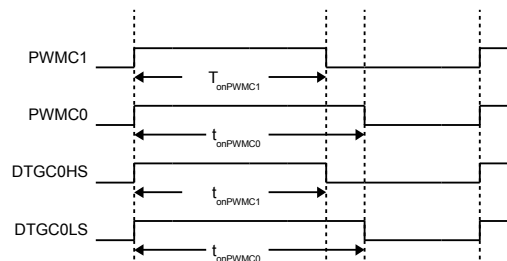
#### 13.2.12.3. Bypass Mode

Set **DTGC0CTL.BYPASS** to 0 to enable dead time insertion.

Set **DTGC0CTL.BYPASS** to 1 to enable bypass mode, no dead time is inserted, PWMC1 is routed to DTGC0HS and PWMC0 is routed to DTGC0LS.

The DTGC0HS and DTGC0LS signals can be inverted in bypass mode.

**Figure 13-4. DTGC0 Bypass Example**



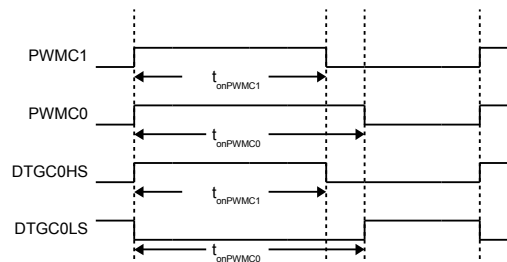
#### 13.2.12.4. Inverting PWM Signal

The DTG output signals DTGC0HS and DTGC0LS can be inverted independently.

Set **DTGC0CTL.INVHS** to invert DTGC0HS signal.

Set **DTGC0CTL.INVLS** to invert DTGC0LS signal.

**Figure 13-5. DTGC0 Bypass and Inverting LS Example**



**13.2.12.5. Dead Time Insertion**

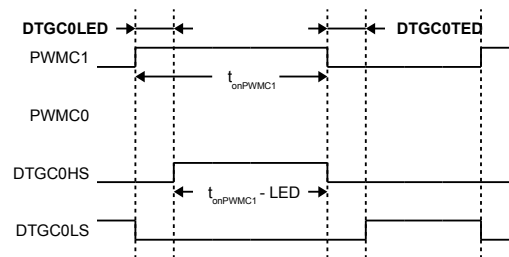
Set **DTGC0CTL.BYPASS** to 0 to enable dead time insertion. In dead time insertion mode only PWM1 signal is used to generate DTGC0HS and DTGC0LS. PWM0 signal is ignored and can be used for other purposes.

Set **DTGC0LED** for desired leading-edge and **DTGC0TED** for desired trailing edge in clock-cycles defined by **TCCTL.DTGCLK** clock source

**NOTE:**

In dead time insertion mode the DTGC0LS signal is automatically inverted compared to PWM1 signal. Set **DTGC0CTL.INVLS** to 0, if this is desired behavior.

**Figure 13-6. DTGC0 LED and TED Example**



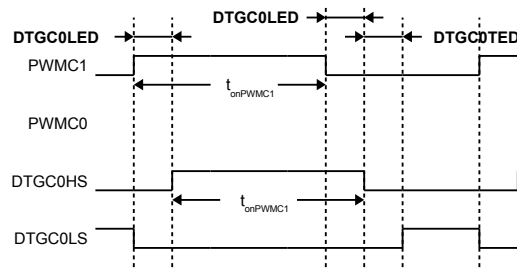
**13.2.12.6. Dead Time Insertion With On Time Preservation**

Set **DTGC0CTL.OTP** to 1 to enable on time preservation. In this mode the DTGC0HS is same as PWM1 on time.

**NOTE:**

In dead time insertion mode the DTGC0LS signal is automatically inverted compared to PWM1 signal. Set **DTGC0CTL.INVLS** to 0, if this is desired behavior.

**Figure 13-7. DTGC0 LED and TED with On Time Preservation Example**



**13.2.13. PWM Output and Capture Input Pin Selection**

Each of the DTGC0HS, DTGC0LS outputs, and CCx inputs can be routed to different I/Os, allowing great flexibility in pin assignments.

In capture mode only one I/O should be enabled as input to the capture. If more than one pin input is enabled, the capture might not work properly.

**Note:**

Not all pins are available pending package option, consult data sheet for available pins and signals.

**Table 13-2. Timer C Signal to Pin Mapping**

PWM	CAPTURE	DEADTIME	PINS
PWMC0	CC0	DTGC0LS	PA4
PWMC1	CC1	DTGC1LS	PA7, PD5

## 14. TIMER D

### 14.1. Register

#### 14.1.1. Register Map

Table 14-1. Timer D Register Map

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>Timer D</b>			
0x4010 0000	<b>TDCTL</b>	Timer D control	0x0000 0000
0x4010 0004	<b>TDPRD</b>	Timer D period	0x0000 0000
0x4010 0008	<b>TDCTR</b>	Timer D counter	0x0000 0000
<b>Timer D PWM Capture and Compare</b>			
0x4010 0040	<b>TDCCTL0</b>	Timer D capture and compare 0 control	0x0000 0000
0x4010 0044	<b>TDCTR0</b>	Timer D counter 0	0x0000 0000
0x4010 0048	<b>TDCCTL1</b>	Timer D capture and compare 1 control	0x0000 0000
0x4010 004C	<b>TDCTR1</b>	Timer D counter 1	0x0000 0000
<b>Timer D Dead Time Generator</b>			
0x4010 00A0	<b>DTGD0CTL</b>	Timer D dead time generator 0 control	0x0000 0080
0x4010 00A4	<b>DTGD0LED</b>	Timer D dead time generator 0 leading edge delay	0x0000 0000
0x4010 00A8	<b>DTGD0TED</b>	Timer D dead time generator 0 trailing edge delay	0x0000 0000

#### 14.1.2. TDCTL

Register 14-1. TDCTL (Timer D Control, 0x4010 0000)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:14	<b>Reserved</b>	RO	0x0	Reserved
13	<b>DTGCLK</b>	RW	0x0	DTGDx clock select 1b: DTGDx use clock after <b>TDCTL.CLKDIV</b> 0b: DTGDx use clock selected by <b>TDCTL.CLK</b>
12	<b>SYNC</b>	RW	0x0	Timer D synchronization 1b: Synchronize Timer D, enable SYNC_IN 0b: Do not synchronize Timer D, disabled SYNC_IN
11:10	<b>MODE</b>	RW	0x0	Timer D Mode 11b: reserved 10b: up / down 01b: up 00b: disabled
9	<b>CLK</b>	RW	0x0	Timer D clock input source 1b: ACLK 0b: HCLK

BIT	NAME	ACCESS	RESET	DESCRIPTION
8:6	<b>CLKDIV</b>	RW	0x0	Timer D input clock divider 111b: / 128 110b: / 64 101b: /32 100b: /16 011b: /8 010b: /4 001b: /2 000b: /1
5	<b>INTEN</b>	RW	0x0	Timer D interrupt enable 1b: enable Timer D interrupt 0b: disable Timer D interrupt
4	<b>INT</b>	RW1C	0x0	Timer D interrupt 1b: interrupt, clear by write 1b 0b: no interrupt
3	<b>SS</b>	RW	0x0	Timer D single shot 1b: single shot mode 0b: continuous timer mode
2	<b>CLR</b>	RW	0x0	Timer D clear 1b: Clear Timer D, hold Timer D in reset and set SYNC_OUT 0b: Do not clear Timer and clear SYNC_OUT
1	<b>Reserved</b>	RO	0x0	Reserved
0	<b>PRDL</b>	RW	0x0	Timer D TDPRD update 1b: Latch new TDPRD value when Timer D counting down, <b>TDCTR</b> value = 0x1 and <b>TDCTL.MODE</b> = 10b 0b: Latch new TDPRD value when Timer D counting up and <b>TDCTR</b> value = <b>TDPRD</b> – 0x1.

#### 14.1.3. TDPRD

##### Register 14-2. TDPRD (Timer D Period, 0x4010 0004)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0	Reserved
15:0	<b>PERIOD</b>	RW	0x0	Timer D period value

#### 14.1.4. TDCTR

##### Register 14-3. TDCTR (Timer D Counter, 0x4010 0008)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CTR</b>	RW	0x0	Current Timer D counter value

#### 14.1.5. TDCC0CTL

##### Register 14-4. TDCC0CTL (Timer D PWMD0 Capture and Compare Control, 0x4010 0040)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved

BIT	NAME	ACCESS	RESET	DESCRIPTION
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMD0 input 0b: Compare mode PWMD0 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW1C	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMD0 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

#### 14.1.6. TDCC0CTR

**Register 14-5. TDCC0CTR (Timer D PWMD0 Capture and Compare Counter, 0x4010 0044)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMD0 compare mode or counter value for PWMD0 capture mode

#### 14.1.7. TDCC1CTL

**Register 14-6. TDCC1CTL (Timer D PWMD1 Capture and Compare Control, 0x4010 0048)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	<b>Reserved</b>	RO	0x0	Reserved
4	<b>CCMODE</b>	RW	0x0	Capture and compare mode 1b: Capture mode PWMD1 input 0b: Compare mode PWMD1 output
3	<b>CCINTEN</b>	RW	0x0	Capture and compare interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>CCINT</b>	RW1C	0x0	Capture and compare interrupt 1b: interrupt, clear by write 1b 0b: no interrupt detected
1:0	<b>CCEDG</b>	RW	0x0	Capture mode edge detect PWMD1 11b: reserved 10b: high to low transition and low to high transition 01b: low to high transition only 00b: high to low transition only

#### 14.1.8. TDCC1CTR

**Register 14-7. TDCC1CTR (Timer D PWMD1 Capture and Compare Counter, 0x4010 004C)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	RO	0x0	Reserved

BIT	NAME	ACCESS	RESET	DESCRIPTION
15:0	<b>CCCTR</b>	RW	0x0	Counter value for PWMD1 compare mode or counter value for PWMD1 capture mode

#### 14.1.9. DTGD0CTL

##### Register 14-8. DTGD0CTL (Timer D Dead Time Generator 0 Control, 0x4010 00A0)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RO	0x0	Reserved
7	<b>BYPASS</b>	RW	0x1	Bypass dead time generator 1b: DTGD0 bypass active, DTGD0LS = PWMD0, DTGD0HS = PWMD1 0b: DTGD0 bypass inactive, dead time inserted to DTGD0LS and DTGD0HS, DTGD0LS = PWMD1, DTGD0HS = PWMD1
6	<b>OTP</b>	RW	0x0	One Time Preservation 1b: DTGD0HS high time is same as PWMD1 high time and is shifted by <b>DTGD0LED</b> 0b: DTGD0HS high time is reduced by <b>DTGD0LED</b>
5	<b>INVHS</b>	RW	0x0	Invert DTGD0HS output signal 1b: invert DTGD0HS 0b: do not invert DTGD0HS
4	<b>INVLS</b>	RW	0x0	Invert DTGD0LS output signal 1b: invert DTGD0LS 0b: do not invert DTGD0LS
3:0	<b>Reserved</b>	RO	0x0	Reserved

#### 14.1.10. DTGD0LED

##### Register 14-9. DTGD0LED (Timer D Dead Time Generator 0 Leading Edge Delay, 0x4010 00A4)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	<b>Reserved</b>	RO	0x0	Reserved
11:0	<b>LED</b>	RW	0x0	Counter value DTGD0 leading edge dead time in clock cycles defined by TDCTL.DTGCLK

#### 14.1.11. DTGD0TED

##### Register 14-10. DTGD0TED (Timer D Dead Time Generator 0 Trailing Edge Delay, 0x4010 00A8)

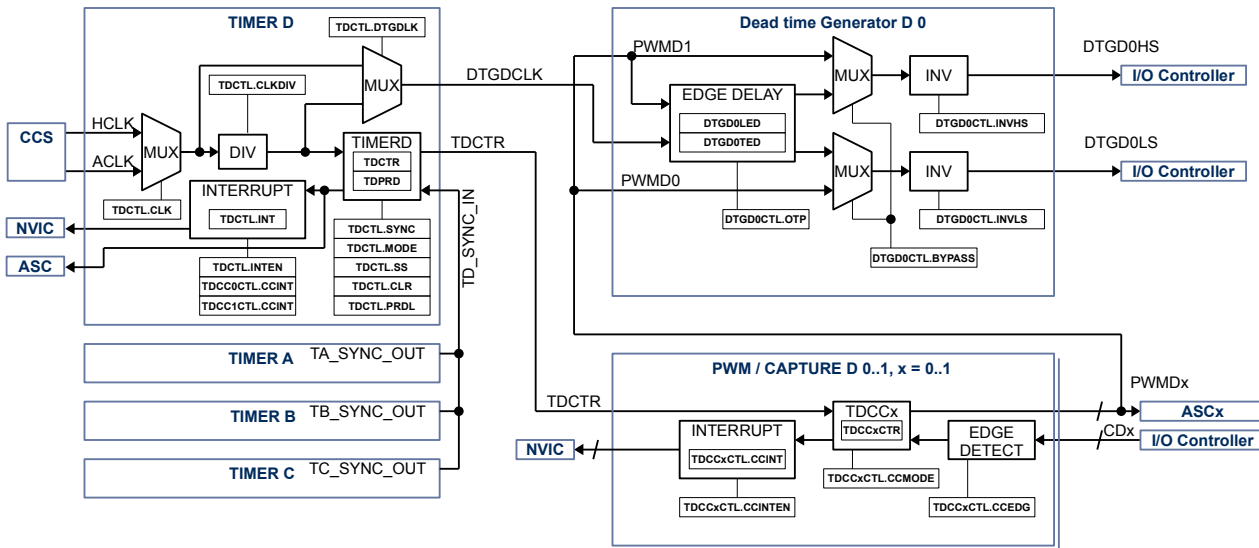
BIT	NAME	ACCESS	RESET	DESCRIPTION
31:12	<b>Reserved</b>	RO	0x0	Reserved
11:0	<b>TED</b>	RW	0x0	Counter value DTGD0 trailing edge dead time in clock cycles defined by TDCTL.DTGCLK



## 14.2. Details of Operation

### 14.2.1. Block Diagram

Figure 14-1. Timer D



### 14.2.2. Configuration

Following blocks need to be configured for correct use of the Timer D:

- Clock Control System (CCS)
- Nested Vectored Interrupt Controller (NVIC)
- IO Controller
- Gate Driver
- Auto sequencing controller (ASC)
- Timer A
- Timer B
- Timer C

### 14.2.3. Timer D Block

The timer D block consist of a 16-bit timer with up mode or up/down mode with 2 PWM/capture units and 1 dead-time generator unit.

### 14.2.4. Timer

Once enabled the timer counts up to the Timer D period value **TDPRD**. The **TDPRD** register can be written to while the timer is running, the new **TDPRD** value will be latched when the counter reaches old **TDPRD** value in up mode. In up/down mode there is the option to latch the new **TDPRD** value when counter counts back to zero. **TDCTL.PRDL** configures when the timer will be updated with the new **TDPRD** value in up/down mode.

The current timer value is accessible with the timer D counter value register **TDCTR**.

#### 14.2.5. Register Update

The **TDPRD**, **TDCCxCTR** register can be written to while the timer is running, the new **TDPRD**, **TDCCxCTR** value will be latched when the counter reaches old **TDPRD** value in up mode. In up/down mode there is the option to latch the new **TDPRD**, **TDCCxCTR** values when counter counts back to zero. **TDCTL.PRDL** configures when the timer will be updated with the new **TDPRD**, **TDCCxCTR** value in up/down mode.

#### 14.2.6. Timer Modes

The timer supports 3 modes of operation: disabled, up and up/down using **TDCTL.MODE**.

By default, the timer mode is disabled. When the timer is disabled, the timer counter does not increment or decrement. If the timer is disabled when previously in up or up/down mode, then the timer counter stops where it is. If the timer is re-enabled by putting it back into up or up/down modes, then the counter continues from the point at which it was disabled. To reset the current counter value **TDCTR** to zero use **TDCTL.CLR**.

In up mode, the timer starts counting from 0 up to the value of **TDPRD**. When the timer counter reaches the value of **TDPRD**, then the timer counter is reset to a value of 0. This mode is typically used for timed events or edge-aligned PWM output.

In up/down mode, the timer starts counting from 0 up to the value of **TDPRD**, and then back down to a value of 0. This timer mode is typically used for center-aligned PWM output. It can also be used for timed events, and will allow a longer timer range due to the fact it counts up and down

#### 14.2.7. Single Shot Mode

The timer can be configured to run either once or continuously.

When the timer is configured in single shot mode using **TDCTL.SS** the timer will only count to **TDPRD** value and stops in up mode. In up/down mode the timer will count to **TDPRD** and back to zero only once.

To start the timer in single-shot mode, **TDCTL.SS** must be set. The timer will start when **TDCTL.CLR** is set. To re-start a single-shot timer, **TDCTL.CLR** must be reset, and then set again.

#### 14.2.8. Input Clock And Pre-Scaler

The timer can be configured to use HCLK or ACLK using **TDCTL.CLK**. The input clock for the can be divided further down from /1 to /128 using the **TDCTL.CLKDIV**.

#### 14.2.9. Timer Synchronization

The Timer A, B, C, D in the system have the ability to have synchronization between them. Each timer has a synchronization in signal (**SYNC\_IN**) and the synchronization out signal (**SYNC\_OUT**).

Timer D can be synchronized with Timer A, B, or C as slave.

The timer asserts the **SYNC\_OUT** pulse when the **TDCTL.CLR** bit is set and de-asserts the **SYNC\_OUT** pulse when the **TDCTL.CLR** bit is cleared.

If timer D that need to be synchronized as slave with master timer C need to set the **TxCTL.SYNC** bit. If this is bit is not set, then the **sync\_in** signal is ignored and the timer operates independently.

When the **TxCTL.SYNC** bit is set and the **SYNC\_IN** signal is asserted, the timer clears the timer counter. The timer counter is also cleared anytime the **TxCTL.CLR** bit is set to a 1. When the **TxCTL.SYNC** bit is set and the **SYNC\_IN** signal is de-asserted and the timer mode is either up or up/down, then the timer will start counting. The timer will not start counting when the mode is set to up or up/down unless the **SYNC\_IN** signal is de-asserted when **TxCTL.SYNC** is set.

#### NOTE:

In order for this feature to work correctly, all timers that are synchronized must be set to the same mode (up or up/down), with the same timer pre-scaler, timer clock input and timer period.

To enable synchronized timers, the following steps should be followed:

1. The slave timer B, C, D is configured with the selected timer input clock, timer pre-scaler, timer period and set the **TxCTL.SYNC** bit. The timer should still be set to disabled at this point.
2. The master timer A, B or C is configured with the same timer input clock, timer pre-scaler, timer period and sets the **TxCTL.CLR** bit. This should clear all timer counters of the master and slave timers.
3. The slave timers set the timer mode to the desired state (either up or up/down).
4. The master timer sets the timer mode to either up or up/down and clears the **TxCTL.CLR** bit. This should start the master and all slave timers simultaneously based on the selected timer clock input.
5. Once configured as above, all timers can be disabled by the master setting **TxCTL.CLR** signal, to assert the SYNC\_OUT signal. The timers can be re-enabled by clearing the **TxCTL.CLR** bit, which de-asserts the SYNC\_OUT signal.

#### 14.2.10. PWM/Compare Units

Timer D supports up to 2 PWM/Capture units PWMD0 to PWMD1. Each PWM/Compare unit can be configured independently in PWM mode or capture mode using **TDCCxCTL.CCMODE**.

##### 14.2.10.1. PWM Mode

The PWM mode is enabled with setting **TDCCxCTL.CCMODE** to 0.

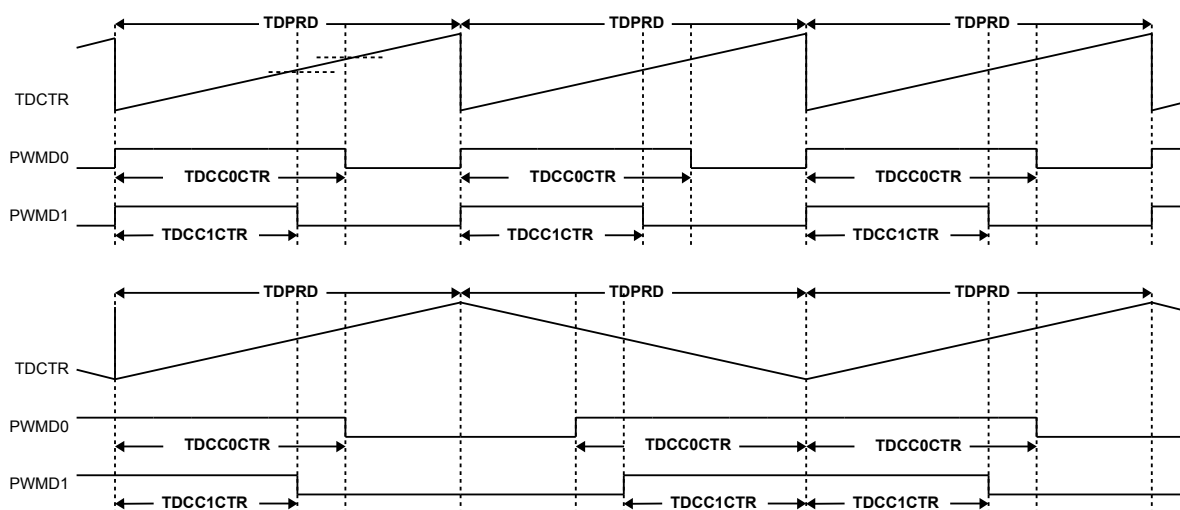
The timer configuration allows either edge-aligned (timer in up mode) or center-aligned (timer in up/down mode) modes of PWM operation.

In both edge-aligned and center-aligned modes of operation, the timer block outputs a PWM waveform that starts out high at a **TDCTR** value of 0 and then transitions to low when **TDCTR** counts up to **TDCCxCTR** compare value.

To configure a duty cycle of 0%, the **TDCCxCTR** should be set to 0; to configure a duty cycle of 100%, the **TDCCxCTR** value should be set to a value greater than or equal to **TDPRD**.

The polarity of the timer PWM outputs are not configurable. Adjustments to the polarity of the PWM outputs may be adjusted in the Dead-Time Generator (DTG) unit connected to the timer peripheral for each output independently.

**Figure 14-2. PWMD0 and PWMD1 Example Using Timer D Up Mode and Up/Down Mode**

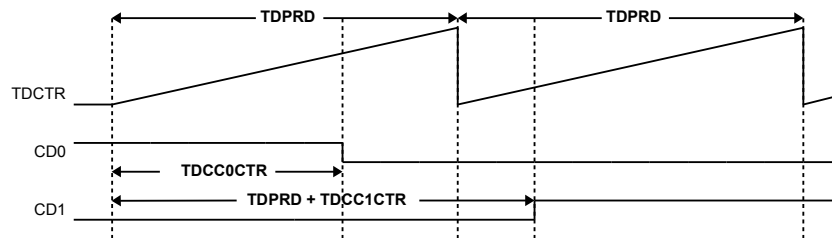


#### 14.2.10.2. Capture Mode

The Capture mode is enabled with setting **TDCCxCTL.CCMODE** to 1. The trip condition for capture mode can be configured using **TDCCxCTL.CCEDGE**, high-to-low signal edge transition, low-to-high signal edge transition or both.

When trip condition is detected the actual **TDCTR** value is copied into **TDCCxCTR**.

**Figure 14-3. CD0 and CD1 Capture Example**



#### 14.2.11. Timer and PWM/Capture Interrupt

The timer may generate interrupt based on the base timer wrap, or when a capture and compare event occurs.

In the base timer both up and up/down timer modes allow an interrupt to be generated when the count reaches 0. Each time the count reaches zero, the **TDCTL.INT** interrupt flag is set. If the interrupt is enabled using the **TDCTL.INTEN**, then the Timer IRQ signal will be asserted to the CPU. The interrupt flag may be cleared by writing a 1 to the **TDCTL.INT** interrupt flag bit.

In the capture and compare PWM units, each time a compare threshold is reached or each time a capture event is detected the **TDCCxCTL.CCINT** bit will be set for that particular timer unit. If the interrupt is enabled via the **TDCCxCTL.CCINTEN**, then the Timer IRQ signal will be asserted to the CPU. The interrupt flag may be cleared by writing a 1 to the **TDCCxCTL.CCINT** interrupt flag bit.

The timer IRQ signal will be asserted if any of the timer interrupt flags **TDCTL.INT** or **TDCCxCTL.CCINT** are set. The Timer IRQ signal will be de-asserted when all of the timer interrupt flags are cleared.

#### 14.2.12. Dead-Time Generator

The dead-time generator can be configured to introduce dead-time for a complementary PWM output. The Timer D block supports up to 1 dead time generator.

##### 14.2.12.1. Dead Time Input Clock Selection

The clock source for the DTGD0 can be selected using **TDCTL.DTGCLK**.

Clear **TDCTL.DTGCLK** to 0 to use clock source selected by **TDCTL.CLK** directly to use higher resolution for dead time insertion.

Set **TDCTL.DTGCLK** to 1 to use divided clock source selected by **TDCTL.CLK** and **TDCTL.CLKDIV** divider to use the same dead time resolution as Timer D.

##### 14.2.12.2. Dead Time Range

The resolution for leading edge and trailing edge dead time is 12bits. Leading and trailing edge can be set independently using **DTGD0LED** and **DTGD0TED**.

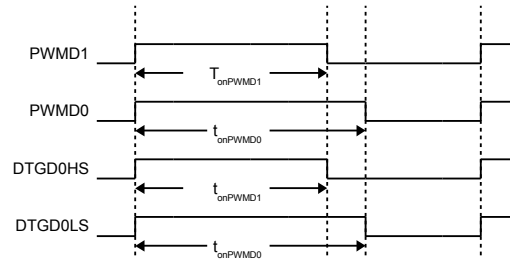
### 14.2.12.3. Bypass Mode

Set **DTGD0CTL.BYPASS** to 0 to enable dead time insertion.

Set **DTGD0CTL.BYPASS** to 1 to enable bypass mode, no dead time is inserted, PWMD1 is routed to DTGD0HS and PWMD0 is routed to DTGD0LS.

The DTGD0HS and DTGD0LS signals can be inverted in bypass mode.

**Figure 14-4. DTGD0 Bypass Example**



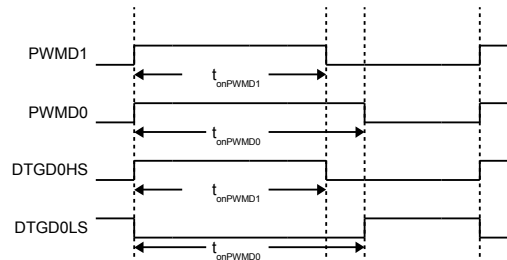
### 14.2.12.4. Inverting PWM Signal

The DTG output signals DTGD0HS and DTGD0LS can be inverted independently.

Set **DTGD0CTL.INVHS** to invert DTGD0HS signal.

Set **DTGD0CTL.INVLS** to invert DTGD0LS signal.

**Figure 14-5. DTGD0 Bypass and Inverting LS Example**



### 14.2.12.5. Dead Time Insertion

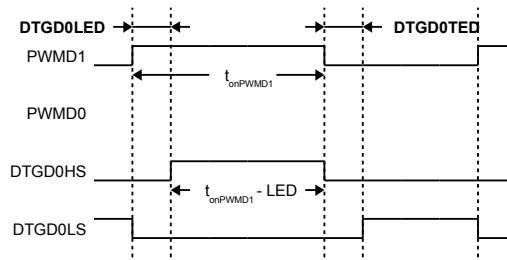
Set **DTGD0CTL.BYPASS** to 0 to enable dead time insertion. In dead time insertion mode only PWMD1 signal is used to generate DTGD0HS and DTGD0LS. PWMD0 signal is ignored and can be used for other purposes.

Set **DTGD0LED** for desired leading-edge and **DTGD0TED** for desired trailing edge in clock-cycles defined by **TDCTL.DTGCLK** clock source

**NOTE:**

In dead time insertion mode the DTGD0LS signal is automatically inverted compared to PWMD1 signal. Set **DTGD0CTL.INVLS** to 0, if this is desired behavior.

**Figure 14-6. DTGD0 LED and TED Example**



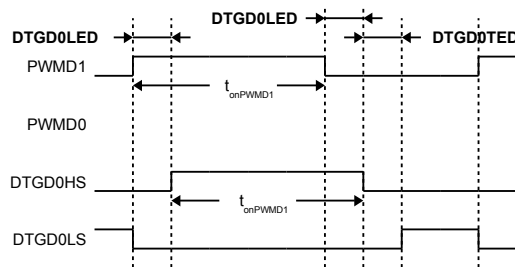
#### 14.2.12.6. Dead Time Insertion with On Time Preservation

Set **DTGD0CTL.OTP** to 1 to enable on time preservation. In this mode the DTGD0HS is same as PWM1 on time.

**NOTE:**

In dead time insertion mode the DTGD0LS signal is automatically inverted compared to PWM1 signal. Set **DTGD0CTL.INVLS** to 0, if this is desired behavior.

**Figure 14-7. DTGD0 LED and TED with On Time Preservation Example**



#### 14.2.13. PWM Output and Capture Input Pin Selection

Each of the DTGD0HS, DTGD0LS outputs, and CDx inputs can be routed to different I/Os, allowing great flexibility in pin assignments.

In capture mode only one I/O should be enabled as input to the capture. If more than one pin input is enabled, the capture might not work properly.

**Note:**

Not all pins are available pending package option, consult data sheet for available pins and signals.

**Table 14-2. Timer D Signal to Pin Mapping**

PWM	CAPTURE	DEADTIME	PINS
PWMD0	CD0	DTGD0LS	PA5, PD7
PWMD1	CD1	DTGD1LS	PD4

## 15. FLASH MEMORY CONTROLLER

### 15.1. Register

#### 15.1.1. Register Map

**Table 15-1. FLASH Memory Controller Register Map**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>FLASH Memory Controller</b>			
0x4002 0000	<b>FLASHLOCK</b>	FLASH lock	0x0000 0000
0x4002 0004	<b>FLASHSTATUS</b>	FLASH status	0x0000 0000
0x4002 0008	<b>FLASHPAGE</b>	FLASH page selection	0x0000 0000
0x4002 000C	<b>Reserved</b>	Reserved	0x0000 0000
0x4002 0010	<b>Reserved</b>	Reserved	0x0000 0000
0x4002 0014	<b>FLASHPERASE</b>	FLASH page erase	0x0000 0000
0x4002 0018	<b>Reserved</b>	Reserved	0x0000 0000
0x4002 001C	<b>Reserved</b>	Reserved	0x0000 0000
0x4002 0020	<b>Reserved</b>	Reserved	0x0000 0000
0x4002 0024	<b>SWDACCESS</b>	SWD access control	0x0000 0000
0x4002 0028	<b>FLASHWSTATE</b>	FLASH wait state control	0x0000 0003
0x4002 002C	<b>FLASHBWRITE</b>	FLASH buffered write enable	0x0000 0000
0x4002 0030	<b>FLASHBWDATA</b>	FLASH buffered write data and address	0x0000 0000

#### 15.1.2. FLASHLOCK

**Register 15-1. FLASHLOCK (FLASH Lock, 0x4002 0000)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:0	<b>LOCK</b>	RW	0x0	Unlock access to FLASH registers and FLASH memory 0xAAAA AAAA: allow write and erase to FLASH pages 0x1234 5678: allow write of FLASHWSTATE register 0xF983 62AB: allow write access to address 0x0010 0008 to disable SWD

### 15.1.3. FLASHSTATUS

#### Register 15-2. FLASHSTATUS (FLASH Status, 0x4002 0004)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	Reserved	R	0x0	Reserved
4	Reserved	R	0x0	Reserved
3	Reserved	R	0x0	Reserved
2	Reserved	R	0x0	Reserved
1	PERASE	R	0x0	Page erase active 1b: page erase in progress 0b: page erase finished or no page erase in progress
0	WRITE	R	0x0	Buffered write active, only used in conjunction with FLASHBWRITE 1b: buffered write in progress 0b: buffered write finished or no buffered write in progress

### 15.1.4. FLASHPAGE

#### Register 15-3. FLASHPAGE (FLASH Page Selector, 0x4002 0008)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:5	Reserved	R	0x0	Reserved
4:0	PAGE	RW	0x0	FLASH page selector for page erase 0x1F: page 31, 0x0000 7C00 to 0x0000 7FFF 0x1E: page 30, 0x0000 7800 to 0x0000 7BFF 0x1D: page 29, 0x0000 7400 to 0x0000 73FF 0x1C: page 28, 0x0000 7000 to 0x0000 73FF 0x1B: page 27, 0x0000 6C00 to 0x0000 6FFF 0x1A: page 26, 0x0000 6800 to 0x0000 6BFF 0x19: page 25, 0x0000 6400 to 0x0000 67FF 0x18: page 24, 0x0000 6000 to 0x0000 63FF 0x17: page 23, 0x0000 5C00 to 0x0000 5FFF 0x16: page 22, 0x0000 5800 to 0x0000 5BFF 0x15: page 21, 0x0000 5400 to 0x0000 57FF 0x14: page 20, 0x0000 5000 to 0x0000 53FF 0x13: page 19, 0x0000 4C00 to 0x0000 4FFF 0x12: page 18, 0x0000 4800 to 0x0000 4BFF 0x11: page 17, 0x0000 4400 to 0x0000 47FF 0x10: page 16, 0x0000 4000 to 0x0000 43FF 0x0F: page 15, 0x0000 3C00 to 0x0000 3FFF 0x0E: page 14, 0x0000 3800 to 0x0000 3BFF 0x0D: page 13, 0x0000 3400 to 0x0000 37FF 0x0C: page 12, 0x0000 3000 to 0x0000 33FF 0x0B: page 11, 0x0000 2C00 to 0x0000 2FFF 0x0A: page 10, 0x0000 2800 to 0x0000 2BFF 0x09: page 9, 0x0000 2400 to 0x0000 27FF 0x08: page 8, 0x0000 2000 to 0x0000 23FF 0x07: page 7, 0x0000 1C00 to 0x0000 1FFF 0x06: page 6, 0x0000 1800 to 0x0000 1BFF 0x05: page 5, 0x0000 1400 to 0x0000 17FF 0x04: page 4, 0x0000 1000 to 0x0000 13FF 0x03: page 3, 0x0000 0C00 to 0x0000 0FFF 0x02: page 2, 0x0000 0800 to 0x0000 0BFF 0x01: page 1, 0x0000 0400 to 0x0000 07FF 0x00: page 0, 0x0000 0000 to 0x0000 03FF



### 15.1.5. FLASHPERASE

#### Register 15-4. FLASHPERASE (FLASH Page Erase, 0x4002 0014)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:0	<b>PERASE</b>	R	0x0	Write of correct key value to this register starts FLASH page erase selected in FLASHPAGE.PAGE 0xA5A5 5A5A: start FLASH page erase selected by FLASHPAGE.PAGE

### 15.1.6. SWDACCESS

#### Register 15-5. SWDACCESS (SDW Access Status, 0x4002 0024)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:0	<b>DEBUG</b>	R	0x0	Status of SWD debug 0xFFFFFFFF: SWD access is enabled 0x69696969: SWD access is disabled

### 15.1.7. FLASHWSTATE

#### Register 15-6. FLASHWSTATE (FLASH Access Wait State, 0x4002 0028)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:2	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0x0
1:0	<b>WSTATE</b>	RW	0x3	FLASH access wait state 11b: 3 wait states for 75MHz < HCLK < 100MHz 10b: 2 wait states for 50MHz < HCLK < 75MHz 01b: 1 wait state for 25MHz < HCLK < 50MHz 00b: 0 wait state for HCLK < 25MHz

### 15.1.8. FLASHBWRITE

#### Register 15-7. FLASHBWRITE (Buffered FLASH Write, 0x4002 002C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:0	<b>BWRITE</b>	RW	0x0	Write of correct key value to this register starts buffered write operation 0xCA72 6B18: start buffered write of FLASHBWDATA

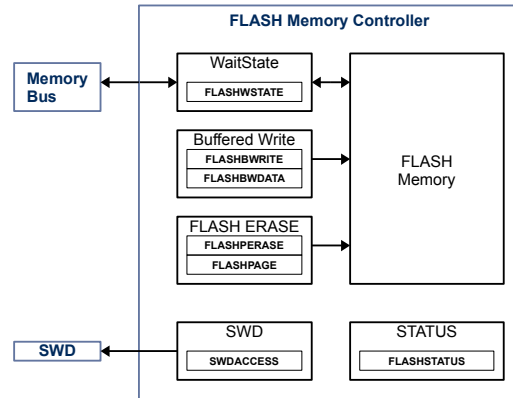
**15.1.9. FLASHBWDATA**
**Register 15-8. FLASHBWDATA (Buffered FLASH Write Data, 0x4002 0030)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0x0
30	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0x0
29:25	<b>PAGE</b>	RW	0x0	FLASH page selector to write to 0x1F: page 31, 0x0000 7C00 to 0x0000 7FFF 0x1E: page 30, 0x0000 7800 to 0x0000 7BFF 0x1D: page 29, 0x0000 7400 to 0x0000 73FF 0x1C: page 28, 0x0000 7000 to 0x0000 73FF 0x1B: page 27, 0x0000 6C00 to 0x0000 6FFF 0x1A: page 26, 0x0000 6800 to 0x0000 6BFF 0x19: page 25, 0x0000 6400 to 0x0000 67FF 0x18: page 24, 0x0000 6000 to 0x0000 63FF 0x17: page 23, 0x0000 5C00 to 0x0000 5FFF 0x16: page 22, 0x0000 5800 to 0x0000 5BFF 0x15: page 21, 0x0000 5400 to 0x0000 57FF 0x14: page 20, 0x0000 5000 to 0x0000 53FF 0x13: page 19, 0x0000 4C00 to 0x0000 4FFF 0x12: page 18, 0x0000 4800 to 0x0000 4BFF 0x11: page 17, 0x0000 4400 to 0x0000 47FF 0x10: page 16, 0x0000 4000 to 0x0000 43FF 0x0F: page 15, 0x0000 3C00 to 0x0000 3FFF 0x0E: page 14, 0x0000 3800 to 0x0000 3BFF 0x0D: page 13, 0x0000 3400 to 0x0000 37FF 0x0C: page 12, 0x0000 3000 to 0x0000 33FF 0x0B: page 11, 0x0000 2C00 to 0x0000 2FFF 0x0A: page 10, 0x0000 2800 to 0x0000 2BFF 0x09: page 9, 0x0000 2400 to 0x0000 27FF 0x08: page 8, 0x0000 2000 to 0x0000 23FF 0x07: page 7, 0x0000 1C00 to 0x0000 1FFF 0x06: page 6, 0x0000 1800 to 0x0000 1BFF 0x05: page 5, 0x0000 1400 to 0x0000 17FF 0x04: page 4, 0x0000 1000 to 0x0000 13FF 0x03: page 3, 0x0000 0C00 to 0x0000 0FFF 0x02: page 2, 0x0000 0800 to 0x0000 0BFF 0x01: page 1, 0x0000 0400 to 0x0000 07FF 0x00: page 0, 0x0000 0000 to 0x0000 03FF
24:16	<b>ADDRESS</b>	RW	0x0	Relative half word address within selected FLASHBWDATA.PAGE
15:0	<b>DATA</b>	RW	0x0	Data to write

## 15.2. Details of Operation

### 15.2.1. Block Diagram

Figure 15-1. FLASH Memory Controller



### 15.2.2. Configuration

Following blocks need to be configured for correct use of the FLASH:

- Clock Control System (CCS)

### 15.2.3. FLASH Memory

The Flash memory controller allows configuration of the FLASH memory. FLASH wait states, FLASH erase, buffered FLASH write, and SWD debug access can be configured. The FLASH memory has up to 32 pages of 1kByte each.

### 15.2.4. Writing to FLASH Controller Registers

The FLASH Controller registers are write protected to reduce chances of accidental erase or modification of FLASH memory. Each write to a FLASH controller register is a 2 step process. The first step is to write the correct key into **FLASHLOCK** followed by a FLASH controller register write.

Without correct key any writes to FLASH controller register will be ignored. Flash controller reads are always possible.

### 15.2.5. FLASH Wait State

After device reset, the **FLASHWSTATE** is set to 0x3. To allow optimal FLASH access time without delay, the FLASH wait state need to be set according to HCLK frequency used. See register table for correct setting.

To write to **FLASHWSTATE** register, **FLASHLOCK** need to be set to 0x1234 5678.

### 15.2.6. FLASH Page Erase

To erase a page of FLASH memory, set **FLASHLOCK** to 0xAAAA AAAA first, then set **FLASHBWDATA.PAGE** to the page to be erased and then set **FLASHPERASE** to 0xA5A5 5A5A. The FLASH page operation will start, **FLASHSTATUS.PERASE** is set to 1b and any access to FLASH memory address space is stalled until erase operation is finished. **FLASHSTATUS.PERASE** is set to 0b when erase operation is finished.

It is not recommended to erase FLASH pages while executing from FLASH as any access to FLASH is stalled until the erase operation is finished. Either execute from SRAM or use SWD debug interface.

### 15.2.7. Write to FLASH

Only half-word address aligned half-word writes are supported. To write a half word to FLASH memory, make sure the memory location is erased by doing a read, it should return 0xFFFF. Set **FLASHLOCK** to 0xAAAA AAAA first, then write a half-word to the memory address directly.

It is not recommended to write to FLASH while executing from FLASH as any access to FLASH is stalled until the erase operation is finished. Either execute from SRAM or use SWD debug interface.

### 15.2.8. Buffered Write to FLASH

Only half-word address aligned half-word writes are supported. The FLASH memory controller also allows buffered write to FLASH, to allow CPU still react to interrupts or perform other tasks while waiting for FLASH write to finish when executing from SRAM.

To write a half word to FLASH memory, make sure the memory location is erased by doing a read, it should return 0xFFFF. Set **FLASHLOCK** to 0xAAAA AAAA first, then write to **FLASHBWDATA**, with **FLASHBWDATA.DATA** the half-word you want to write, and **FLASHBWDATA.PAGE** the page where the memory location resides and **FLASHBWDATA.ADDRESS** the relative address of the memory location.

The FLASH page operation will start, **FLASHSTATUS.WRITE** is set to 1b, AHB bus control will be given back to CPU to allow execution of other commands. Any access to FLASH memory while buffered write operation is active will stall. **FLASHSTATUS.WRITE** is set to 0b when buffered write operation is finished.

To calculate **FLASHBWDATA.PAGE** use:

$$PAGE = \frac{Memoryaddress}{pagesize} \quad (4)$$

Where:

PAGE: integer value for **FLASHBWDATA.PAGE**

Memoryaddress: Word memory address

pagesize: FLASH page size: 0x400

Then to calculate **FLASHBWDATA.ADDRESS** use:

$$ADDRESS = \frac{(Memoryaddress - PAGE * pagesize)}{2} \quad (5)$$

Where:

ADDRESS: integer value for **FLASHBWDATA.ADRESS**

PAGE: integer value for **FLASHBWDATA.PAGE**

Memoryaddress: Word memory address  
pagesize: FLASH page size: 0x400

Example:

memoryaddress: 0x0000 0438

PAGE =  $0x0000\ 0438 / 0x400 = 0x01$

ADDRESS =  $(0x0000\ 0438 - 0x01 * 0x400) / 0x2 = 0x38 / 0x2 = 0x1C$

It is not recommended to write to FLASH while executing from FLASH as any access to FLASH is stalled until the erase operation is finished. Either execute from SRAM or use SWD debug interface.

### 15.2.9. SWD Debug Access Disable

The SWD debug access is enabled by default and can be disabled by a FUSE to prevent access of device memory.

**Caution need to be taken. This action is not reversible.**

To disable SWD set **FLASHLOCK** to 0xF983 62AB first, then write 0x6969 6969 to address 0x0010 0008 will disable the SWD debug access.

## 16. ADC AND AUTO SEQUENCER

### 16.1. Register

#### 16.1.1. Register Map

**Table 16-1. Register Map – EMUX**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>EMUX</b>			
0x4015 0000	EMUXCTL	ADC external MUX control register	0x0000 0000
0x4015 0004	EMUXDATA	ADC external MUX data register	0x0000 0000

**Table 16-2. Register Map – ADC**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>ADC</b>			
0x4015 0008	ADCCTL	ADC control register	0x0000 0000
0x4015 000C	ADCR	ADC conversion result register	0x0000 0000
0x4015 0010	ADCINT	ADC Interrupt register	0x0000 0000

**Table 16-3. Register Map – ADC Auto Sequencer 0**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>ADC Auto Sequencer -0</b>			
0x4015 0040	AS0CTL	Automated ADC sampling 0 control register	0x0000 0000
0x4015 0044	AS0S0	Automated Sampling 0 Sequence 0 register	0x0000 0000
0x4015 0048	AS0R0	Automated Sampling 0 Sample result 0 register	0x0000 0000
0x4015 004C	AS0S1	Automated Sampling 0 Sequence 1 register	0x0000 0000
0x4015 0050	AS0R1	Automated Sampling 0 Sample result 1 register	0x0000 0000
0x4015 0054	AS0S2	Automated Sampling 0 Sequence 2 register	0x0000 0000
0x4015 0058	AS0R2	Automated Sampling 0 Sample result 2 register	0x0000 0000
0x4015 005C	AS0S3	Automated Sampling 0 Sequence 3 register	0x0000 0000
0x4015 0060	AS0R3	Automated Sampling 0 Sample result 3 register	0x0000 0000
0x4015 0064	AS0S4	Automated Sampling 0 Sequence 4 register	0x0000 0000
0x4015 0068	AS0R4	Automated Sampling 0 Sample result 4 register	0x0000 0000
0x4015 006C	AS0S5	Automated Sampling 0 Sequence 5 register	0x0000 0000
0x4015 0070	AS0R5	Automated Sampling 0 Sample result 5 register	0x0000 0000
0x4015 0074	AS0S6	Automated Sampling 0 Sequence 6 register	0x0000 0000
0x4015 0078	AS0R6	Automated Sampling 0 Sample result 6 register	0x0000 0000
0x4015 007C	AS0S7	Automated Sampling 0 Sequence 7 register	0x0000 0000
0x4015 0080	AS0R7	Automated Sampling 0 Sample result 7 register	0x0000 0000

**Table 16-4. Register Map – ADC Auto Sequencer 1**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>ADC Auto Sequencer -1</b>			
0x4015 0100	AS1CTL	Automated ADC sampling 1 control register	0x0000 0000
0x4015 0104	AS1S0	Automated Sampling 1 Sequence 0 register	0x0000 0000
0x4015 0108	AS1R0	Automated Sampling 1 Sample result 0 register	0x0000 0000
0x4015 010C	AS1S1	Automated Sampling 1 Sequence 1 register	0x0000 0000
0x4015 0110	AS1R1	Automated Sampling 1 Sample result 1 register	0x0000 0000
0x4015 0114	AS1S2	Automated Sampling 1 Sequence 2 register	0x0000 0000
0x4015 0118	AS1R2	Automated Sampling 1 Sample result 2 register	0x0000 0000
0x4015 011C	AS1S3	Automated Sampling 1 Sequence 3 register	0x0000 0000
0x4015 0120	AS1R3	Automated Sampling 1 Sample result 3 register	0x0000 0000
0x4015 0124	AS1S4	Automated Sampling 1 Sequence 4 register	0x0000 0000
0x4015 0128	AS1R4	Automated Sampling 1 Sample result 4 register	0x0000 0000
0x4015 012C	AS1S5	Automated Sampling 1 Sequence 5 register	0x0000 0000
0x4015 0130	AS1R5	Automated Sampling 1 Sample result 5 register	0x0000 0000
0x4015 0134	AS1S6	Automated Sampling 1 Sequence 6 register	0x0000 0000
0x4015 0138	AS1R6	Automated Sampling 1 Sample result 6 register	0x0000 0000
0x4015 013C	AS1S7	Automated Sampling 1 Sequence 7 register	0x0000 0000
0x4015 0140	AS1R7	Automated Sampling 1 Sample result 7 register	0x0000 0000

### 16.1.2. EMUXCTL

**Register 16-1. EMUXCTL (ADC external MUX control register 0x4015 0000)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:6	<b>Reserved</b>	R	0x0	Reserved
5	<b>EMUXC</b>	RW	0x0	ADC external MUX control 1b: EMUX is controlled by auto-sequencer unit 0b: EMUX manual control
4	<b>EMUXBUSY</b>	RW	0x0	ADC external MUX status 1b: ADC external MUX sending data 0b: ADC external MUX not busy or not used
3	<b>EMUXDONE</b>	RW	0x0	ADC external MUX data send done, auto-clear with read or when new data is sent over EMUX 1b: ADC external MUX data sent 0b: ADC external MUX busy
2:0	<b>EMUXCDIV</b>	RW	0x0	ADC external mux clock to FCLK divider 111b: FCLK/8 110b: FCLK/7 101b: FCLK/6 100b: FCLK/5 011b: FCLK/4 010b: FCLK/3 001b: FCLK/2 000b: FCLK/1

### 16.1.3. EMUXDATA

#### Register 16-2. EMUXDATA (EMUX data register 0x4015 0004)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	R	0x0	Reserved
7:0	<b>DATA</b>	RW	0x0	EMUX data, writing this register will start transmission over EMUX

### 16.1.4. ADCCTL

#### Register 16-3. ADCCTL (ADC control register 0x4015 0008)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	R	0x0	Reserved
15	<b>ADCEN</b>	RW	0x0	ADC module enable 1b: enable ADC module 0b: turn off ADC module
14	<b>ADCSTART</b>	RW	0x0	Start ADC conversion. A write of 1b will start ADC conversion if ADCCTL.ADCBUSY = 0b and ADCCTL.ADCMODE = 00b. 1b: start ADC conversion 0b: stop ADC conversion, also stop repeated ADC conversions ADCCTL.ADCREPEAT = 1b.
13	<b>ADCREPEAT</b>	RW	0x0	ADC repeat mode 1b: repeated conversion, also auto rearms auto sequencer 0b: single shot conversion
12:10	<b>ADCMODE</b>	RW	0x0	ADC conversion mode 111b: automated sequencer 0 and 1 independently triggered 110b: automated sequencer 0 and 1 daisy chained trigger on AS0 101b: automated sequencer 1 only trigger condition 100b: automated sequencer 0 only trigger condition 011b: automated sequencer 0 and 1 daisy chained 010b: automated sequencer 1 only 001b: automated sequencer 0 only 000b: single channel
9:8	<b>Reserved</b>	R	0x0	Reserved
7	<b>ADCBUSY</b>	R	0x0	ADC busy 1b: ADC conversion or auto sequencer active 0b: ADC no operation
6:4	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 111b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: AD0
3	<b>Reserved</b>	R	0x0	Reserved
2:0	<b>ADCCDIV</b>	RW	0x0	ADC input clock FCLK divider 111b: FCLK/8 110b: FCLK/7 101b: FCLK.6 100b: FCLK/5 011b: FCLK/4 010b: FCLK/3 001b: FCLK/2 000b: FCLK/1



### 16.1.5. ADCCR

**Register 16-4. ADCCR (ADC conversion result register 0x4015 000C)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	Reserved	R	0x0	Reserved
9:0	ADCRESULT	R	0x0	ADC conversion result

### 16.1.6. ADCINT

**Register 16-5. ADCINT (ADC Interrupt register 0x4015 0010)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:13	Reserved	R	0x0	Reserved
19:18	ASCINTTR	RW	0x0	Last Auto sequencer to run 11b: reserved 10b: auto sequencer 1 to run 01b: auto sequencer 0 to run 00b: no sequencer to r
17:16	ASCINTSEQ	RW	0x0	Last Auto sequencer to trigger ADCINT.ASCINT 11b: Auto sequencer 1 and 0 triggered interrupt 10b: Auto sequencer 1 triggered interrupt 01b: Auto sequencer 0 triggered interrupt 00b: no trigger
15:13	Reserved	R	0x0	Reserved
12	ASCINT_EN	RW	0x0	Enable auto sequencer collision interrupt 1b: enable ASCINT 0b: disable ASCINT
11	AS1INT_EN	RW	0x0	Enable auto sequencer 1 conversions finished interrupt 1b: enable AS1INT 0b: disable AS1INT
10	AS0INT_EN	RW	0x0	Enable auto sequencer 0 conversions finished interrupt 1b: enable AS0INT 0b: disable AS0INT
9	EMUXINT_EN	RW	0x0	Enable EMUX transfer finished interrupt 1b: enable EMUXINT 0b: disable EMUXINT
8	ADCINT_EN	RW	0x0	Enable ADC conversion finished interrupt 1b: enable ADCINT 0b: disable ADCINT
7:5	Reserved	RW	0x0	Reserved, must be set to 0
4	ASCINT	RW	0x0	Auto sequencer collision interrupt 1b: interrupt, clear by writing 1b to it 0b: no interrupt  NOTE: This bit is cleared by writing a 1b to it.
3	AS1INT	RW	0x0	Auto sequencer 1 conversions finished interrupt 1b: interrupt, clear by writing 1b to it 0b: no interrupt  NOTE: This bit is cleared by writing a 1b to it.

BIT	NAME	ACCESS	RESET	DESCRIPTION
2	<b>AS0INT</b>	RW	0x0	Auto sequencer 0 conversions finished interrupt 1b: interrupt, clear by writing 1b to it 0b: no interrupt  NOTE: This bit is cleared by writing a 1b to it.
1	<b>EMUXINT</b>	RW	0x0	EMUX data transfer finished interrupt 1b: interrupt, clear by writing 1b to it 0b: no interrupt  NOTE: This bit is cleared by writing a 1b to it.
0	<b>ADCINT</b>	RW	0x0	ADC conversion finished interrupt 1b: interrupt, clear by writing 1b to it 0b: no interrupt  NOTE: This bit is cleared by writing a 1b to it.

### 16.1.7. AS0CTL

#### Register 16-6. AS0CTL (Auto Sequencer 0 control register 0x4015 0040)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:13	<b>Reserved</b>	R	0x0	Reserved
12	<b>AS0BUSY</b>	RW	0x0	Auto sequencer 0 busy 1b: auto sequencer 0 sampling active 0b: auto sequencer 0 not active
11	<b>AS0EN</b>	RW	0x0	Auto sequencer 0 enable 1b: auto sequencer 0 enabled 0b: auto sequencer 0 not enabled
10:8	<b>AS0D</b>	RW	0x0	Auto sequencer 0 sampling depth 111b: 8 samples 110b: 7 samples 101b: 6 samples 100b: 5 samples 011b: 4 samples 010b: 3 samples 001b: 2 samples 000b: 1 sample
7	<b>AS0TR</b>	RW	0x0	Auto sequencer 0 trigger source 1b: Timer, as defined by AS0CTL.AS0TRTMR 0b: PWM, as defined by AS0CTL.AS0TRPWM
6	<b>AS0TRE</b>	RW	0x0	Auto sequencer 0 trigger source AS0CTL.AS0TR edge 1b: high to low edge 0b: low to high edge
5:4	<b>AS0TRTMR</b>	RW	0x0	Auto sequencer 0 timer trigger source 11b: Timer D 10b: Timer C 01b: Timer B 00b: Timer A

BIT	NAME	ACCESS	RESET	DESCRIPTION
3:0	<b>ASA0TRPWM</b>	RW	0x0	Auto sequencer 0 PWM trigger source 1111b: reserved 1110b: reserved 1101b: PWMD1 1100b: PWMD0 1011b: PWMC1 1010b: PWMC0 1001b: PWMB1 1000b: PWMB0 0111b: PWMA7 0110b: PWMA6 0101b: PWMA5 0100b: PWMA4 0011b: PWMA3 0010b: PWMA2 0001b: PWMA1 0000b: PWMA0

### 16.1.8. AS0S0

**Register 16-7. AS0S0 (Auto sequencer 0-sample 0 control 0x4015 0044)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 111b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS0S0.EMUXD data after S/H of ADC 01b: send AS0S0.EMUXD data at beginning of this sample sequence 00b: do not send AS0S0.EMUXD data
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

### 16.1.9. AS0R0

**Register 16-8. AS0R0 (Auto sequencer 0-sample 0 result register 0x4015 0048)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCRESULT</b>	R	0x0	ADC conversion result

### 16.1.10. AS0S1

#### Register 16-9. AS0S1 (Auto sequencer 0-sample 1 control 0x4015 004C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 111b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS0S1.EMUXD data after S/H of ADC 01b: send AS0S1.EMUXD data at beginning of this sample sequence 00b: do not send AS0S1.EMUXD data
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

### 16.1.11. AS0R1

#### Register 16-10. AS0R1 ( Auto sequencer 0-sample 1 result register 0x4015 0050)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCRESLT</b>	R	0x0	ADC conversion result

### 16.1.12. AS0S2

#### Register 16-11. AS0S2 (Auto sequencer 0-sample 2 control 0x4015 0054)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 111b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock

BIT	NAME	ACCESS	RESET	DESCRIPTION
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS0S2.EMUXD data after S/H of ADC 01b: send AS0S2.EMUXD data at beginning of this sample sequence 00b: do not send AS0S2.EMUXD data
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

### 16.1.13. AS0R2

#### Register 16-12. AS0R2 ( Auto sequencer 0-sample 2 result register 0x4015 0058)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCRESULT</b>	R	0x0	ADC conversion result

### 16.1.14. AS0S3

#### Register 16-13. AS0S3 (Auto sequencer 0-sample 3 control 0x4015 005C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 111b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS0S3.EMUXD data after S/H of ADC 01b: send AS0S3.EMUXD data at beginning of this sample sequence 00b: do not send AS0S3.EMUXD data
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

### 16.1.15. AS0R3

#### Register 16-14. AS0R3 ( Auto sequencer 0-sample 3 result register 0x4015 0060)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCRESULT</b>	R	0x0	ADC conversion result

### 16.1.16. AS0S4

#### Register 16-15. AS0S4 (Auto sequencer 0-sample 4 control 0x4015 0064)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 11b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS0S4.EMUXD data after S/H of ADC 01b: send AS0S4.EMUXD data at beginning of this sample sequence 00b: do not send AS0S4.EMUXD data
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

### 16.1.17. AS0R4

#### Register 16-16. AS0R4 ( Auto sequencer 0-sample 4 result register 0x4015 0068)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCRESLT</b>	R	0x0	ADC conversion result

### 16.1.18. AS0S5

#### Register 16-17. AS0S5 (Auto sequencer 0-sample 5 control 0x4015 006C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 111b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock

BIT	NAME	ACCESS	RESET	DESCRIPTION
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS0S5.EMUXD data after S/H of ADC 01b: send AS0S5.EMUXD data at beginning of this sample sequence 00b: do not send AS0S5.EMUXD data
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

#### 16.1.19. AS0R5

##### Register 16-18. AS0R5 ( Auto sequencer 0-sample 5 result register 0x4015 0070)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCRESULT</b>	R	0x0	ADC conversion result

#### 16.1.20. AS0S6

##### Register 16-19. AS0S6 (Auto sequencer 0-sample 6 control 0x4015 0074)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 111b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS0S6.EMUXD data after S/H of ADC 01b: send AS0S6.EMUXD data at beginning of this sample sequence 00b: do not send AS0S6.EMUXD data
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

#### 16.1.21. AS0R6

##### Register 16-20. AS0R6 ( Auto sequencer 0-sample 6 result register 0x4015 0078)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCRESULT</b>	R	0x0	ADC conversion result

### 16.1.22. AS0S7

#### Register 16-21. AS0S7 (Auto sequencer 0-sample 7 control 0x4015 007C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	Reserved	R	0x0	Reserved
14:12	ADCMUX	RW	0x0	ADC MUX input select 11b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	DELAY	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	EMUXS	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS0S7.EMUXD data after S/H of ADC 01b: send AS0S7.EMUXD data at beginning of this sample sequence 00b: do not send AS0S7.EMUXD data
7:0	EMUXD	RW	0x0	EMUX data to transmit

### 16.1.23. AS0R7

#### Register 16-22. AS0R7 ( Auto sequencer 0-sample 7 result register 0x4015 0080)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	Reserved	R	0x0	Reserved
9:0	ADCRESLT	R	0x0	ADC conversion result

### 16.1.24. AS1CTL

#### Register 16-23. AS1CTL (Auto Sequencer 1 control register 0x4015 0100)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:13	Reserved	R	0x0	Reserved
12	AS1BUSY	RW	0x0	Auto sequencer 1 busy 1b: auto sequencer 1 sampling active 0b: auto sequencer 1 not active
11	AS1EN	RW	0x0	Auto sequencer 1 enable 1b: auto sequencer 1 enabled 0b: auto sequencer 1 not enabled
10:8	AS1D	RW	0x0	Auto sequencer 1 sampling depth 111b: 8 samples 110b: 7 samples 101b: 6 samples 100b: 5 samples 011b: 4 samples 010b: 3 samples 001b: 2 samples 000b: 1 sample



BIT	NAME	ACCESS	RESET	DESCRIPTION
7	<b>AS1TR</b>	RW	0x0	Auto sequencer 1 trigger source 1b: Timer, as defined by AS1CTL.AS1TRTMR 0b: PWM, as defined by AS1CTL.AS1TRPWM
6	<b>AS1TRE</b>	RW	0x0	Auto sequencer 1 trigger source AS1CTL.AS1TR edge 1b: high to low edge 0b: low to high edge
5:4	<b>AS1TRTMR</b>	RW	0x0	Auto sequencer 1 timer trigger source 11b: Timer D 10b: Timer C 01b: Timer B 00b: Timer A
3:0	<b>AS1TRPWM</b>	RW	0x0	Auto sequencer 1 PWM trigger source 1111b: reserved 1110b: reserved 1101b: PWMD1 1100b: PWMD0 1011b: PWMC1 1010b: PWMC0 1001b: PWMB1 1000b: PWMB0 0111b: PWMA7 0110b: PWMA6 0101b: PWMA5 0100b: PWMA4 0011b: PWMA3 0010b: PWMA2 0001b: PWMA1 0000b: PWMA0

### 16.1.25. AS1S0

#### Register 16-24. AS1S0 (Auto sequencer 1-sample 0 control 0x4015 0104)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 111b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS1S0.EMUXD data after S/H of ADC 01b: send AS1S0.EMUXD data at beginning of this sample sequence 00b: do not send AS1S0.EMUXD data
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

### 16.1.26. AS1R0

#### Register 16-25. AS1R0 ( Auto sequencer 1-sample 0 result register 0x4015 0108)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	Reserved	R	0x0	Reserved
9:0	ADCRESULT	R	0x0	ADC conversion result

### 16.1.27. AS1S1

#### Register 16-26. AS1S1 (Auto sequencer 1-sample 1 control 0x4015 010C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	Reserved	R	0x0	Reserved
14:12	ADCMUX	RW	0x0	ADC MUX input select 111b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	DELAY	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	EMUXS	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS1S1.EMUXD data after S/H of ADC 01b: send AS1S1.EMUXD data at beginning of this sample sequence 00b: do not send AS1S1.EMUXD data
7:0	EMUXD	RW	0x0	EMUX data to transmit

### 16.1.28. AS1R1

#### Register 16-27. AS1R1 ( Auto sequencer 1-sample 1 result register 0x4015 0110)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	Reserved	R	0x0	Reserved
9:0	ADCRESULT	R	0x0	ADC conversion result

### 16.1.29. AS1S2

#### Register 16-28. AS1S2 (Auto sequencer 1-sample 2 control 0x4015 0114)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	Reserved	R	0x0	Reserved

BIT	NAME	ACCESS	RESET	DESCRIPTION
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 11b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS1S2.EMUXD data after S/H of ADC 01b: send AS1S2.EMUXD data at beginning of this sample sequence 00b: do not send AS1S2.EMUXD data
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

### 16.1.30. AS1R2

#### Register 16-29. AS1R2 ( Auto sequencer 1-sample 2 result register 0x4015 0118)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCREULT</b>	R	0x0	ADC conversion result

### 16.1.31. AS1S3

#### Register 16-30. AS1S3 (Auto sequencer 1-sample 3 control 0x4015 011C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 11b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS1S3.EMUXD data after S/H of ADC 01b: send AS1S3.EMUXD data at beginning of this sample sequence 00b: do not send AS1S3.EMUXD data

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

### 16.1.32. AS1R3

#### Register 16-31. AS1R3 ( Auto sequencer 1-sample 3 result register 0x4015 0120)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCRESULT</b>	R	0x0	ADC conversion result

### 16.1.33. AS1S4

#### Register 16-32. AS1S4 (Auto sequencer 1-sample 4 control 0x4015 0124)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 111b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS1S4.EMUXD data after S/H of ADC 01b: send AS1S4.EMUXD data at beginning of this sample sequence 00b: do not send AS1S4.EMUXD data
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

### 16.1.34. AS1R4

#### Register 16-33. AS1R4 ( Auto sequencer 1-sample 4 result register 0x4015 0128)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCRESULT</b>	R	0x0	ADC conversion result

### 16.1.35. AS1S5

#### Register 16-34. AS1S5 (Auto sequencer 1-sample 5 control 0x4015 012C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved

BIT	NAME	ACCESS	RESET	DESCRIPTION
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 11b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS1S5.EMUXD data after S/H of ADC 01b: send AS1S5.EMUXD data at beginning of this sample sequence 00b: do not send AS1S5.EMUXD data
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

#### 16.1.36. AS1R5

##### Register 16-35. AS1R5 ( Auto sequencer 1-sample 5 result register 0x4015 0130)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCRESULT</b>	R	0x0	ADC conversion result

#### 16.1.37. AS1S6

##### Register 16-36. AS1S6 (Auto sequencer 1-sample 6 control 0x4015 0134)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 11b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS1S6.EMUXD data after S/H of ADC 01b: send AS1S6.EMUXD data at beginning of this sample sequence 00b: do not send AS1S6.EMUXD data

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

### 16.1.38. AS1R6

#### Register 16-37. AS1R6 ( Auto sequencer 1-sample 6 result register 0x4015 0138)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCRESULT</b>	R	0x0	ADC conversion result

### 16.1.39. AS1S7

#### Register 16-38. AS1S7 (Auto sequencer 1-sample 7 control 0x4015 013C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>ADCMUX</b>	RW	0x0	ADC MUX input select 111b: VSSA 110b: reserved 101b: AD5 100b: AD4 011b: AD3 010b: AD2 001b: reserved 000b: EMUX
11:10	<b>DELAY</b>	RW	0x0	Delay between start of sample sequence and start of ADC conversion in ADC input clocks FLCK/ADCCTL.ADCDIV 11b: 16 ADC input clock cycles 10b: 8 ADC input clock cycles 01b: 4 ADC input clock cycles 00b: 0 ADC input clock
9:8	<b>EMUXS</b>	RW	0x0	EMUX transmission start 11b: reserved 10b: send AS1S7.EMUXD data after S/H of ADC 01b: send AS1S7.EMUXD data at beginning of this sample sequence 00b: do not send AS1S7.EMUXD data
7:0	<b>EMUXD</b>	RW	0x0	EMUX data to transmit

### 16.1.40. AS1R7

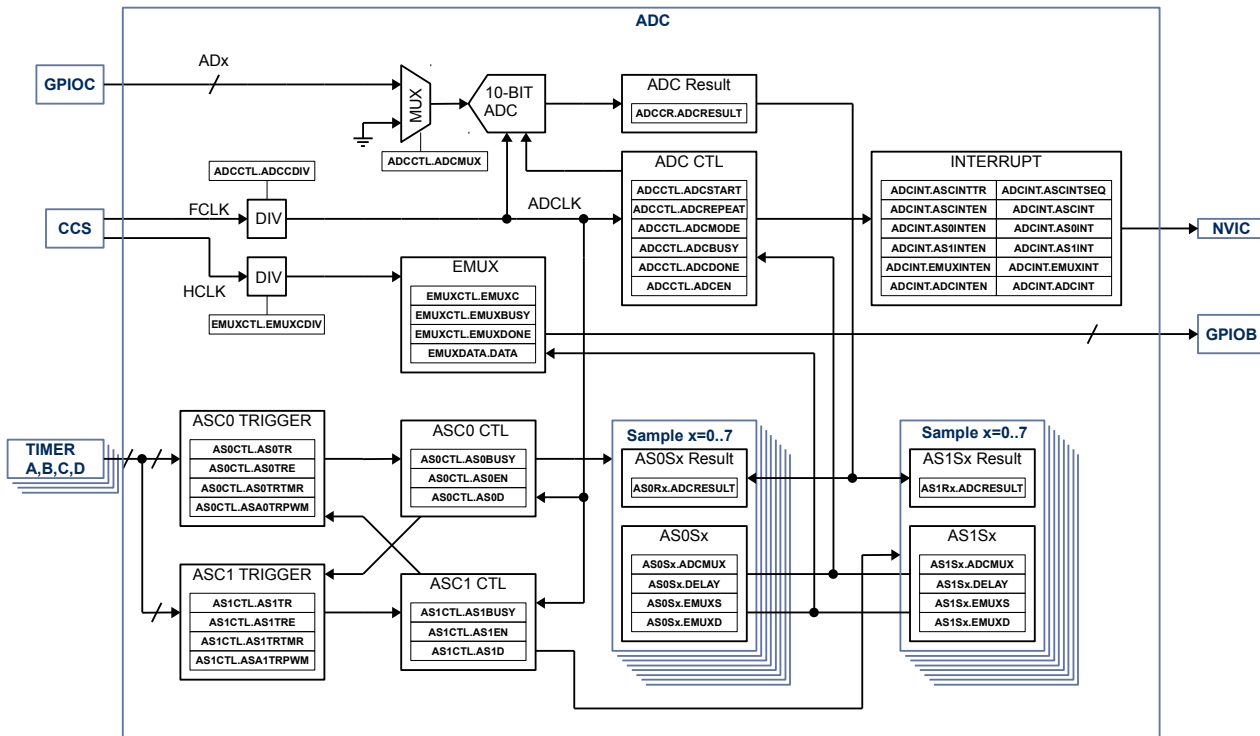
#### Register 16-39. AS1R7 ( Auto sequencer 1-sample 7 result register 0x4015 0140)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	<b>Reserved</b>	R	0x0	Reserved
9:0	<b>ADCRESULT</b>	R	0x0	ADC conversion result

## 16.2. Details of Operation

### 16.2.1. Block Diagram

**Figure 16-1. ADC, EMUX, ASC0, ASC1**



## 16.3. Details of Operation

### 16.3.1. Basic Configuration

Following blocks need to be configured for correct operation of the ADC

- CCS
- Timer A, B, C or D
- GPIOB
- GPIOC
- NVIC

### 16.3.2. ADC, Autosequencer and EMUX

The ADC is a 10-bit SAR ADC. It can be used standalone or together with up to 2 independent low latency auto sequencer state machines to take series of up to 8 samples each into dedicated sample result registers, triggered by either PWM or timer signals. Each sample setup can be programmed with dedicated ADC-MUX setting and settling time delay. A dedicated, programmable high speed low latency communication interface is available to set analog both MUX, sample and hold circuits in the analog peripherals.

### 16.3.3. Clock Setting

The ADC clock is derived from FCLK and can be set with **ADCCTL.ADCCDIV**. The ADC clock should not exceed 16MHz for correct operation.

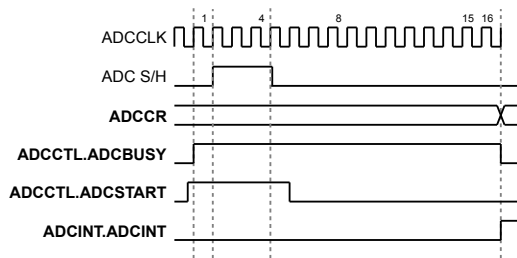
The EMUX clock is derived from HCLK and can be set with **EMUXCTL.EMUXCDIV**.

### 16.3.4. ADC

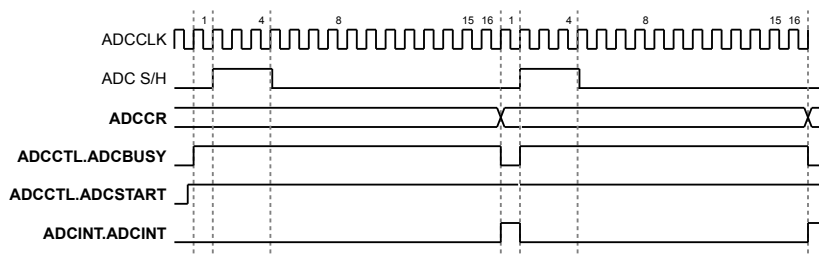
The ADC, ASC0, ASC1 and EMUX block is enabled with **ADCCTL.ADCEN**. In manual conversion mode set **ADCCTL.ADCMODE** to 0b. Set the ADx channel with **ADCCTL.ADCMUX**. For single conversion, set **ADCCTL.REPEAT** to 0b, for repeated conversion set **ADCCTL.REPEAT** to 1b. To start a conversion set **ADCCTL.ADCSTART** to 1b. The ADC will start sampling the analog input channel for 3 clock cycles and holds the analog value in it's internal S/H for conversion. It is safe to switch ADC input channel 4 clocks after ADC start without affecting the ADC result.

One complete ADC conversion will take 16 ADC clock cycles and the ADC conversion result will be available in **ADCCR**. In repeated mode **ADCCR** will be overwritten every 16 ADC clock cycles.. The **ADCCTL.ADCBUSY** flag will 1b as long as conversions are active. **ADCCTL.ADCSTART** will auto clear in single conversion mode. To stop a conversion manually clear **ADCCTL.ADCSTART**.

**Figure 16-2. ADC Conversion (Single Shot)**



**Figure 16-3. ADC Conversion (Repeat Mode)**



### 16.3.5. EMUX

The EMUX is a low latency high speed serial interface with 8-bit data message to control the external ADC MUX and S/H in the analog front end. The EMUX interface is independent from the SOC BUS bridge.

The clock frequency of the EMUX can be adjusted with **EMUXCTL.EMUXCDIV** from HCLK/1 to HCLK/8.

To allow use of EMUX with auto sequencer ASC0 and ASC1, **EMUXCTL.EMUXC** must be set to 1b.



In manual mode with **EMUXCTL.EMUXC** = 0b, the EMUX will start sending the message MSB first as soon as a 8-bit message is written to **EMUXDATA**. While the message is transferred, **EMUXCTL.EMUXDONE** is cleared and is set to 1b when the message transfer is complete.

### 16.3.6. Auto Sequencer ASC0, ASC1

The ADC and EMUX can be controlled with 2 independent auto sequencer state machines ASC0 and ASC1 to offload the CPU from high speed, low latency sampling. Each sequencer can be programmed to take up to 8 consecutive ADC samples from different analog inputs.

#### 16.3.6.1. Auto Sequencer Modes

The AC0, ASC1 support 8 different modes, configurable with **ADCCTL.ADCMODE**.

With **ADCCTL.ADCMODE** = 000b ASC0 and ASC1 are disabled and the ADC is used in manual mode.

With **ADCCTL.ADCMODE** = 001b only ASC0 is active and manually triggered with **AS0CTL.AS0EN**.

With **ADCCTL.ADCMODE** = 010b only ASC1 is enabled and manually triggered with **AS1CTL.AS1EN**.

With **ADCCTL.ADCMODE** = 011b ASC0 and ASC1 are enabled and daisy chained. When manually triggered with **AS0CTL.AS0EN**. ASC0 will convert all programmed samples, when finished ASC0 will automatically trigger ASC1.

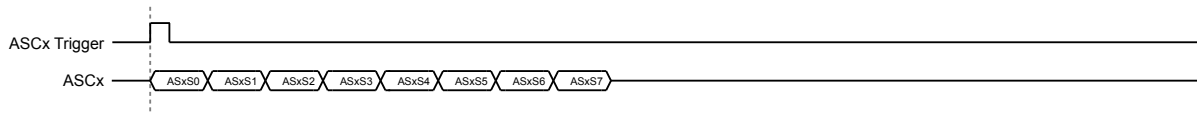
With **ADCCTL.ADCMODE** = 100b only ASC0 is active and triggered with trigger source configured in **AS0CTL.AS0TR**.

With **ADCCTL.ADCMODE** = 101b only ASC1 is active and triggered with trigger source configured in **AS1CTL.AS1TR**.

With **ADCCTL.ADCMODE** = 110b ASC0 and ASC1 are enabled and daisy chained. When triggered with source defined in **AS0CTL.AS0TR**, ASC0 will convert all programmed samples, when finished ASC0 will automatically trigger ASC1.

With **ADCCTL.ADCMODE** = 111b ASC0 and ASC1 are enabled and run independently. ASC0 is triggered with **AS0CTL.AS0TR**, ASC1 is triggered with **AS1CTL.AS1TR**.

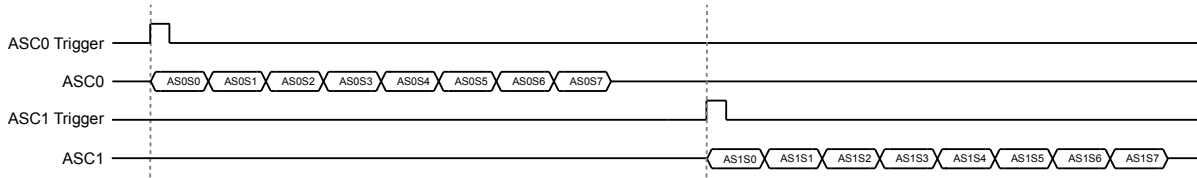
**Figure 16-4. ASCx, ADCCTL.ADCMODE = 001b, 010b, 100b, 101b**



**Figure 16-5. ASCx, ADCCTL.ADCMODE = 011b, 110b**



**Figure 16-6. ASCx, ADCCTL.ADCMODE = 111b**



### 16.3.6.2. Sequencer trigger

Each sequencer ASC0 and ASC1 can use 2 different trigger modes, manual mode or automated mode.

In automated mode use **ASxCTL.ASxTR** to set the trigger source either to timer A, B, C, or D or PWMAx, PWMBx, PWMCx or PWMDx. Use **AS0xCTL.ASxTRE** to set rising or falling edge trigger. Use **AS0xCTL.ASxTMR** to select timer source or **AS0xCTL.ASxTRPWM** to PWM source.

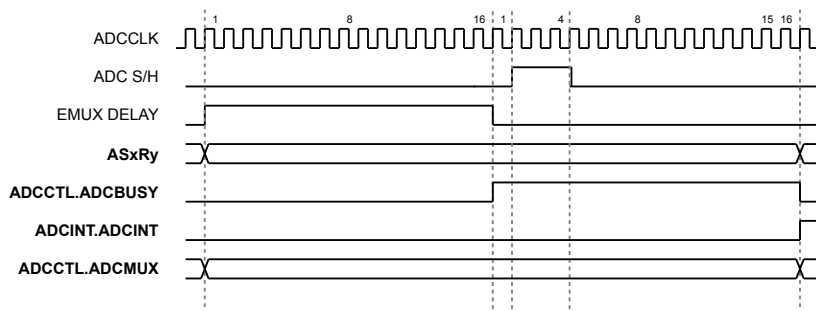
### 16.3.6.3. ASC Samples

Each sequencer can be programmed to take 1 to 8 samples up on triggering using **ASxCTL.AsxD**. For each sample, the ADC channel can be programmed with **ASxSy.ADCMUX**, a delay between MUX change and ADC start using **ASxSy.DELAY**, a EMUX message to be send with **ASxSy.EMUXD**, and a configuration with **ASxSy.EMUXS** to not send EMUXD, send right after ADCMUX change or send right after start of delay.

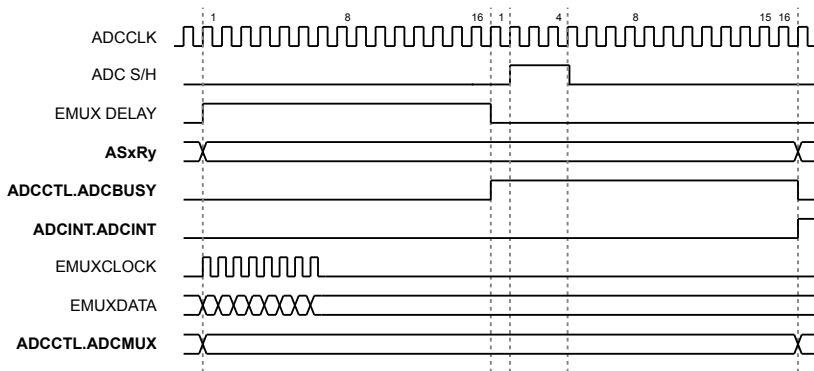
#### **NOTE:**

Make sure that the EMUX transmission is finished within delay time or ADC conversion time by choosing the correct EMUX clock divider setting.

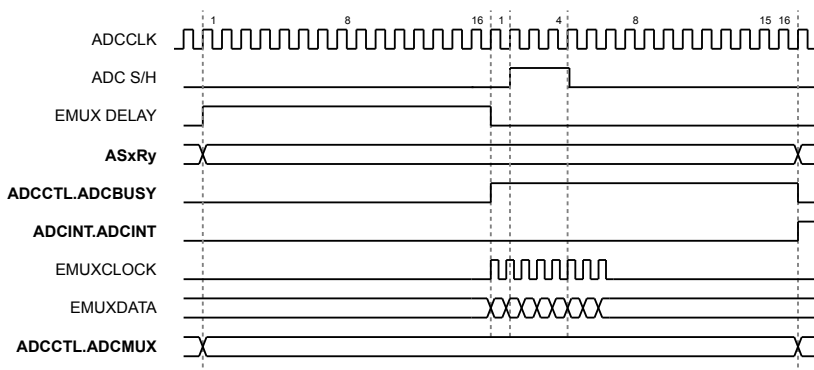
**Figure 16-7. ASxSy Sample with ASxSy.EMUXS = 00b and ASxSy.DELAY = 11b**



**Figure 16-8. ASxSy Sample with ASxSy.EMUXS = 01b and ASxSy.DELAY = 11b**



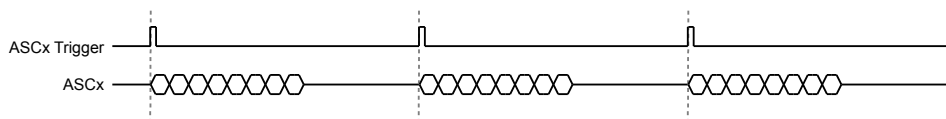
**Figure 16-9. ASxSy Sample with ASxSy.EMUXS = 10b and ASxSy.DELAY = 11b**



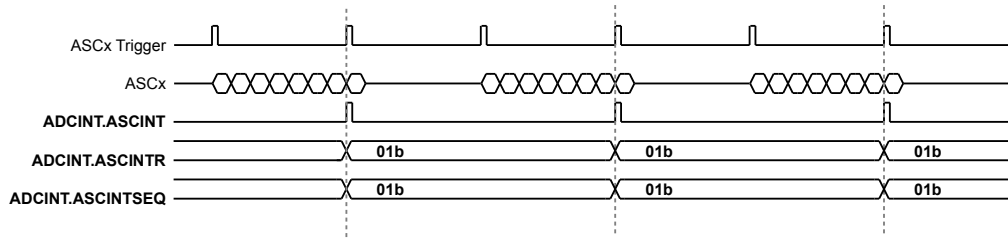
#### 16.3.6.4. ASC0, ASC1 Priority and Collision

In **ADCCTL.ADCMODE** = 100b, 101b, 110b the ASC are triggered with external trigger timer or PWM. Care has to be taken to space the trigger wide enough to allow sequencer ASCx to finish all samples. In case the sequencer trigger event happens before ASC sequencer finishes all samples, the **ADCINT.ADCINT** collision interrupt will be set and the trigger will be ignored. When **ADCINT.ADCINT** is set, **ADCINT.ADCINTSEQ** shows the ASC0 or ASC1 trigger causing the collision interrupt and **ADCINT.ADCINTR** the actual running sequencer.

**Figure 16-10. ASCx, 8 samples, No Collision**



**Figure 16-11. ASCx 8 samples, Collision**

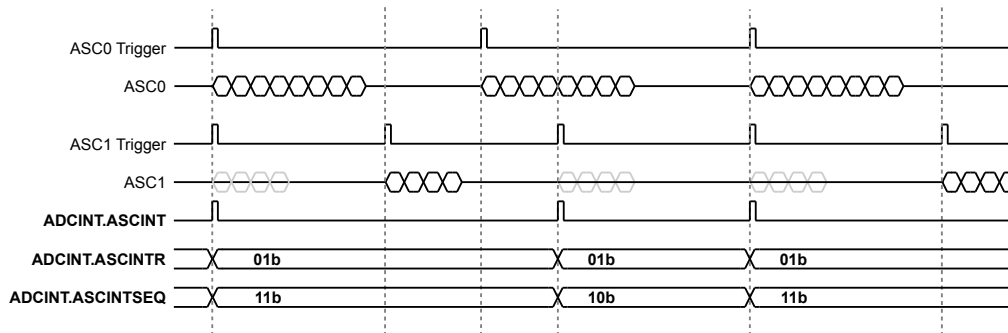


In **ADCCTL.ADCMODE** = 111b, the ASC0 and ASC1 sequencer are triggered independently but are accessing the same ADC.

In case of both ASC0 and ASC1 are triggered at the same time, ASC0 has always higher priority and will be executed while ASC1 is skipped and ignored. **ADCINT.ASCINT** will be set, **ADCINT.ASCINTSEQ** shows the ASC0 or ASC1 trigger causing the collision interrupt and **ADCINT.ASCINTR** the actual running sequencer.

In case of ASC0 or ASC1 sequencer running while the other is triggered, the second sequencer trigger is skipped and ignored, **ADCINT.ASCINT** will be set, **ADCINT.ASCINTSEQ** shows the ASC0 or ASC1 trigger causing the collision interrupt and **ADCINT.ASCINTR** the actual running sequencer.

**Figure 16-12. ASC0 8 samples, ASC1 4 samples, Collision**



## 17. I<sup>2</sup>C

### 17.1. Register

#### 17.1.1. Register Map

**Table 17-1. I<sup>2</sup>C Register Map**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>I<sup>2</sup>C</b>			
0x401B 0000	<b>I2CCFG</b>	I <sup>2</sup> C configuration	0x0000 0000
0x401B 0004	<b>I2CINTSTAT</b>	I <sup>2</sup> C interrupt and status	0x0000 0000
0x401B 0008	<b>I2CINTEN</b>	I <sup>2</sup> C interrupt enable	0x0000 0000
0x401B 0030	<b>I2CMCTL</b>	I <sup>2</sup> C master access control	0x0000 0000
0x401B 0034	<b>I2CMRXDATA</b>	I <sup>2</sup> C master receive data	0x0000 0000
0x401B 0038	<b>I2CMTXDATA</b>	I <sup>2</sup> C master transmit data	0x0000 0000
0x401B 0040	<b>I2CBAUD</b>	I <sup>2</sup> C master baud rate	0x01EC 01EC
0x401B 0070	<b>I2CSRXDATA</b>	I <sup>2</sup> C slave receive data	0x0000 0000
0x401B 0074	<b>I2CSTXDATA</b>	I <sup>2</sup> C slave transmit data	0x0000 0000
0x401B 0078	<b>I2CSADDR</b>	I <sup>2</sup> C slave address	0x0000 0000

#### 17.1.2. I2CCFG

**Register 17-1. I2CCFG (I<sup>2</sup>C Configuration, 0x401B 0000)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:6	<b>Reserved</b>	R	0x0	Reserved
5	<b>Reserved</b>	RW	0x1	Reserved, must be set to 1b
4	<b>ADDRMODE</b>	RW	0x0	Address Mode 1b: 10-bit addressing 0b: 7-bit addressing
3	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0b
2	<b>MAEN</b>	RW	0x0	Master 1b: I2C Master enable 0b: I2C Master enable
1	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0b
0	<b>SLEN</b>	RW	0x0	Slave Enable 1b: I2C Slave enable 0b: I2C Slave disable

#### 17.1.3. I2CINT

**Register 17-2. I2CINT (I<sup>2</sup>C Interrupt, 0x401B 0004)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:25	<b>Reserved</b>	R	0x0	Reserved

BIT	NAME	ACCESS	RESET	DESCRIPTION
24	<b>SLXFERDONEINT</b>	R	0x0	Slave Transfer 1b = Slave transfer complete, clears on read 0b = Slave transfer not done
23:19	<b>Reserved</b>	R	0x0	Reserved
18	<b>SLRXFINT</b>	R	0x0	Slave receive data register SLRXDATA full 1b: SLRXDATA received data from I2C bus, clears on read 0b: SLRXDATA did not receive data since last read of I2CINT
17	<b>SLTXEINT</b>	R	0x0	Slave transmit data register SLTXDATA empty 1b: SLTXDATA transmitted to I2C bus, clears on read 0b: SLTXDATA not transmitted since last read of I2CINT
16	<b>SLADDRMINT</b>	R	0x0	Slave Address match 1b: Slave address match detected, clears on read 0b: no match
15:12	<b>Reserved</b>	R	0x0	Reserved
11	<b>MADACKINT</b>	R	0x0	Master data acknowledge 1b: Master data NACK'd, clears on read 0b: Master data ACK'd
10	<b>MAARBLINT</b>	R	0x0	Master lost arbitration 1b: Master lost arbitration, clear on read 0b: no error
9	<b>MAADDRACKINT</b>	R	0x0	Master address acknowledge 1b: Master address NACK'd, clears on read 0b: Master address ACK'd
8	<b>MAXFERDONEINT</b>	RW	0x0	Master transfer complete 1b: Master transfer complete, clears on read 0b: not done
7:3	<b>Reserved</b>	R	0x0	Reserved
2	<b>MARXF</b>	R	0x0	Master receive data register MARXDATA full 1b: MARXDATA received data from I2C bus, clears on read 0b: MARXDATA did not receive data since last read of I2CINT
1	<b>MACTLE</b>	RW	0x0	MACCTL access register accessed 1b: I2CMACTL processed by I2C engine, clears on read 0b: I2CMACTL not accessed by I2C engine since last read of I2CINT
0	<b>MATXE</b>	R	0x0	Master transmit data register MATXDATA empty 1b: MATXDATA transmitted to I2C bus, clears on read 0b: MATXDATA not transmitted since last read of I2CINT

#### 17.1.4. I2CINTEN

**Register 17-3. I2CINTEN (I<sup>2</sup>C Interrupt Enable, 0x401B 0008)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:25	Reserved	R	0x0	Reserved
24	SLXFERDONEIN TEN	RW	0x0	SLXFERDONE Interrupt enable 1b: interrupt enable 0b: interrupt disabled
23:19	Reserved	R	0x0	Reserved
18	SLRXF	R	0x0	SLRXF Interrupt enable 1b: interrupt enable 0b: interrupt disabled
17	SLTXE	R	0x0	SLTXE Interrupt enable 1b: interrupt enable 0b: interrupt disabled
16	SLADDRM	R	0x0	SLADDRM Interrupt enable 1b: interrupt enable 0b: interrupt disabled
15:9	Reserved	R	0x0	Reserved
8	MAXFERDONE	R	0x0	MAXFERDONE Interrupt enable 1b: interrupt enable 0b: interrupt disabled
7:3	Reserved	R	0x0	Reserved
2	MARXF	R	0x0	MARXF Interrupt enable 1b: interrupt enable 0b: interrupt disabled
1	MACTLE	R	0x0	MACTLE Interrupt enable 1b: interrupt enable 0b: interrupt disabled
0	MATXE	R	0x0	MATXE Interrupt enable 1b: interrupt enable 0b: interrupt disabled

#### 17.1.5. I2CMACTL

**Register 17-4. I2CMACTL (I<sup>2</sup>C Master Access Control, 0x401B 0030)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:14	Reserved	R	0x0	Reserved
13	I2CMACTLF	R	0x0	I2CMACTL full 1b: I2CMACTL full, write not allowed, read to clear 0b: I2CMACTL processed, write allowed
12	Reserved	R	0x0	Reserved
11	XFERTYPE	RW	0x0	Master transfer type 1b: I <sup>2</sup> C Master Read 0b: I <sup>2</sup> C Master Write
10	RSTART	RW	0x0	Repeated start 1b: No STOP at end of transfer Repeated START 0b: STOP at end of transfer
9:7	I2CADDRU	RW	0x0	Upper I <sup>2</sup> C address bit 9:7
6:0	I2CADDRL	RW	0x0	Lower I <sup>2</sup> C address bit 6:0

### 17.1.6. I2CMARXDATA

#### Register 17-5. I2CMARXDATA (I<sup>2</sup>C Master Receive Data, 0x401B 0034)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:9	Reserved	RW	0x0	Reserved
8	I2CMARXDATAF	R	0x0	I2CMARXDATA full 1b: I2CMARXDATA register full, clear by read 0b: I2CMARXDATA register empty
7:0	MARXDATA	RW	0x0	Master Data Byte received

### 17.1.7. I2CMATXDATA

#### Register 17-6. I2CMATXDATA (I<sup>2</sup>C Master Transmit Data, 0x401B 0038)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:9	Reserved	RW	0x0	Reserved
9	LBYTE	RW	0x0	Last Byte of Transfer 1b: Last byte of READ or WRITE indicator, initiate STOP after data transfer
8	I2CMATXDATAF	R	0x0	I2CMATXDATA full 1b: I2CMATXDATA register full, data not transmitted 0b: I2CMATXDATA register empty
7:0	MATXDATA	RW	0x0	Master Data Byte to transmit

### 17.1.8. I2CBAUD

#### Register 17-7. I2CBAUD (I<sup>2</sup>C Baud Rate, 0x401B 0040)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:27	Reserved	R	0x0	Reserved
26:16	SCLH	RW	0x1EC	Number of HCLK cycles for I2CCL high time
15:11	Reserved	R	0x0	Reserved
10:0	SCLL	RW	0x1EC	Number of HCLK cycles for I2CCL low time

### 17.1.9. I2CSLRXDATA

#### Register 17-8. I2CSLRXDATA (I<sup>2</sup>C Slave Receive Data, 0x401B 0070)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:9	Reserved	RW	0x0	Reserved
8	I2CSLRXDATAF	R	0x0	I2CSLRXDATA full 1b: I2CSLRXDATA register full, data not transmitted 0b: I2CSLRXDATA register empty
7:0	SLRXDATA	RW	0x0	Slave Data Byte received



### 17.1.10. I2CSLTXDATA

#### Register 17-9. I2CSLTXDATA (I<sup>2</sup>C Slave Transmit Data, 0x401B 0074)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:9	Reserved	RW	0x0	Reserved
9	I2CSLTXDATAF	R	0x0	I2CSLTXDATA full 1b: I2CSLTXDATA register full, data not transmitted 0b: I2CSLTXDATA register empty
8	NACK	RW	0x0	Slave ACK or NACK 1b: Issue NACK on I <sup>2</sup> C Write 0b: Issue ACK on I <sup>2</sup> C Write
7:0	SLTXDATA	RW	0x0	Slave Data Byte to transmit

### 17.1.11. I2CSLADDR

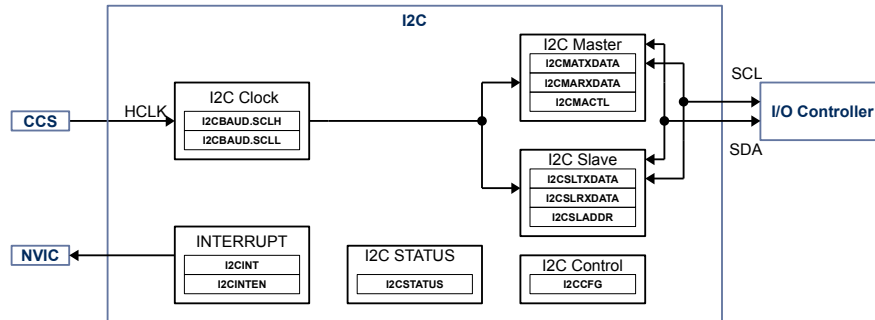
#### Register 17-10. I2CSLADDR (I<sup>2</sup>C Slave Address, 0x401B 0074)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:10	Reserved	RW	0x0	Reserved
9:7	SLADDRH	R	0x0	Higher Slave address bit 9:7
6:0	SLADDRL	RW	0x0	Lower Slave address bit 6:0

## 17.2. Details of Operation

### 17.2.1. Block Diagram

Figure 17-1. I2C



### 17.2.2. Configuration

Following blocks need to be configured for correct use of the I2C:

- Clock Control System (CCS)
- Nested Vectored Interrupt Controller (NVIC)
- IO Controller

### 17.2.3. I2C

The I2C Controller has one master and one slave connected to the same I/O that can be configured to be master only, slave only or concurrent master/slave. The I2C controller supports Normal mode (100kHz), Fast mode (400kHz), and Fast Mode+ (1MHz) operation. And 7-bit/ 10-bit addressing.

The master supports both single master and multi-master, multi-master sync and multi-master arbitration.

The slave supports clock stretching as well.

### 17.2.4. I2C Clock setting

The I2C SCLK frequency is derived from HCLK, **I2CBAUD.SCLH** sets the SCLK high pulse and **I2CBAUD.SCLL** sets the SCLK low pulse in HCLK cycles and need to be set correctly for different I2C mode.

The minimum HCLK for correct function of the I2C block is: 2.8MHz for normal mode, 3.2MHz for fast mode and 6.14MHz for fast+ mode.

The table below shows pre-calculated **I2CBAUD** settings for normal, fast and fast+ mode with 50MHz HCLK.

**Table 17-11. I2CBAUD settings for different HCLK**

I2C Mode	SCLK Frequency	HLCK	I2CBAUD.SCLH	I2CBAUD.SCLL
Normal	100kHz	50MHz	0xFA	0xFA
Normal	100kHz	4MHz	0x14	0x14
Normal	100kHz	2.8MHz	0x0E	0x0E
Fast	400kHz	50MHz	0x3E	0x3E
Fast	400kHz	4MHz	0x3E	0x3E
Fast	400kHz	3.2MHz	0x04	0x04
Fast+	1000kHz	50MHz	0x18	0x18
Fast+	1000kHz	6.14MHz	0x03	0x03

### 17.2.5. I2C Addressing

The I2C address for I2C master is set in **I2CMACTL.I2CADDRL** and **I2CMACTL.I2CADDRH**. The slave address is set **I2CSLADDR.SLADDRL** and **I2CSLADDR.SLADDRH**.

## 18. UART

### 18.1. Register

#### 18.1.1. Register Map

**Table 18-1. UART Register Map**

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>UART</b>			
0x401D 0000	<b>UARTRTX</b>	UART receive/transmit FIFO	0x0000 0000
0x401D 0004	<b>UARTIEN</b>	UART interrupt enable	0x0000 0000
0x401D 0008	<b>UARTII</b>	UART interrupt identification	0x0000 0001
0x401D 000C	<b>UARTLC</b>	UART line control	0x0000 0000
0x401D 0010	<b>Reserved</b>	Reserved	0x0000 0000
0x401D 0014	<b>UARTLS</b>	UART line status	0x0000 0060
0x401D 0018	<b>Reserved</b>	Reserved	unknown
0x401D 001C	<b>Reserved</b>	Reserved	0x0000 0000
0x401D 0020	<b>UARTFCTL</b>	UART FIFO control	0x0000 0000
0x401D 0024	<b>UARTIEN</b>	UART interrupt enable	0x0000 0000
0x401D 0028	<b>UARTDL_L</b>	UART divisor latch low byte	0x0000 0000
0x401D 002C	<b>UARTDL_H</b>	UART divisor latch high byte	0x0000 0000
0x401D 0038	<b>UARTFD_F</b>	UART fractional divisor value	0x0000 0000
0x401D 003C	<b>Reserved</b>	Reserved	0x0000 0000
0x401D 0040	<b>UARTFSTAT</b>	UART FIFO status	0x0000 0005

#### 18.1.2. UARTRTX/UARTDL\_L

**Register 18-1. UARTRTX (UART Receive/Transmit FIFO, 0x401D 0000)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	R	0x0	Reserved
7:0	<b>RTX</b>	RW	0x0	Receive and Transmit FIFO buffer on READ: RX FIFO on WRITE: TX FIFO

### 18.1.3. UARTIEN/UARTDL\_H

#### Register 18-2. UARTIEN (UART Interrupt Enable, 0x401D 0004)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:4	<b>Reserved</b>	R	0x0	Reserved
3	<b>MSINTEN</b>	RW	0x0	Modem Status interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>RSINTEN</b>	RW	0x0	Receive interrupt enable 1b: enable interrupt 0b: disable interrupt
1	<b>TXINTEN</b>	RW	0x0	TX register empty interrupt enable 1b: enable interrupt 0b: disable interrupt
0	<b>RXINTEN</b>	RW	0x0	RX register data available interrupt enable 1b: enable interrupt 0b: disable interrupt

### 18.1.4. UARTII

#### Register 18-3. UARTII (UART Interrupt Identification, 0x401D 0008)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	R	0x0	Reserved
7	<b>RXFEN</b>	R	0x0	RX FIFO enable flag 1b: enabled 0b: disabled
6	<b>TXFEN</b>	R	0x0	TX FIFO enable flag 1b: enabled 0b: disabled
5:4	<b>Reserved</b>	R	0x0	Reserved
3:1	<b>UARTINTID</b>	R	0x0	UART Interrupt type 111b: reserved 110b: Timeout 101b: reserved 100b: reserved 011b: RX Line Status 010b: RX Data Available 001b: TX Hold register empty 000b: Modem Status
0	<b>UARTINT</b>	R	0x1	UART Interrupt 1b: UART interrupt 0b: no UART interrupt

### 18.1.5. UARTLC

#### Register 18-4. UARTLC (UART Line Control, 0x401D 000C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	R	0x0	Reserved
7	<b>Reserved</b>	RW	0x0	Reserved, write to 0x0
6	<b>SB</b>	RW	0x0	Break Control 1b: force TX to 0b 0b: normal operation
5	<b>SP</b>	RW	0x0	Stick Parity 1b: enable 0b: disable
4	<b>EPS</b>	RW	0x0	Parity type 1b: generate EVEN parity 0b: generate ODD parity
3	<b>PEN</b>	RW	0x0	Parity Bit 1b: enable Parity 0b: disable Parity
2	<b>STB</b>	RW	0x0	Stop Bits 1b: 2 STOP bits (1.5 STOP bits for BPC=00) 0b: 1 STOP bit
1:0	<b>BPC</b>	RW	0x0	Bits per Character 11b: 8 bits 10b: 7 bits 01b: 6 bits 00b: 5 bits

### 18.1.6. UARTLS

#### Register 18-5. UARTLS (UART Line Status, 0x401D 0014)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	R	0x0	Reserved
7	<b>RXE</b>	R	0x0	RX FIFO Error 1b: at least 1 parity, framing or break error active in FIFO 0b: no error in RX FIFO
6	<b>TXE</b>	R	0x1	TX Empty 1b: TX shift register and TX FIFO are empty 0b: not empty
5	<b>TCFE</b>	R	0x1	TX FIFO Empty 1b: TX FIFO are empty 0b: not empty
4	<b>RXBE</b>	R	0x0	RX Break 1b: entry on top of RX FIFO has break error, bit clears on read 0b: error cleared
3	<b>RXFE</b>	R	0x0	RX Framing Error 1b: entry on top of RX FIFO has framing error, bit clears on read 0b: error cleared
2	<b>RXPE</b>	R	0x0	RX Parity Error 1b: entry on top of RX FIFO has parity error, bit clears on read 0b: error cleared
1	<b>RXOE</b>	R	0x0	RX Overrun error 1b: RX FIFO full and last entry overwritten, bit clears on read 0b: error cleared

BIT	NAME	ACCESS	RESET	DESCRIPTION
0	<b>RXDR</b>	R	0x0	RX Data ready 1b: at least 1 entry in RX FIFO 0b: RX FIFO empty

### 18.1.7. UARTFCTL

#### Register 18-6. UARTFCTL (FIFO Control, 0x401D 0020)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	R	0x0	Reserved
7:6	<b>RXFT</b>	RW	0x1	RX FIFO Threshold 11b: 14 Bytes in FIFO 10b: 8 Bytes in FIFO 01b: 4 Bytes in FIFO 00b: 1 Byte in FIFO
5:3	<b>Reserved</b>	R	0x0	Reserved
2	<b>TXFRESET</b>	RW	0x0	TX FIFO reset 1b: clear TX FIFO, bit auto clears 0b: no action
1	<b>RXFRESET</b>	RW	0x0	RX FIFO reset 1b: clear RX FIFO, bit auto clears 0b: no action
0	<b>FEN</b>	RW	0x0	FIFO enable 1b: enable RX, TX FIFO 0b: disable RX, TX FIFO

### 18.1.8. UARTIEN

#### Register 18-7. UARTIEN (UART Interrupt Enable, 0x401D 0024)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:4	<b>Reserved</b>	R	0x0	Reserved
3	<b>MSINTEN</b>	RW	0x0	Modem Status interrupt enable 1b: enable interrupt 0b: disable interrupt
2	<b>RSINTEN</b>	RW	0x0	Receive interrupt enable 1b: enable interrupt 0b: disable interrupt
1	<b>TXINTEN</b>	RW	0x0	TX register empty interrupt enable 1b: enable interrupt 0b: disable interrupt
0	<b>RXINTEN</b>	RW	0x0	RX register data available interrupt enable 1b: enable interrupt 0b: disable interrupt

### 18.1.9. UARTDL\_L

#### Register 18-8. UARTDL\_L (UART Divisor Latch Low Byte, 0x401D 0028)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	R	0x0	Reserved
7:0	<b>DL_L</b>	RW	0x0	Divisor value, low byte

#### 18.1.10. UARTDL\_H

##### Register 18-9. UARTDL\_H (UART Divisor Latch High Byte, 0x401D 002C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	R	0x0	Reserved
7:0	<b>DL_H</b>	RW	0x0	Divisor value, high byte

#### 18.1.11. UARTFD\_F

##### Register 18-10. UARTFD\_F (UART Fractional Divisor Value, 0x401D 0038)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	R	0x0	Reserved
7:0	<b>FRAC</b>	RW	0x0	Fractional divisor value

#### 18.1.12. UARTFSTAT

##### Register 18-11. UARTFSTAT (UART FIFO Status, 0x401D 0040)

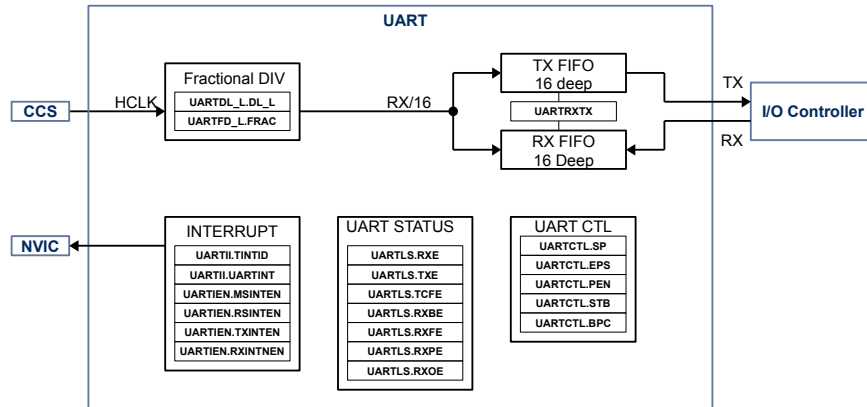
BIT	NAME	ACCESS	RESET	DESCRIPTION
31:4	<b>Reserved</b>	R	0x0	Reserved
3	<b>RXFF</b>	R	0x0	RX FIFO full 1b: RX FIFO full 0b: RX FIFO not full
2	<b>RXFE</b>	R	0x1	RX FIFO empty 1b: RX FIFO empty 0b: RX FIFO not empty
1	<b>TXFF</b>	R	0x0	TX FIFO full 1b: TX FIFO full 0b: TX FIFO not full
0	<b>TXFE</b>	R	0x1	TX FIFO empty 1b: TX FIFO empty 0b: TX FIFO not empty



## 18.2. Details of Operation

### 18.2.1. Block Diagram

Figure 18-1. UART



### 18.2.2. Configuration

Following blocks need to be configured for correct use of the UART:

- Clock Control System (CCS)
- Nested Vectored Interrupt Controller (NVIC)
- I/O Controller

### 18.2.3. UART

The UART supports up to 115,200 k-bit communication speed, has programmable clock selection with fractional divider, loop back mode for testing, 16 Byte transmit and 16 Byte receive FIFO with programmable receive interrupt threshold.

### 18.2.4. UART Clock Rate Setting

The UART block has a fractional divider to set up the baud rate. The UART clock is fed by the HCLK. The UART clock must be set to 16x the desired RX TX baud rate setting for correct functioning.

To calculate settings for **UARTDL\_H**, **UARTDL\_L**, **UARTFD\_F**, first calculate the desired divider setting using following formula:

$$UARTDivisor = \frac{HCLK}{BAUDRATE * 16} \quad (6)$$

Where:

UARTDivisor: calculated divisor

HCLK: HCLK frequency in Hz

BAUDRATE: desired Baud rate

The integer portion of UARTDivisor is used to set **UARTDL\_H**, **UARTDL\_L**.

To calculate the value of **UARTFD\_F**, use formula below and round to the nearest integer.

$$UARTFD = UARTDivisor_{frac} * 256 \quad (7)$$

Where:

UARTFD: calculated UARTFD value

UARTDIVISOR\_FRAC: UARTDivisor fractional value

To calculate Baud rate error use following:

$$BAUDRATEERROR = BAUDRATE - (HCLK / (UARTDivisor + UARTDivisor_{frac} / 256)) / 16 \quad (8)$$

Where:

BAUDRATEERROR: absolute baud rate error

BAUDRATE: Baudrate

HCLK: HCLK frequency in Hz

UARTDivisor: UART divisor integer value

UARTDivisor\_frac: UART divisor fractional value

To calculate relative Baud rate error use following:

$$Relative\ BAUDRATEERROR = BAUDRATEERROR / BAUDRATE * 100 \quad (9)$$

Where:

Relative BAUDRATE ERROR: relative BAUD rate error in %.

BAUDRATEERROR: absolute Baudrateerror

BAUDRATE: desired baudrate setting

The table below shows pre-calculated divisor settings for common Baud rates with 50MHz HCLK.

#### Register 18-12. UART Divisor Settings for 50 MHz HCLK

Baud Rate	Desired Divisor	Int	frac	UARTDL_H	UARTDL_L	UARTFD_F	Absolute BAUD rate error	BAUD rate error %
300	10416.667	10416	171	0x28	0xB0	0xAB	0.000010	0.0000%
600	5208.333	5208	85	0x14	0x58	0x55	-0.000038	0.0000%
900	3472.222	3472	57	0x0D	0x90	0x39	-0.000058	0.0000%
1200	2604.167	2604	43	0x0A	0x2C	0x2B	0.000154	0.0000%
2400	1302.083	1302	21	0x05	0x16	0x15	-0.000614	0.0000%
4800	651.042	651	11	0x02	0x8B	0x0B	0.002458	0.0001%
9600	325.521	325	133	0x01	0x45	0x85	0.004915	0.0001%
19200	162.760	162	195	0x00	0xA2	0xC3	-0.049152	-0.0003%
38400	81.380	81	97	0x00	0x51	0x61	-0.1	-0.0003%
57600	54.253	54	65	0x00	0x41	0x41	-0.501355	-0.0009%
115200	27.127	27	33	0x00	0x21	0x21	1.120655	0.0010%

### 18.2.5. Data settings

The **UARTLC** register defines the character settings like number of data bits, parity and number of stop bits

### 18.2.6. FIFO Settings

The FIFO can be configured with the **UARTFCTL** register. FIFO enable, depth and TX/RX FIFO reset can be configured. The FIFO status can be monitored in the **UARTLS** register. A write to **UARTRTX** writes to the TX FIFO, while a read from **UARTRTX** reads from RX FIFO.

### 18.2.7. Error Checking on Received Data

The character specific error does not show up in **UARTLS** until the character is at the top of the RX FIFO.

Each captured character is checked for following errors

- Break Error **UARTLS.RXBE** – there was a logic '0' detected on the RX input for more than one character transmission period ( BAUD RATE \*(1 startbit + **UARTLC.BPC** data bits + 1 parity bit + **UARTLC.STB** stopbits)
- Framing Error **UARTLS.RXFE** – there was a logic '0' detected where there should be a STOP bit
- Parity Error **UARTLS.PE** – received parity and calculated parity do not match.

## 19. SOC BUS BRIDGE

### 19.1. Register

#### 19.1.1. Register Map

Table 19-1. SOC Bus Bridge Register Map

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>SOC Bus Bridge</b>			
0x4020 0000	<b>SOCBCTL</b>	SOC Bus Bridge control	0x0000 0000
0x4020 0004	<b>SOCBCFG</b>	SOC Bus Bridge configuration	0x0000 0200
0x4020 0008	<b>SOCBCLKDIV</b>	SOC Bus Bridge clock divider	0x0000 0008
0x4000 000C	<b>Reserved</b>	Reserved	0x0000 0000
0x4000 0010	<b>Reserved</b>	Reserved	0x0000 0000
0x4020 0014	<b>SOCBSTAT</b>	SOC Bus Bridge status	0x0000 0000
0x4020 0018	<b>Reserved</b>	Reserved	0x0000 0000
0x4020 001C	<b>SOCBD</b>	SOC Bus Bridge data	0x0000 0000
0x4020 0020	<b>SOCBINT_EN</b>	SOC Bus Bridge interrupt enable	0x0000 0034

#### 19.1.2. SOCBCTL

Register 19-1. SOCBCTL (SOC Bus Bridge Control, 0x4020 0000)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:9	<b>Reserved</b>	R	0x0	Reserved
8	<b>Reserved</b>	RW	0x0	Reserved, set to 0x0
7	<b>Reserved</b>	RW	0x0	Reserved, set to 0x1
6	<b>Reserved</b>	RW	0x0	Reserved, set to 0x1
5	<b>MTRARM</b>	W	0x0	MTRANS re-arm Writing a 1b to this bit re-arms the <b>SOCBCTL.MTRANS</b> operation by de-asserting the CSx chip select and returning the master mode state machine to IDLE.
3	<b>Reserved</b>	RW	0x0	Reserved, set to 0x0
2	<b>Reserved</b>	RW	0x0	Reserved, set to 0x0
1	<b>SIE</b>	RW	0x0	SOC Bus Bridge interrupt enable. 1b = Enable the interrupt 0b = Disable the interrupt
0	<b>SSEN</b>	RW	0x0	SOC Bus Bridge enable: 1b = Enable this module. 0b = Disable this module.

#### 19.1.3. SOCBCFG

Register 19-2. SOCBCFG (SOC Bus Bridge Configuration, 0x4020 0004)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:14	<b>Reserved</b>	R	0x0	Reserved

BIT	NAME	ACCESS	RESET	DESCRIPTION
13	Reserved	RW	0x0	Reserved, set to 0x0
12	Reserved	RW	0x0	Reserved, be set to 0x0
11	Reserved	RW	0x0	Reserved, be set to 0x1
10	Reserved	RW	0x0	Reserved, be set to 0x1
9:6	Reserved	RW	0x0	Reserved, be set to 0x0
5	Reserved	RW	0x0	Reserved, be set to 0x0
4	Reserved	RW	0x0	Reserved, be set to 0x0
3	Reserved	RW	0x0	Reserved, mbe set to 0x0
2	MRST	RW	0x0	Module reset. 1b: Force soft reset of module. The internal state machines are reset; Status register is cleared; However, the soft reset doesn't affect control register values. 0b: do not hold the module in reset.
1:0	Reserved	RW	0x0	Reserved, set to 0x0

#### 19.1.4. SOCBCLKDIV

##### Register 19-3. SOCBCLK (SOC Bus Bridge Clock Divider, 0x4020 0008)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	Reserved	R	0x0	Reserved
15:0	CLKDIV	RW	0x8	Clock divisor for SCLK: $SCLK = HCLK / [(CLKDIV+1)*2]$ the minimum divider is /2

#### 19.1.5. SOCBSTAT

##### Register 19-4. SOCBSTAT (SOC Bus Bridge Status, 0x4020 0014)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	Reserved	R	0x0	Reserved
14:12	CURSTATE	RW	0x0	Raw status of the SOC bus bridge master state machine's "current_state" register. 111b: CSBEGIN 110b: MTRANS 101b: CKWAIT 100b: CSWAIT 011b: CSHOLD 010b: TRANSFER 001b: CSSETUP 000b: IDLE
11	Reserved	R	0x0	Reserved
10	RXFULL	R	0x0	Raw indicator that the Rx incoming holding register contains a valid data word. 1b: Rx incoming holding register contains a valid data word. 0b: Rx incoming holding register contains no valid data word.
9	TXFULL	R	0x0	Raw indicator that the Tx outgoing holding register is still in use, and not ready to accept another data word. 1b: Tx outgoing holding register is still in use not ready to accept another data word. 0b: Tx outgoing register is ready to accept another data word

BIT	NAME	ACCESS	RESET	DESCRIPTION
8	<b>WRUFL</b>	RW	0x0	Write Buffer Underflow: set on the start of a second outgoing transfer if data hasn't been written to the <b>SOCBD</b> after the previous transfer 1b: Write Underflow detected, clear by writing 1b to it 0b: No Write Underflow since this bit was cleared  Note: This bit is cleared by writing a 1b to it. This bit is a sticky status bit, and will set upon meeting the condition regardless of the state of its corresponding interrupt enable <b>SOCBINT_EN.WRUFL_EN</b> .
7:6	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0x0
5	<b>CYC_DONE</b>	RW	0x0	Cycle Done: this bit will set when the current transfer of 8 bits is complete. It indicates that 8 bits were sent on the transmit port and 8 bits were sampled on the receive port. 1b: Cycle done detected, clear by writing 1b to it 0b: No Cycle Done detected since this bit was cleared  Note: This bit is cleared by writing a 1b to it. This bit is a sticky status bit, and will set upon meeting the condition regardless of the state of its corresponding interrupt enable <b>SOCBINT_EN.CYC_DONE_EN</b> .
4:3	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0x0
2	<b>RDOFL</b>	RW	0x0	Read Buffer Overflow: set on the completion of a second incoming transfer if data hasn't been read from the <b>SOCBD</b> from the previous transfer 1b: Read Overflow detected, clear by writing 1b to it 0b: No Read Overflow since this bit was cleared  Note: This bit is cleared by writing a 1b to it. This bit is a sticky status bit, and will set upon meeting the condition regardless of the state of its corresponding interrupt enable <b>SOCBINT_EN.RDOFL_EN</b> .
1	<b>Reserved</b>	R	0x0	Reserved
0	<b>SOCB_INT</b>	R	0x0	SOC bus bridge Interrupt Logical OR of each raw status bit WRUFL, RDOFL, and CYC_DONE, qualified with its corresponding SOCBINT_EN enable. 1b: interrupt 0b: no interrupt  Note that if the corresponding SOCBINT_EN of those status bits is reset to '0', those status bits themselves will still assert upon meeting the condition, but will not contribute to the assertion of SOCB_INT. The status bits are true "raw" status bits, and the corresponding SOCBINT_EN simply allows them to cause an interrupt.

### 19.1.6. SOCBD

#### Register 19-5. SOCBD (SOC Bus Bridge Data, 0x4020 001C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:8	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0x0
7:0	<b>DATA</b>	RW	0x0	SOC bus bridge data On READ: retrieve received data word from the incoming holding buffer. On WRITE: write a address or data word to the outgoing holding buffer.

### 19.1.7. SOCBINT\_EN

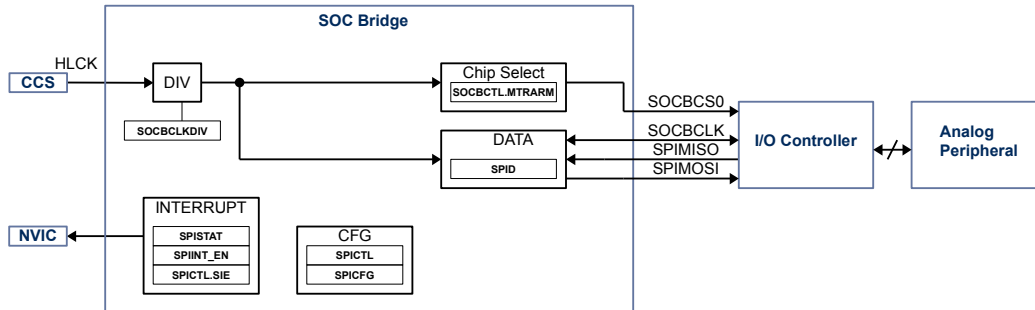
#### Register 19-6. SOCBINT\_EN (SOC Bus Bridge Interrupt Enable, 0x4020 0020)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:9	Reserved	RW	0x0	Reserved
8	WRUFL_EN	RW	0x0	Write buffer underflow WRUFL interrupt enable 1b: enable <b>SOCBSTAT.WRUFL</b> interrupt 0b: disable <b>SOCBSTAT.WRUFL</b> interrupt
7:6	Reserved	RW	0x0	Reserved, set to 0x0
5	Reserved	RW	0x1	Reserved, set to 0x0
4:3	Reserved	RW	0x1	Reserved, set to 0x0
2	Reserved	RW	0x1	Read buffer overflow RDOFL interrupt enable 1b: enable <b>SOCBSTAT.RDOFL</b> interrupt 0b: disable <b>SOCBSTAT.RDOFL</b> interrupt
1:0	Reserved	RW	0x0	Reserved

## 19.2. Details of Operation

### 19.2.1. Block Diagram

Figure 19-1. SOC Bridge



### 19.2.2. Configuration

Following blocks need to be configured for correct use of the SPI:

- Clock Control System (CCS)
- Nested Vectored Interrupt Controller (NVIC)
- IO Controller

### 19.2.3. SOC Bridge

The SOC bridge is used to set and read registers in the analog section of the device.

### 19.2.4. SOC Bridge Clock Rate Setting

The SOC bridge Module SOCCLK is derived from HCLK to drive the SPI logic, generate setup, hold and wait timings for CSx signals, and SOCCLK.

The SPICLK clock is derived from HCLK using a clock divider configurable with **SOCCLKDIV**. The lowest clock allowable divider in SPI slave mode is HCLK/8. In SPI Master mode the lowest allowable clock divider is HCLK/2.

To calculate SOCCLK use

$$SOCCLK = \frac{HCLK}{(SOCCLKDIV + 1) * 2} \quad (10)$$

Where:

SOCCLK: SOCCLK in Hz

HCLK: HCLK in Hz

SOCCLKDIV: **SOCCLKDIV** setting



### 19.2.5. Enable and Setup of SOC Bridge

### 19.2.6. SOC Interrupt

The SOC bridge engine interrupt is enabled with **SOCCTL.SIE**. Then any sub interrupts are enabled in **SINT\_EN**.

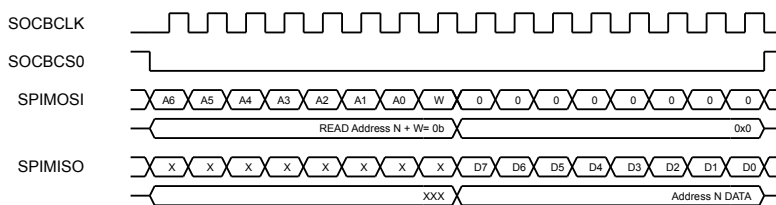
### 19.2.7. SOC Bridge Protocol

The SOC bridge protocol is a 2 byte protocol, the first byte is the address packet including a 7-bit address [7:1] and a write bit [0], the second packet is an 8bit data packet.

### 19.2.8. Reading from SOC Bridge

To read to the SOC bridge, start with writing the address packet to **SOCBD** first. Wait for **SOCBSTAT.CYC\_DONE** = 1b. Clear **SOCBSTAT.CYC\_DONE** by set to 1b. Then write a second dummy address packet to **SOCBD** to clock out the data packet. Wait for **SOCBSTAT.CYC\_DONE** = 1b. Clear **SOCBSTAT.CYC\_DONE** by set to 1b. Set **SOCBCTL.MTRARM** to 1b to deassert SOCBCS0. Then read from **SOCBD** to get the data packet content

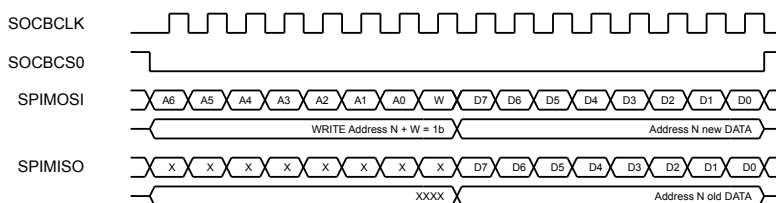
**Figure 19-2. Single Read from SOC Bridge**



### 19.2.9. Writing to SOC Bridge

To write to the SOC bridge, start with writing the address packet to **SOCBD** first. Wait for **SOCBSTAT.CYC\_DONE** = 1b. Clear **SOCBSTAT.CYC\_DONE** by set to 1b. Then write the data packet to **SOCBD**. Wait for **SOCBSTAT.CYC\_DONE** = 1b. Clear **SOCBSTAT.CYC\_DONE** by set to 1b. Set **SOCBCTL.MTRARM** to 1b to deassert SOCBCS0. Optional you read from **SOCBD** to get the data packet content with old data content of the register address.

**Figure 19-3. Single Write to SOC Bridge**



## 20. SPI

### 20.1. Register

#### 20.1.1. Register Map

Table 20-1. SPI Register Map

ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>SPI</b>			
0x4021 0000	<b>SPICTL</b>	SPI control	0x0000 0000
0x4021 0004	<b>SPICFG</b>	SPI configuration	0x0000 0200
0x4021 0008	<b>SPICLKDIV</b>	SPI clock divider	0x0000 0008
0x4021 000C	<b>Reserved</b>	Reserved	0x0000 0000
0x4021 0010	<b>Reserved</b>	Reserved	0x0000 0000
0x4021 0014	<b>SPISTAT</b>	SPI status	0x0000 0000
0x4021 0018	<b>SPICSSSTR</b>	SPI chip select steering	0x0000 0000
0x4021 001C	<b>SPID</b>	SPI data	0x0000 0000
0x4021 0020	<b>SPIINT_EN</b>	SPI interrupt enable	0x0000 0000

#### 20.1.2. SPICTL

Register 20-1. SPICTL (SPI Control, 0x4021 0000)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:9	<b>Reserved</b>	R	0x0	Reserved
8	<b>RTRANS</b>	RW	0x0	Auto-retransmit on UCLK error. 1b: On a UCLK error, the transmit holding register does NOT get reset, so the word that was transmitting when the UCLK error occurred remains queued. 0b: On a UCLK error, the transmit holding register is reset, and the word that was transmitting when the UCLK error occurred is lost.
7	<b>MMST_N</b>	RW	0x0	Multi-master mode (MASTER ONLY) 1b: Single-Master mode, always drive a value onto CSx, SPICLK and MOSI. 0b: Multi-master mode, always tri-state the CSx, SPICLK and MOSI lines when a transfer is complete.
6	<b>MTRANS</b>	RW	0x0	Multiple Transfer Mode (MASTER ONLY) 1b: Generate multiple transfers of [SPICFG.WL] bits within a single CSx assertion. 0b: Generate single transfers (assert CSx, transfer data word of [SPICFG.WL] bits, de-assert CSx).
5	<b>MTRARM</b>	W	0x0	MTRANS re-arm (MASTER ONLY): Writing a 1b to this bit re-arms the SPICTL.MTRANS operation by de-asserting the CSx chip select and returning the master mode state machine to IDLE.
4	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0x0
3	<b>SE</b>	RW	0x0	Slave Enable 1b: SPI is a SLAVE. 0b: SPI is a MASTER.

BIT	NAME	ACCESS	RESET	DESCRIPTION
2	<b>LPBK</b>	RW	0x0	Internal loop back Mode 1b: Tie the serial out source to the serial in line (internal signaling, does not traverse the chip IO bi-di buffers). 0b = Normal operation.
1	<b>SIE</b>	RW	0x0	SPI Interrupt enable. 1b = Enable the interrupt 0b = Disable the interrupt
0	<b>SSEN</b>	RW	0x0	SPI enable: 1b = Enable this module. 0b = Disable this module.

### 20.1.3. SPICFG

**Register 20-2. SPICFG (SPI Configuration, 0x4021 0004)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:14	<b>Reserved</b>	R	0x0	Reserved
13	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0x0
12	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0x0
11	<b>MTURBO</b>	RW	0x0	Master “turbo” operation mode: 1b: Enable master turbo mode, using HCLK-based bit count, allowing operation down to max 2:1 HCLK:SPICLK ratio 0b: Disable master turbo mode, legacy operation down to max 8:1 HCLK:SPICLK ratio.
10	<b>TXDBUF</b>	RW	0x0	Transmit Double-Buffer mode: 1b: enable double-buffer “ping-pong” on shift register transmit output path (keep up with back-to-back words at faster HCLK:SPICLK ratios) 0b: disable double-buffer, legacy operation with a single shift register buffer and single queuing buffer
9	<b>TXDATPH</b>	RW	0x0	Early Transmit Data Phase. 1b: Enable: MISO (slave mode) or MOSI(master mode) transitions occur ½ an SPICLK period sooner than the normal protocol (i.e. transition on capture edge instead of launch edge) 0b: Disable: normal transmit data phase, transitions are on the launch edge of SPICLK
8	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0x0
7	<b>RCVCPH</b>	RW	0x0	SLAVE MODE Clock Phase 1b: Second clock transition of a new transfer is used to sample data 0b: First clock transition of a new transfer is used to sample data
6	<b>RCVCP</b>	RW	0x0	SLAVE MODE Clock Polarity 1b: SPICLK is HI in its inactive state 0b: SPICLK is LO in its inactive state
5	<b>CPH</b>	RW	0x0	MASTER MODE Clock Phase 1b: Second clock transition of a new transfer is used to sample data 0b: First clock transition of a new transfer is used to sample data
4	<b>CP</b>	RW	0x0	MASTER MODE Clock Polarity 1b: SPICLK is HI in its inactive state 0b: SPICLK is LO in its inactive state
3	<b>LB1ST</b>	RW	0x0	Least Bit First 1b: LSB is the first serial bit of a transfer 0b: MSB is the first serial bit of a transfer

BIT	NAME	ACCESS	RESET	DESCRIPTION
2	<b>MRST</b>	RW	0x0	Module reset. 1b: Force soft reset of module. The internal state machines are reset; Status register is cleared; However, the soft reset doesn't affect control register values. 0b: do not hold the module in reset.
1:0	<b>WL</b>	RW	0x0	Word Length select 11b: Word Length = 32-bits 10b: Word Length = 24-bits 01b: Word Length = 16-bits 00b: Word Length = 8-bits

#### 20.1.4. SPICLKDIV

##### Register 20-3. SPICLKDIV (SPI Clock Divider, 0x4021 0008)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:16	<b>Reserved</b>	R	0x0	Reserved
15:0	<b>CLKDIV</b>	RW	0x8	Clock divisor for SCLK: $SCLK = HCLK / [(CLKDIV+1)*2]$ In Master/Slave mode with <b>SPICFG.MTURBO</b> = 0b, minimum divider is /8. In Master mode with <b>SPICFG.MTURBO</b> = 1b, minimum divider is /2

#### 20.1.5. SPISTAT

##### Register 20-4. SPISTAT (SPI Status, 0x4021 0014)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:15	<b>Reserved</b>	R	0x0	Reserved
14:12	<b>CURSTATE</b>	RW	0x0	Raw status of the SOC bus bridge master state machine's "current_state" register. 111b: CSBEGIN 110b: MTRANS 101b: CKWAIT 100b: CSWAIT 011b: CSHOLD 010b: TRANSFER 001b: CSSETUP 000b: IDLE
11	<b>Reserved</b>	R	0x0	Reserved
10	<b>RXFULL</b>	R	0x0	Raw indicator that the Rx incoming holding register contains a valid data word. 1b: Rx incoming holding register contains a valid data word. 0b: Rx incoming holding register contains no valid data word.
9	<b>TXFULL</b>	R	0x0	Raw indicator that the Tx outgoing holding register is still in use, and not ready to accept another data word. 1b: Tx outgoing holding register is still in use not ready to accept another data word. 0b: Tx outgoing register is ready to accept another data word

BIT	NAME	ACCESS	RESET	DESCRIPTION
8	<b>WRUFL</b>	RW	0x0	<p>Write Buffer Underflow</p> <p>Set on the start of a second outgoing transfer if data hasn't been written to the SD register after the previous transfer</p> <p>1b: Write Underflow detected, clear by writing 1b to it</p> <p>0b: No Write Underflow since this bit was cleared</p> <p>Note: This bit is cleared by writing a 1b to it. This bit is a sticky status bit, and will set upon meeting the condition regardless of the state of its corresponding interrupt enable SPIINT_EN.WRUFL_EN.</p>
7	<b>Reserved</b>	RW	0x0	Reserved, must be set to 0x0
6	<b>TE</b>	RW	0x0	<p>Chip Select Trailing Edge Detect</p> <p>1b: a chip select de-assertion was detected, clear by writing 1b to it</p> <p>0b: no chip select de-assertion detected since this bit was cleared</p> <p>NOTE: This bit is cleared by writing a 1b to it. This bit is a sticky status bit, and will set upon meeting the condition regardless of the state of its corresponding interrupt enable SPIINT_EN.TE_EN.</p>
5	<b>CYC_DONE</b>	RW	0x0	<p>Cycle Done: this bit will set when the current transfer of 8 bits is complete. It indicates that 8 bits were sent on the transmit port and 8 bits were sampled on the receive port.</p> <p>1b: Cycle done detected, clear by writing 1b to it</p> <p>0b: No Cycle Done detected since this bit was cleared</p> <p>NOTE: This bit is cleared by writing a 1b to it. This bit is a sticky status bit, and will set upon meeting the condition regardless of the state of its corresponding interrupt enable SPIINT_EN.CYC_DONE_EN.</p>
4	<b>UCLK</b>	RW	0x0	<p>Underclock Condition</p> <p>Set if the current transfer received less than word-length "WL" clocks on the SPICLK line prior to CSx de-assertion.</p> <p>1b: Underclock condition detected, clear by writing 1b to it</p> <p>0b: No underclock condition detected since this bit was cleared.</p> <p>NOTE: This bit is cleared by writing a 1b to it. This bit is a sticky status bit, and will set upon meeting the condition regardless of the state of its corresponding interrupt enable SPIINT_EN.UCLK_EN.</p>
3	<b>LE</b>	RW	0x0	<p>Chip Select Leading Edge Detect:</p> <p>1b: a chip select assertion was detected, clear by writing 1 to it</p> <p>0b: no chip select assertion detected since this bit was cleared</p> <p>NOTE: This bit is cleared by writing a 1b to it. This bit is a sticky status bit, and will set upon meeting the condition regardless of the state of its corresponding interrupt enable SPIINT_EN.LE_EN.</p>
2	<b>RDOFL</b>	RW	0x0	<p>Read Buffer Overflow: set on the completion of a second incoming transfer if data hasn't been read from the SD register from the previous transfer</p> <p>1b: Read Overflow detected, clear by writing 1b to it</p> <p>0b: No Read Overflow since this bit was cleared</p> <p>NOTE: This bit is cleared by writing a '1' to it. This bit is a sticky status bit, and will set upon meeting the condition regardless of the state of its corresponding interrupt enable SPIINT_EN.RDOFL_EN.</p>
1	<b>Reserved</b>	R	0x0	Reserved

BIT	NAME	ACCESS	RESET	DESCRIPTION
0	SPI_INT	R	0x0	SPI Interrupt Logical OR of each raw status bit WRUFL, RDOFL, and CYC_DONE, qualified with its corresponding INT_EN enable. 1b: interrupt 0b: no interrupt  NOTE: that if the corresponding INT_EN of those status bits is reset to '0', those status bits themselves will still assert upon meeting the condition, but will not contribute to the assertion of SPI_INT. The status bits are true "raw" status bits, and the corresponding SPIINT_EN simply allows them to cause an interrupt.

### 20.1.6. SPICSSTR

**Register 20-5. SPICSSTR (SPI Chip Select Steering, 0x4021 0018)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:24	Reserved	RW	0x0	Reserved
23:20	CKWAIT	RW	0x0	SPI Clock Wait (MASTER ONLY): only applies if SPICTL.MTRANS=1b(multiple transfers with one chip select assertion). This value determines the minimum number of SPICLK periods to wait between back-to-back transfers. During this wait time, SPICLK does not toggle but CSx remains active.
19:16	CSWAIT	RW	0x0	Chip Select Wait: This value determines the minimum number of SPICLK periods to wait between the de-assertion of CSx and the re-assertion of CSx
15:12	CSHOLD	RW	0x0	Chip Select Hold : minimum number of SPICLK periods to wait from the last SPICLK transition to de-assertion of CSx
11:8	CSSETUP	RW	0x0	Chip Select Setup: minimum number of SPICLK periods to wait from the assertion of Csx to the first SPICLK transition.
7:3	Reserved	R	0x0	Reserved
2	CSL	RW	0x0	Chip Select active level select (MASTER or SLAVE mode): 1b: active HI outgoing (master) or incoming (slave) chip select 0b: active LO outgoing (master) or incoming (slave) chip select
1:0	CSNUM	RW	0x0	Chip Select Number: Outgoing (MASTER, GPIO modes) - select which one of the possible four chip selects are asserted (the other three are driven de-asserted). Incoming (SLAVE, GPIO modes) – select which one of the possible four chip selects are actively used. 11b: reserved 10b: SPICS2 01b: SPICS1 00b: SPICS0

### 20.1.7. SPID

**Register 20-6. SPID (SPI Data, 0x4021 001C)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:0	DATA	RW	0x0	SOC bus bridge data On READ: retrieve received data word from the incoming holding buffer. On WRITE: write a data word to the outgoing holding buffer.

### 20.1.8. SPIINT\_EN

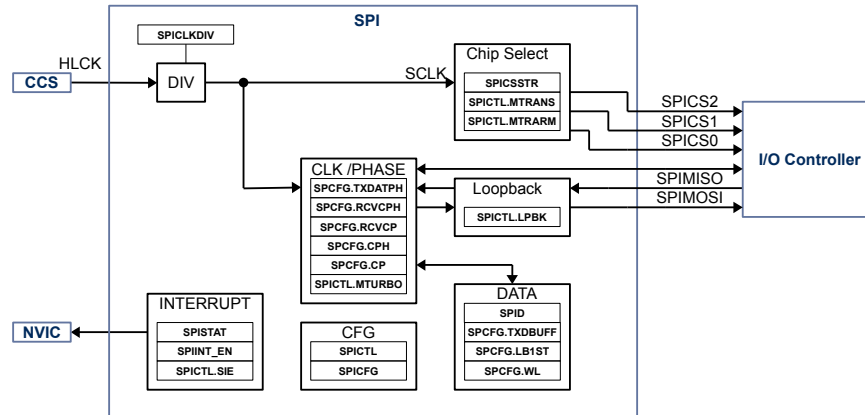
#### Register 20-7. SPIINT\_EN (SPI Interrupt Enable, 0x4021 0020)

BIT	NAME	ACCESS	RESET	DESCRIPTION
31:9	<b>Reserved</b>	RW	0x0	Reserved
8	<b>WRUFL_EN</b>	RW	0x0	Write buffer underflow WRUFL interrupt enable 1b: enable SPISTAT.WRUFL interrupt 0b: disable SPISTAT.WRUFL interrupt
7	<b>BP_DONE</b>	RW	0x0	Byte-packing BP_DONE interrupt enable 1b: enable SPISTAT.BP_DONE interrupt 0b: disable SPISTAT.BP_DONE interrupt
6	<b>TE_EN</b>	RW	0x0	Trailing Edge detect TE interrupt enable 1b: enable SPISTAT.TE interrupt 0b: disable SPISTAT.TE interrupt
5	<b>CYC_DONE_EN</b>	RW	0x0	Cycle done CYC_DONE interrupt enable 1b: enable SPISTAT.CYC_DONE interrupt 0b: disable SPISTAT.CYC_DONE interrupt
4	<b>UCLK_EN</b>	RW	0x1	Underclock condition detect UCLK interrupt enable 1b: enable SPISTAT.UCLK interrupt 0b: disable SPISTAT.UCLK interrupt
3	<b>LE_EN</b>	RW	0x0	Leading Edge detect LE interrupt enable 1b: enable SPISTAT.LE interrupt 0b: disable SPISTAT.LE interrupt
2	<b>RDOFL_EN</b>	RW	0x1	Read buffer overflow RDOFL interrupt enable 1b: enable SPISTAT.RDOFL interrupt 0b: disable SPISTAT.RDOFL interrupt
1:0	<b>Reserved</b>	RW	0x0	Reserved

## 20.2. Details of Operation

### 20.2.1. Block Diagram

Figure 20-1. SPI



### 20.2.2. Configuration

Following blocks need to be configured for correct use of the SPI:

- Clock Control System (CCS)
- Nested Vectored Interrupt Controller (NVIC)
- IO Controller

### 20.2.3. SPI

The SPI engine has selectable data byte ordering LSB or MSB first, 4 different data / clock modes, can send / receive packets 8, 16, 32, 64 boundaries, selectable CS polarity, soft reset, and auto-retransmit.

In master mode it supports up to 3 different slaves using chip select. The master mode also allows sending multiple packets per CS.

### 20.2.4. SPI Clock Rate Setting

The SPI Module SPICLK is derived from HCLK to drive the SPI logic, generate setup, hold and wait timings for CSx signals, and SPICLK.

The SPICLK clock is derived from HCLK using a clock divider configurable with **SPICLKDIV**. The lowest clock allowable divider in SPI slave mode is HCLK/8. In SPI Master mode the lowest allowable clock divider is HCLK/2 if **SPICFG.MTURBO** is 1b, HCLK/8 if **SPICFG.MTURBO** is 0b. **SPICFG.MTURBO** works only in Master mode.

To calculate SCLK use

$$SCLK = \frac{HCLK}{(SPICLKDIV + 1) * 2} \quad (11)$$



Where:

SCLK: SCLK in Hz

HCLK: HLCK in Hz

SPICLKDIV: **SPICLKDIV** setting

### 20.2.5. Master Slave Mode

The master mode is selected with **SPICTL.SE**= 0b. In master mode a write to **SPID** will initiate SPI data transfer.

When **SPICFG.TXDBUF** is set to 1b, the **SPID** is double buffering is enabled, allowing queuing of the next data word while the current one is transferred.

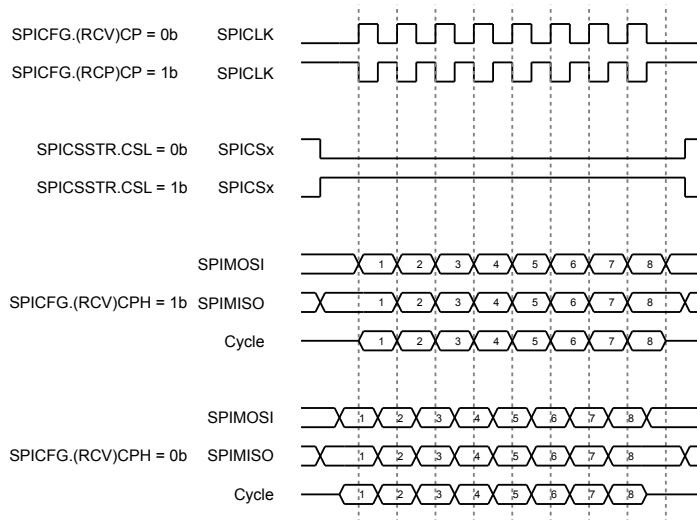
### 20.2.6. Clock Phase, Polarity

The clock and phase can be programmed independently for master and slave.

The master mode clock polarity is configured with **SPICFG.CP** and the phase is configured with **SPCFG.CPH**.

The slave mode clock polarity is configured with **SPICFG.RCVCP** and the phase is configured with **SPICFG.RCVCPH**.

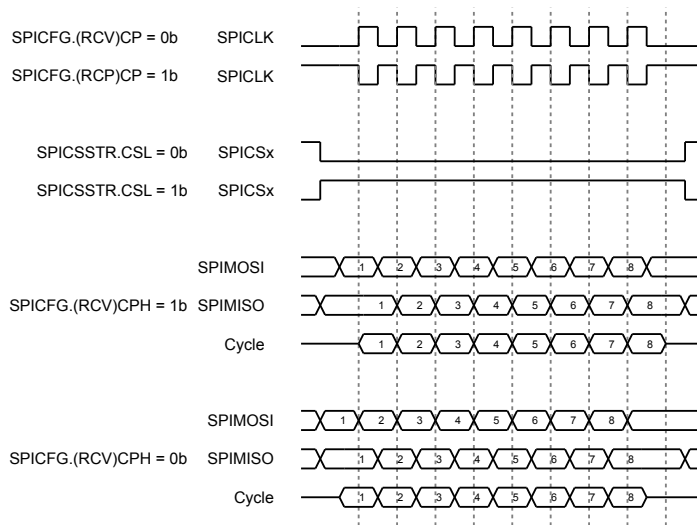
**Figure 20-2. SPI clock polarity and phase**



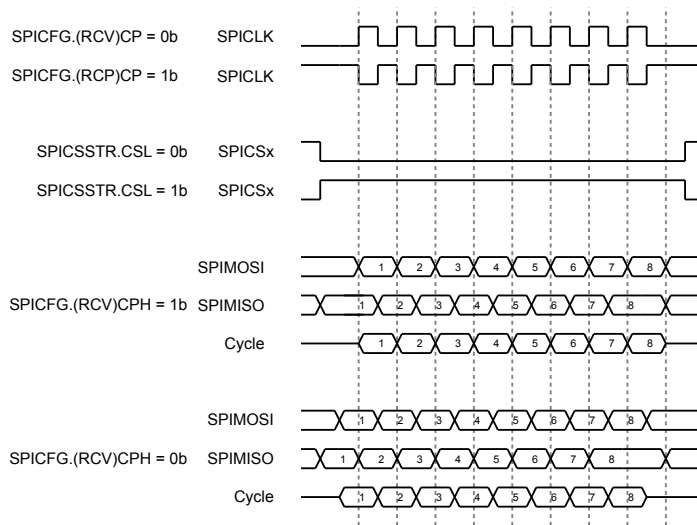
### 20.2.7. SPI Early Data Transmit

For cases when the SPI is running at high speed the transmitted data can be transmitted 1/2 a SPICLK early to compensate for delays on the receiver side. When **SPICFG.TXDATPH** is 1b, SPIMOSI is transmitted 1/2 SPICLK early in master mode. In slave mode SPIMISO is transmitted 1/2 SPICLK early.

**Figure 20-3. SPIMOSI early transmit in master mode**



**Figure 20-4. SPIMISO early transmit in slave mode**



### 20.2.8. Data Format

The **SPICFG.WL** sets the word length from 8bit to 32bit in 8bit steps.

The **SPICFG.LB1ST** configures the data format to transmit, LSB ior MSB first.

### 20.2.9. Chip Select Settings

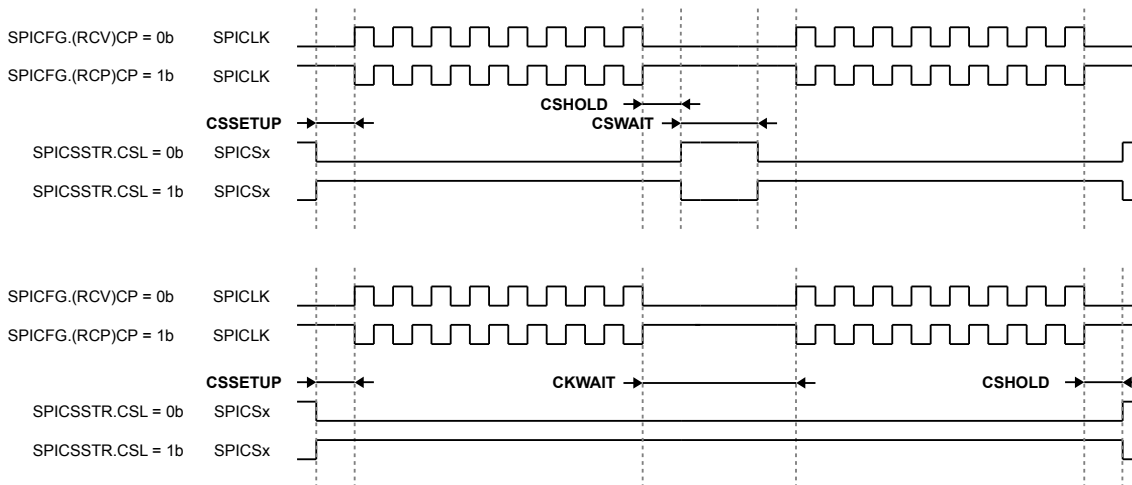
The SPI engine supports up to 3 chip select signals SPICS0, SPICS1 and SPICS2, selectable with **SPICSSTR.CSNUM**.

The CS polarity can be set with **SPICSSTR.CSL**, 0b for active low, 1b for active high.

In SPI engine can automatically assert and deassert SPICSx with each data transaction with **SPICTL.MTRANS** = 0b. To allow multiple data words within a SPICSx assertion set **SPICTL.MTRANS** to 1b. The first data word will assert SPICSx. SPICSx will remain low until it is deasserted with writing **SPICTL.MTARM** to 1b.

Timings for CS behavior can be programmed with granularity of SPICLK. The period between assertion of SPICSx and SPICLK transition can be set with **SPICSSTR.CSSETUP**. The period between last SPICLK transition and deassertion of SPICSx can be set with **SPICSSTR.CSHOLD**. The period between deassertion and assertion of SPICSx can be set with **SPICSSTR.CSWAIT**. The period between SPICLK transitions between multi word packages can be set with **SPICSSTR.CKWAIT**.

**Figure 20-5. SPICSx**



### 20.2.10. Auto Retransmit Data Word

Upon detection of an undercount **SPICSTAT.UCLK** error, the default behavior of the SPI module is to reset the SERDES bit counters and the transmit side holding buffers, assuming that software must re-load the word that did not complete transmission because of the UCLK error.

Note that the receive side holding buffers are *\*not\** reset, and contain the data word received from the last *\*good\** transfer.

An optional mode is provided by setting the RTRANS bit in the **SPICTL.RTRANS**. When set, the reset of the transmit side holding buffers because of UCLK error is disabled, and the word that did not complete transmission remains queued for transmit.

### 20.2.11. Loop Back Mode

The SPI engine has a loop back mode for test purposes, enabled with **SPICTL.LPBK**. The loop back mode is only available in master mode. In loop back mode, SPIMOSI and SPIMISO are connected together internally

and SPICLK, SPIMOSI, SPCSx can be still observed on pins.

#### **20.2.12. SPI Interrupt**

The SPI engine interrupt is enabled with **SPICTL.SIE**. Then any sub interrupts are enabled in **SPIINT\_EN**.

#### **20.2.13. SPI Enable**

The SPI engine is enabled with **SPICTL.SSEN**. To soft reset the state machine and clear all status bits use **SPICFG.MRST**.

## 21. MULTI-MODE POWER MANAGER

### 21.1. Register

#### 21.1.1. Register Map

Table 21-1. Multi-Mode Power Manager Register Map

SOC ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>Multi-Mode Power Manager</b>			
0x00	<b>SOC.SYSSTAT</b>	System Status	0x00
0x01 – 0x0F	<b>Reserved</b>	Reserved	N/A
0x10	<b>SOC.PWRCTL</b>	Power Manager Control	0x00
0x11	<b>SOC.PWRSTAT</b>	Power Manager Status	0x00
0x12	<b>SOC.PWRSTATCFG</b>	Power Manager Status Configuration	0x00
0x13	<b>Reserved</b>	Reserved	0xFF
0x14	<b>SOC.SCFG</b>	Power Manager DC-DC Switcher configuration	0x00
0x15	<b>SOC.PWRCFG</b>	Power Manager Configuration	N/A
0x16 – 0x1F	<b>Reserved</b>	Reserved	N/A

#### 21.1.2. SOC.SYSSTAT

Register 21-1. SOC.SYSSTAT (Analog System Status, SOC 0x00)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7	<b>PBINT</b>	R/W	0x0	Push button status. If <b>PWRSTATCFG.PBEN</b> = 1b and <b>SYSSTAT.PBINTEN</b> = 1b, show push button status of AIO6. 1b: AIO6 toggled, write 1b to clear. 0b: no AIO6 push button event detected or disabled or masked
6	<b>OTWINT</b>	R/W	0x0	Over temperature warning to indicate the temperature is over 140°C. When <b>SYSSTAT.OTWINTEN</b> = 1b, this interrupt is enabled 1b: Over temperature > 140°C detected. Write 1b to clear 0b: No over temperature detected.
5:4	<b>VTHREF</b>	R/W	0x0	Programmable threshold voltage. Used for AMP6 through AMP9 in comparator mode. 11b: 1.25V 10b: 0.5V 01b: 0.2V 00b: 0.1V
3	<b>Reserved</b>	R/W	0x0	Reserved, write to 0b
2	<b>PBINTEN</b>	R/W	0x0	Push button interrupt enable. 1b: enable <b>SYSSTAT.PBINT</b> 0b: mask <b>SYSSTAT.PBINT</b>
1	<b>OTSD</b>	R/W	0x0	Over temperature shutdown. 1b: if over temperature shutdown reached, do not shutdown device 0b: if over temperature shutdown reached, shutdown device
0	<b>OTWINTEN</b>	R/W	0x0	Over temperature Warning Interrupt enable 1b: <b>SYSSTAT.OTWINT</b> enabled 0b: <b>SYSSTAT.OTWINT</b> disabled

### 21.1.3. SOC.PWRCTL

#### Register 21-2. SOC.PWRCTL (Power Control, SOC 0x10)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7	HIB	R/W	0x0	Hibernate Mode. 1b: Hibernate mode. Bit is cleared after wake from Wake Up timer or push button event or power up event 0b: Normal mode  <b>NOTE:</b> Setting SOC.PWRCTL.HIB =1b without <b>SOC.PWRSTATCFG.PBEN</b> or <b>SOC.PWRCTL.WUTIMER</b> enabled will not allow device exit hibernate mode. Only way to exit is power cycle device.
6	MCUALIVE	R/W	0x0	Indicator for MCU that power manager is alive and released all pull down resistors on internal databus. 1b: Power manager alive 0b: Power manager in hibernate mode
5:3	PWRMUX	R/W	0x0	Power rail monitoring MUX for AB11 111b: VCOMP 110b: VHM * 1/30 101b: VP * 1/10 100b: REGI * 1/10 011b: VSYS * 4/10 010b: VCCIO * 4/10 001b: VCC33 * 4/10 000b: VCC18
2:0	WUTIMER	R/W	0x0	Wake up from hibernate mode timer setting 111b: 8s 110b: 4s 101b: 2s 100b: 1s 011b: 500ms 010b: 250ms 001b: 125ms 000b: disabled

### 21.1.4. SOC.PWRSTAT

#### Register 21-3. SOC.PWRSTAT (Power Manager Status, Persistent In Hibernate mode, SOC 0x11)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7	HWRESET	R/W	0x0	HW reset status 1b: HW reset occurred, clear by writing to 1b 0b: Normal
6	WDTRESET	R/W	0x0	Watchdog timer status 1b: Watchdog timer reset occurred, clear by writing to 1b 0b: Normal
5	DCDC	R/W	0x0	DC-DC fault 1b: DC-DC fault, clear by writing to 1b 0b: no fault
4	TMP	R/W	0x0	Over Temperature fault 1b: Over temperature fault, clear by writing to 1b 0b: no fault
3	VSYS	R/W	0x0	VSYS under voltage fault 1b: VSYS under voltage fault, clear by writing to 1b 0b: no fault  <b>Note:</b> VSYS will be 1b at initial power up. Check with <b>SOC.PWRSTATCFG.SU</b> to determine fail or power up condition.

BIT	NAME	ACCESS	RESET	DESCRIPTION
2	VCCIO	R/W	0x0	VCCIO under voltage fault 1b: VCCIO under voltage fault, clear by writing to 1b 0b: no fault  <b>Note:</b> VCCIO will be 1b at initial power up. Check with <b>SOC.PWRSTATCFG.SU</b> to determine fail or power up condition.
1	VCC33	R/W	0x0	VCC33 under voltage fault 1b: VCC33 under voltage fault, clear by writing to 1b 0b: no fault  <b>Note:</b> VCC33 will be 1b at initial power up. Check with <b>SOC.PWRSTATCFG.SU</b> to determine fail or power up condition.
0	VCC18	R/W	0x0	VCC18 under voltage fault 1b: VCC18 under voltage fault, clear by writing to 1b 0b: no fault  <b>Note:</b> VCC18 will be 1b at initial power up. Check with <b>SOC.PWRSTATCFG.SU</b> to determine fail or power up condition.

### 21.1.5. SOC.PWRSTATCFG

#### Register 21-4. SOC.PWRSTATCFG (Power Status Configuration, SOC 0x12)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7	PWRUNLOCK	R/W	0x0	Unlock for <b>SOC.SCFG</b> and <b>SOC.PWRCFG</b> 1b: Unlock, enable write access to <b>SOC.SCFG</b> and <b>SOC.PWRCFG</b> 0b: no write access to <b>SOC.SCFG</b> and <b>SOC.PWRCFG</b>
6	SU	R/W	0x0	Start up status, bit is cleared at power up, write to 1b to indicate start up from power up is recognized. As long as VHM has power this bit condition is retained. 1b: write to 1b to store power up condition 0b: power up detected
5	VPUVINT	R/W	0x0	vp UV Interrupt 1b: Interrupt, clear by writing to 1b 0b: no interrupt
4	Reserved	R/W	0x0	Reserved, write to 0b
3	VPUVINTEN	R/W	0x0	VP under voltage interrupt enable 1b: enabled 0b: disabled
2	WDTCFG	R/W	0x0	Watchdog timer setting 1b: 8s 0b: 4s
1	WDTEN	R/W	0x0	Watchdog timer enable 1b: enabled 0b: disabled
0	PBEN	R/W	0x0	AIO6 Push button enable 1b: Push button enabled 0b: Push button disabled

### 21.1.6. SOC.SCFG

#### Register 21-5. SOC.SCFG (Power Switcher Configuration, SOC 0x14)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7	<b>SRESET</b>	R/W	0x0	Soft reset 1b: Write to 1b to assert nRST to reset CPU 0b: normal
6	<b>Reserved</b>	R/W	0x0	Reserved, write to 0b
5	<b>VHMCLAMP</b>	R/W	0x0	VHM voltage clamp 1b: Clamp disabled 0b: Clamp VHM to 20V
4	<b>FMODE</b>	R/W	0x0	DC-DC Switching frequency range for <b>SOC.SCFG.FSWM</b> 1b: 181kHz to 500kHz 0b: 45kHz to 125kHz
3:1	<b>FSWM</b>	R/W	0x0	DC-DC frequency configuration 111b: <b>FMODE</b> = 0b: 125kHz, <b>FMODE</b> = 1b: 500kHz 110b: <b>FMODE</b> = 0b: 100kHz, <b>FMODE</b> = 1b: 400kHz 101b: <b>FMODE</b> = 0b: 82.5kHz, <b>FMODE</b> = 1b: 330kHz 100b: <b>FMODE</b> = 0b: 72.25kHz, <b>FMODE</b> = 1b: 289kHz 011b: <b>FMODE</b> = 0b: 62.5kHz, <b>FMODE</b> = 1b: 250kHz 010b: <b>FMODE</b> = 0b: 55kHz, <b>FMODE</b> = 1b: 220kHz 001b: <b>FMODE</b> = 0b: 50kHz, <b>FMODE</b> = 1b: 200kHz 000b: <b>FMODE</b> = 0b: 45kHz, <b>FMODE</b> = 1b: 181kHz
0	<b>MAXDUTY</b>	R/W	0x0	Maximum DC-DC duty 1b: Min off time 820ns 0b: <b>FMODE</b> = 0b: Min. off time 25% duty, <b>FMODE</b> = 1b: Min off time 440ns

### 21.1.7. SOC.PWRCFG

#### Register 21-6. SOC.PWRCFG (Power Status Configuration, SOC 0x15)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	<b>VP</b>	R/W	Varies <sup>1</sup>	VP voltage setting 11b: 15V 10b: 12V 01b: 9V 00b: 5V
5	<b>SMPSOFF</b>	R/W	0x0	DC-DC disable 1b: disabled 0b: enabled
4	<b>VSYSILIM</b>	R/W	0x0	VSYS current limit 1b: 450mA 0b: 300mA
3	<b>TRESET</b>	R/W	0x0	nRESET low time 1b: 32ms 0b: 1ms
2	<b>TON</b>	R/W	0x0	Pre-startup DC-DC on time 1b: 1.2μs 0b: 0.8μs
1	<b>VCC18</b>	R/W	0x0	VCC18 Voltage 1b: 1.8V 0b: 1.9V
0	<b>Reserved</b>	R/W	0x0	Reserved, write to 0b

<sup>1</sup> See the data sheet for the reset value for this parameter



## 22. CONFIGURABLE ANALOG FRONT END

### 22.1. Register

#### 22.1.1. Register Map

**Table 22-1. Configurable Analog Front End Register Map**

SOC ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>Configurable Analog Front End</b>			
0x20	<b>SOC.AIO0CFG</b>	AIO0 Configuration	0x00
0x21	<b>SOC.AIO1CFG</b>	AIO1 Configuration	0x00
0x22	<b>SOC.AIO2CFG</b>	AIO2 Configuration	0x00
0x23	<b>SOC.AIO3CFG</b>	AIO3 Configuration	0x00
0x24	<b>SOC.AIO4CFG</b>	AIO4 Configuration	0x00
0x25	<b>SOC.AIO5CFG</b>	AIO5 Configuration	0x00
0x26	<b>SOC.AIO6CFG</b>	AIO6 Configuration	0x00
0x27	<b>SOC.AIO7CFG</b>	AIO7 Configuration	0x00
0x28	<b>SOC.AIO8CFG</b>	AIO8 Configuration	0x00
0x29	<b>SOC.AIO9CFG</b>	AIO9 Configuration	0x00
0x2A	<b>SOC.SMCFG</b>	Signal manager Configuration	0x00
0x2B	<b>SOC.HPDAC</b>	High Protection Threshold	0x00
0x2C	<b>SOC.LPDAC0</b>	Low Protection Threshold	0x00
0x2D	<b>SOC.LPDAC1</b>	Low Protection Threshold	0x00
0x2E	<b>SOC.ADCSCANCFG</b>	ADC Scan Control	0x00
0x2F	<b>SOC.ADMUX</b>	ADC MUX Select	0x00
0x30	<b>SOC.PROTINTEN</b>	Protection Interrupt Enable	0x00
0x31	<b>SOC.PROTINT</b>	Protection Status	0x00
0x32	<b>SOC.AIOOUT0</b>	AIO0, AIO1, AIO2, AIO3, AIO4, AIO5 digital output control	0x00
0x33	<b>SOC.AIOOUT1</b>	AIO6, AIO7, AIO8, AIO9 digital output control	0x00
0x34	<b>SOC.AIOIN0</b>	AIO0, AIO1, AIO2, AIO3, AIO4, AIO5 digital input status	0x00
0x35	<b>SOC.AIOIN1</b>	AIO6, AIO7, AIO8, AIO9 digital input status	0x00
0x36	<b>SOC.AIOINTEN</b>	AIO6, AIO7, AIO8, AIO9 Interrupt enable	0x00
0x37	<b>SOC.AIOINT</b>	AIO6, AIO7, AIO8, AIO9 Interrupt flag	0x00
0x38	<b>SOC.SMCTL</b>	Signal Manager Control	0x00
0x39 – 0x5F	<b>Reserved</b>	Reserved	N/A

#### 22.1.2. SOC.AIO0CFG

##### Register 22-1. SOC.AIO0CFG (AIO0 Configuration, SOC 0x20)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	<b>MODE10</b>	R/W	0x0	AIO1, AIO0 Mode 11b: reserved 10b: reserved 01b: Differential amplifier mode mode, AIO1 and AIO0 build a differential amplifier input 00b: IO Mode, AIO1 and AIO0 work in I/O mode
5:4	<b>OPT0</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 00b, AIO0 IO Setting 11b: Reserved 10b: Open-drain output 01b: Reserved 00b: Input, input state available at <b>SOC.AIOIN0.AIO0DIN</b>
3	<b>POLO</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 00b and <b>CFGAI00.OPT0</b> = 00b, AIO0 Input polarity setting. If <b>AIO0CFG.MODE10</b> = 00b and <b>CFGAI00.OPT0</b> = 10b, AIO0 Output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX0</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 00b and <b>CFGAI00.OPT0</b> = 00b, Reserved If <b>AIO0CFG.MODE10</b> = 00b and <b>CFGAI00.OPT0</b> = 10b, Output polarity, set AIO0 state according to MUX0 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: DOUT0
5:3	<b>GAIN10</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 01b, Differential amplifier gain setting 111b: 48x 110b: 32x 101b: 16x 100b: 8x 011b: 4x 010b: 2x 001b: 1x 000b: 1x
2	<b>Reserved</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 01b, Reserved, Write to 0b
1:0	<b>LP10EN</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 01b, LP10 comparator setting 11b: LP10 comparator enabled with 4 $\mu$ s blanking time 10b: LP10 comparator enabled with 2 $\mu$ s blanking time 01b: LP10 comparator enabled with 1 $\mu$ s blanking time 00b: LP10 comparator disabled

### 22.1.3. SOC.AIO1CFG

#### Register 22-2. SOC.AIO1CFG (AIO1 Configuration, SOC 0x21)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	<b>Reserved</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 00b, Reserved, write to 00b
5:4	<b>OPT1</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 00b, AIO1 IO Setting 11b: Reserved 10b: Open-drain output 01b: Reserved 00b: Input, input state available at <b>SOC.AIOIN0.AIO1DIN</b>

BIT	NAME	ACCESS	RESET	DESCRIPTION
3	<b>POL1</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 00b and <b>AIO0CFG.OPT0</b> = 00b, AIO1 Input polarity setting. If <b>AIO0CFG.MODE10</b> = 00b and <b>AIO0CFG.OPT0</b> = 10b, AIO1 Output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX1</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 00b and <b>AIO0CFG.OPT0</b> = 00b, Reserved If <b>AIO0CFG.MODE10</b> = 00b and <b>AIO0CFG.OPT0</b> = 10b, set AIO1 state according to MUX1 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: DOUT1
7	<b>HP10PR1EN</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 01b, HPROT10 PR1 Protection enable 1b: HP10 output to PR1 enabled 0b: HP10 output to PR1 disabled
6	<b>HP10PR2EN</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 01b, HPROT10 PR2 Protection enable 1b: HP10 output to PR2 enabled 0b: HP10 output to PR2 disabled
5	<b>LP10PR1EN</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 01b, LPROT10 PR1 Protection enable 1b: LP10 output to PR1 enabled 0b: LP10 output to PR1 disabled
4	<b>LP10PR2EN</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 01b, LPROT10 PR2 Protection enable 1b: LP10 output to PR2 enabled 0b: LP10 output to PR2 disabled
3	<b>OS10EN</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 01b, Differential Amplifier Offset 1b: Offset enabled, input signal shifted by VREF/2 0b: Offset disabled
2	<b>CAL10EN</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 01b, Differential Amplifier Offset Calibration 1b: enabled 0b: disabled
1:0	<b>HP10EN</b>	R/W	0x0	If <b>AIO0CFG.MODE10</b> = 01b, HP10 comparator setting 11b: HP10 comparator enabled with 4μs blanking time 10b: HP10 comparator enabled with 2μs blanking time 01b: HP10 comparator enabled with 1μs blanking time 00b: HP10 comparator disabled

#### 22.1.4. SOC.AIO2CFG

**Register 22-3. SOC.AIO2CFG (AIO2 Configuration, SOC 0x22)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	<b>MODE32</b>	R/W	0x0	AIO3, AIO2 Mode 11b: reserved 10b: reserved 01b: Differential amplifier mode mode, AIO3 and AIO2 build a differential amplifier input 00b: IO Mode, AIO3 and AIO2 work in I/O mode
5:4	<b>OPT2</b>	R/W	0x0	If <b>AIO2CFG.MODE32</b> = 00b, AIO2 IO Setting 11b: Reserved 10b: Open-drain output 01b: Reserved 00b: Input, input state available at <b>SOC.AIOIN0.AIO2DIN</b>

BIT	NAME	ACCESS	RESET	DESCRIPTION
3	<b>POL2</b>	R/W	0x0	If <b>AIO2CFG.MODE32</b> = 00b and <b>CFGAI02.OPT2</b> = 00b, AIO2 Input polarity setting. If <b>AIO2CFG.MODE32</b> = 00b and <b>CFGAI02.OPT2</b> = 10b, AIO2 Output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX2</b>	R/W	0x0	If <b>AIO2CFG.MODE32</b> = 00b and <b>CFGAI02.OPT2</b> = 00b, Reserved If <b>AIO2CFG.MODE32</b> = 00b and <b>CFGAI02.OPT2</b> = 10b, set AIO2 state according to MUX2 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: DOUT2
5:3	<b>GAIN32</b>	R/W	0x0	If <b>AIO2CFG.MODE32</b> = 01b, Differential amplifier gain setting 111b: 48x 110b: 32x 101b: 16x 100b: 8x 011b: 4x 010b: 2x 001b: 1x 000b: 1x
2	<b>Reserved</b>	R/W	0x0	If <b>AIO2CFG.MODE32</b> = 01b, Reserved, Write to 0b
1:0	<b>LP32EN</b>	R/W	0x0	If <b>AIO2CFG.MODE32</b> = 01b, LP32 comparator setting 11b: LP32 comparator enabled with 4μs blanking time 10b: LP32 comparator enabled with 2μs blanking time 01b: LP32 comparator enabled with 1μs blanking time 00b: LP32 comparator disabled

### 22.1.5. SOC.AIO3CFG

#### Register 22-4. SOC.AIO3CFG (AIO3 Configuration, SOC 0x23)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	<b>Reserved</b>	R/W	0x0	If <b>AIO2CFG.MODE32</b> = 00b, Reserved, write to 00b
5:4	<b>OPT3</b>	R/W	0x0	If <b>AIO2CFG.MODE32</b> = 00b, AIO3 IO Setting 11b: Reserved 10b: Open-drain output 01b: Reserved 00b: Input, input state available at <b>SOC.AIOIN0.AIO3DIN</b>
3	<b>POL3</b>	R/W	0x0	If <b>AIO2CFG.MODE32</b> = 00b and <b>AIO2CFG.OPT2</b> = 00b, AIO3 Input polarity setting. If <b>AIO2CFG.MODE32</b> = 00b and <b>AIO2CFG.OPT2</b> = 10b, AIO3 Output polarity setting. 1b: active-low 0b: active-high

BIT	NAME	ACCESS	RESET	DESCRIPTION
2:0	<b>MUX3</b>	R/W	0x0	If <b>AIO2CFG.MODE32 = 00b</b> and <b>AIO2CFG.OPT2 = 00b</b> , Reserved If <b>AIO2CFG.MODE32 = 00b</b> and <b>AIO2CFG.OPT2 = 10b</b> , set AIO3 state according to MUX3 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: DOUT3
7	<b>HP32PR1EN</b>	R/W	0x0	If <b>AIO2CFG.MODE32 = 01b</b> , HPROT32 PR1 Protection enable 1b: HP32 output to PR1 enabled 0b: HP32 output to PR1 disabled
6	<b>HP32PR2EN</b>	R/W	0x0	If <b>AIO2CFG.MODE32 = 01b</b> , HPROT32 PR2 Protection enable 1b: HP32 output to PR2 enabled 0b: HP32 output to PR2 disabled
5	<b>LP32PR1EN</b>	R/W	0x0	If <b>AIO2CFG.MODE32 = 01b</b> , LPROT32 PR1 Protection enable 1b: LP32 output to PR1 enabled 0b: LP32 output to PR1 disabled
4	<b>LP32PR2EN</b>	R/W	0x0	If <b>AIO2CFG.MODE32 = 01b</b> , LPROT32 PR2 Protection enable 1b: LP32 output to PR2 enabled 0b: LP32 output to PR2 disabled
3	<b>OS32EN</b>	R/W	0x0	If <b>AIO2CFG.MODE32 = 01b</b> , Differential Amplifier Offset 1b: Offset enabled, input signal shifted by VREF/2 0b: Offset disabled
2	<b>CAL32EN</b>	R/W	0x0	If <b>AIO2CFG.MODE32 = 01b</b> , Differential Amplifier Offset Calibration 1b: enabled 0b: disabled
1:0	<b>HP32EN</b>	R/W	0x0	If <b>AIO2CFG.MODE32 = 01b</b> , HP32 comparator setting 11b: HP32 comparator enabled with 4 $\mu$ s blanking time 10b: HP32 comparator enabled with 2 $\mu$ s blanking time 01b: HP32 comparator enabled with 1 $\mu$ s blanking time 00b: HP32 comparator disabled

### 22.1.6. SOC.AIO4CFG

#### Register 22-5. SOC.AIO4CFG (AIO4 Configuration, SOC 0x24)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	<b>MODE54</b>	R/W	0x0	AIO5, AIO4 Mode 11b: reserved 10b: reserved 01b: Differential amplifier mode mode, AIO5 and AIO4 build a differential amplifier input 00b: IO Mode, AIO5 and AIO4 work in I/O mode
5:4	<b>OPT4</b>	R/W	0x0	If <b>AIO4CFG.MODE54 = 00b</b> , AIO4 IO Setting 11b: Reserved 10b: Open-drain output 01b: Reserved 00b: Input, input state available at <b>SOC.AIOIN0.AIO4DIN</b>

BIT	NAME	ACCESS	RESET	DESCRIPTION
3	<b>POL4</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = 00b and <b>CFGAI04.OPT4</b> = 00b, AIO4 Input polarity setting. If <b>AIO4CFG.MODE54</b> = 00b and <b>CFGAI04.OPT4</b> = 10b, AIO4 Output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX4</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = 00b and <b>CFGAI04.OPT4</b> = 00b, Reserved If <b>AIO4CFG.MODE54</b> = 00b and <b>CFGAI04.OPT4</b> = 10b, set AIO4 state according to MUX4 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: DOUT4
5:3	<b>GAIN54</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = 01b, Differential amplifier gain setting 111b: 48x 110b: 32x 101b: 16x 100b: 8x 011b: 4x 010b: 2x 001b: 1x 000b: 1x
2	<b>Reserved</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = 01b, Reserved, Write to 0b
1:0	<b>LP54EN</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = 01b, LP54 comparator setting 11b: LP54 comparator enabled with 4μs blanking time 10b: LP54 comparator enabled with 2μs blanking time 01b: LP54 comparator enabled with 1μs blanking time 00b: LP54 comparator disabled

### 22.1.7. SOC.AIO5CFG

**Register 22-6. SOC.AIO5CFG (AIO5 Configuration, SOC 0x25)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	<b>Reserved</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = 00b, Reserved, write to 00b
5:4	<b>OPT5</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = 00b, AIO5 IO Setting 11b: Reserved 10b: Open-drain output 01b: Reserved 00b: Input, input state available at <b>SOC.AIOIN0.AIO5DIN</b>
3	<b>POL5</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = 00b and <b>AIO4CFG.OPT4</b> = 00b, AIO5 Input polarity setting. If <b>AIO4CFG.MODE54</b> = 00b and <b>AIO4CFG.OPT4</b> = 10b, AIO5 Output polarity setting. 1b: active-low 0b: active-high

BIT	NAME	ACCESS	RESET	DESCRIPTION
2:0	<b>MUX5</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = 00b and <b>AIO4CFG.OPT4</b> = 00b, Reserved If <b>AIO4CFG.MODE32</b> = 00b and <b>AIO4CFG.OPT4</b> = 10b, Output polarity, set AIO3 state according to MUX3 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: DOUT5
7	<b>HP54PR1EN</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = <b>01b</b> , HPROT54 PR1 Protection enable 1b: HP54 output to PR1 enabled 0b: HP54 output to PR1 disabled
6	<b>HP54PR2EN</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = <b>01b</b> , HPROT32 PR2 Protection enable 1b: HP32 output to PR2 enabled 0b: HP32 output to PR2 disabled
5	<b>LP54PR1EN</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = <b>01b</b> , LPROT54 PR1 Protection enable 1b: LP32 output to PR1 enabled 0b: LP32 output to PR1 disabled
4	<b>LP54PR2EN</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = <b>01b</b> , LPROT54 PR2 Protection enable 1b: LP32 output to PR2 enabled 0b: LP32 output to PR2 disabled
3	<b>OS54EN</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = <b>01b</b> , Differential Amplifier Offset 1b: Offset enabled, input signal shifted by VREF/2 0b: Offset disabled
2	<b>CAL54EN</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = <b>01b</b> , Differential Amplifier Offset Calibration 1b: enabled 0b: disabled
1:0	<b>HP54EN</b>	R/W	0x0	If <b>AIO4CFG.MODE54</b> = <b>01b</b> , HP54 comparator setting 11b: HP54 comparator enabled with 4 $\mu$ s blanking time 10b: HP54 comparator enabled with 2 $\mu$ s blanking time 01b: HP54 comparator enabled with 1 $\mu$ s blanking time 00b: HP54 comparator disabled

### 22.1.8. SOC.AIO6CFG

#### Register 22-7. SOC.AIO6CFG (AIO6 Configuration, SOC 0x26)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	<b>MODE6</b>	R/W	0x0	AIO6 Mode 11b: Special mode 10b: Comparator mode 01b: Gain Amplifier mode 00b: IO Mode
5:4	<b>OPT6</b>	R/W	0x0	If <b>AIO6CFG.MODE6</b> = 00b, AIO6 IO Setting 11b: Reserved 10b: Open-drain output 01b: Reserved 00b: Input, input state available at <b>SOC.AIOIN1.AIO6DIN</b>
3	<b>POL6</b>	R/W	0x0	If <b>AIO6CFG.MODE6</b> = 00b and <b>CFGAI06.OPT6</b> = 00b, AIO6 Input polarity setting. If <b>AIO6CFG.MODE6</b> = 00b and <b>CFGAI06.OPT6</b> = 10b, AIO6 Output polarity setting. 1b: active-low 0b: active-high

BIT	NAME	ACCESS	RESET	DESCRIPTION
2:0	<b>MUX6</b>	R/W	0x0	If <b>AIO6CFG.MODE6</b> = 00b and <b>CFGAI06.OPT6</b> = 00b, Reserved If <b>AIO6CFG.MODE6</b> = 00b and <b>CFGAI06.OPT6</b> = 10b, Output polarity, set AIO6 state according to MUX6 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: DOUT6
5:3	<b>GAIN6</b>	R/W	0x0	If <b>AIO6CFG.MODE6</b> = 01b, amplifier gain setting 111b: 48x 110b: 32x 101b: 16x 100b: 8x 011b: 4x 010b: 2x 001b: 1x 000b: Gain Amplifier Bypass, Direct mode
2:0	<b>MUX6</b>	R/W	0x0	If <b>AIO6CFG.MODE6</b> = 01b, Analog MUX Setting 111b: AB7 110b: AB6 101b: AB5 100b: AB4 011b: AB3 010b: AB2 001b: AB1 000b: AB6
5:4	<b>OPT6</b>	R/W	0x0	If <b>AIO6CFG.MODE6</b> = 10b, AIO6 Comparator Reference select 11b: AB3 10b: AB2 01b: AB1 00b: VTHREF
3	<b>POL6</b>	R/W	0x0	If <b>AIO6CFG.MODE6</b> = 10b, AIO6 Comparator output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX6</b>	R/W	0x0	If <b>AIO6CFG.MODE6</b> = 10b, send AIO6 comparator output state to internal digital bus and <b>SOC.AIOIN1.AIO6DIN</b> 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: Reserved
5	<b>ADMUX</b>	R/W	0x0	If <b>AIO6CFG.MODE6</b> = 11b 1b: MUX ADMUX output to AB7 0b: do not MUX ADMUX output to AB7
4	<b>SWAP</b>	R/W	0x0	If <b>AIO6CFG.MODE6</b> = 11b 1b: Swap buffer offset 0b: do not swap buffer offset
3	<b>Reserved</b>	R/W	0x0	If <b>AIO6CFG.MODE6</b> = 11b, Reserved, write to 0b



BIT	NAME	ACCESS	RESET	DESCRIPTION
2:0	<b>MUX6</b>	R/W	0x0	If <b>AIO6CFG.MODE6</b> = 11b, send AIO6 comparator output state to internal digital bus and <b>SOC.AIOIN1.AIO6DIN</b> 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: Reserved

### 22.1.9. SOC.AIO7CFG

#### Register 22-8. SOC.AIO7CFG (AIO7 Configuration, SOC 0x27)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	<b>MODE7</b>	R/W	0x0	AIO7 Mode 11b: Special mode 10b: Comparator mode 01b: Gain Amplifier mode 00b: IO Mode
5:4	<b>OPT7</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 00b, AIO7 IO Setting 11b: Reserved 10b: Open-drain output 01b: Reserved 00b: Input, input state available at <b>SOC.AIOIN1.AIO7DIN</b>
3	<b>POL7</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 00b and <b>CFGAI07.OPT7</b> = 00b, AIO7 Input polarity setting. If <b>AIO7CFG.MODE7</b> = 00b and <b>CFGAI07.OPT7</b> = 10b, AIO7 Output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX7</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 00b and <b>CFGAI07.OPT7</b> = 00b, Reserved If <b>AIO7CFG.MODE7</b> = 00b and <b>CFGAI07.OPT7</b> = 10b, Output polarity, set AIO7 state according to MUX7 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: DOUT7
5:3	<b>GAIN7</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 01b, amplifier gain setting 111b: 48x 110b: 32x 101b: 16x 100b: 8x 011b: 4x 010b: 2x 001b: 1x 000b: Gain Amplifier Bypass, Direct mod

BIT	NAME	ACCESS	RESET	DESCRIPTION
2:0	<b>MUX7</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 01b, Analog Mux Setting 111b: AB7 110b: AB6 101b: AB5 100b: AB4 011b: AB3 010b: AB2 001b: AB1 000b: AB7
5:4	<b>OPT7</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 10b, AIO7 Comparator Reference select 11b: AB3 10b: AB2 01b: AB1 00b: VTHREF
3	<b>POL7</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 10b, AIO7 Comparator output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX7</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 10b, send AIO7 comparator output state to internal digital bus and <b>SOC.AIOIN1.AIO7DIN</b> 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: Reserved
5:4	<b>OPT7</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 11b, Select reference for AIO7, AIO8, AIO9 comparator in special mode. 11b: AB3 10b: AB2 01b: AB1 00b: Reserved
3	<b>POL7</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 11b, AIO7 comparator output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX7</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 11b Reserved

### 22.1.10. SOC.AIO8CFG

#### Register 22-9. SOC.AIO8CFG (AIO8 Configuration, SOC 0x28)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	<b>MODE8</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> not 11b, AIO8 Mode 11b: Reserved 10b: Comparator mode 01b: Gain Amplifier mode 00b: IO Mode
5:4	<b>OPT8</b>	R/W	0x0	If <b>AIO8CFG.MODE8</b> = 00b, AIO8 IO Setting 11b: Reserved 10b: Open-drain output 01b: Reserved 00b: Input, input state available at <b>SOC.AIOIN1.AIO8DIN</b>

BIT	NAME	ACCESS	RESET	DESCRIPTION
3	<b>POL8</b>	R/W	0x0	If <b>AIO8CFG.MODE8</b> = 00b and <b>CFGAI08.OPT8</b> = 00b, AIO8 Input polarity setting. If <b>AIO8CFG.MODE8</b> = 00b and <b>CFGAI08.OPT8</b> = 10b, AIO8 Output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX8</b>	R/W	0x0	If <b>AIO8CFG.MODE8</b> = 00b and <b>CFGAI08.OPT8</b> = 00b, Reserved If <b>AIO8CFG.MODE8</b> = 00b and <b>CFGAI08.OPT8</b> = 10b, Output polarity, set AIO8 state according to MUX8 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: DOUT8
5:3	<b>GAIN8</b>	R/W	0x0	If <b>AIO8CFG.MODE8</b> = 01b, amplifier gain setting 111b: 48x 110b: 32x 101b: 16x 100b: 8x 011b: 4x 010b: 2x 001b: 1x 000b: Gain Amplifier Bypass, Direct mod
2:0	<b>MUX8</b>	R/W	0x0	If <b>AIO8CFG.MODE8</b> = 01b, Analog Mux Setting 111b: AB7 110b: AB6 101b: AB5 100b: AB4 011b: AB3 010b: AB2 001b: AB1 000b: AB8
5:4	<b>OPT8</b>	R/W	0x0	If <b>AIO8CFG.MODE8</b> = 10b, AIO8 Comparator Reference select 11b: AB3 10b: AB2 01b: AB1 00b: VTHREF
3	<b>POL8</b>	R/W	0x0	If <b>AIO8CFG.MODE8</b> = 10b, AIO8 Comparator output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX8</b>	R/W	0x0	If <b>AIO8CFG.MODE8</b> = 10b, send AIO8 comparator output state to internal digital bus 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: Reserved
5	<b>OPT8[1]</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 11b 1b: do not bypass S/H for POS signal 0b: bypass S/H for POS signal
4	<b>OPT8[0]</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 11b, 1b: select nIRQ2/POS output to output POS 0b: select nIRQ2/POS output nIRQ2

BIT	NAME	ACCESS	RESET	DESCRIPTION
3	<b>POL8</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 11b, AIO8 comparator output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX8</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 11b Reserved

### 22.1.11. SOC.AIO9CFG

#### Register 22-10. SOC.AIO9CFG (AIO9 Configuration, SOC 0x29)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	<b>MODE9</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> not 11b, AIO9 Mode 11b: Reserved 10b: Comparator mode 01b: Gain Amplifier mode 00b: IO Mode
5:4	<b>OPT9</b>	R/W	0x0	If <b>AIO9CFG.MODE9</b> = 00b, AIO9 IO Setting 11b: Reserved 10b: Open-drain output 01b: Reserved 00b: Input, input state available at <b>SOC.AIOIN1.AIO9DIN</b>
3	<b>POL9</b>	R/W	0x0	If <b>AIO9CFG.MODE9</b> = 00b and <b>CFGAI09.OPT9</b> = 00b, AIO9 Input polarity setting. If <b>AIO9CFG.MODE9</b> = 00b and <b>CFGAI09.OPT9</b> = 10b, AIO9 Output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX9</b>	R/W	0x0	If <b>AIO9CFG.MODE9</b> = 00b and <b>CFGAI09.OPT9</b> = 00b, Reserved If <b>AIO9CFG.MODE9</b> = 00b and <b>CFGAI09.OPT9</b> = 10b, Output polarity, set AIO9 state according to MUX9 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: DOUT9
5:3	<b>GAIN9</b>	R/W	0x0	If <b>AIO9CFG.MODE9</b> = 01b, amplifier gain setting 111b: 48x 110b: 32x 101b: 16x 100b: 8x 011b: 4x 010b: 2x 001b: 1x 000b: Gain Amplifier Bypass, Direct mod
2:0	<b>MUX9</b>	R/W	0x0	If <b>AIO9CFG.MODE9</b> = 01b, Analog Mux Setting 111b: AB7 110b: AB6 101b: AB5 100b: AB4 011b: AB3 010b: AB2 001b: AB1 000b: AB9

BIT	NAME	ACCESS	RESET	DESCRIPTION
5:4	<b>OPT9</b>	R/W	0x0	If <b>AIO9CFG.MODE9</b> = 10b, AIO9 Comparator Reference select 11b: AB3 10b: AB2 01b: AB1 00b: VTHREF
3	<b>POL9</b>	R/W	0x0	If <b>AIO9CFG.MODE9</b> = 10b, AIO9 Comparator output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX9</b>	R/W	0x0	If <b>AIO9CFG.MODE9</b> = 10b and <b>CFGAI09.OPT9</b> = 00b, send AIO9 comparator output state to internal digital bus and input state available at <b>SOC.AIOIN1.AIO9DIN</b> 111b: DB7 110b: DB6 101b: DB5 100b: DB4 011b: DB3 010b: DB2 001b: DB1 000b: Reserved
7	<b>MODE9[1]</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 11b 1b: Connect AIO7 to AB7, AIO8 to AB8, AIO9 to AB9 with 40% attenuation 0b: do not MUX AIO7, AIO8, AIO9 inputs to ABUS
6	<b>MODE9[0]</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 11b 1b: connect AIO7, AIO8, AIO9 to AB1 using matched 100Ω resistors 0b: keep positive comparator input of AIO7, AIO8, AIO9 separate
5:4	<b>OPT9</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 11b 11b: MUX AIO9 comparator output to POS 10b: MUX AIO8 comparator output to POS 01b: MUX AIO7 comparator output to POS 00b: not connected
3	<b>POL9</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 11b and if <b>AIO9CFG.MODE9</b> = 11b, AIO9 comparator output polarity setting. 1b: active-low 0b: active-high
2:0	<b>MUX9</b>	R/W	0x0	If <b>AIO7CFG.MODE7</b> = 11b Reserved

## 22.1.12. SOC.SMCFG

### Register 22-11. SOC.SMCFG (Signal Manager Configuration, SOC 0x2A)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7	<b>AIO9HYS</b>	R/W	0x0	AIO9 Comparator Hysteresis 1b: Comparator Hysteresis enabled 0b: Comparator Hysteresis disabled
6	<b>AIO8HYS</b>	R/W	0x0	AIO8 Comparator Hysteresis 1b: Comparator Hysteresis enabled 0b: Comparator Hysteresis disabled
5	<b>AIO7HYS</b>	R/W	0x0	AIO7 Comparator Hysteresis 1b: Comparator Hysteresis enabled 0b: Comparator Hysteresis disabled
4	<b>AIO6HYS</b>	R/W	0x0	AIO6 Comparator Hysteresis 1b: Comparator Hysteresis enabled 0b: Comparator Hysteresis disabled

BIT	NAME	ACCESS	RESET	DESCRIPTION
3	<b>HPROTHYS</b>	R/W	0x0	HPx Hysteresis 1b: Comparator Hysteresis enabled 0b: Comparator Hysteresis disabled
2	<b>LPROTHYS</b>	R/W	0x0	LPx Hysteresis 1b: Comparator Hysteresis enabled 0b: Comparator Hysteresis disabled
1	<b>LPDACAB3</b>	R/W	0x0	Connect LPDAC output to AB3 1b: LPDAC output connected to AB3 0b: LPDAC output not connected to AB3
0	<b>HPDACAB2</b>	R/W	0x0	Connect HPDAC output to AB2 1b: HPDAC output connected to AB2 0b: HPDAC output not connected to AB2

### 22.1.13. SOC.HPDAC

#### Register 22-12. SOC.HPDAC (HPDAC Setting, SOC 0x2B)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:0	<b>HPDAC</b>	R/W	0x0	HPDAC Setting 0xFF: 255/256*VREF .. 0x00: 0/256*VREF

### 22.1.14. SOC.LPDAC0

#### Register 22-13. SOC.LPDAC0 (LPDAC Setting [9:2], SOC 0x2C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:0	<b>LPDAC[9:2]</b>	R/W	0x0	LPDAC Setting bit 9 to bit 2

### 22.1.15. SOC.LPDAC1

#### Register 22-14. SOC.LPDAC1 (LPDAC Setting [1:0], SOC 0x2C)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:2	<b>Reserved</b>	R/W	0x0	Reserved, write to 0x0
1:0	<b>LPDAC[1:0]</b>	R/W	0x0	LPDAC Setting bit 1 to bit 0

### 22.1.16. SOC.ADCSCANCFG

#### Register 22-15. SOC.ADCSCANCFG (ADCSCAN Configuration, SOC 0x2E)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:5	<b>Reserved</b>	R/W	0x0	Reserved, write to 0x0
4	<b>SCANEN</b>	R/W	0x0	ADC Scan Control Enable 1b: enabled 0b: disabled
3	<b>ADCBUFEN</b>	R/W	0x0	ADC Buffer Enable 1b: enabled 0b: disabled

BIT	NAME	ACCESS	RESET	DESCRIPTION
2	DAO10SH	R/W	0x0	DAO10 Sample and Hold buffer enable 1b: enable S/H 0b: disable and bypass S/H
1	DAO32SH	R/W	0x0	DAO32 Sample and Hold buffer enable 1b: enable S/H 0b: disable and bypass S/H
0	DAO54SH	R/W	0x0	DAO54 Sample and Hold buffer enable 1b: enable S/H 0b: disable and bypass S/H

### 22.1.17. SOC.ADMUX

#### Register 22-16. SOC.ADMUX (AD Mux Selector, SOC 0x2F)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:4	Reserved	R/W	0x0	Reserved, write to 0x0
3:0	MUXA	R/W	0x0	AD Mux Channel Selector 1111b: VREF / 2 1110b: AB12 1101b: AB11 1100b: AB10 1011b: AB9 1010b: AB8 1001b: AB7 1000b: AB6 0111b: AB5 0110b: AB4 0101b: AB3 0100b: AB2 0011b: AB1 0010b: DAO54 0001b: DAO32 0000b: DAO10

### 22.1.18. SOC.PROTINTEN

#### Register 22-17. SOC.PROTINTEN (Protection Interrupt Enable, SOC 0x30)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7	Reserved	R/W	0x0	Reserved, write to 0x0
6	HP54INTEN	R/W	0x0	HPROT54 Interrupt enable 1b: enable 0b: disabled
5	HP32INTEN	R/W	0x0	HPROT32 Interrupt enable 1b: enable 0b: disabled
4	HP10INTEN	R/W	0x0	HPROT10 Interrupt enable 1b: enable 0b: disabled
3	Reserved	R/W	0x0	Reserved, write to 0x0
2	LP54INTEN	R/W	0x0	LPROT54 Interrupt enable 1b: enable 0b: disabled

BIT	NAME	ACCESS	RESET	DESCRIPTION
1	LP32INTEN	R/W	0x0	LPROT32 Interrupt enable 1b: enable 0b: disabled
0	LP10INTEN	R/W	0x0	LPROT10 Interrupt enable 1b: enable 0b: disabled

### 22.1.19. SOC.PROTINT

#### Register 22-18. SOC.PROTINT (Protection Interrupt, SOC 0x31)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7	Reserved	R/W	0x0	Reserved, write to 0x0
6	HP54INT	R/W	0x0	HPROT54 Interrupt 1b: Interrupt, write 1 to clear 0b: No interrupt
5	HP32INT	R/W	0x0	HPROT32 Interrupt 1b: Interrupt, write 1 to clear 0b: No interrupt
4	HP10INT	R/W	0x0	HPROT10 Interrupt 1b: Interrupt, write 1 to clear 0b: No interrupt
3	Reserved	R/W	0x0	Reserved, write to 0x0
2	LP54INT	R/W	0x0	LPROT54 Interrupt 1b: Interrupt, write 1 to clear 0b: No interrupt
1	LP32INT	R/W	0x0	LPROT32 Interrupt 1b: Interrupt, write 1 to clear 0b: No interrupt
0	LP10INT	R/W	0x0	LPROT10 Interrupt 1b: Interrupt, write 1 to clear 0b: No interrupt

### 22.1.20. SOC.AIOOUT0

#### Register 22-19. SOC.AIOOUT0 (AIO Digital Output, SOC 0x32)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	Reserved	R/W	0x0	Reserved, write to 0x0
5	AIO5DOUT	R/W	0x0	AIO5 digital out 1b: Output High Z 0b: Output VSSA
4	AIO4DOUT	R/W	0x0	AIO4 digital out 1b: Output High Z 0b: Output VSSA
3	AIO3DOUT	R/W	0x0	AIO3 digital out 1b: Output High Z 0b: Output VSSA
2	AIO2DOUT	R/W	0x0	AIO2 digital out 1b: Output High Z 0b: Output VSSA



BIT	NAME	ACCESS	RESET	DESCRIPTION
1	AIO1DOUT	R/W	0x0	AIO1 digital out 1b: Output High Z 0b: Output VSSA
0	AIO0DOUT	R/W	0x0	AIO0 digital out 1b: Output High Z 0b: Output VSSA

### 22.1.21. SOC.AIOOUT1

#### Register 22-20. SOC.AIOOUT0 (AIO Digital Output, SOC 0x33)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:4	Reserved	R/W	0x0	Reserved, write to 0x0
3	AIO9DOUT	R/W	0x0	AIO9 digital out 1b: Output High Z 0b: Output VSSA
2	AIO8DOUT	R/W	0x0	AIO8 digital out 1b: Output High Z 0b: Output VSSA
1	AIO7DOUT	R/W	0x0	AIO7 digital out 1b: Output High Z 0b: Output VSSA
0	AIO6DOUT	R/W	0x0	AIO6 digital out 1b: Output High Z 0b: Output VSSA

### 22.1.22. SOC.AIOIN0

#### Register 22-21. SOC.AIOIN0 (AIO Digital Input, SOC 0x34)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	Reserved	R/W	0x0	Reserved, write to 0x0
5	AIO5DIN	R	0x0	AIO5 digital input 1b: Input high 0b: Input low
4	AIO4DIN	R	0x0	AIO4 digital input 1b: Input high 0b: Input low
3	AIO3DIN	R	0x0	AIO3 digital input 1b: Input high 0b: Input low
2	AIO2DIN	R	0x0	AIO2 digital input 1b: Input high 0b: Input low
1	AIO1DIN	R	0x0	AIO1 digital input 1b: Input high 0b: Input low
0	AIO0DIN	R	0x0	AIO0 digital input 1b: Input high 0b: Input low

### 22.1.23. SOC.AIOIN1

#### Register 22-22. SOC.AIOIN1 (AIO Digital Input 1, SOC 0x35)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:4	<b>Reserved</b>	R/W	0x0	Reserved, write to 0x0
3	<b>AIO9DIN</b>	R	0x0	AIO9 digital input 1b: Input high 0b: Input low
2	<b>AIO8DIN</b>	R	0x0	AIO8 digital input 1b: Input high 0b: Input low
1	<b>AIO7DIN</b>	R	0x0	AIO7 digital input 1b: Input high 0b: Input low
0	<b>AIO6DIN</b>	R	0x0	AIO6 digital input 1b: Input high 0b: Input low

### 22.1.24. SOC.AIOINTEN

#### Register 22-23. SOC.AIOINTEN (AIO6,7,8,9 Interrupt Enable, SOC 0x36)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7	<b>AIO9REINTEN</b>	R/W	0x0	AIO9 digital input rising edge interrupt enable 1b: enabled 0b: disabled
6	<b>AIO8REINTEN</b>	R/W	0x0	AIO8 digital input rising edge interrupt enable 1b: enabled 0b: disabled
5	<b>AIO7REINTEN</b>	R/W	0x0	AIO7 digital input rising edge interrupt enable 1b: enabled 0b: disabled
4	<b>AIO6REINTEN</b>	R/W	0x0	AIO6 digital input rising edge interrupt enable 1b: enabled 0b: disabled
3	<b>AIO9FEINTEN</b>	R/W	0x0	AIO9 digital input falling edge interrupt enable 1b: enabled 0b: disabled
2	<b>AIO8FEINTEN</b>	R/W	0x0	AIO8 digital input falling edge interrupt enable 1b: enabled 0b: disabled
1	<b>AIO7FEINTEN</b>	R/W	0x0	AIO7 digital input falling edge interrupt enable 1b: enabled 0b: disabled
0	<b>AIO6FEINTEN</b>	R/W	0x0	AIO6 digital input falling edge interrupt enable 1b: enabled 0b: disabled

### 22.1.25. SOC.AIOINT

#### Register 22-24. SOC.AIOINT (AIO6,7,8,9 Interrupt, SOC 0x37)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:4	<b>Reserved</b>	R/W	0x0	Reserved, write to 0x0
3	<b>AIO9INT</b>	R/W	0x0	AIO9 Interrupt 1b: Interrupt, clear by writing 1b 0b: No Interrupt
2	<b>AIO8INT</b>	R/W	0x0	AIO8 Interrupt 1b: Interrupt, clear by writing 1b 0b: No Interrupt
1	<b>AIO7INT</b>	R/W	0x0	AIO7 Interrupt 1b: Interrupt, clear by writing 1b 0b: No Interrupt
0	<b>AIO6INT</b>	R/W	0x0	AIO6 Interrupt 1b: Interrupt, clear by writing 1b 0b: No Interrupt

### 22.1.26. SOC.SMCTL

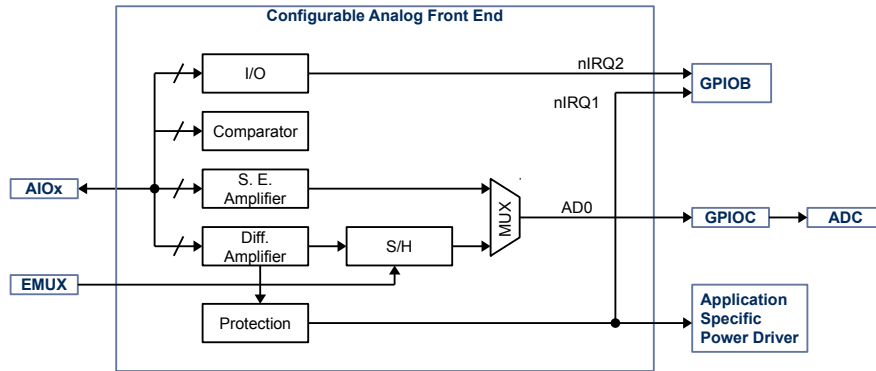
#### Register 22-25. SOC.SMCTL (Signal Manager Control SOC 0x38)

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:1	<b>Reserved</b>	R/W	0x0	Reserved, write to 0x0
0	<b>SMEN</b>	R/W	0x0	Signal Manager Enable 1b: Enabled 0b: Disabled

## 22.2. Details of Operation

### 22.2.1. Block Diagram

Figure 22-1. Configurable Analog Front End



### 22.2.2. Configuration

Following blocks need to be configured for correct use of the Configurable analog frontend.

- Clock Control System (CCS)
- Nested Vectored Interrupt Controller (NVIC)
- IO Controller
- SOC Bridge

### 22.2.3. Configurable Analog Front End

The configurable analog front end registers are accessible through the SOC Bridge. AIOx can be configured as digital open drain I/Os or analog inputs.

Programmable comparators and protection circuits can be connected to the analog signals.

The analog signals can be conditioned with programmable gain single ended amplifiers or programmable gain differential amplifiers before muxed to the ADC.

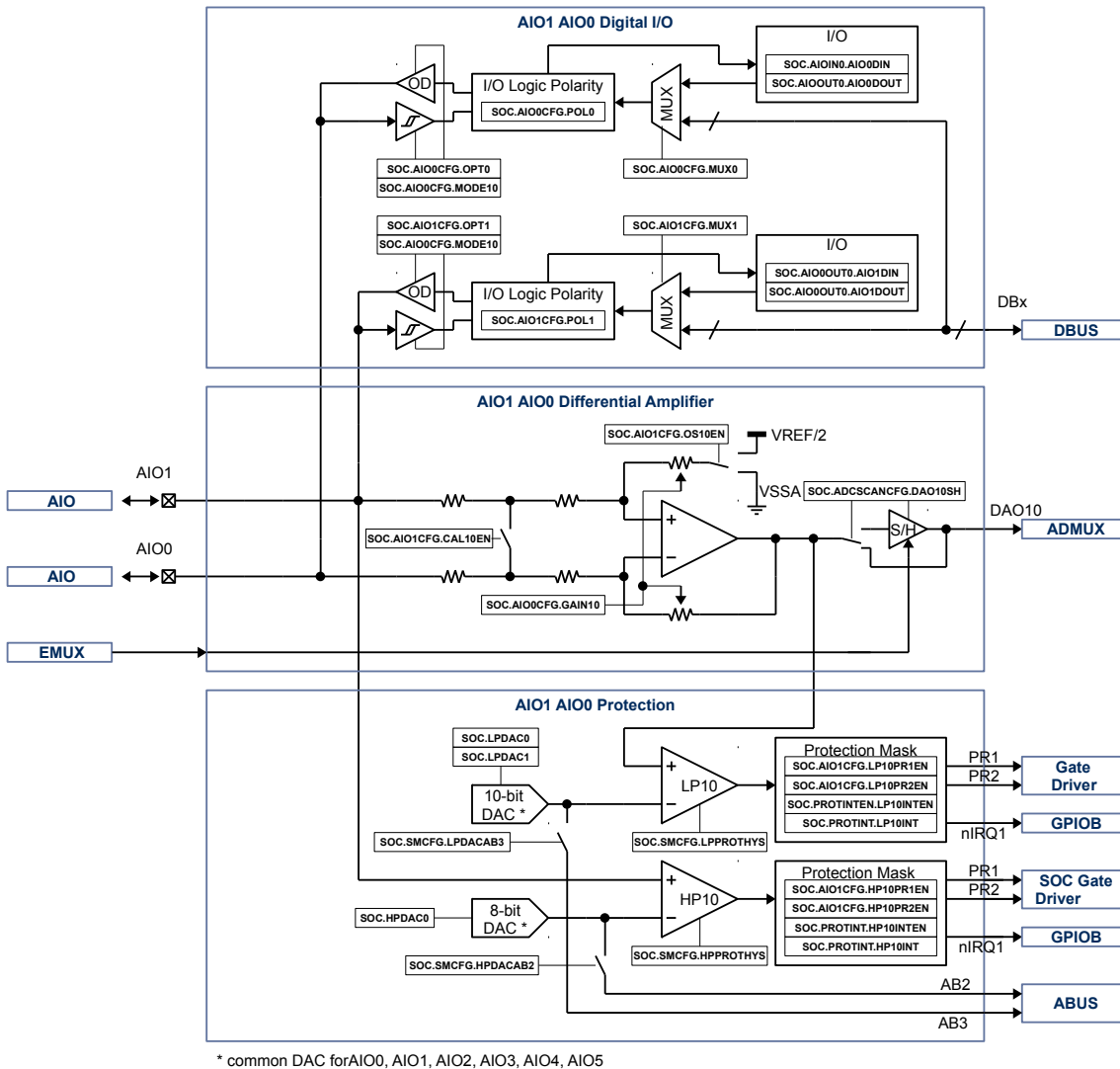
#### 22.2.3.1. Configurable Analog Front End Enable

Use **SOC.SMCTL.SMEN** to enable the configurable analog front end.

## 22.3. AIO1, AIO0

### 22.3.1. Block Diagram

Figure 22-2. AIO1, AIO0



### 22.3.2. AIO1, AIO0

AIO1 and AIO0 can be configured as digital inputs or as differential amplifier pair with additional protection.

### 22.3.3. AIO1, AIO0 digital I/O Mode

Set **SOC.AIO0CFG.MODE10** = 00b to use AIO1 and AIO0 as digital inputs.

#### 22.3.3.1. AIO0 IO

Set **SOC.AIO0CFG.OPT0** = 00b to use AIO0 as input. The input state can be read at **SOC.AIOIN0.AIO0DIN**.

Set **SOC.AIO0CFG.OPT0** = 10b to use AIO0 as open drain output. Set **SOC.AIO0CFG.MUX0** = 00b to mux the output state from **SOC.AIOOUT0.AIO0DOUT**. Use **SOC.AIO0CFG.MUX0** to mux the output signal from the internal digital bus DBUS DB1 to DB7.

#### 22.3.3.2. AIO1 IO

Set **SOC.AIO1CFG.OPT1** = 00b to use AIO1 as input. The input state can be read at **SOC.AIOIN0.AIO1DIN**.

Set **SOC.AIO1CFG.OPT1** = 10b to use AIO1 as open drain output. Set **SOC.AIO1CFG.MUX1** = 00b to MUX the output state from **SOC.AIOOUT0.AIO1DOUT**. Use **SOC.AIO1CFG.MUX1** to MUX the output signal from the internal digital bus DBUS DB1 to DB7.

#### 22.3.3.3. AIO0 Polarity

Use **SOC.AIO0CFG.POL0** to set logic polarity of the signal between AIO0 input/output and MUX0.

#### 22.3.3.4. AIO1 Polarity

Use **SOC.AIO0CFG.POL1** to set logic polarity of the signal between AIO1 input/output and MUX1.

### 22.3.4. AIO1, AIO0 differential Amplifier Mode

Set **SOC.AIO0CFG.MODE10** = 01b to use AIO1 and AIO0 as input to a differential amplifier.

#### 22.3.4.1. AIO1, AIO0 Differential Amplifier Gain

Use **SOC.AIO0CFG.GAIN10** to set to gain between 1x to 48x.

#### 22.3.4.2. AIO1, AIO0 Differential Amplifier Reference

Use **SOC.AIO1CFG.OS10EN** to set the amplifier reference either VSSA or VREF/2.

#### 22.3.4.3. AIO1, AIO0 Differential Amplifier Calibration

Use **SOC.AIO1CFG.CAL10EN** to short the input of the differential amplifier to allow reading of the amplifier offset.

### 22.3.5. AIO1, AIO0 Protection

In **SOC.AIO0CFG.MODE10** = 01b differential amplifier mode, a high side comparator protector HP10 and a low

side comparator protector LP10 are also active that can be configured to disabled high-side or low-side drivers in the application specific power driver section.

#### 22.3.5.1. HP10 Comparator

The HP10 comparator takes the AIO1 voltage referenced to VSSA and compares it against the HP-DAC voltage. The 8-bit HP-DAC is programmable with **SOC.HPDAC0**.

Use **SOC.AIO1CFG.HP10EN** to enable HP10 comparator with different blanking times.

Use **SOC.SMCFGHPPROTHYS** to enable HP10 comparator hysteresis.

The output of HP10 comparator can be configured to trigger protection signal PR1 using **SOC.AIO1CFG.HP10PR1EN** or protection signal PR2 using **SOC.AIO1CFG.HP10PR2EN**.

The output of HP10 can also trigger the nIRQ1 interrupt using **SOC.PROTINTEN.HP10INTEN**. The interrupt status can be observed with **SOC.PROTINT.HP10INT**.

#### 22.3.5.2. LP10 Comparator

The LP10 comparator takes the output of the differential amplifier and compares it against the LP-DAC voltage. The 10-bit LP-DAC is programmable with **SOC.LPDAC0** and **SOC.LPDAC1**.

Use **SOC.AIO0CFG.LP10EN** to enable LP10 comparator with different blanking times.

Use **SOC.SMCFGGLPPROTHYS** to enable LP10 comparator hysteresis.

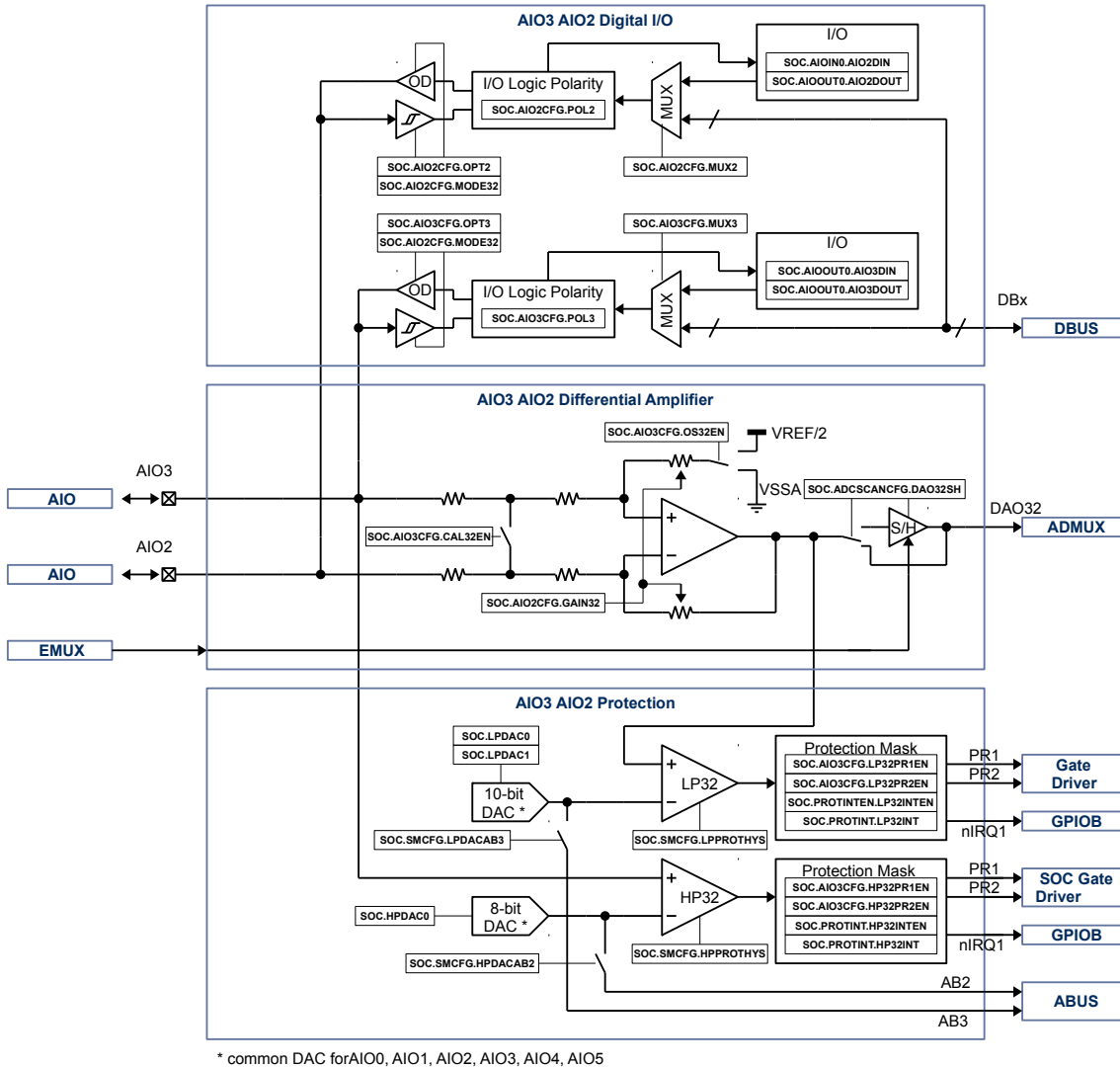
The output of LP10 comparator can be configured to trigger protection signal PR1 using **SOC.AIO1CFG.LP10PR1EN** or protection signal PR2 using **SOC.AIO1CFG.LP10PR2EN**.

The output of LP10 can also trigger the nIRQ1 interrupt using **SOC.PROTINTEN.LP10INTEN**. The interrupt status can be observed with **SOC.PROTINT.LP10INT**.

## 22.4. AIO3, AIO2

### 22.4.1. Block Diagram

Figure 22-3. AIO3, AIO2



### 22.4.2. AIO3, AIO2

AIO3 and AIO2 can be configured as digital inputs or as differential amplifier pair with additional protection.

### 22.4.3. AIO3, AIO2 digital I/O Mode

Set **SOC.AIO2CFG.MODE32** = 00b to use AIO3 and AIO2 as digital inputs.



#### 22.4.3.1. AIO2 IO

Set **SOC.AIO2CFG.OPT2** = 00b to use AIO2 as input. The input state can be read at **SOC.AIOIN0.AIO2DIN**.

Set **SOC.AIO2CFG.OPT2** = 10b to use AIO2 as open drain output. Set **SOC.AIO2CFG.MUX2** = 00b to MUX the output state from **SOC.AIOOUT0.AIO2DOUT**. Use **SOC.AIO2CFG.MUX2** to MUX the output signal from the internal digital bus DBUS DB1 to DB7.

#### 22.4.3.2. AIO3 IO

Set **SOC.AIO3CFG.OPT3** = 00b to use AIO3 as input. The input state can be read at **SOC.AIOIN0.AIO3DIN**.

Set **SOC.AIO3CFG.OPT3** = 10b to use AIO3 as open drain output. Set **SOC.AIO3CFG.MUX3** = 00b to MUX the output state from **SOC.AIOOUT0.AIO3DOUT**. Use **SOC.AIO3CFG.MUX3** to MUX the output signal from the internal digital bus DBUS DB1 to DB7.

#### 22.4.3.3. AIO2 Polarity

Use **SOC.AIO2CFG.POL2** to set logic polarity of the signal between AIO2 input/output and MUX2.

#### 22.4.3.4. AIO3 Polarity

Use **SOC.AIO3CFG.POL3** to set logic polarity of the signal between AIO3 input/output and MUX3.

### 22.4.4. AIO3, AIO2 differential Amplifier Mode DAO32

Set **SOC.AIO2CFG.MODE32** = 01b to use AIO3 and AIO2 as input to a differential amplifier DAO32.

#### 22.4.4.1. DAO32 Differential Amplifier Gain

Use **SOC.AIO2CFG.GAIN32** to set to gain between 1x to 48x.

#### 22.4.4.2. DAO32 Differential Amplifier Reference

Use **SOC.AIO3CFG.OS32EN** to set the amplifier reference either VSSA or VREF/2.

#### 22.4.4.3. DAO32 Differential Amplifier Calibration

Use **SOC.AIO3CFG.CAL32EN** to short the input of the differential amplifier to allow reading of the amplifier offset.

### 22.4.5. AIO3, AIO2 Protection

In **SOC.AIO2CFG.MODE32** = 01b differential amplifier mode, a high side comparator protector HP10 and a low

side comparator protector LP10 are also active that can be configured to disabled high-side or low-side drivers in the application specific power driver section.

#### 22.4.5.1. HP32 Comparator

The HP32 comparator takes the AIO3 voltage referenced to VSSA and compares it against the HP-DAC voltage. The 8-bit HP-DAC is programmable with **SOC.HPDAC0**.

Use **SOC.AIO3CFG.HP32EN** to enable HP32 comparator with different blanking times.

Use **SOC.SMCFGHPPROTHYS** to enable HP32 comparator hysteresis.

The output of HP32 comparator can be configured to trigger protection signal PR1 using **SOC.AIO3CFG.HP32PR1EN** or protection signal PR2 using **SOC.AIO3CFG.HP32PR2EN**.

The output of HP32 can also trigger the nIRQ1 interrupt using **SOC.PROTINTEN.HP32INTEN**. The interrupt status can be observed with **SOC.PROTINT.HP32INT**.

#### 22.4.5.2. LP32 Comparator

The LP32 comparator takes the output of the differential amplifier and compares it against the LP-DAC voltage. The 10-bit LP-DAC is programmable with **SOC.LPDAC0** and **SOC.LPDAC1**.

Use **SOC.AIO2CFG.LP32EN** to enable LP32 comparator with different blanking times.

Use **SOC.SMCFGGLPPROTHYS** to enable LP32 comparator hysteresis.

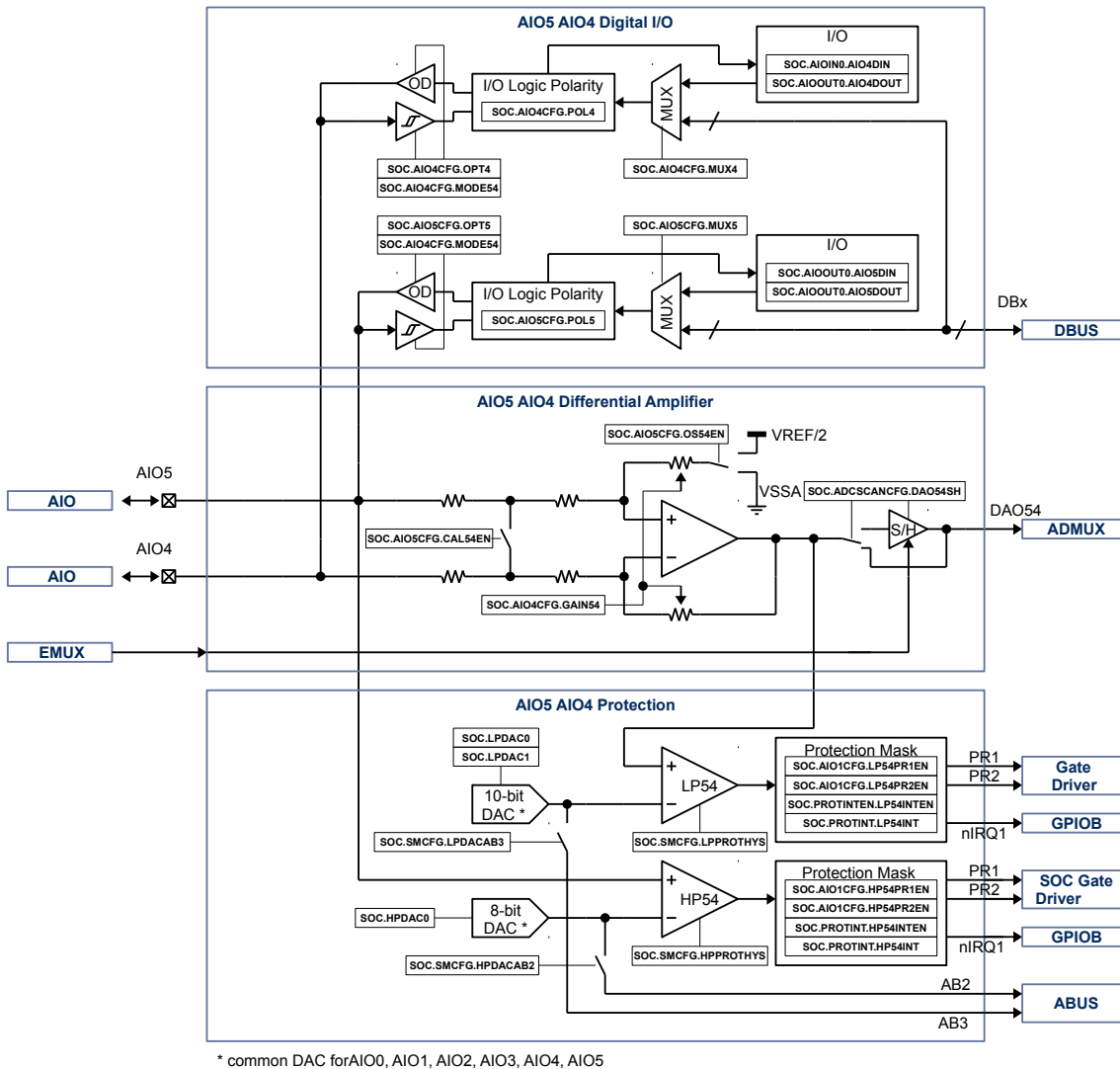
The output of LP32 comparator can be configured to trigger protection signal PR1 using **SOC.AIO3CFG.LP32PR1EN** or protection signal PR2 using **SOC.AIO3CFG.LP32PR2EN**.

The output of LP32 can also trigger the nIRQ1 interrupt using **SOC.PROTINTEN.LP32INTEN**. The interrupt status can be observed with **SOC.PROTINT.LP32INT**.

## 22.5. AIO5, AIO4

### 22.5.1. Block Diagram

Figure 22-4. AIO5, AIO4



### 22.5.2. AIO5, AIO4

AIO5 and AIO4 can be configured as digital inputs or as differential amplifier pair with additional protection.

### 22.5.3. AIO5, AIO4 digital I/O Mode

Set **SOC.AIO4CFG.MODE54** = 00b to use AIO5 and AIO4 as digital inputs.

#### 22.5.3.1. AIO4 IO

Set **SOC.AIO4CFG.OPT4** = 00b to use AIO4 as input. The input state can be read at **SOC.AIOIN0.AIO4DIN**.

Set **SOC.AIO4CFG.OPT4** = 10b to use AIO4 as open drain output. Set **SOC.AIO4CFG.MUX4** = 00b to MUX the output state from **SOC.AIOOUT0.AIO4DOUT**. Use **SOC.AIO4CFG.MUX4** to MUX the output signal from the internal digital bus DBUS DB1 to DB7.

#### 22.5.3.2. AIO5 IO

Set **SOC.AIO5CFG.OPT5** = 00b to use AIO5 as input. The input state can be read at **SOC.AIOIN0.AIO5DIN**.

Set **SOC.AIO5CFG.OPT5** = 10b to use AIO5 as open drain output. Set **SOC.AIO5CFG.MUX5** = 00b to MUX the output state from **SOC.AIOOUT0.AIO5DOUT**. Use **SOC.AIO5CFG.MUX5** to MUX the output signal from the internal digital bus DBUS DB1 to DB7.

#### 22.5.3.3. AIO4 Polarity

Use **SOC.AIO4CFG.POL4** to set logic polarity of the signal between AIO4 input/output and MUX4.

#### 22.5.3.4. AIO5 Polarity

Use **SOC.AIO5CFG.POL5** to set logic polarity of the signal between AIO5 input/output and MUX5.

### 22.5.4. AIO5, AIO4 differential Amplifier Mode DAO54

Set **SOC.AIO4CFG.MODE54** = 01b to use AIO5 and AIO4 as input to a differential amplifier DAO54.

#### 22.5.4.1. DAO54 Differential Amplifier Gain

Use **SOC.AIO4CFG.GAIN54** to set to gain between 1x to 48x.

#### 22.5.4.2. DAO54 Differential Amplifier Reference

Use **SOC.AIO5CFG.OS54EN** to set the amplifier reference either VSSA or VREF/2.

#### 22.5.4.3. DAO54 Differential Amplifier Calibration

Use **SOC.AIO5CFG.CAL54EN** to short the input of the differential amplifier to allow reading of the amplifier offset.

### 22.5.5. AIO5, AIO4 Protection

In **SOC.AIO4CFG.MODE54** = 01b differential amplifier mode, a high side comparator protector HP54 and a low

side comparator protector LP54 are also active that can be configured to disabled high-side or low-side drivers in the application specific power driver section.

#### 22.5.5.1. HP54 Comparator

The HP54 comparator takes the AIO5 voltage referenced to VSSA and compares it against the HP-DAC voltage. The 8-bit HP-DAC is programmable with **SOC.HPDAC0**.

Use **SOC.AIO5CFG.HP54EN** to enable HP54 comparator with different blanking times.

Use **SOC.SMCFGHPPROTHYS** to enable HP54 comparator hysteresis.

The output of HP54 comparator can be configured to trigger protection signal PR1 using **SOC.AIO5CFG.HP54PR1EN** or protection signal PR2 using **SOC.AIO5CFG.HP54PR2EN**.

The output of HP54 can also trigger the nIRQ1 interrupt using **SOC.PROTINTEN.HP54INTEN**. The interrupt status can be observed with **SOC.PROTINT.HP54INT**.

#### 22.5.5.2. LP54 Comparator

The LP54 comparator takes the output of the differential amplifier and compares it against the LP-DAC voltage. The 10-bit LP-DAC is programmable with **SOC.LPDAC0** and **SOC.LPDAC1**.

Use **SOC.AIO4CFG.LP54EN** to enable LP54 comparator with different blanking times.

Use **SOC.SMCFGGLPPROTHYS** to enable LP54 comparator hysteresis.

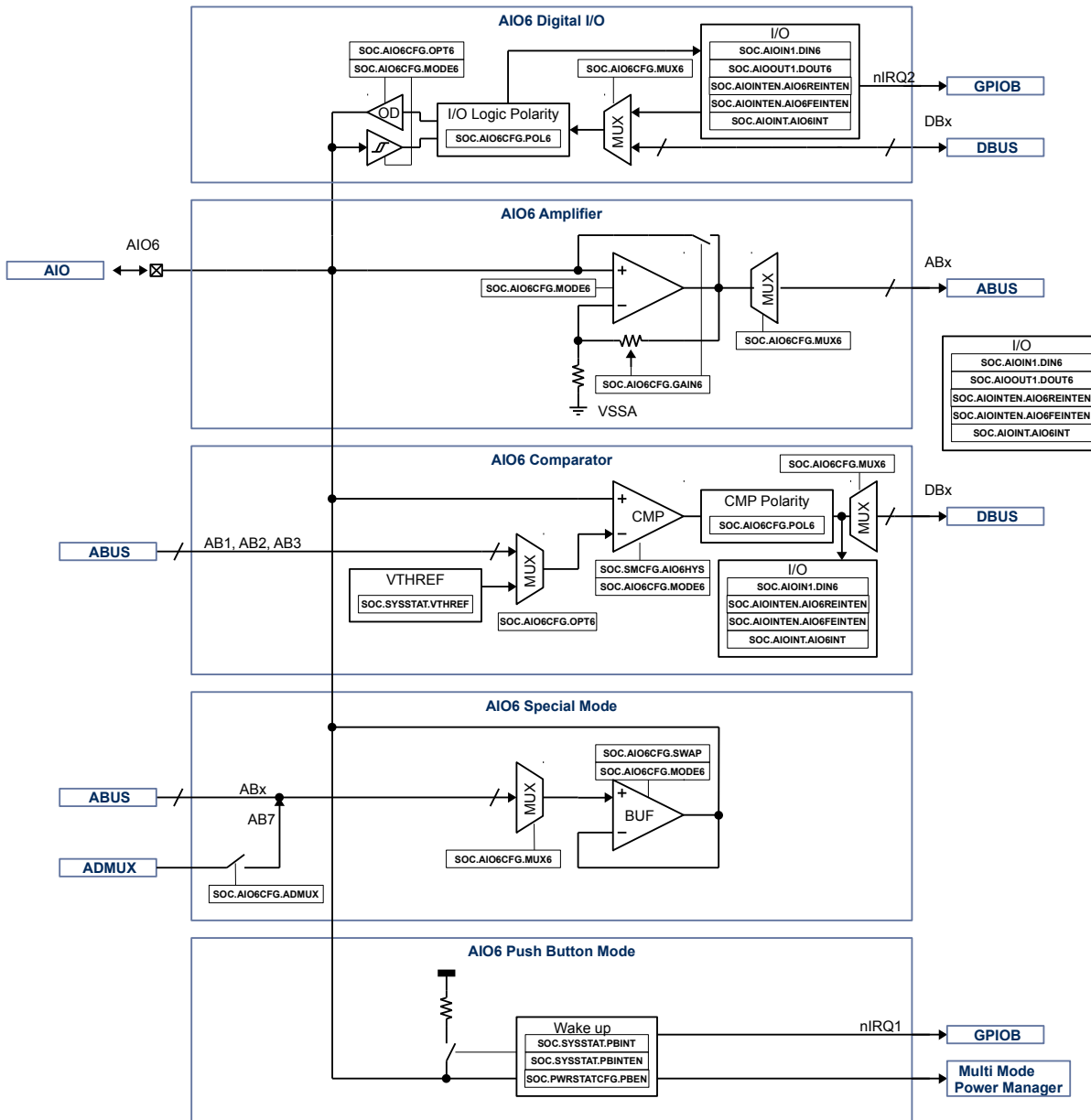
The output of LP54 comparator can be configured to trigger protection signal PR1 using **SOC.AIO5CFG.LP54PR1EN** or protection signal PR2 using **SOC.AIO5CFG.LP54PR2EN**.

The output of LP54 can also trigger the nIRQ1 interrupt using **SOC.PROTINTEN.LP54INTEN**. The interrupt status can be observed with **SOC.PROTINT.LP54INT**.

## 22.6. AIO6

### 22.6.1. Block Diagram

Figure 22-5. AIO6



### 22.6.2. AIO6

AIO6 can be configured as digital input, as singled ended programmable gain amplifier, as comparator, as output from the analog ABUS or as push button input to wake up the device from low power hibernate mode.

### 22.6.3. AIO6 digital I/O Mode

Set **SOC.PWRSTAT.PBEN** = 0b and **SOC.AIO6CFG.MODE6** = 00b to use AIO6 as digital IO.

#### 22.6.3.1. AIO6 IO

Set **SOC.AIO6CFG.OPT6** = 00b to use AIO6 as input. The input state can be read at **SOC.AIOIN1.AIO6DIN**.

Set **SOC.AIO6CFG.OPT6** = 10b to use AIO6 as open drain output. Set **SOC.AIO6CFG.MUX6** = 00b to MUX the output state from **SOC.AIOOUT1.AIO6DOUT**. Use **SOC.AIO6CFG.MUX6** to MUX the output signal from the internal digital bus DBUS DB1 to DB7.

#### 22.6.3.2. AIO6 IO Interrupt

Set **SOC.AIOINTEN.AIO6REINTEN** to allow nIRQ2 interrupt on AIO6 low high transition.

Set **SOC.AIOINTEN.AIO6FEINTEN** to allow nIRQ2 interrupt on AIO6 high low transition.

The interrupt status can be monitored with **SOC.AIOINT.AIO6INT** and cleared by writing **SOC.AIOINT.AIO6INT** to 1b.

#### 22.6.3.3. AIO6 Polarity

Use **SOC.AIO6CFG.POL6** to set logic polarity of the signal between AIO6 input/output and MUX6.

### 22.6.4. AIO6 Single Ended Amplifier Mode

Set **SOC.PWRSTAT.PBEN** = 0b and **SOC.AIO6CFG.MODE6** = 01b to use AIO6 as input to a programmable gain amplifier.

#### 22.6.4.1. AIO6 Amplifier Gain

Use **SOC.AIO6CFG.GAIN6** to set to gain between 1x to 48x or bidirectional amplifier bypassmode.

#### 22.6.4.2. AIO6 Analog MUX

Use **SOC.AIO6CFG.MUX6** to switch the output of the amplifier to analog channel AB1 to AB7 on the analog bus ABUS.

### 22.6.5. AIO6 Comparator Mode

Set **SOC.PWRSTAT.PBEN** = 0b and **SOC.AIO6CFG.MODE6** = 10b to use AIO6 in comparator mode.

#### 22.6.5.1. AIO6 Comparator Hysteresis

Use **SOC.SMCFG.AIO6HYS** to enable AIO6 comparator hysteresis.

#### 22.6.5.2. AIO6 Comparator setting

Use **SOC.AIO6CFG.OPT6** to set the compare value of the comparator to AB1, AB2, AB3 or VTHREF, settable with **SOC.SYSSTAT.VTHREF**.

#### 22.6.5.3. AIO6 Comparator Polarity

Use **SOC.AIO6CFG.POL6** to set the output polarity of the comparator.

#### 22.6.5.4. AIO6 Comparator Output MUX

Use **SOC.AIO6CFG.MUX6** to set the output of the comparator to the digital bus DB1 to DB7 or **SOC.AIOIN1.AIO6DIN**.

### 22.6.6. AIO6 Special Mode

Set **SOC.PWRSTAT.PBEN** = 0b and **SOC.AIO6CFG.MODE6** = 11b to use AIO6 in special mode. In special mode the AIO6 can output a buffered signal from the internal ABUS, AB1 to AB7.

#### 22.6.6.1. AIO6 Special Mode MUX

Use **SOC.AIO6CFG.MUX6** to set the ABx channel output to AIO6.

#### 22.6.6.2. AIO6 Special Mode ADMUX

Use **SOC.AIO6CFG.ADMUX** set the MUX the ADMUX output to AB7.

#### 22.6.6.3. AIO6 Special Mode OFFSET SWAP

Use **SOC.AIO6CFG.SWAP** to swap the random offset of the buffer for calibration reasons.

### 22.6.7. AIO6 Push Button Mode

Set **SOC.PWRSTAT.PBEN** = 1b to enable AIO6 Hibernate push button mode, where AIO6 has an internal weak pull up also active in hibernate mode. Set **SOC.PWRSTAT.PBINTEN** = 1b to enable nIRQ1 interrupt. Use **SOC.PWRSTAT.PBINT** to monitor interrupt status and write 1b to clear interrupt.



#### *22.6.7.1. AIO6 Push Button Wake Up*

In Hibernate Wake Mode and enabled push button mode, if AIO6 is pulled low for the de-bouncing time period, the **SOC.PWRCTL.HIB** is cleared and the device powers up.

#### *22.6.7.2. AIO6 Push Button Power Down*

In normal mode the **SOC.PWRSTAT.PBINT** is set when AIO6 is pulled low for the de-bouncing time period. The system then can be put into hibernate mode by setting **SOC.PWRCTL.HIB** = 1b.

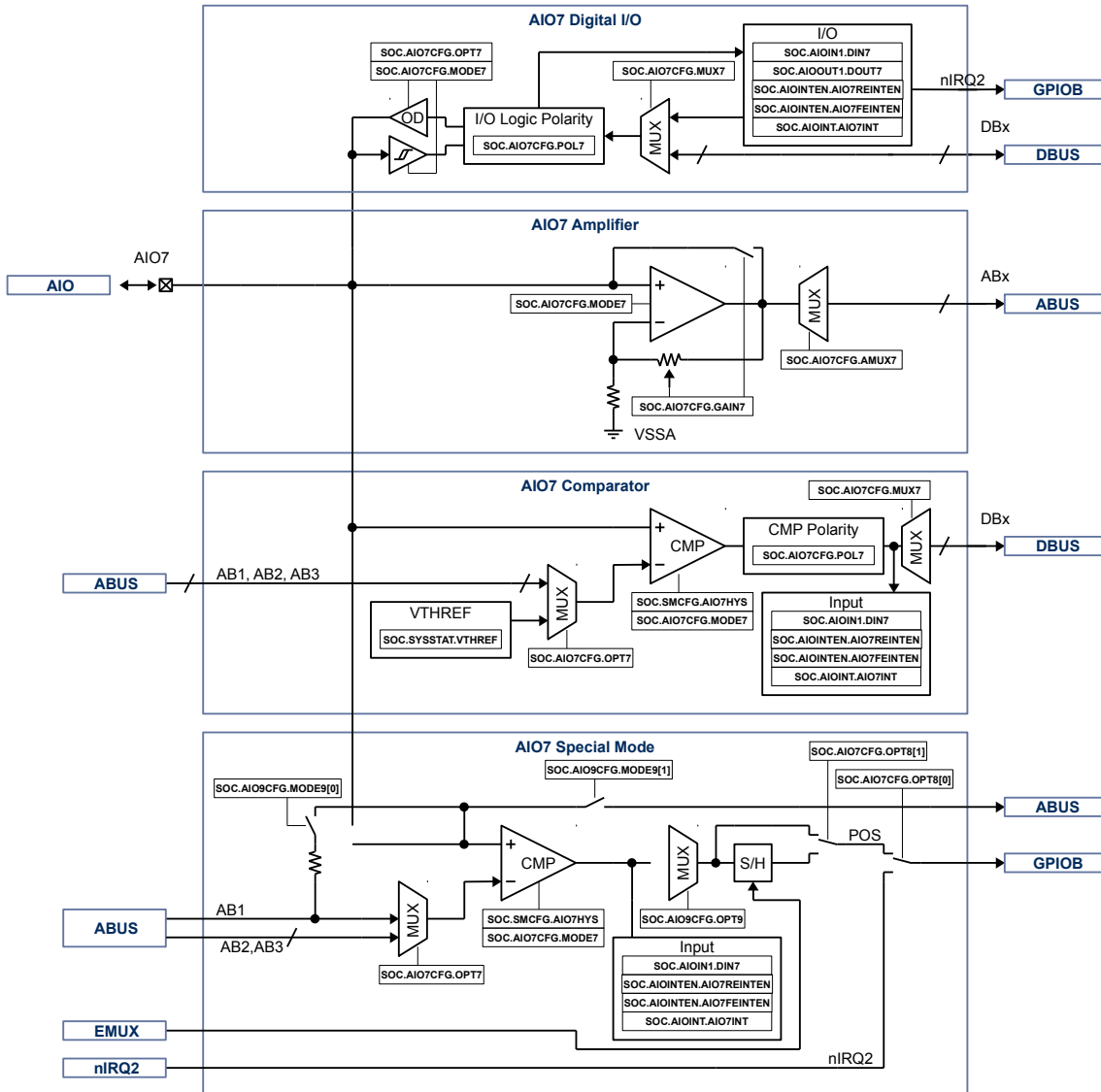
#### *22.6.7.3. AIO6 Push Button Hard Reset*

In normal operation if the AIO6 is pulled low for more than 8s, the nRST signal will be asserted and the MCU is reset. **SOC.PWRSTAT.HWRESET** is set to indicate this condition.

## 22.7. AIO7

### 22.7.1. Block Diagram

Figure 22-6. AIO7



### 22.7.2. AIO7

AIO7 can be configured as digital input, as singled ended programmable gain amplifier, as comparator, as output from the analog ABUS or as push button input to wake up the device from low power hibernate mode.

### 22.7.3. AIO7 digital I/O Mode

Set **SOC.AIO7CFG.MODE7** = 00b to use AIO7 as digital IO.

### 22.7.3.1. AIO7 IO

Set **SOC.AIO7CFG.OPT7** = 00b to use AIO7 as input. The input state can be read at **SOC.AIOIN1.AIO7DIN**.

Set **SOC.AIO7CFG.OPT7** = 10b to use AIO7 as open drain output. Set **SOC.AIO7CFG.MUX7** = 00b to MUX the output state from **SOC.AIOOUT1.AIO7DOUT**. Use **SOC.AIO7CFG.MUX7** to MUX the output signal from the internal digital bus DBUS DB1 to DB7.

### 22.7.3.2. AIO7 IO Interrupt

Set **SOC.AIOINTEN.AIO7REINTEN** to allow nIRQ2 interrupt on AIO7 low high transition.

Set **SOC.AIOINTEN.AIO7FEINTEN** to allow nIRQ2 interrupt on AIO7 high low transition.

The interrupt status can be monitored with **SOC.AIOINT.AIO7INT** and cleared by writing **SOC.AIOINT.AIO7INT** to 1b.

### 22.7.3.3. AIO7 Polarity

Use **SOC.AIO7CFG.POL7** to set logic polarity of the signal between AIO7 input/output and MUX7.

## 22.7.4. AIO7 Single Ended Amplifier Mode

Set **SOC.AIO7CFG.MODE7** = 01b to use AIO7 as input to a programmable gain amplifier.

### 22.7.4.1. AIO7 Amplifier Gain

Use **SOC.AIO7CFG.GAIN7** to set to gain between 1x to 48x or bidirectional amplifier bypassmode.

### 22.7.4.2. AIO7 Analog MUX

Use **SOC.AIO7CFG.MUX7** to switch the output of the amplifier to analog channel AB1 to AB7 on the analog bus ABUS.

## 22.7.5. AIO7 Comparator Mode

Set **SOC.AIO7CFG.MODE7** = 10b to use AIO7 in comparator mode.

### 22.7.5.1. AIO7 Comparator Hysteresis

Use **SOC.SMCFG.AIO7HYS** to enable AIO7 comparator hysteresis.

#### 22.7.5.2. AIO7 Comparator Reference

Use **SOC.AIO7CFG.OPT7** to set the compare value of the comparator to AB1, AB2, AB3 or VTHREF, settable with **SOC.SYSSTAT.VTHREF**.

#### 22.7.5.3. AIO7 Comparator Polarity

Use **SOC.AIO7CFG.POL7** to set the output polarity of the comparator.

#### 22.7.5.4. AIO7 Comparator Output

The comparator output can be observed with **SOC.AIOIN1.AIO7DIN**.

### 22.7.6. AIO7 Special Mode

Set **SOC.AIO7CFG.MODE7** = 11b to use AIO7 in special mode. In special mode the AIO7 comparator is enabled.

#### 22.7.6.1. AIO7 Comparator Hysteresis

Use **SOC.SMCFG.AIO7HYS** to enable AIO7 comparator hysteresis.

#### 22.7.6.2. AIO7 Comparator Reference

Use **SOC.AIO7CFG.OPT7** to set the compare value of the comparator to AB1, AB2 or AB3.

#### 22.7.6.3. AIO7 Comparator Reference Star Point

Use **SOC.AIO9CFG.MODE9[0]** = 1b to connect AIO7, AIO8 and AIO9 to AB1 with a 100kOhm resistor to create a star point reference for the comparator.

With **SOC.AIO9CFG.MODE9[0]** = 0b, the AB1 reference could for example come from AIO6 in amplifier mode **SOC.AIO6CFG.MODE6** = 01b and **SOC.AIO6CFG.GAIN6** = 000b direct mode and **SOC.AIO6CFG.MUX6** = 1b.

#### 22.7.6.4. AIO7 Voltage Reading

Use **SOC.AIO9CFG.MODE9[1]** = 1b to MUX AIO7 to AB7 with 40% attenuation, so ADC can read out AIO7 voltage.

#### 22.7.6.5. AIO7 Comparator Output

Use **SOC.AIO9CFG.OPT9** to select AIO7, AIO8 or AIO9 comparator output signal for POS.

#### 22.7.6.6. AIO7 POS S/H Bypass

Set **SOC.AIO8CFG.OPT8[1]** =0b bypass the POS S/H.

Set **SOC.AIO8CFG.OPT8[1]** =1b use POS S/H for use with EMUX.

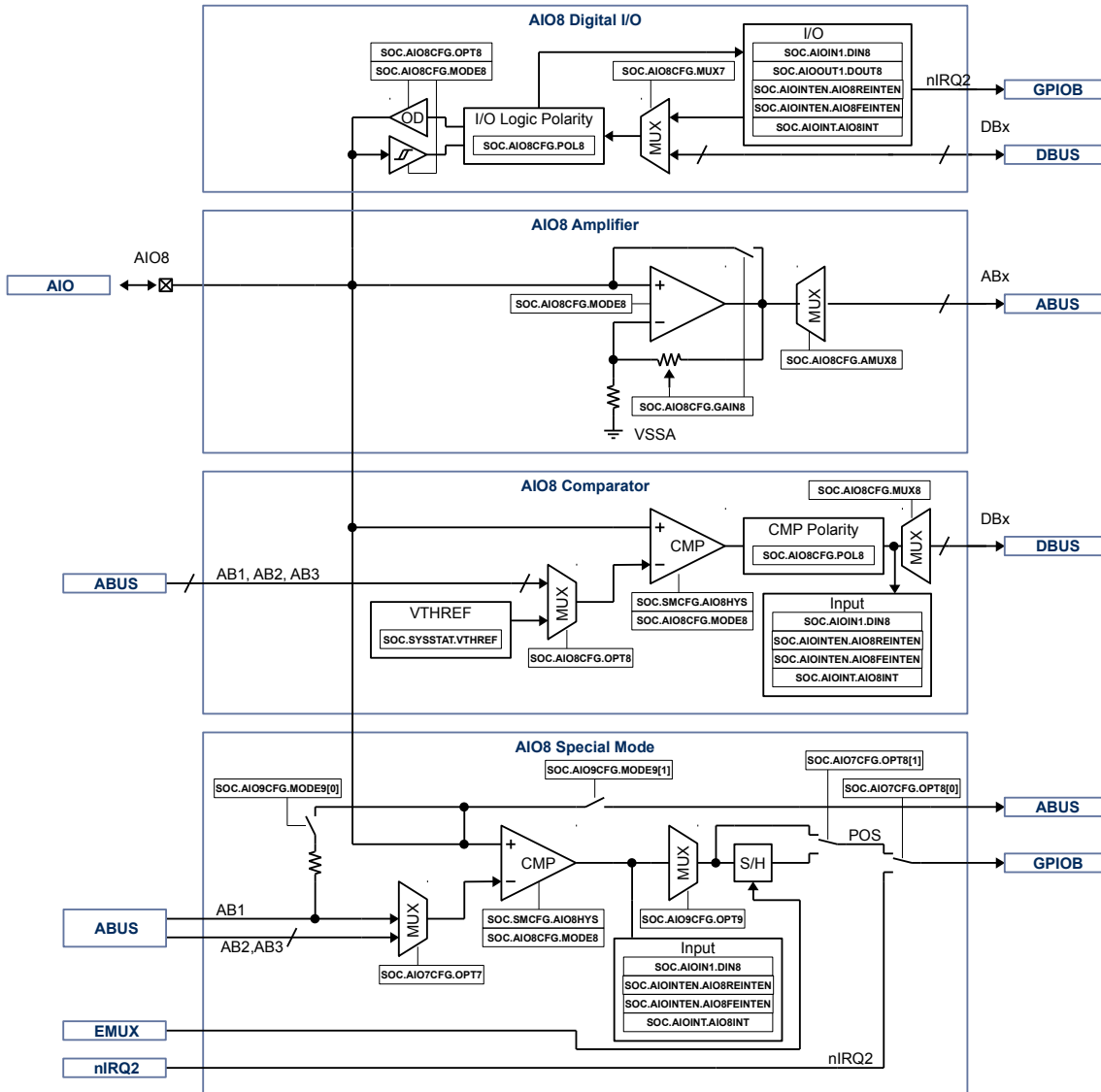
#### 22.7.6.7. AIO7 nIRQ2/POS Selector

Use **SOC.AIO8CFG.OPT8[0]** to select POS or nIRQ2 output.

## 22.8. AIO8

### 22.8.1. Block Diagram

Figure 22-7. AIO8



### 22.8.2. AIO8

AIO8 can be configured as digital input, as single ended programmable gain amplifier, as comparator, as output from the analog ABUS or as push button input to wake up the device from low power hibernate mode.

### 22.8.3. AIO8 digital I/O Mode

Set **SOC.AIO8CFG.MODE8** = 00b to use AIO8 as digital IO.

#### 22.8.3.1. AIO8 IO

Set **SOC.AIO8CFG.OPT8** = 00b to use AIO8 as input. The input state can be read at **SOC.AIOIN1.AIO8DIN**.

Set **SOC.AIO8CFG.OPT8** = 10b to use AIO8 as open drain output. Set **SOC.AIO8CFG.MUX8** = 00b to MUX the output state from **SOC.AIOOUT1.AIO8DOUT**. Use **SOC.AIO8CFG.MUX8** to MUX the output signal from the internal digital bus DBUS DB1 to DB7.

#### 22.8.3.2. AIO8 IO Interrupt

Set **SOC.AIOINTEN.AIO8REINTEN** to allow nIRQ2 interrupt on AIO8 low high transition.

Set **SOC.AIOINTEN.AIO8FEINTEN** to allow nIRQ2 interrupt on AIO8 high low transition.

The interrupt status can be monitored with **SOC.AIOINT.AIO8INT** and cleared by writing **SOC.AIOINT.AIO8INT** to 1b.

#### 22.8.3.3. AIO8 Polarity

Use **SOC.AIO8CFG.POL8** to set logic polarity of the signal between AIO8 input/output and MUX8.

### 22.8.4. AIO8 Single Ended Amplifier Mode

Set **SOC.AIO8CFG.MODE8** = 01b to use AIO8 as input to a programmable gain amplifier.

#### 22.8.4.1. AIO8 Amplifier Gain

Use **SOC.AIO8CFG.GAIN8** to set to gain between 1x to 48x or bidirectional amplifier bypassmode.

#### 22.8.4.2. AIO8 Analog MUX

Use **SOC.AIO8CFG.MUX8** to switch the output of the amplifier to analog channel AB1 to AB7 on the analog bus ABUS.

### 22.8.5. AIO8 Comparator Mode

Set **SOC.AIO8CFG.MODE8** = 10b to use AIO8 in comparator mode.

#### 22.8.5.1. AIO8 Comparator Hysteresis

Use **SOC.SMCFG.AIO8HYS** to enable AIO8 comparator hysteresis.

#### 22.8.5.2. AIO8 Comparator Reference

Use **SOC.AIO8CFG.OPT8** to set the compare value of the comparator to AB1, AB2, AB3 or VTHREF, settable with **SOC.SYSSTAT.VTHREF**.

#### 22.8.5.3. AIO8 Comparator Polarity

Use **SOC.AIO8CFG.POL8** to set the output polarity of the comparator.

#### 22.8.5.4. AIO8 Comparator Output

The comparator output can be observed with **SOC.AIOIN1.AIO8DIN**.

### 22.8.6. AIO8 Special Mode

Set **SOC.AIO7CFG.MODE7** = 11b to use AIO8 in special mode. In special mode the AIO8 comparator is enabled.

#### 22.8.6.1. AIO8 Comparator Hysteresis

Use **SOC.SMCFG.AIO8HYS** to enable AIO8 comparator hysteresis.

#### 22.8.6.2. AIO8 Comparator Reference

Use **SOC.AIO7CFG.OPT7** to set the compare value of the comparator to AB1, AB2 or AB3.

#### 22.8.6.3. AIO8 Comparator Reference Star Point

Use **SOC.AIO9CFG.MODE9[0]** = 1b to connect AIO7, AIO8 and AIO9 to AB1 with a 100kOhm resistor to create a star point reference for the comparator.

With **SOC.AIO9CFG.MODE9[0]** = 0b, the AB1 reference could for example come from AIO6 in amplifier mode **SOC.AIO6CFG.MODE6** = 01b and **SOC.AIO6CFG.GAIN6** = 000b direct mode and **SOC.AIO6CFG.MUX6** = 1b.

#### 22.8.6.4. AIO8 Voltage Reading

Use **SOC.AIO9CFG.MODE9[1]** = 1b to MUX AIO8 to AB8 with 40% attenuation, so ADC can read out AIO8 voltage.



#### 22.8.6.5. AIO8 Comparator Output

Use **SOC.AIO9CFG.OPT9** to select AIO7, AIO8 or AIO9 comparator output signal for POS.

#### 22.8.6.6. AIO8 POS S/H Bypass

Set **SOC.AIO8CFG.OPT8[1]** =0b bypass the POS S/H.

Set **SOC.AIO8CFG.OPT8[1]** =1b use POS S/H for use with EMUX.

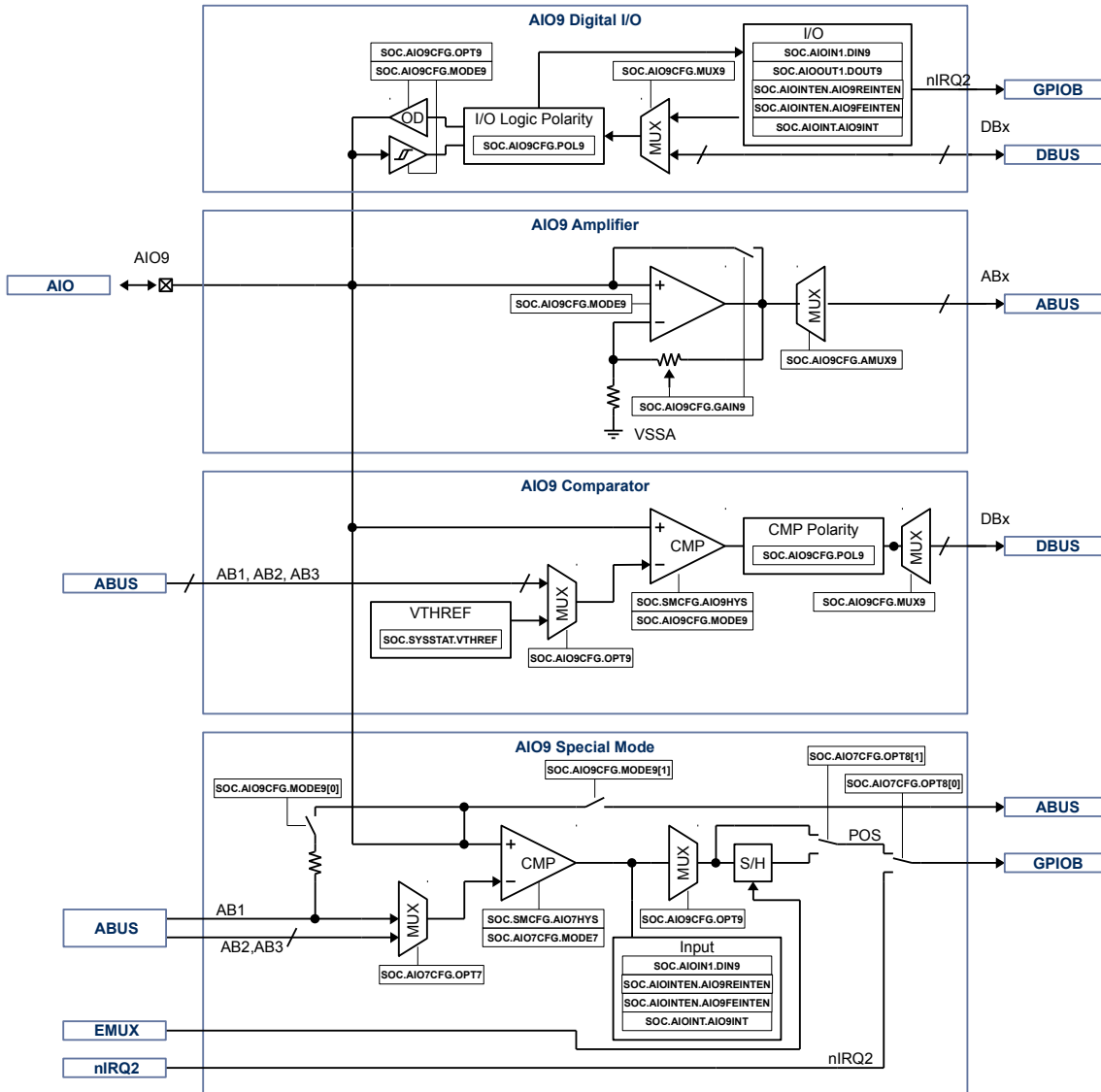
#### 22.8.6.7. AIO8 nIRQ2/POS Selector

Use **SOC.AIO8CFG.OPT8[0]** to select POS or nIRQ2 output.

## 22.9. AIO9

### 22.9.1. Block Diagram

Figure 22-8. AIO9



### 22.9.2. AIO9

AIO9 can be configured as digital input, as single ended programmable gain amplifier, as comparator, as output from the analog ABUS or as push button input to wake up the device from low power hibernate mode.

### 22.9.3. AIO9 digital I/O Mode

Set **SOC.AIO9CFG.MODE9** = 00b to use AIO9 as digital IO.

#### 22.9.3.1. AIO9 IO

Set **SOC.AIO9CFG.OPT9** = 00b to use AIO9 as input. The input state can be read at **SOC.AIOIN1.AIO9DIN**.

Set **SOC.AIO9CFG.OPT9** = 10b to use AIO9 as open drain output. Set **SOC.AIO9CFG.MUX9** = 00b to MUX the output state from **SOC.AIOOUT1.AIO9DOUT**. Use **SOC.AIO9CFG.MUX9** to MUX the output signal from the internal digital bus DBUS DB1 to DB7.

#### 22.9.3.2. AIO9 IO Interrupt

Set **SOC.AIOINTEN.AIO9REINTEN** to allow nIRQ2 interrupt on AIO8 low high transition.

Set **SOC.AIOINTEN.AIO9FEINTEN** to allow nIRQ2 interrupt on AIO8 high low transition.

The interrupt status can be monitored with **SOC.AIOINT.AIO9INT** and cleared by writing **SOC.AIOINT.AIO9INT** to 1b.

#### 22.9.3.3. AIO9 Polarity

Use **SOC.AIO9CFG.POL9** to set logic polarity of the signal between AIO9 input/output and MUX9.

### 22.9.4. AIO9 Single Ended Amplifier Mode

Set **SOC.AIO9CFG.MODE9** = 01b to use AIO9 as input to a programmable gain amplifier.

#### 22.9.4.1. AIO9 Amplifier Gain

Use **SOC.AIO9CFG.GAIN9** to set to gain between 1x to 48x or bidirectional amplifier bypass mode.

#### 22.9.4.2. AIO9 Analog MUX

Use **SOC.AIO9CFG.MUX9** to switch the output of the amplifier to analog channel AB1 to AB7 on the analog bus ABUS.

### 22.9.5. AIO9 Comparator Mode

Set **SOC.AIO9CFG.MODE9** = 10b to use AIO9 in comparator mode.

#### 22.9.5.1. AIO9 Comparator Hysteresis

Use **SOC.SMCFG.AIO9HYS** to enable AIO9 comparator hysteresis.

#### 22.9.5.2. AIO9 Comparator Reference

Use **SOC.AIO9CFG.OPT9** to set the compare value of the comparator to AB1, AB2, AB3 or VTHREF, settable with **SOC.SYSSTAT.VTHREF**.

#### 22.9.5.3. AIO9 Comparator Polarity

Use **SOC.AIO9CFG.POL9** to set the output polarity of the comparator.

#### 22.9.5.4. AIO9 Comparator Output

The comparator output can be observed with **SOC.AIOIN1.AIO9DIN**.

### 22.9.6. AIO9 Special Mode

Set **SOC.AIO7CFG.MODE7** = 11b to use AIO9 in special mode. In special mode the AIO9 comparator is enabled.

#### 22.9.6.1. AIO9 Comparator Hysteresis

Use **SOC.SMCFG.AIO9HYS** to enable AIO9 comparator hysteresis.

#### 22.9.6.2. AIO9 Comparator Reference

Use **SOC.AIO7CFG.OPT7** to set the compare value of the comparator to AB1, AB2 or AB3.

#### 22.9.6.3. AIO9 Comparator Reference Star Point

Use **SOC.AIO9CFG.MODE9[0]** =1b to connect AIO7, AIO8 and AIO9 to AB1 with a 100kOhm resistor to create a star point reference for the comparator.

With **SOC.AIO9CFG.MODE9[0]** =0b, the AB1 reference could for example come from AIO6 in amplifier mode **SOC.AIO6CFG.MODE6** = 01b and **SOC.AIO6CFG.GAIN6** = 000b direct mode and **SOC.AIO6CFG.MUX6** = 1b.

#### 22.9.6.4. AIO9 Voltage Reading

Use **SOC.AIO9CFG.MODE9[1]** =1b to MUX AIO9 to AB9 with 40% attenuation, so ADC can read out AIO9 voltage.

#### 22.9.6.5. AIO9 Comparator Output

Use **SOC.AIO9CFG.OPT9** to select AIO7, AIO8 or AIO9 comparator output signal for POS.

#### 22.9.6.6. AIO9 POS S/H Bypass

Set **SOC.AIO8CFG.OPT8[1]** =0b bypass the POS S/H.

Set **SOC.AIO8CFG.OPT8[1]** =1b use POS S/H for use with EMUX.

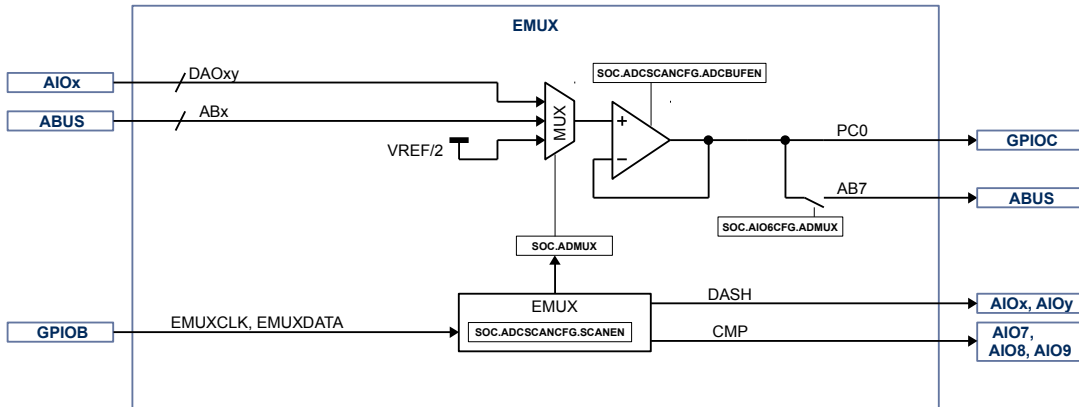
#### 22.9.6.7. AIO9 nIRQ2/POS Selector

Use **SOC.AIO8CFG.OPT8[0]** to select POS or nIRQ2 output.

## 22.10. EMUX and ADMUX

### 22.10.1. Block Diagram

Figure 22-9. EMUX



### 22.10.2. EMUX

The EMUX is a dedicated high speed low latency serial interface to control the ADMUX, the AIO7, AIO8, AIO9 POS S/H and the DAOxy S/H using the ASC0, ASC1 sequencer.

#### 22.10.2.1. EMUX Enable

Set **SOC.ADCSCANCFG.SCANEN** = 01b enable the EMUX control of the **SOC.ADMUX** and the DAOxy S/H.

#### 22.10.2.2. EMUXD Packet Structure

The EMUXD packet is transmitted MSB first.

### Register 22-26. SOC.AIOIN0 (AIO Digital Input, SOC 0x34)

BIT	NAME	DESCRIPTION
7:6	<b>C</b>	Header, write to 01b
5	<b>CMP</b>	Comparator toggle 1b: Hold POS value 0b: Sample POS value
4	<b>DASH</b>	DAxy SH toggle 1b: Hold DAOxy value 0b: Sample DAOxy

BIT	NAME	DESCRIPTION
3:0	MUX	AD MUX Channel Selector 1111b: VREF / 2 1110b: AB12 1101b: AB11 1100b: AB10 1011b: AB9 1010b: AB8 1001b: AB7 1000b: AB6 0111b: AB5 0110b: AB4 0101b: AB3 0100b: AB2 0011b: AB1 0010b: DAO54 0001b: DAO32 0000b: DAO10

### 22.10.2.3. EMUX Timing

The EMUXD packet is transmitted MSB first.

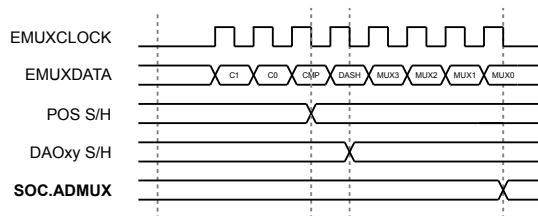
The header C1, C2 is fixed and need to be written to 01b.

The POS S/H is toggled based on the CMP bit in the EMUXD packet with the falling edge of the 3<sup>th</sup> clock cycle.

The DAOxy S/H is toggled based on the DASH bit in the EMUXD packet with the falling edge of the 4<sup>th</sup> clock cycle.

The ADMUX is switched with the falling edge of the 8<sup>th</sup> clock based on the data of bits 3:1 of the EMUXD packet.

**Figure 22-10. EMUX Timing**



### 22.10.3. ADMUX

The ADMUX is a dedicated analog MUX in the configurable analog front-end.

#### 22.10.3.1. ADMUX Control

Use **SOC.ADMUX** to set the channel for the ADMUX.

When EMUX is enabled, the SOC.ADMUX can be controlled directly from the ASC0, ASC1 sequencers using EMUXD.

### 22.10.3.2. ADMUX Buffer

Use **SOC.ADCSCANCFG.ADCBUFEN** to enable the ADMUX buffer.

### 22.10.3.3. ADMUX

Use **SOC.AIO6CFG.ADMUX** to route the output of the ADMUX to AB7 for debug purposes.



## 23. APPLICATION SPECIFIC POWER DRIVER

### 23.1. Register

#### 23.1.1. Register Map

**Table 23-1. Application Specific Power Driver Register Map**

SOC ADDRESS	NAME	DESCRIPTION	RESET VALUE
<b>Application Specific Power Driver</b>			
0x60	<b>Reserved</b>	Reserved	0x00
0x61	<b>SOC.PRCTL</b>	PR1, PR2 Control	0x00
0x62	<b>Reserved</b>	Reserved	0x00
0x63	<b>Reserved</b>	Reserved	0x00
0x64	<b>SOC.OMOUT</b>	OM0/2/4 output control	0x00
0x65	<b>Reserved</b>	Reserved	0x00
0x66	<b>SOC.DRVCTL</b>	Gate Driver Control	0x00
0x67	<b>SOC.PROTCTL</b>	PROT Control	0x00
0x68	<b>Reserved</b>	Reserved	0x00
0x69 – 0x7F	<b>Reserved</b>	Reserved	N/A

#### 23.1.2. SOC.PRCTL

**Register 23-1. SOC.PRCTL (PR1, PR2 Control, SOC 0x61)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
7	<b>HSPR1EN</b>	R/W	0x0	High side PR1 protection enable 1b: PR1 enabled 0b: PR1 disabled
6	<b>HSPR2EN</b>	R/W	0x0	High side PR2 protection enable 1b: PR2 enabled 0b: PR2 disabled
5:0	<b>Reserved</b>	R/W	0x0	Reserved, write to 0x0

#### 23.1.3. SOC.OMOUT

**Register 23-2. SOC.OMOUT (OM0/2/4 Output, SOC 0x64)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:6	<b>Reserved</b>	R/W	0x0	Reserved, write to 0x0
5	<b>PROTDB7EN</b>	R/W	0x0	DB7 signal enable for PROT 1b: enable DB7 0b: disable DB7
4	<b>OM4DOUT</b>	R/W	0x0	OM4 output in open drain mode 1b: High-Z 0b: VSS
3	<b>PROTDB6EN</b>	R/W	0x0	DB6 signal enable for PROT 1b: enable DB6 0b: disable DB6
2	<b>OM2DOUT</b>	R/W	0x0	OM2 output in open drain mode 1b: High-Z 0b: VSS
1	<b>PROTDB5EN</b>	R/W	0x0	DB5 signal enable for PROT 1b: enable DB5 0b: disable DB5
0	<b>OM0DOUT</b>	R/W	0x0	OM0 output in open drain mode 1b: High-Z 0b: VSS

#### 23.1.4. SOC.DRVCTL

**Register 23-3. SOC.DRVCTL (Application Specific Power Driver Control, SOC 0x66)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:1	<b>Reserved</b>	R/W	0x0	Reserved, write to 0x0
0	<b>DRVEN</b>	R/W	0x0	Application Specific Power Driver Enable 1b: enabled 0b: disabled

#### 23.1.5. SOC.PROTCTL

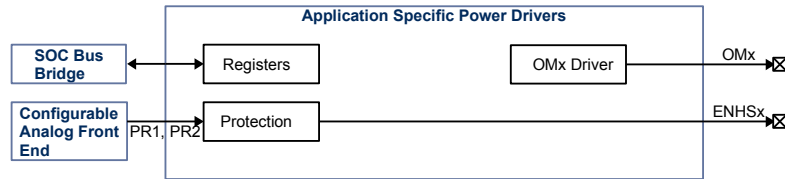
**Register 23-4. SOC.PROTCTL (PROT Control, SOC 0x67)**

BIT	NAME	ACCESS	RESET	DESCRIPTION
7:1	<b>Reserved</b>	R/W	0x0	Reserved, write to 0x0
0	<b>PROTSEL</b>	R/W	0x0	Select input signal for PROT 1b: nIRQ2, DB5, DB6 or DB7 0b: PR1

## 23.2. Details of Operation

### 23.2.1. Block Diagram

Figure 23-1. Application Specific Power Driver



### 23.2.2. Configuration

Following blocks need to be configured for correct use of the Application specific power driver

- Clock Control System (CCS)
- Nested Vectored Interrupt Controller (NVIC)
- IO Controller
- SOC Bridge

### 23.2.3. Application Specific Power Driver

The application specific power driver registers are accessible through the SOC Bridge. The OMx open drain drivers are controlled by SOC registers. The ENHS0 and ENHS1 protection outputs can be configured to toggle based on protection from the configurable analog front end.

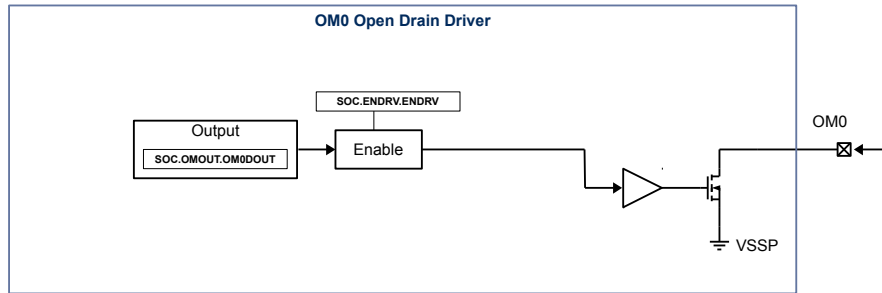
#### 23.2.3.1. Application Specific Power Driver Enable

Use **SOC.DRVCTL.DRVEN** to enable the application specific power driver.

## 23.3. OM0

### 23.3.1. Block Diagram

Figure 23-2. OM0



### 23.3.2. OM0

OM0 is a medium voltage open drain output.

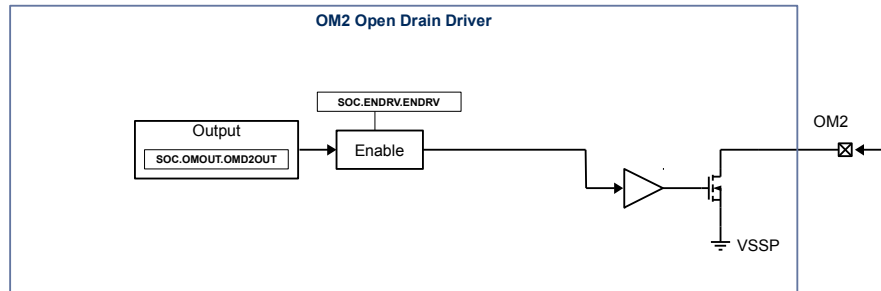
#### 23.3.2.1. OM0 Open Drain Output

Use `SOC.OMOUT.OM0DOUT` to toggle output state of OM0.

## 23.4. OM2

### 23.4.1. Block Diagram

Figure 23-3. OM2



### 23.4.2. OM2

OM2 is a medium voltage open drain output.

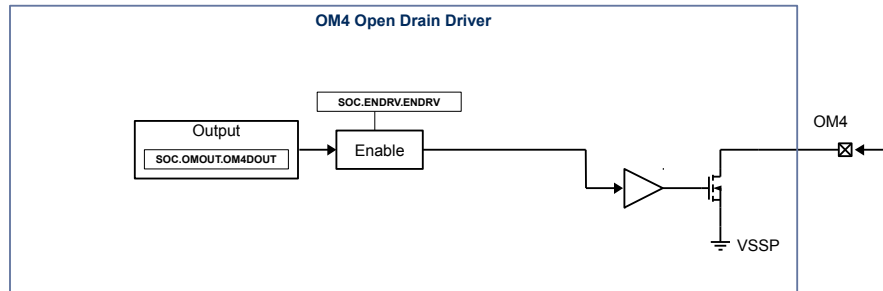
#### 23.4.2.1. OM2 Open Drain Output

Use `SOC.OMOUT.OM2DOUT` to toggle output state of OM2.

## 23.5. OM4

### 23.5.1. Block Diagram

Figure 23-4. OM4



### 23.5.2. OM4

OM4 is a medium voltage open drain output.

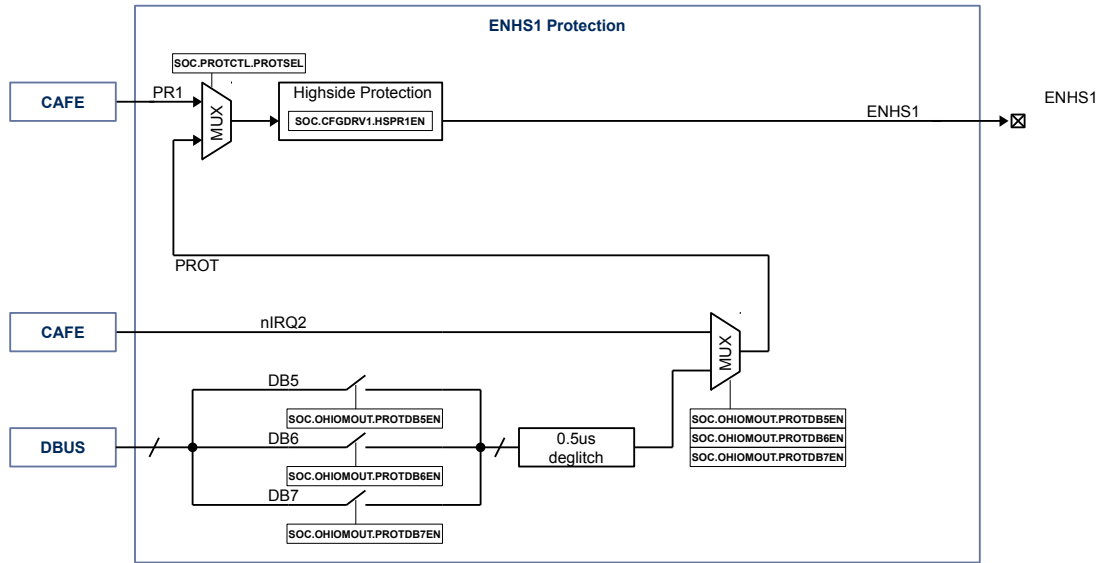
#### 23.5.2.1. OM4 Open Drain Output

Use `SOC.OMOUT.OM4DOUT` to toggle output state of OM4.

## 23.6. ENHS1 Protection

### 23.6.1. Block Diagram

Figure 23-5. ENHS1 Protection



### 23.6.2. ENHS1 Protection

The PR1, nIRQ2, DB5, DB6, DB7 signals from the AIOx inputs in the configurable analog front-end can be used to drive the ENHS1 pin to signal external components.

#### 23.6.2.1. Protection Enable

Use **SOC.DRV1CFG.HSPR1EN** to enable protection signal ENHS1 for ENHS1 output.

#### 23.6.2.2. Input Signal Selection

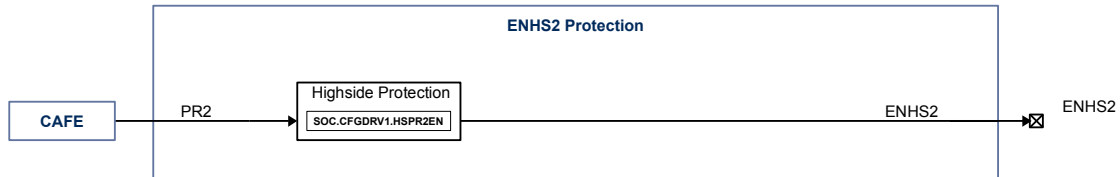
Use **SOC.PROTCTL.PROTSEL** to select between PR1 or PROT signal as input to ENHS1 protection signal.

Use **SOC.OMOUT.PROTDB5EN**, **SOC.OMOUT.PROTDB6EN**, and **SOC.OMOUT.PROTDB7EN** to select input signal to PROT.

## 23.7. ENHS2 Protection

### 23.7.1. Block Diagram

Figure 23-6. ENHS2 Protection



### 23.7.2. ENHS2 Protection

The PR2 signal from the AIOx inputs in the configurable analog front-end can be used to toggle ENHS2 output.

#### 23.7.2.1. Protection Enable

Use **SOC.DRV1CFG.HSPR2EN** to enable protection signal PR2 to ENHS2 output.



## 24. ARM CORTEX-M0 REFERENCE

### 24.1. Introduction

#### 24.1.1. Overview

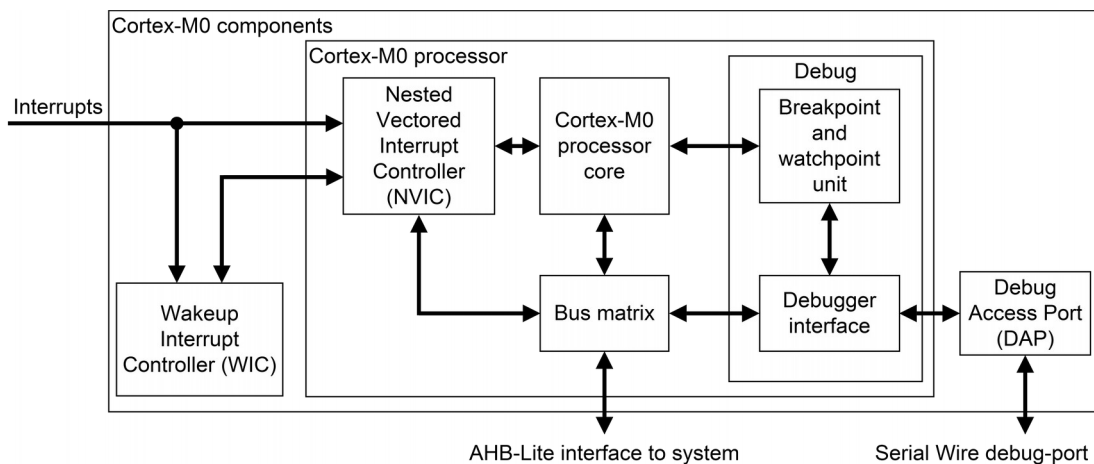
This chapter is taken from the ARM Cortex-M0 User Guide with minimal modifications made to account for the specific Cortex-M0 implementation.

#### 24.1.2. About the Cortex-M0 processor and core peripherals

The Cortex™-M0 processor is an entry-level 32-bit ARM Cortex processor designed for a broad range of embedded applications. It offers significant benefits to developers, including:

- a simple architecture that is easy to learn and program
- ultra-low power, energy efficient operation
- excellent code density
- deterministic, high-performance interrupt handling
- upward compatibility with Cortex-M processor family.

**Figure 24-1. Cortex-M0 implementation**



The Cortex-M0 processor is built on a highly area and power optimized 32-bit processor core, with a 3-stage pipeline von Neumann architecture. The processor delivers exceptional energy efficiency through a small but powerful instruction set and extensively optimized design, providing high-end processing hardware including a single-cycle multiplier.

The Cortex-M0 processor implements the ARMv6-M architecture, which is based on the 16-bit Thumb® instruction set and includes Thumb-2 technology. This provides the exceptional performance expected of a modern 32-bit architecture, with a higher code density than other 8-bit and 16-bit microcontrollers.

The Cortex-M0 processor closely integrates a configurable Nested Vectored Interrupt Controller (NVIC), to deliver industry-leading interrupt performance. The NVIC:

- includes a non-maskable interrupt (NMI)
- provides zero jitter interrupt option
- provides four interrupt priority levels.

The tight integration of the processor core and NVIC provides fast execution of interrupt service routines (ISRs), dramatically reducing the interrupt latency. This is achieved through the hardware stacking of registers, and the ability to abandon and restart load-multiple and store-multiple operations. Interrupt handlers do not require any assembler wrapper code, removing any code overhead from the ISRs. Tail-chaining optimization also significantly reduces the overhead when switching from one ISR to another.

To optimize low-power designs, the NVIC integrates with the sleep modes, that include a deep sleep function that enables the entire device to be rapidly powered down.

#### *24.1.2.1. System-level interface*

The Cortex-M0 processor provides a single system-level interface using AMBA® technology to provide high speed, low latency memory accesses.

#### *24.1.2.2. Integrated configurable debug*

The Cortex-M0 processor implements a complete hardware debug solution, with extensive hardware breakpoint and watchpoint options. This provides high system visibility of the processor, memory and peripherals through a Serial Wire Debug (SWD) port that is ideal for microcontrollers and other small package devices. The device supports 4 hardware breakpoints.

#### *24.1.2.3. Cortex-M0 processor features summary*

- high code density with 32-bit performance
- tools and binary upwards compatible with Cortex-M processor family
- integrated ultra low-power sleep modes
- efficient code execution permits slower processor clock or increases sleep mode time
- single-cycle 32-bit hardware multiplier
- zero jitter interrupt handling
- extensive debug capabilities.

#### *24.1.2.4. Cortex-M0 core peripherals*

These are:

##### **24.1.2.4.1. NVIC**

The NVIC is an embedded interrupt controller that supports low latency interrupt processing.

#### 24.1.2.4.2. System Control Block

The System Control Block (SCB) is the programmers model interface to the processor. It provides system implementation information and system control, including configuration, control, and reporting of system exceptions.

#### 24.1.2.4.3. System timer

The system timer, SysTick, is a 24-bit count-down timer. Use this as a Real Time Operating System (RTOS) tick timer or as a simple counter.

### 24.2. The Cortex-M0 Processor

#### 24.2.1. Programmers Model

This section describes the Cortex-M0 programmers model. In addition to the individual core register descriptions, it contains information about the processor modes and stacks.

##### 24.2.1.1. Processor modes

The processor modes are:

##### **Thread mode**

Used to execute application software. The processor enters Thread mode when it comes out of reset.

##### **Handler mode**

Used to handle exceptions. The processor returns to Thread mode when it has finished all exception processing.

##### 24.2.1.2. Stacks

The processor uses a full descending stack. This means the stack pointer indicates the last stacked item on the stack memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks, the main stack and the process stack, with independent copies of the stack pointer, see Stack Pointer in chapter 24.2.1.3.2 on page 281.

In Thread mode, the CONTROL register controls whether the processor uses the main stack or the process stack, see CONTROL register in chapter 24.2.1.3.12 on page 284. In Handler mode, the processor always uses the main stack. The options for processor operations are:

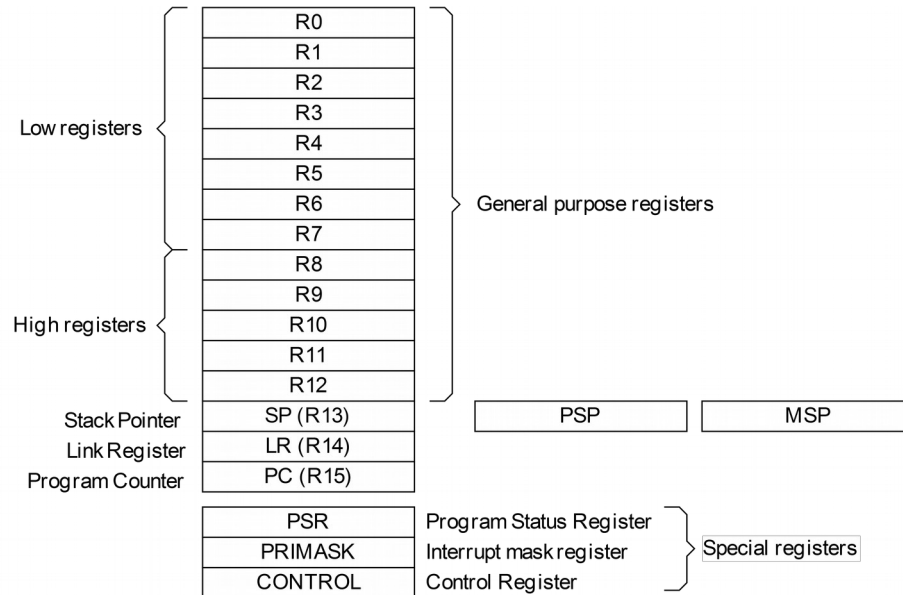
**Table 24-1. Summary of processor mode and stack use options**

PROCESSOR MODE	USED TO EXECUTE	STACK USED
Thread	Applications	Main stack or process stack*
Handler	Exception handlers	Main stack

\* See CONTROL Register in chapter 24.2.1.3.12 on page 284

24.2.1.3. Core Registers

**Figure 24-2. Core Registers**



**Table 24-2. Core register set summary**

NAME	TYPE*	RESET VALUE	DESCRIPTION
R0 – R12	RW	Unknown	General Purpose Register, see chapter 24.2.1.3.1 on page 280
MSP	RW	See description	Stack pointer, see chapter 24.2.1.3.2 on page 281
PSP	RW	Unknown	Stack pointer, see chapter 24.2.1.3.2 on page 281
LR	RW	Unknown	Link Register, see chapter 24.2.1.3.3 on page 281
PC	RW	See description	Program Counter, see chapter 28224.2.1.3.4 on page 281
PSR	RW	Unknown **	Program Status Register, see chapter 24.2.1.3.5 on page 281
APSR	RW	Unknown	Application Program status register, see chapter 24.2.1.3.6 on page 282
IPSR	RW	0x0000 0000	Interrupt Program status register, see chapter 24.2.1.3.7 on page 282
ESPR	RW	Unknown **	Execution Program status register, see chapter 24.2.1.3.8 on page 283
PRIMASK	RW	0x0000 0000	Priority Mask Register, see chapter 24.2.1.3.11 on page 284
CONTROL	RW	0x0000 0000	Control Register, see chapter 24.2.1.3.12 on page 284

\*. Describes access type during program execution in thread mode and Handler mode. Debug access can differ.

\*\* . Bit[24] is the T-bit and is loaded from bit[0] of the reset vector.

**24.2.1.3.1. General-purpose registers**

R0-R12 are 32-bit general-purpose registers for data operations.

### 24.2.1.3.2. Stack Pointer

The Stack Pointer (SP) is register R13. In Thread mode, bit[1] of the CONTROL register indicates the stack pointer to use:

- 0 = Main Stack Pointer (MSP). This is the reset value.
- 1 = Process Stack Pointer (PSP).

On reset, the processor loads the MSP with the value from address 0x0000 0000

### 24.2.1.3.3. Link Register

The Link Register (LR) is register R14. It stores the return information for subroutines, function calls, and exceptions. On reset, the LR value is Unknown.

### 24.2.1.3.4. Program Counter

The Program Counter (PC) is register R15. It contains the current program address. On reset, the processor loads the PC with the value of the reset vector, which is at address 0x0000 0004. Bit[0] of the value is loaded into the EPSR T-bit at reset and must be 1.

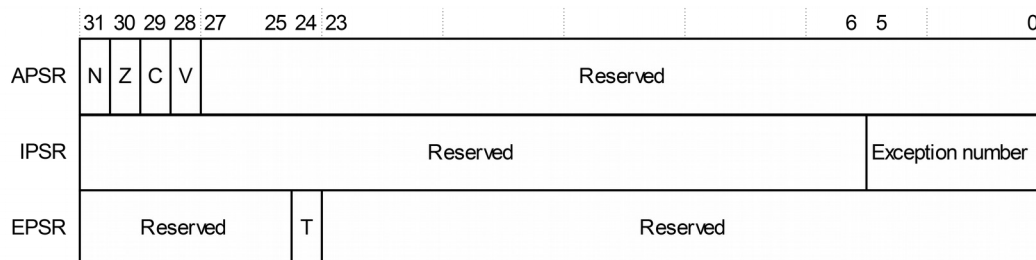
### 24.2.1.3.5. Program Status Register

The Program Status Register (PSR) combines:

- Application Program Status Register (APSR)
- Interrupt Program Status Register (IPSR)
- Execution Program Status Register (EPSR).

These registers are mutually exclusive bitfields in the 32-bit PSR. The PSR bit assignments are:

**Figure 24-3. PSR**



Access these registers individually or as a combination of any two or all three registers, using the register name as an argument to the `MSR` or `MRS` instructions. For example:

- read all of the registers using PSR with the `MRS` instruction

- write to the APSR using APSR with the MSR instruction.

The PSR combinations and attributes are:

**Table 24-3. Core register set summary**

REGISTER	TYPE*	COMBINATION
PSR	RW *, **	APSR, EPSR, and IPSR
IEPSR	RO	EPSR and IPSR
IAPSR	RW,*	APSR and IPSR
EAPST	RW,**	APSR and EPSR

\* The processor ignores writes to the IPSR bits.

\*\* Reads of the EPSR bits return zero, and the processor ignores writes to the these bits

See the instruction descriptions MRS in chapter 24.3.7.6 on page 333 and MSR in chapter 24.3.7.7 on page 334 for more information about how to access the program status registers.

### 24.2.1.3.6. Application Program Status Register

The APSR contains the current state of the condition flags, from previous instruction executions. See the register summary in Table 24-2. Core register set summary on page 280 for its attributes. The bit assignments are:

**Table 24-4. APSR bit assignments**

BIT	NAME	FUNCTION
31	<b>N</b>	Negative Flag
30	<b>Z</b>	Zero Flag
29	<b>C</b>	Carry or Borrow Flag
28	<b>V</b>	Overflow Flag
27:0	<b>Reserved</b>	Reserved

See The condition flags in chapter 24.3.3.6.1 on page 307 for more information about the APSR negative, zero, carry or borrow, and overflow flags.

### 24.2.1.3.7. Interrupt Program Status Register

The IPSR contains the exception number of the current Interrupt Service Routine (ISR). See the register summary in Table 24-2. Core register set summary on page 280 for its attributes. The bit assignments are:

**Table 24-5. IPSR bit assignments**

BIT	NAME	FUNCTION
31:6	<b>Reserved</b>	Reserved

BIT	NAME	FUNCTION
5:0	<b>Exception Number</b>	This is the number of the current exception: 63-48: Reserved 47: IRQ31 . . . 16: IRQ0 15: SysTick 14: PendSV 13-12: Reserved 11: SVCALL 10-4: Reserved 3: HardFault 2: NMI 1: Reserved 0: Thread mode see Exception types in chapter 24.2.3.2 on page 291

### 24.2.1.3.8. Execution Program Status Register

The EPSR contains the Thumb state bit.

See the register summary in Table 24-2. Core register set summary on page 280 for ESPR attributes. The bit assignments are:

**Table 24-6. EPSR bit assignments**

BIT	NAME	FUNCTION
31:25	<b>Reserved</b>	Reserved
24	<b>T</b>	Thumb state bit
23:0	<b>Reserved</b>	Reserved

Attempts by application software to read the EPSR directly using the `MRS` instruction always return zero. Attempts to write the EPSR using the `MSR` instruction are ignored. Fault handlers can examine the EPSR value in the stacked PSR to determine the cause of the fault. See Exception entry and return in chapter 24.2.3.6 on page 295. The following can clear the T bit to 0:

- instructions `BLX`, `BX` and `POP{PC}`
- restoration from the stacked xPSR value on an exception return
- `bit[0]` of the vector value on an exception entry.

Attempting to execute instructions when the T bit is 0 results in a HardFault or lockup.

See Lockup in chapter 24.2.4.1 on page 297 for more information.

### 24.2.1.3.9. Interruptible-restartable instructions

The interruptible-restartable instructions are `LDM` and `STM`, and the multiply instruction. When an interrupt occurs during the execution of one of these instructions, the processor abandons execution of the instruction.

After servicing the interrupt, the processor restarts execution of the instruction from the beginning.

### 24.2.1.3.10. Exception mask register

The exception mask register disables the handling of exceptions by the processor. Disable exceptions where they might impact on timing critical tasks or code sequences requiring atomicity.

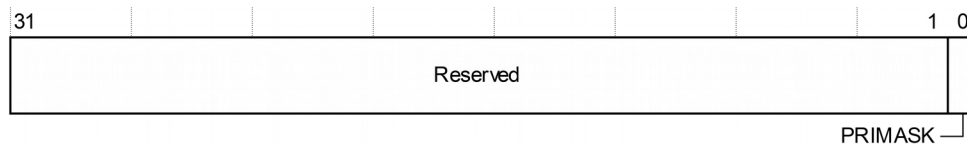
To disable or re-enable exceptions, use the `MSR` and `MRS` instructions, or the `CPS` instruction, to change the value of `PRIMASK`. See `MRS` in chapter 24.3.7.6 on page 333, `MSR` in chapter 24.3.7.7 on page 334, and `CPS` in chapter 24.3.7.2 on page 330 for more information.

### 24.2.1.3.11. Priority Mask Register

The `PRIMASK` register prevents activation of all exceptions with configurable priority.

See the register summary in Table 24-2. Core register set summary on page 280 for its attributes. The bit assignments are:

**Figure 24-4. PRIMASK**



**Table 24-7. PRIMASK register bit assignments**

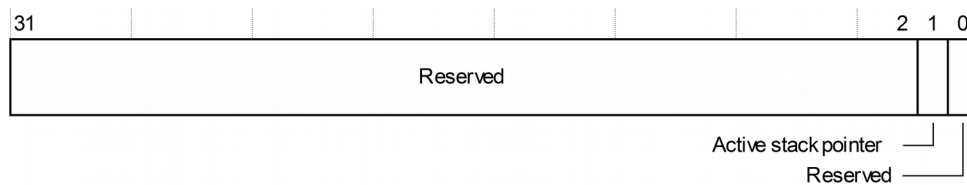
BIT	NAME	FUNCTION
31:1	<b>Reserved</b>	Reserved
0	<b>PRIMASK</b>	1: prevents activation of all exceptions with configurable priority 0: no effect

### 24.2.1.3.12. Control Register

The `CONTROL` register controls the stack used when the processor is in Thread mode.

See the register summary in Table 24-2. Core register set summary on page 280 for its attributes. The bit assignments are:

**Figure 1-6. CONTROL**





**Table 24-8. CONTROL register bit assignments**

BIT	NAME	FUNCTION
31:1	<b>Reserved</b>	Reserved
0	<b>Active Stack Pointer</b>	Defines the current stack: 1: MSP is the current stack pointer 0: PSP is the current stack pointer In Handler mode this bit reads as zero and ignores writes.
0	<b>Reserved</b>	Reserved

Handler mode always uses the MSP, so the processor ignores explicit writes to the active stack pointer bit of the CONTROL register when in Handler mode. The exception entry and return mechanisms update the CONTROL register.

In an OS environment, it is recommended that threads running in Thread mode use the process stack and the kernel and exception handlers use the main stack.

By default, Thread mode uses the MSP. To switch the stack pointer used in Thread mode to the PSP, use the MSR instruction to set the Active stack pointer bit to 1, see MSR in chapter 24.3.7.7 on page 334.

#### Note

When changing the stack pointer, software must use an ISB instruction immediately after the MSR instruction. This ensures that instructions after the ISB execute using the new stack pointer. See ISB in chapter 24.3.7.5 on page 332.

#### 24.2.1.4. Exceptions and interrupts

The Cortex-M0 processor supports interrupts and system exceptions. The processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. An interrupt or exception changes the normal flow of software control. The processor uses handler mode to handle all exceptions except for reset. See Exception entry in chapter 24.2.3.6.5 on page 296 and Exception return in chapter 24.2.3.6.6 on page 296 for more information.

The NVIC registers control interrupt handling. See Nested Vectored Interrupt Controller in chapter 24.4.2 on page 339 for more information.

#### 24.2.1.5. Data Types

The processor:

- supports the following data types:
  - 32-bit words
  - 16-bit halfwords
  - 8-bit bytes
- manages all data memory accesses as little-endian. Instruction memory and Private Peripheral Bus (PPB) accesses are always little-endian. See Memory regions, types and attributes in chapter 24.2.2.1 on page 287 for more information.

#### 24.2.1.6. *The Cortex Microcontroller Software Interface Standard*

ARM provides the Cortex Microcontroller Software Interface Standard (CMSIS) for programming Cortex-M0 microcontrollers. The CMSIS is an integrated part of the device driver library. For a Cortex-M0 microcontroller system, CMSIS defines:

- a common way to:
  - access peripheral registers
  - define exception vectors
- the names of:
  - the registers of the core peripherals
  - the core exception vectors
- a device-independent interface for RTOS kernels.

The CMSIS includes address definitions and data structures for the core peripherals in the Cortex-M0 processor. It also includes optional interfaces for middleware components comprising a TCP/IP stack and a Flash file system.

The CMSIS simplifies software development by enabling the reuse of template code, and the combination of CMSIS-compliant software components from various middleware vendors. Software vendors can expand the CMSIS to include their peripheral definitions and access functions for those peripherals.

This document includes the register names defined by the CMSIS, and gives short descriptions of the CMSIS functions that address the processor core and the core peripherals.

#### **Note**

This document uses the register short names defined by the CMSIS. In a few cases these differ from the architectural short names that might be used in other documents.

The following sections give more information about the CMSIS:

- Power management programming hints in chapter 24.2.5.5 on page 300
- Intrinsic functions in chapter 24.3.2 on page 302
- Accessing the Cortex-M0 NVIC registers using CMSIS in chapter 24.4.2.1 on page 340
- NVIC programming hints in chapter 24.4.2.8.1 on page 344

#### **24.2.2. Memory model**

This section describes the processor memory map and the behavior of memory accesses. The processor has a fixed memory map that provides up to 4GB of addressable memory. The memory map is:

**Figure 24-6. Memory Map**

Device	511MB	0xFFFFFFFF
Private peripheral bus	1MB	0xE0100000 0xE00FFFFFF 0xE0000000 0xDFFFFFFF
External device	1.0GB	
External RAM	1.0GB	0xA0000000 0x9FFFFFFF
Peripheral	0.5GB	0x60000000 0x5FFFFFFF
SRAM	0.5GB	0x40000000 0x3FFFFFFF
Code	0.5GB	0x20000000 0x1FFFFFFF
		0x00000000

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers, see About the Cortex-M0 processor and core peripherals in chapter 24.1.2 on page 277

#### 24.2.2.1. Memory regions, types and attributes

The memory map is split into regions. Each region has a defined memory type, and some regions have additional memory attributes. The memory type and attributes determine the behavior of accesses to the region.

The memory types are:

##### 24.2.2.1.1. Normal

The processor can re-order transactions for efficiency, or perform speculative reads.

### 24.2.2.1.2. Device

The processor preserves transaction order relative to other transactions to Device or Strongly-ordered memory.

### 24.2.2.1.3. Strongly-ordered

The processor preserves transaction order relative to all other transactions.

The different ordering requirements for Device and Strongly-ordered memory mean that the memory system can buffer a write to Device memory, but must not buffer a write to Strongly-ordered memory.

The additional memory attributes include.

### 24.2.2.1.4. Execute Never (XN)

Means the processor prevents instruction accesses. A HardFault exception is generated on executing an instruction fetched from an XN region of memory.

#### 24.2.2.2. Memory system ordering of memory accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not guarantee that the order in which the accesses complete matches the program order of the instructions, providing any re-ordering does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions, see Software ordering of memory accesses in chapter 24.2.2.4 on page 289.

However, the memory system does guarantee some ordering of accesses to Device and Strongly-ordered memory. For two memory access instructions A1 and A2, if A1 occurs before A2 in program order, the ordering of the memory accesses caused by two instructions is:

**Figure 24-7. Memory Ordering Restrictions**

A1 \ A2	Normal access	Device access		Strongly-ordered access
		Non-shareable	Shareable	
Normal access	-	-	-	-
Device access, non-shareable	-	<	-	<
Device access, shareable	-	-	<	<
Strongly-ordered access	-	<	<	<

Where:

- Means that the memory system does not guarantee the ordering of the accesses.

< Means that accesses are observed in program order, that is, A1 is always observed before A2.

### 24.2.2.3. Behavior of memory accesses

The behavior of accesses to each region in the memory map is:

**Table 24-9. Memory Access Behavior**

ADDRESS RANGE	MEMORY REGION	MEMORY TYPE	XN*	DESCRIPTION
0xFFFF FFFF – 0xE010 0000	<b>Device</b>	Device	XN	Reserved
0xE00F FFFF – 0xE000 0000	<b>Private Peripheral Bus</b>	Strongly -ordered	XN	This region includes the NVIC, System timer, and System Control Block. Only word accesses can be used in this region.
0xDFFF FFFF – 0xA000 0000	<b>External Device</b>	Device	XN	External device memory
0x9FFF FFFF – 0x6000 0000	<b>External RAM</b>	Normal	-	Executable region for data
0x5FFF FFFF – 0x4000 0000	<b>Peripheral</b>	Device	XN	External device memory
0x3FFF FFFF – 0x2000 0000	<b>SRAM</b>	Normal	-	Executable region for data. You can also put code here
0x1FFF FFFF – 0x0000 0000	<b>Code</b>	Normal	-	Executable region for program code. You can also put data here

\* See Memory regions, types and attributes in chapter 24.2.2.1 on page 287 for more information.

The Code, SRAM, and external RAM regions can hold programs.

### 24.2.2.4. Software ordering of memory accesses

The order of instructions in the program flow does not always guarantee the order of the corresponding memory transactions. This is because:

- the processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence
- memory or devices in the memory map might have different wait states
- some memory accesses are buffered or speculative.

Memory system ordering of memory accesses in chapter 24.2.2.2 on page 288 describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The processor provides the following memory barrier instructions:

#### 24.2.2.4.1. DMB

The Data Memory Barrier (DMB) instruction ensures that outstanding memory transactions complete before subsequent memory transactions. See DMB in chapter 24.3.7.3 on page 331.

#### 24.2.2.4.2. DSB

The Data Synchronization Barrier (DSB) instruction ensures that outstanding memory transactions complete

before subsequent instructions execute. See `DSB` in chapter 24.3.7.4 on page 332.

#### **24.2.2.4.3. ISB**

The Instruction Synchronization Barrier (`ISB`) ensures that the effect of all completed memory transactions is recognizable by subsequent instructions. See `ISB` in chapter 24.3.7.5 on page 332.

The following are examples of using memory barrier instructions:

#### **24.2.2.4.4. Vector table**

If the program changes an entry in the vector table, and then enables the corresponding exception, use a `DMB` instruction between the operations. This ensures that if the exception is taken immediately after being enabled the processor uses the new exception vector.

#### **24.2.2.4.5. Self-modifying code**

If a program contains self-modifying code, use an `ISB` instruction immediately after the code modification in the program. This ensures subsequent instruction execution uses the updated program.

#### **24.2.2.4.6. Memory map switching**

If the system contains a memory map switching mechanism, use a `DSB` instruction after switching the memory map. This ensures subsequent instruction execution uses the updated memory map.

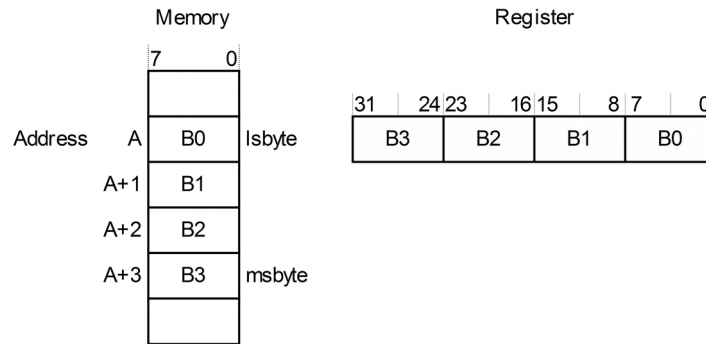
Memory accesses to Strongly-ordered memory, such as the System Control Block, do not require the use of `DMB` instructions.

#### *24.2.2.5. Memory endianness*

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. Little-endian format in chapter 24.2.2.5.1 on page 290 describes how words of data are stored in memory.

#### **24.2.2.5.1. Little-endian format**

In little-endian format, the processor stores the least significant byte (lsbyte) of a word at the lowest-numbered byte, and the most significant byte (msbyte) at the highest-numbered byte. For example:



**Figure 24-8. Little Endian Format**

### 24.2.3. Exception model

This section describes the exception model.

#### 24.2.3.1. Exception states

Each exception is in one of the following states:

##### 24.2.3.1.1. Inactive

The exception is not active and not pending.

##### 24.2.3.1.2. Pending

The exception is waiting to be serviced by the processor.

An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.

##### 24.2.3.1.3. Active

An exception that is being serviced by the processor but has not completed.

#### Note

An exception handler can interrupt the execution of another exception handler. In this case both exceptions are in the active state.

##### 24.2.3.1.4. Active and pending

The exception is being serviced by the processor and there is a pending exception from the same source.

#### 24.2.3.2. Exception types

The exception types are:

### 24.2.3.2.1. Reset

Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts in Thread mode.

### 24.2.3.2.2. NMI

A NonMaskable Interrupt (NMI) can be signaled by a peripheral or triggered by software. This is the highest priority exception other than reset. It is permanently enabled and has a fixed priority of -2. NMIs cannot be:

- masked or prevented from activation by any other exception
- preempted by any exception other than Reset.

### 24.2.3.2.3. HardFault

A HardFault is an exception that occurs because of an error during normal or exception processing. HardFaults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority.

### 24.2.3.2.4. SVCcall

A supervisor call (SVC) is an exception that is triggered by the SVC instruction. In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers.

### 24.2.3.2.5. PendSV

PendSV is an interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active.

### 24.2.3.2.6. SysTick

A SysTick exception is an exception the system timer generates when it reaches zero. Software can also generate a SysTick exception. In an OS environment, the processor can use this exception as system tick.

### 24.2.3.2.7. Interrupt (IRQ)

An interrupt, or IRQ, is an exception signaled by a peripheral, or generated by a software request. All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor

**Table 24-10. Properties of the different exception types**

EXCEPTION NUMBER*	IRQ NUMBER*	EXCEPTION TYPE	PRIORITY	VECTOR ADDRESS**	ACTIVATION
1	-	Reset	-3, the highest	0x0000 0004	Asynchronous
2	-14	NMI	-2	0x0000 0008	Asynchronous
3	-13	HardFault	-1	0x0000 000C	Synchronous
4-10	-	Reserved	-	-	-
11	-5	SVCcall	Configurable***	0x0000 002C	Synchronous



EXCEPTION NUMBER*	IRQ NUMBER*	EXCEPTION TYPE	PRIORITY	VECTOR ADDRESS**	ACTIVATION
12-13	-	Reserved	-	-	-
14	-2	PendSV	Configurable***	0x0000 0038	Asynchronous
15	-1	SysTick	Configurable***	0x0000 003C	Asynchronous
16 and above	0 and above	Interrupt (IRQ)	Configurable***	0x0000 0040 and above****	Asynchronous

\*To simplify the software layer, the CMSIS only uses IRQ numbers and therefore uses negative values for exceptions other than interrupts. The IPSR returns the Exception number, see Interrupt Program Status Register in chapter 24.2.1.3.5 on page 281.

\*\*See Vector table for more information.

\*\*\*See Interrupt Priority Registers in chapter 24.4.2.6 on page 342.

\*\*\*\*Increasing in steps of 4.

For an asynchronous exception, other than reset, the processor can execute additional instructions between when the exception is triggered and when the processor enters the exception handler.

Privileged software can disable the exceptions that Table 24-10. Properties of the different exception types on page 292 shows as having configurable priority, see Interrupt Clear-enable Register in chapter 24.4.2.3 on page 341.

For more information about HardFaults, see Fault handling in chapter 24.2.4 on page 297.

### 24.2.3.3. Exception handlers

The processor handles exceptions using:

#### 24.2.3.3.1. Interrupt Service Routines (ISRs)

Interrupts IRQ0 to IRQ31 are the exceptions handled by ISRs.

#### 24.2.3.3.2. Fault handler

HardFault is the only exception handled by the fault handler.

#### 24.2.3.3.3. System handlers

NMI, PendSV, SVCall SysTick, and HardFault are all system exceptions handled by system handlers.

### 24.2.3.4. Vector table

The vector table contains the reset value of the stack pointer, and the start addresses, also called exception vectors, for all exception handlers. Figure 24-9. Vector Table on page 294 shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is written in Thumb code.

**Figure 24-9. Vector Table**

Exception number	IRQ number	Vector	Offset
47	31	IRQ31	0xBC
.		.	.
.		.	.
18	2	IRQ2	0x48
17	1	IRQ1	0x44
16	0	IRQ0	0x40
15	-1	SysTick   Reserved	0x3C
14	-2	PendSV	0x38
13		Reserved	
12			
11	-5	SVCall	0x2C
10			
9			
8			
7		Reserved	
6			
5			
4			
3	-13	HardFault	0x10
2	-14	NMI	0x0C
			0x08
1		Reset	0x04
			0x00
		Initial SP value	0x00

The vector table is fixed at address 0x0000 0000.

#### 24.2.3.5. Exception priorities

As Table 24-10. Properties of the different exception types on page 292 shows, all exceptions have an associated priority, with:

- a lower priority value indicating a higher priority
- configurable priorities for all exceptions except Reset, HardFault, and NMI.

If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities see

- System Handler Priority Registers in chapter 24.4.3.7 on page 350
- Interrupt Priority Registers in chapter 24.4.2.6 on page 342.

#### Note

Configurable priority values are in the range 0-192, in steps of 64. The Reset, HardFault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.

Assigning a higher priority value to IRQ[0] and a lower priority value to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

#### 24.2.3.6. Exception entry and return

Descriptions of exception handling use the following terms:

##### 24.2.3.6.1. Preemption

When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled.

When one exception preempts another, the exceptions are called nested exceptions. See Exception entry in chapter 24.2.3.6.5 on page 296 for more information.

##### 24.2.3.6.2. Return

This occurs when the exception handler is completed, and:

- there is no pending exception with sufficient priority to be serviced
- the completed exception handler was not handling a late-arriving exception.

The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See Exception return in chapter 24.2.3.6.6 on page 296 for more information.

##### 24.2.3.6.3. Tail-chaining

This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.

##### 24.2.3.6.4. Late-arriving

This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved would be the same for both exceptions. On return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.

### 24.2.3.6.5. Exception entry

Exception entry occurs when there is a pending exception with sufficient priority and either:

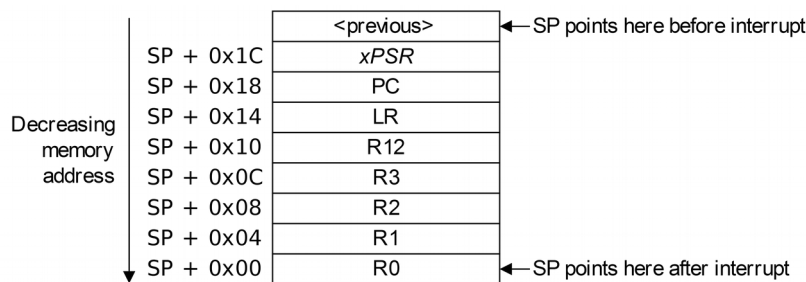
- the processor is in Thread mode
- the new exception is of higher priority than the exception being handled, in which case the new exception preempts the exception being handled.

When one exception preempts another, the exceptions are nested.

Sufficient priority means the exception has greater priority than any limit set by the mask register, see Exception mask register in chapter 24.2.1.3.10 on page 284. An exception with less priority than this is pending but is not handled by the processor.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred to as stacking and the structure of eight data words is referred to as a stack frame. The stack frame contains the following information:

**Figure 24-10. Exception Entry Stack Contents**



Immediately after stacking, the stack pointer indicates the lowest address in the stack frame. The stack frame is aligned to a double-word address.

The stack frame includes the return address. This is the address of the next instruction in the interrupted program. This value is restored to the PC at exception return so that the interrupted program resumes.

The processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC\_RETURN value to the LR. This indicates which stack pointer corresponds to the stack frame and what operation mode the processor was in before the entry occurred.

If no higher priority exception occurs during exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active.

If another higher priority exception occurs during exception entry, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception. This is the late arrival case.

### 24.2.3.6.6. Exception return

Exception return occurs when the processor is in Handler mode and execution of one of the following instructions attempts to set the PC to an EXC\_RETURN value:

- a POP instruction that loads the PC
- a BX instruction using any register.

The processor saves an EXC\_RETURN value to the LR on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. Bits[31:4] of an EXC\_RETURN value are 0xFFFF FFFF. When the processor loads a value matching this pattern to the PC it detects that the operation is a not a normal branch operation and, instead, that the exception is complete. Therefore, it starts the exception return sequence. Bits[3:0] of the EXC\_RETURN value indicate the required return stack and processor mode, as Table 24-11. Execution return behavior on page 297 shows.

**Table 24-11. Execution return behavior**

EXEC_RETURN	DESCRIPTION
0xFFFF FFF1	Return to Handler mode. Exception return gets state from the main stack. Execution uses MSP after return.
0xFFFF FFF9	Return to Thread mode. Exception return gets state from MSP. Execution uses MSP after return.
0xFFFF FFFD	Return to Thread mode. Exception return gets state from PSP. Execution uses PSP after return.
All other values	Reserved

#### 24.2.4. Fault handling

Faults are a subset of exceptions, see Exception model in chapter 24.2.3 on page 291. All faults result in the HardFault exception being taken or cause lockup if they occur in the NMI or HardFault handler. The faults are:

- execution of an SVC instruction at a priority equal or higher than SVCall
- execution of a BKPT instruction without a debugger attached
- a system-generated bus error on a load or store
- execution of an instruction from an XN memory address
- execution of an instruction from a location for which the system generates a bus fault
- a system-generated bus error on a vector fetch
- execution of an Undefined instruction
- execution of an instruction when not in Thumb-State as a result of the T-bit being previously cleared to 0
- • an attempted load or store to an unaligned address.

#### Note

Only Reset and NMI can preempt the fixed priority HardFault handler. A HardFault can preempt any exception other than Reset, NMI, or another hard fault.

##### 24.2.4.1. Lockup

The processor enters a lockup state if a fault occurs when executing the NMI or HardFault handlers, or if the system generates a bus error when unstacking the PSR on an exception return using the MSP. When the

processor is in lockup state it does not execute any instructions. The processor remains in lockup state until one of the following occurs:

- it is reset
- a debugger halts it
- an NMI occurs and the current lockup is in the HardFault handler.
- 

**Note**

If lockup state occurs in the NMI handler a subsequent NMI does not cause the processor to leave lockup state.

**24.2.5. Power management**

The Cortex-M0 processor sleep modes reduce power consumption:

- a sleep mode, that stops the processor clock
- a deep sleep mode, that stops the system clock and switches off the PLL and flash memory.

The SLEEPDEEP bit of the SCR selects which sleep mode is used, see System Control Register in chapter 24.4.3.5 on page 349.

This section describes the mechanisms for entering sleep mode, and the conditions for waking up from sleep mode.

*24.2.5.1. Entering sleep mode*

This section describes the mechanisms software can use to put the processor into sleep mode.

The system can generate spurious wakeup events, for example a debug operation wakes up the processor. Therefore software must be able to put the processor back into sleep mode after such an event. A program might have an idle loop to put the processor back in to sleep mode.

**24.2.5.1.1. Wait for interrupt**

The Wait For Interrupt instruction, WFI, causes immediate entry to sleep mode. When the processor executes a WFI instruction it stops executing instructions and enters sleep mode. See WFI in chapter 24.3.7.12 on page 338 for more information.

**24.2.5.1.2. Wait for event**

The Wait For Event instruction, WFE, causes entry to sleep mode conditional on the value of a one-bit event register. When the processor executes a WFE instruction, it checks the value of the event register:

**0:** The processor stops executing instructions and enters sleep mode

**1:** The processor sets the register to zero and continues executing instructions without entering sleep mode.

See WFE in chapter 24.3.7.11 on page 337 for more information.

If the event register is 1b, this indicates that the processor must not enter sleep mode on execution of a `WFE` instruction. Typically, this is because of the assertion of an external event, or because another processor in the system has executed a `SEV` instruction, see `SEV` in chapter 24.3.7.9 on page 335. Software cannot access this register directly.

### 24.2.5.1.3. Sleep-on-exit

If the `SLEEPONEXIT` bit of the `SCR` is set to 1, when the processor completes the execution of an exception handler and returns to Thread mode it immediately enters sleep mode. Use this mechanism in applications that only require the processor to run when an interrupt occurs.

#### 24.2.5.2. Wakeup from sleep mode

The conditions for the processor to wakeup depend on the mechanism that caused it to enter sleep mode.

#### 24.2.5.2.1. Wakeup from WFI or sleep-on-exit

Normally, the processor wakes up only when it detects an exception with sufficient priority to cause exception entry.

Some embedded systems might have to execute system restore tasks after the processor wakes up, and before it executes an interrupt handler. To achieve this set the `PRIMASK` bit to 1. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor sets `PRIMASK` to zero. For more information about `PRIMASK`, see Exception mask register in chapter 24.2.1.3.10 on page 284.

#### 24.2.5.2.2. Wakeup from WFE

The processor wakes up if:

- it detects an exception with sufficient priority to cause exception entry.
- it detects an external event signal, see The external event input in chapter 24.2.5.4 on page 300.
- in a multiprocessor system, another processor in the system executes a `SEV` instruction.

In addition, if the `SEVONPEND` bit in the `SCR` is set to 1, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause exception entry. For more information about the `SCR` see System Control Register in chapter 24.4.3.5 on page 349.

#### 24.2.5.3. The Wakeup Interrupt Controller

The Wakeup Interrupt Controller (WIC) is a peripheral that can detect an interrupt and wake the processor from deep sleep mode. The WIC is enabled only when the `DEEPSLEEP` bit in the `SCR` is set to 1b, see System Control Register in chapter 24.4.3.5 on page 349.

The WIC is not programmable, and does not have any registers or user interface. It operates entirely from hardware signals.

When the WIC is enabled and the processor enters deep sleep mode, the power management unit in the system can power down most of the Cortex-M0 processor. This has the side effect of stopping the SysTick timer. When the WIC receives an interrupt, it takes a number of clock cycles to wakeup the processor and restore its state, before it can process the interrupt. This means interrupt latency is increased in deep sleep mode.

#### 24.2.5.4. *The external event input*

The processor provides an external event input signal. This signal is not available on this device.

#### 24.2.5.5. *Power management programming hints*

ISO/IEC C cannot directly generate the WFI, WFE, and SEV instructions. The CMSIS provides the following intrinsic functions for these instructions:

```
void __WFE(void) // Wait for Event
void __WFI(void) // Wait for Interrupt
void __SEV(void) // Send Event
```

## 24.3. The Cortex-M0 Instruction Set

This chapter is the reference material for the Cortex-M0 instruction set description in a User Guide. The following sections give general information:

- Instruction set summary in chapter 24.3.1 on page 300.
- Intrinsic functions in chapter 24.3.2 on page 302.
- About the instruction descriptions in chapter 24.3.3 on page 303.

Each of the following sections describes a functional group of Cortex-M0 instructions.

Together they describe all the instructions supported by the Cortex-M0 processor:

- Memory access instructions in chapter 24.3.4 on page 308.
- General data processing instructions in chapter 24.3.5 on page 315.
- Branch and control instructions in chapter 24.3.6 on page 327.
- Miscellaneous instructions in chapter 24.3.7 on page 329.

### 24.3.1. Instruction set summary

The processor implements a version of the Thumb instruction set. Table 24-12. Cortex-M0 instructions lists the supported instructions.

#### Note



In Table 24-12. Cortex-M0 instructions:

- angle brackets, <>, enclose alternative forms of the operand
- braces, {}, enclose optional operands and mnemonic parts
- the Operands column is not exhaustive.

For more information on the instructions and operands, see the instruction descriptions.

**Table 24-12. Cortex-M0 instructions**

MNEMONIC	OPERANDS	BRIEF DESCRIPTION	FLAGS	CHAPTER, PAGE
ADCS	{Rd,} Rn, Rm	Add with Carry	N, Z, C, V	Chapter 24.3.5.1, page 316
ADD{S}	{Rd,} Rn, <Rm #imm>	Add	N, Z, C, V	Chapter 24.3.5.1, page 316
ADR	Rd, label	PC-relative Address to Register	-	Chapter 24.3.4.1, page 309
ANDS	{Rd,} Rn, Rm	Bitwise AND	N, Z	Chapter 24.3.5.1, page 316
ASRS	{Rd,} Rn, <Rm #imm>	Arithmetic Shift Right	N, Z, C	Chapter 24.3.5.3, page 320
B{cc}	label	Branch {conditionally}	-	Chapter 24.3.6.1, page 328
BICS	{Rd,} Rn, Rm	Bit Clear	N, Z	Chapter 24.3.5.2, page 318
BKPT	#imm	Breakpoint	-	Chapter 24.3.7.1, page 330
BL	label	Branch with Link	-	Chapter 24.3.6.1, page 328
BLX	Rm	Branch indirect with Link	-	Chapter 24.3.6.1, page 328
BX	Rm	Branch indirect	-	Chapter 24.3.6.1, page 328
CMN	Rn, RM	Compare Negative	N, Z, C, V	Chapter 24.3.5.4, page 321
CMP	Rn, <Rm #imm>	Compare	N, Z, C, V	Chapter 24.3.5.4, page 321
CPSID	i	Change Processor State, Disable Interrupts	-	Chapter 24.3.7.2, page 330
CPSIE	i	Change Processor State, Enable Interrupts	-	Chapter 24.3.7.2, page 330
DMB	-	Data Memory Barrier	-	Chapter 24.3.7.3, page 331
DSB	-	Data Synchronization Barrier	-	Chapter 24.3.7.4, page 332
EORS	{Rd,} Rn, Rm	Exclusive OR	N, Z	Chapter 24.3.5.2, page 318
ISB	-	Instruction Synchronization Barrier	-	Chapter 24.3.7.5, page 332
LDM	Rn{!}, reglist	Load Multiple Registers, increment after	-	Chapter 24.3.4.5, page 313
LDR	Rt, label	Load Register from PC-relative Address	-	Chapter 24.3.4.4, page 312
LDR	Rt, [Rn, <Rm #imm>]	Load Register with Word	-	Chapter 24.3.4, page 308
LDRB	Rt, [Rn, <Rm #imm>]	Load Register with Byte	-	Chapter 24.3.4, page 308
LDRH	Rt, [Rn, <Rm #imm>]	Load Register with Half-Word	-	Chapter 24.3.4, page 308
LDRSB	Rt, [Rn, <Rm #imm>]	Load Register with signed Byte	-	Chapter 24.3.4, page 308
LDRSH	Rt, [Rn, <Rm #imm>]	Load Register with signed Half-Word	-	Chapter 24.3.4, page 308
LSLS	{Rd,} Rn, <Rs #imm>	Logical Shift Left	N, Z, C	Chapter 24.3.5.3, page 320
LSRS	{Rd,} Rn, <Rs #imm>	Logical Shift Right	N, Z, C	Chapter 24.3.5.3, page 320
MOV{S}	Rd, Rm	Move	N, Z	Chapter 24.3.5.5, page 322
MRS	Rd, spec_reg	Move to General Register from Special Register	-	Chapter 24.3.7.6, page 333
MSR	spec_reg, Rm	Move to special register from General Register	N, Z, C, V	Chapter 24.3.7.7, page 334

MNEMONIC	OPERANDS	BRIEF DESCRIPTION	FLAGS	CHAPTER, PAGE
MULS	<i>Rd, Rn, Rm</i>	Multiply, 32-bit result	N, Z	Chapter 24.3.5.6, page 323
MVNS	<i>Rd, Rm</i>	Bitwise NOT	N, Z	Chapter 24.3.5.5, page 322
NOP	-	No Operation	-	Chapter 24.3.7.8, page 335
ORRS	<i>{Rd,} Rn, Rm</i>	Logical OR	N, Z	Chapter 24.3.5.2, page 318
POP	<i>reglist</i>	Pop registers from stack	-	Chapter 24.3.4.6, page 314
PUSH	<i>reglist</i>	Push registers onto stack	-	Chapter 24.3.4.6, page 314
REV	<i>Rd, Rm</i>	Byte-Reverse word	-	Chapter 24.3.5.7, page 324
REV16	<i>Rd, Rm</i>	Byte-Reverse packed halfwords	-	Chapter 24.3.5.7, page 324
REVSH	<i>Rd, Rm</i>	Byte-Reverse signed halfword	-	Chapter 24.3.5.7, page 324
RORS	<i>{Rd,} Rn, Rs</i>	Rotate Right	N, Z, C	Chapter 24.3.5.3, page 320
RSBS	<i>{Rd,} Rn, #0</i>	Reverse Subtract	N, Z, C, V	Chapter 24.3.5.1, page 316
SBCS	<i>{Rd,} Rn, Rm</i>	Subtract with Carry	N, Z, C, V	Chapter 24.3.5.1, page 316
SEV	-	Send Event	-	Chapter 24.3.7.9, page 335
STM	<i>Rn!, reglist</i>	Store Multiple Registers, Increment After	-	Chapter 24.3.4.5, page 313
STR	<i>Rt, [Rn, &lt;Rm&gt;#imm]</i>	Store Register as word	-	Chapter 24.3.4, page 308
STRB	<i>Rt, [Rn, &lt;Rm&gt;#imm]</i>	Store Register as byte	-	Chapter 24.3.4, page 308
STRH	<i>Rt, [Rn, &lt;Rm&gt;#imm]</i>	Store Register as half word	-	Chapter 24.3.4, page 308
SUB{S}	<i>Rt, Rn, &lt;Rm&gt;#imm</i>	Subtract	N,Z,C,V	Chapter 24.3.5.1, page 316
SVC	<i>#imm</i>	Supervisor Call	-	Chapter 24.3.7.10, page 336
SXTB	<i>Rd, Rm</i>	Sign extend byte	-	Chapter 24.3.5.8, page 325
SXTH	<i>Rd, Rm</i>	Sign extend half word	-	Chapter 24.3.5.8, page 325
TST	<i>Rd, Rm</i>	Logical AND based test	N, Z	Chapter 24.3.5.9, page 326
UXTB	<i>Rd, Rm</i>	Zero extend a byte	-	Chapter 24.3.5.8, page 325
UXTH	<i>Rd, Rm</i>	Zero extend a halfword	-	Chapter 24.3.5.8, page 325
WFE	-	Wait for Event	-	Chapter 24.3.7.11, page 337
WFI	-	Wait for Interrupt	-	Chapter 24.3.7.12, page 338

### 24.3.2. Intrinsic Functions

ISO/IEC C code cannot directly access some Cortex-M0 instructions. This section describes intrinsic functions that can generate these instructions, provided by the CMSIS and that might be provided by a C compiler. If a C compiler does not support an appropriate intrinsic function, you might have to use inline assembler to access the relevant instruction.

The CMSIS provides the following intrinsic functions to generate instructions that ISO/IEC C code cannot directly access:

**Table 24-13. CMSIS intrinsic functions to generate some Cortex-M0 instructions**

INSTRUCTION	CMSIS INTRINSIC FUNCTION
CPSIE <i>i</i>	<code>void __enable_irq (void)</code>
CPSID <i>i</i>	<code>void __disable_irq (void)</code>
ISB	<code>void __ISB(void)</code>
DSB	<code>void __DSB(void)</code>

INSTRUCTION	CMSIS INTRINSIC FUNCTION
DMB	void __DMB(void)
NOP	void __NOP(void)
REV	uint32_t REV(uint32_t int value)
REV16	uint32_t REV16(uint32_t int value)
REVSH	uint32_t REVSH(uint32_t int value)
SEV	void __SEV(void)
WFE	void __WFE(void)
WFI	void __WFI(void)

The CMSIS also provides a number of functions for accessing the special registers using MRS and MSR instructions:

**Table 24-14. CMSIS intrinsic functions to access special registers**

SPECIAL REGISTER	ACCESS	CMSIS FUNCTION
PRIMASK	Read	uint32_t __get_PRIMASK (void)
	Write	void __set_PRIMASK (uint32_t value)
CONTROL	Read	uint32_t __get_CONTROL (void)
	Write	void __set_CONTROL (uint32_t value)
MSP	Read	uint32_t __get_MSP (void)
	Write	void __set_MSP (uint32_t TopOfMainStack)
PSP	Read	uint32_t __get_PSP (void)
	Write	void __set_PSP (uint32_t TopOfMainStack)

### 24.3.3. About the Instruction Descriptions

The following sections give more information about using the instructions:

- Operands in chapter 24.3.3.1 on page 303
- Restrictions when using PC or SP in chapter 24.3.3.2 on page 304
- Shift Operations in chapter 24.3.3.3 on page 304
- Address alignment in chapter 24.3.3.4 on page 306
- PC-relative expressions in chapter 24.3.3.5 on page 306
- Conditional execution in chapter 24.3.3.6 on page 307

#### 24.3.3.1. Operands

An instruction operand can be an ARM register, a constant, or another instruction-specific parameter. Instructions act on the operands and often store the result in a destination register. When there is a destination register in the instruction, it is usually specified before the other operands.

### 24.3.3.2. Restrictions when using PC or SP

Many instructions are unable to use, or have restrictions on whether you can use, the Program Counter (PC) or Stack Pointer (SP) for the operands or destination register. See instruction descriptions for more information.

#### Note

When you update the PC with a `BX`, `BLX`, or `POP` instruction, bit[0] of any address must be 1 for correct execution. This is because this bit indicates the destination instruction set, and the Cortex-M0 processor only supports Thumb instructions. When a `BL` or `BLX` instruction writes the value of bit[0] into the LR it is automatically assigned the value 1.

### 24.3.3.3. Shift Operations

Register shift operations move the bits in a register left or right by a specified number of bits, the shift length. Register shift can be performed directly by the instructions `ASR`, `LSR`, `LSL`, and `ROR` and the result is written to a destination register.

The permitted shift lengths depend on the shift type and the instruction, see the individual instruction description. If the shift length is 0, no shift occurs. Register shift operations update the carry flag except when the specified shift length is 0. The following sub-sections describe the various shift operations and how they affect the carry flag. In these descriptions, *Rm* is the register containing the value to be shifted, and *n* is the shift length.

#### 24.3.3.3.1. ASR

Arithmetic shift right by *n* bits moves the left-hand 32-*n* bits of the register *Rm*, to the right by *n* places, into the right-hand 32-*n* bits of the result, and it copies the original bit[31] of the register into the left-hand *n* bits of the result. See Figure 24-11. `ASR #3` on page 304.

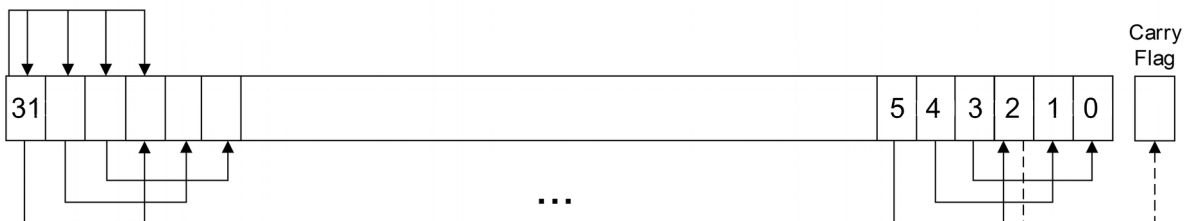
You can use the `ASR` operation to divide the signed value in the register *Rm* by  $2^n$ , with the result being rounded towards negative-infinity.

When the instruction is `ASRS` the carry flag is updated to the last bit shifted out, bit[*n*-1], of the register *Rm*.

#### Note

- If *n* is 32 or more, then all the bits in the result are set to the value of bit[31] of *Rm*.
- If *n* is 32 or more and the carry flag is updated, it is updated to the value of bit[31] of *Rm*.

**Figure 24-11. ASR #3**



### 24.3.3.3.2. LSR

Logical shift right by  $n$  bits moves the left-hand  $32-n$  bits of the register  $Rm$ , to the right by  $n$  places, into the right-hand  $32-n$  bits of the result, and it sets the left-hand  $n$  bits of the result to 0. See Figure 24-12. LSR #3 on page 305.

You can use the LSR operation to divide the value in the register  $Rm$  by  $2^n$ , if the value is regarded as an unsigned integer.

When the instruction is LSRs, the carry flag is updated to the last bit shifted out, bit[ $n-1$ ], of the register  $Rm$ .

**Note**

- If  $n$  is 32 or more, then all the bits in the result are cleared to 0.
- If  $n$  is 33 or more and the carry flag is updated, it is updated to 0.

**Figure 24-12. LSR #3**



### 24.3.3.3.3. LSL

Logical shift left by  $n$  bits moves the right-hand  $32-n$  bits of the register  $Rm$ , to the left by  $n$  places, into the left-hand  $32-n$  bits of the result, and it sets the right-hand  $n$  bits of the result to 0. See Figure 24-13. LSL #3 on page 306.

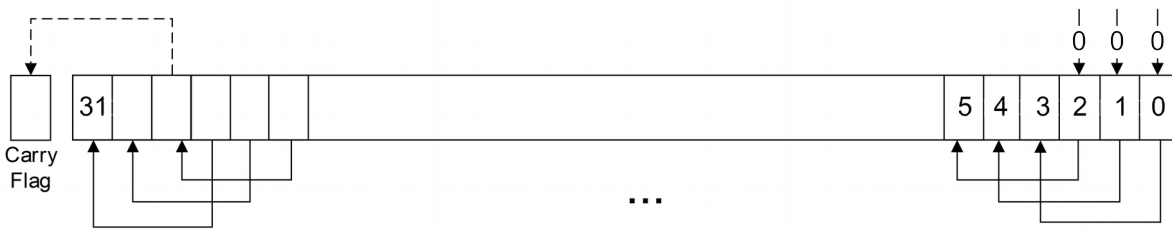
You can use the LSL operation to multiply the value in the register  $Rm$  by  $2^n$ , if the value is regarded as an unsigned integer or a two’s complement signed integer. Overflow can occur without warning.

When the instruction is LSLs the carry flag is updated to the last bit shifted out, bit[ $32-n$ ], of the register  $Rm$ . These instructions do not affect the carry flag when used with LSL #0.

**Note**

- If  $n$  is 32 or more, then all the bits in the result are cleared to 0.
- If  $n$  is 33 or more and the carry flag is updated, it is updated to 0.

**Figure 24-13. LSL #3**



#### 24.3.3.3.4. ROR

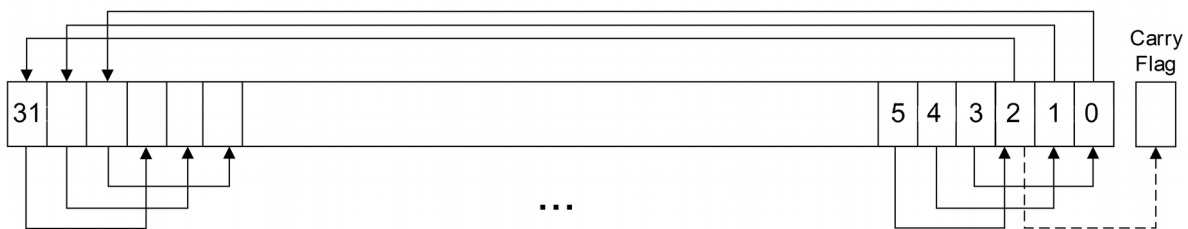
Rotate right by  $n$  bits moves the left-hand  $32-n$  bits of the register  $Rm$ , to the right by  $n$  places, into the right-hand  $32-n$  bits of the result, and it moves the right-hand  $n$  bits of the register into the left-hand  $n$  bits of the result. See Figure 24-14. ROR #3 on page 306.

When the instruction is `RORS` the carry flag is updated to the last bit rotation,  $\text{bit}[n-1]$ , of the register  $Rm$ .

**Note**

- If  $n$  is 32, then the value of the result is same as the value in  $Rm$ , and if the carry flag is updated, it is updated to  $\text{bit}[31]$  of  $Rm$ .
- ROR with shift length,  $n$ , greater than 32 is the same as ROR with shift length  $n-32$ .

**Figure 24-14. ROR #3**



#### 24.3.3.4. Address alignment

An aligned access is an operation where a word-aligned address is used for a word, or multiple word access, or where a halfword-aligned address is used for a halfword access. Byte accesses are always aligned.

There is no support for unaligned accesses on the Cortex-M0 processor. Any attempt to perform an unaligned memory access operation results in a HardFault exception.

#### 24.3.3.5. PC-relative expressions

A PC-relative expression or label is a symbol that represents the address of an instruction or literal data. It is represented in the instruction as the PC value plus or minus a numeric offset. The assembler calculates the required offset from the label and the address of the current instruction. If the offset is too big, the assembler produces an error.

**Note**

- For most instructions, the value of the PC is the address of the current instruction plus 4 bytes.
- Your assembler might permit other syntaxes for PC-relative expressions, such as a label plus or minus a number, or an expression of the form *[PC, #imm]*.

#### 24.3.3.6. Conditional execution

Most data processing instructions update the condition flags in the Application Program Status Register (APSR) according to the result of the operation, see Application Program Status Register in chapter 24.2.1.3.6 on page 282. Some instructions update all flags, and some only update a subset. If a flag is not updated, the original value is preserved. See the instruction descriptions for the flags they affect.

You can execute a conditional branch instruction, based on the condition flags set in another instruction, either:

- immediately after the instruction that updated the flags
- after any number of intervening instructions that have not updated the flags.

On the Cortex-M0 processor, conditional execution is available by using conditional branches.

This section describes:

- The condition flags in chapter 24.3.3.6.1 on page 307
- Condition code suffixes in chapter 24.3.3.6.2 on page 308

#### 24.3.3.6.1. The condition flags

The APSR contains the following condition flags:

- N      Set to 1 when the result of the operation was negative, cleared to 0 otherwise.
- Z      Set to 1 when the result of the operation was zero, cleared to 0 otherwise.
- C      Set to 1 when the operation resulted in a carry, cleared to 0 otherwise.
- V      Set to 1 when the operation caused overflow, cleared to 0 otherwise.

For more information about the APSR see Program Status Register in chapter 24.2.1.3.5 on page 281

A carry occurs:

- if the result of an addition is greater than or equal to  $2^{32}$
- if the result of a subtraction is positive or zero
- as the result of a shift or rotate instruction.

Overflow occurs when the sign of the result, in bit[31], does not match the sign of the result had the operation been performed at infinite precision, for example:

- if adding two negative values results in a positive value
- if adding two positive values results in a negative value
- if subtracting a positive value from a negative value generates a positive value
- if subtracting a negative value from a positive value generates a negative value.

The Compare operations are identical to subtracting, for `CMP`, or adding, for `CMN`, except that the result is discarded. See the instruction descriptions for more information.

### 24.3.3.6.2. Condition code suffixes

Conditional branch is shown in syntax descriptions as *B{cond}*. A branch instruction with a condition code is only taken if the condition code flags in the APSR meet the specified condition, otherwise the branch instruction is ignored. Table 24-15. Condition code suffixes on page 308 shows the condition codes to use.

Table 24-15. Condition code suffixes on page 308 also shows the relationship between condition code suffixes and the N, Z, C, and V flags.

**Table 24-15. Condition code suffixes**

SUFFIX	FLAGS	MEANING
EQ	Z = 1	Equal, last flag setting result was zero
NE	Z = 0	Not equal, last flag setting result was non-zero
CS or HS	C = 1	Higher or same, unsigned
CC or LO	C = 0	Lower, unsigned
MI	N = 1	Negative
PL	N = 0	Positive or zero
VS	V = 1	Overflow
VC	V = 0	No overflow
HI	C = 1 and Z = 0	Higher, unsigned
LS	C = 0 or Z = 1	Lower or same, unsigned
GE	N = V	Greater than or equal, signed
LT	N != V	Less than, signed
GT	Z = 0 and N = V	Greater than, signed
LE	Z = 1 and N != V	Less than or equal, signed
AL	Can have any value	Always. This is the default when no suffix is specified

### 24.3.4. Memory access instructions

Table 24-16. Memory access instructions on page 308 shows the memory access instructions:

**Table 24-16. Memory access instructions**

MNEMONIC	BRIEF DESCRIPTION	SEE
ADR	Generate PC-Relative Address	ADR in chapter 24.3.4.1 on page 309
LDM	Load Multiple Registers	LDM and STM in chapter 24.3.4.5 on page 313
LDR{type}	Load Register using Immediate Offset	LDR and STR, immediate offset in chapter 24.3.4.2 on page 310
LDR{type}	Load Register Using Register Offset	LDR and STR, register offset in chapter 24.3.4.3 on page 311
LDR	Load Register from PC-Relative Address	LDR, PC-relative in chapter 24.3.4.4 on page 312
POP	Pop Registers From Stack	PUSH and POP in chapter 24.3.4.6 on page 314



MNEMONIC	BRIEF DESCRIPTION	SEE
PUSH	Push Registers To Stack	PUSH and POP in chapter 24.3.4.6 on page 314
STM	Store Multiple Registers	LDM and STM in chapter 24.3.4.5 on page 313
STR{type}	Store Register Using Immediate Offset	LDR and STR in chapter 24.3.4.2 on page 310
STR{type}	Store Register Using Register Offset	LDR and STR in chapter 24.3.4.3 on page 311

#### 24.3.4.1. ADR

Generates a PC-relative address.

##### 24.3.4.1.1. Syntax

ADR *Rd*, *label*

where:

*Rd* is the destination register.

*Label* is a PC-relative expression. See PC-relative expressions in chapter 24.3.3.5 on page 306.

##### 24.3.4.1.2. Operation

ADR generates an address by adding an immediate value to the PC, and writes the result to the destination register.

ADR facilitates the generation of position-independent code, because the address is PC-relative.

If you use ADR to generate a target address for a BX or BLX instruction, you must ensure that bit[0] of the address you generate is set to 1 for correct execution.

##### 24.3.4.1.3. Restrictions

In this instruction *Rd* must specify R0-R7. The data-value addressed must be word aligned and within 1020 bytes of the current PC.

##### 24.3.4.1.4. Condition flags

This instruction does not change the flags.

##### 24.3.4.1.5. Examples

```
ADR R1, TextMessage           ; Write address value of a location labeled as
                               ; TextMessage to R1
```

ADR R3, [PC, #996] ; Set R3 to value of PC + 996.

#### 24.3.4.2. LDR and STR, immediate offset

Load and Store with immediate offset.

##### 24.3.4.2.1. Syntax

LDR *Rt*, [<*Rn* | *SP*> {, #*imm*}]

LDR<B|H> *Rt*, [*Rn* {, #*imm*}]

STR *Rt*, [<*Rn* | *SP*>, {, #*imm*}]

STR<B|H> *Rt*, [*Rn* {, #*imm*}]

where:

*Rt* is the register to load or store.

*Rn* is the register on which the memory address is based.

*imm* is an offset from *Rn*. If *imm* is omitted, it is assumed to be zero.

##### 24.3.4.2.2. Operation

LDR, LDRB and LDRH instructions load the register specified by *Rt* with either a word, byte or halfword data value from memory. Sizes less than word are zero extended to 32-bits before being written to the register specified by *Rt*.

STR, STRB and STRH instructions store the word, least-significant byte or lower halfword contained in the single register specified by *Rt* in to memory. The memory address to load from or store to is the sum of the value in the register specified by either *Rn* or *SP* and the immediate value *imm*.

##### 24.3.4.2.3. Restrictions

In these instructions:

- *Rt* and *Rn* must only specify R0-R7.
- *imm* must be between:
  - 0 and 1020 and an integer multiple of four for LDR and STR using SP as the base register
  - 0 and 124 and an integer multiple of four for LDR and STR using R0-R7 as the base register
  - 0 and 62 and an integer multiple of two for LDRH and STRH
  - 0 and 31 for LDRB and STRB
- The computed address must be divisible by the number of bytes in the transaction, see Address alignment in chapter 24.3.3.4 on page 306.

#### 24.3.4.2.4. Condition flags

These instructions do not change the flags.

#### 24.3.4.2.5. Examples

```
LDR R4, [R7] ; Loads R4 from the address in R7.  
STR R2, [R0,#const-struct] ; const-struct is an expression evaluating  
; to a constant in the range 0-1020.
```

#### 24.3.4.3. LDR and STR, register offset

Load and Store with register offset.

##### 24.3.4.3.1. Syntax

```
LDR Rt, [Rn, Rm]  
LDR<B|H> Rt, [Rn, Rm]  
LDR<SB|SH> Rt, [Rn, Rm]  
STR Rt, [Rn, Rm]  
STR<B|H> Rt, [Rn, Rm]
```

where:

*Rt* is the register to load or store.  
*Rn* is the register on which the memory address is based.  
*Rm* is a register containing a value to be used as the offset.

##### 24.3.4.3.2. Operation

LDR, LDRB, LDRH, LDRSB and LDRSH load the register specified by *Rt* with either a word, zero extended byte, zero extended halfword, sign extended byte or sign extended halfword value from memory.

STR, STRB and STRH store the word, least-significant byte or lower halfword contained in the single register specified by *Rt* into memory.

The memory address to load from or store to is the sum of the values in the registers specified by *Rn* and *Rm*.

##### 24.3.4.3.3. Restrictions

In these instructions:

- *Rt*, *Rn*, and *Rm* must only specify R0-R7.

- the computed memory address must be divisible by the number of bytes in the load or store, see Address alignment in chapter 24.3.3.4 on page 306.

#### 24.3.4.3.4. Condition flags

These instructions do not change the flags.

#### 24.3.4.3.5. Examples

```
STR R0, [R5, R1]      ; Store value of R0 into an address equal to
                       ; sum of R5 and R1
LDRSH R1, [R2, R3]    ; Load a halfword from the memory address
                       ; specified by (R2 + R3), sign extend to 32-bits
                       ; and write to R1.
```

#### 24.3.4.4. LDR, PC-relative

Load register (literal) from memory.

##### 24.3.4.4.1. Syntax

```
LDR Rt, label
```

where:

*Rt* is the register to load.

*label* is a PC-relative expression. See PC-relative expressions in chapter 24.3.3.5 on page 306.

##### 24.3.4.4.2. Operation

Loads the register specified by *Rt* from the word in memory specified by *label*.

##### 24.3.4.4.3. Restrictions

In these instructions, *label* must be within 1020 bytes of the current PC and word aligned.

##### 24.3.4.4.4. Condition flags

These instructions do not change the flags.

#### 24.3.4.4.5. Examples

```
LDR R0, LookUpTable           ; Load R0 with a word of data from an address
                               ; labeled as LookUpTable.
LDR R3, [PC, #100]           ; Load R3 with memory word at (PC + 100).
```

#### 24.3.4.5. LDM and STM

Load and Store Multiple registers.

##### 24.3.4.5.1. Syntax

```
LDM Rn{!}, reglist
STM Rn!, reglist
```

where:

*Rn* is the register on which the memory addresses are based.

! writeback suffix.

*reglist* is a list of one or more registers to be loaded or stored, enclosed in braces.

It can contain register ranges. It must be comma separated if it contains more than one register or register range, see Examples in chapter 24.3.4.5.5 on page 314.

LDMIA and LDMFD are synonyms for LDM. LDMIA refers to the base register being Incremented After each access. LDMFD refers to its use for popping data from Full Descending stacks.

STMIA and STMEA are synonyms for STM. STMIA refers to the base register being Incremented After each access. STMEA refers to its use for pushing data onto Empty Ascending stacks.

##### 24.3.4.5.2. Operation

LDM instructions load the registers in *reglist* with word values from memory addresses based on *Rn*.

STM instructions store the word values in the registers in *reglist* to memory addresses based on *Rn*.

The memory addresses used for the accesses are at 4-byte intervals ranging from the value in the register specified by *Rn* to the value in the register specified by  $Rn + 4 * (n-1)$ , where *n* is the number of registers in *reglist*. The accesses happens in order of increasing register numbers, with the lowest numbered register using the lowest memory address and the highest number register using the highest memory address. If the writeback suffix is specified, the value in the register specified by  $Rn + 4 * n$  is written back to the register specified by *Rn*.

### 24.3.4.5.3. Restrictions

In these instructions:

- *reglist* and *Rn* are limited to R0-R7.
- the writeback suffix must always be used unless the instruction is an `LDM` where *reglist* also contains *Rn*, in which case the writeback suffix must not be used.
- the value in the register specified by *Rn* must be word aligned. See Address alignment in chapter 24.3.3.4 on page 306.
- for `STM`, if *Rn* appears in *reglist*, then it must be the first register in the list.

### 24.3.4.5.4. Condition flags

These instructions do not change the flags.

### 24.3.4.5.5. Examples

```
LDM R0, {R0,R3,R4}           ; LDMIA is a synonym for LDM
STMIA R1!, {R2-R4,R6}
```

### 24.3.4.5.6. Incorrect examples

```
STM R5!, {R4,R5,R6}         ; Value stored for R5 is unpredictable
LDM R2, {}                  ; There must be at least one register in the list
```

#### 24.3.4.6. PUSH and POP

Push registers onto, and pop registers off a full-descending stack.

#### 24.3.4.6.1. Syntax

```
PUSH reglist
```

```
POP reglist
```

where:

*reglist* is a non-empty list of registers, enclosed in braces. It can contain register ranges. It must be comma separated if it contains more than one register or register range.

#### 24.3.4.6.2. Operation

`PUSH` stores registers on the stack, with the lowest numbered register using the lowest memory address and the

highest numbered register using the highest memory address.

`POP` loads registers from the stack, with the lowest numbered register using the lowest memory address and the highest numbered register using the highest memory address.

`PUSH` uses the value in the `SP` register minus four as the highest memory address, `POP` uses the value in the `SP` register as the lowest memory address, implementing a full-descending stack. On completion, `PUSH` updates the `SP` register to point to the location of the lowest store value, `POP` updates the `SP` register to point to the location above the highest location loaded.

If a `POP` instruction includes `PC` in its *reglist*, a branch to this location is performed when the `POP` instruction has completed. Bit[0] of the value read for the `PC` is used to update the `APSR T-bit`. This bit must be 1 to ensure correct operation.

#### 24.3.4.6.3. Restrictions

In these instructions:

- *reglist* must use only `R0-R7`.
- The exception is `LR` for a `PUSH` and `PC` for a `POP`.

#### 24.3.4.6.4. Condition flags

These instructions do not change the flags.

#### 24.3.4.6.5. Examples

```

PUSH {R0,R4-R7}           ; Push R0,R4,R5,R6,R7 onto the stack
PUSH {R2,LR}              ; Push R2 and the link-register onto the stack
POP {R0,R6,PC}           ; Pop r0,r6 and PC from the stack, then branch to
                          ; the new PC.
  
```

### 24.3.5. General data processing instructions

Table 24-17. Data processing instructions on page 315 shows the data access instructions:

**Table 24-17. Data processing instructions**

MNEMONIC	BRIEF DESCRIPTION	SEE
ADCS	Add with Carry	ADC, ADD, RSB, SBC, and SUB in chapter 24.3.5.1 on page 316
ADD{S}	Add	ADC, ADD, RSB, SBC, and SUB in chapter 24.3.5.1 on page 316
ANDS	Logical AND	AND, ORR, EOR, and BIC on page
ASRS	Arithmetic Shift Right	ASR, LSL, LSR, and ROR in chapter 24.3.5.3 on page 320

MNEMONIC	BRIEF DESCRIPTION	SEE
BICS	Bit Clear	AND, ORR, EOR, and BIC in chapter 24.3.5.2 on page 318
CMN	Compare Negative	CMP and CMN in chapter 24.3.5.4 on page 321
CMP	Compare	CMP and CMN in chapter 24.3.5.4 on page 321
EORS	Exclusive OR	AND, ORR, EOR, and BIC in chapter 24.3.5.2 on page 318
LSLS	Logical Shift Left	ASR, LSL, LSR, and ROR in chapter 24.3.5.3 on page 320
LSRS	Logical Shift Right	ASR, LSL, LSR, and ROR in chapter 24.3.5.3 on page 320
MOV{S}	Move	MOV and MVN in chapter 24.3.5.5 on page 322
MULS	Multiply	MULS in chapter 24.3.5.6 on page 323
MVNS	Move NOT	MOV and MVN in chapter 24.3.5.5 on page 322
ORRS	Logical OR	AND, ORR, EOR, and BIC in chapter 24.3.5.2 on page 318
REV	Reverse Byte Order In A Word	REV, REV16, and REVSH in chapter 24.3.5.7 on page 324
REV16	Reverse Byte Order In each Half Word	REV, REV16, and REVSH in chapter 24.3.5.7 on page 324
REVSH	Reverse Byte Order In Bottom Half Word and Sign Extend	REV, REV16, and REVSH in chapter 24.3.5.7 on page 324
RORS	Rotate Right	ASR, LSL, LSR, and ROR in chapter 24.3.5.3 on page 320
RSBS	Reverse Subtract	ADC, ADD, RSB, SBC, and SUB in chapter 24.3.5.1 on page 316
SBCS	Subtract with Carry	ADC, ADD, RSB, SBC, and SUB in chapter 24.3.5.1 on page 316
SUBS	Subtract	ADC, ADD, RSB, SBC, and SUB in chapter 24.3.5.1 on page 316
SXTB	Sign Extend A Byte	SXT and UXT in chapter 24.3.5.8 on page 325
SXTH	Sign Extend a Halfword	SXT and UXT in chapter 24.3.5.8 on page 325
UXTB	Zero Extend a Byte	SXT and UXT in chapter 24.3.5.8 on page 325
UXTH	Zero Extend a Halfword	SXT and UXT in chapter 24.3.5.8 on page 325
TST	Test	TST in chapter 24.3.5.9 on page 326

#### 24.3.5.1. ADC, ADD, RSB, SBC, and SUB

Add with carry, Add, Reverse Subtract, Subtract with carry, and Subtract.

##### 24.3.5.1.1. Syntax

ADCS {Rd, } Rn, Rm



ADD{S} {*Rd*, } *Rn*, <*Rm*|#*imm*>

RSBS {*Rd*, } *Rn*, *Rm*, #0

SBCS {*Rd*, } *Rn*, *Rm*

SUB{S} {*Rd*, } *Rn*, <*Rm*|#*imm*>

Where:

*S* causes an ADD or SUB instruction to update flags

*Rd* specifies the result register

*Rn* specifies the first source register

*Rm* specifies the second source register

*imm* specifies a constant immediate value.

When the optional *Rd* register specifier is omitted, it is assumed to take the same value as *Rn*, for example ADDS *R1*,*R2* is identical to ADDS *R1*,*R1*,*R2*.

### 24.3.5.1.2. Operation

The ADCS instruction adds the value in *Rn* to the value in *Rm*, adding a further one if the carry flag is set, places the result in the register specified by *Rd* and updates the N, Z, C, and V flags.

The ADD instruction adds the value in *Rn* to the value in *Rm* or an immediate value specified by *imm* and places the result in the register specified by *Rd*.

The ADDS instruction performs the same operation as ADD and also updates the N, Z, C and V flags.

The RSBS instruction subtracts the value in *Rn* from zero, producing the arithmetic negative of the value, and places the result in the register specified by *Rd* and updates the N, Z, C and V flags.

The SBCS instruction subtracts the value of *Rm* from the value in *Rn*, deducts a further one if the carry flag is set. It places the result in the register specified by *Rd* and updates the N, Z, C and V flags.

The SUB instruction subtracts the value in *Rm* or the immediate specified by *imm*. It places the result in the register specified by *Rd*.

The SUBS instruction performs the same operation as SUB and also updates the N, Z, C and V flags.

Use ADC and SBC to synthesize multiword arithmetic, see Examples in chapter 24.3.5.1.4 on page 318.

See also ADR in chapter 24.3.4.1 on page 309.

### 24.3.5.1.3. Restrictions

Table 24-18. ADC, ADD, RSB, SBC, and SUB operand restrictions on page 317 lists the legal combinations of register specifiers and immediate values that can be used with each instruction.

#### Table 24-18. ADC, ADD, RSB, SBC, and SUB operand restrictions

INSTRUCTION	RD	Rn	Rm	imm	RESTRICTIONS
ADCS	R0-R7	R0-R7	R0-R7	-	<i>Rd</i> and <i>Rn</i> must specify the same register
ADD	R0-R15	R0-R15	R0-PC	-	<i>Rd</i> and <i>Rn</i> must specify the same register <i>Rn</i> and <i>Rm</i> must not both specify PC
	R0-R7	SP or PC	-	0-1020	Immediate value must be an integer multiple of four
	SP	SP	-	0-508	Immediate value must be an integer multiple of four
ADDS	R0-R7	R0-R7	-	0-7	-
	R0-R7	R0-R7	-	0-255	<i>Rd</i> and <i>Rn</i> must specify the same register
	R0-R7	R0-R7	R0-R7	-	-
RSBS	R0-R7	R0-R7	-	-	-
SBCS	R0-R7	R0-R7	R0-R7	-	<i>Rd</i> and <i>Rn</i> must specify the same register
SUB	SP	SP	-	0-508	Immediate value must be an integer multiple of four
SUBS	R0-R7	R0-R7	-	0-7	-
	R0-R7	R0-R7	-	0-255	<i>Rd</i> and <i>Rn</i> must specify the same register
	R0-R7	R0-R7	R0-R7	-	-

#### 24.3.5.1.4. Examples

Example below shows two instructions that add a 64-bit integer contained in R0 and R1 to another 64-bit integer contained in R2 and R3, and place the result in R0 and R1.

```
ADDS R0, R0, R2      ; add the least significant words
ADCS R1, R1, R3      ; add the most significant words with carry
```

Multiword values do not have to use consecutive registers. Example below shows instructions that subtract a 96-bit integer contained in R1, R2, and R3 from another contained in R4, R5, and R6. The example stores the result in R4, R5, and R6.

```
SUBS R4, R4, R1      ; subtract the least significant words
SBCS R5, R5, R2      ; subtract the middle words with carry
SBCS R6, R6, R3      ; subtract the most significant words with carry
```

Example below the RSBS instruction used to perform a 1's complement of a single register.

```
RSBS R7, R7, #0      ; subtract R7 from zero
```

#### 24.3.5.2. AND, ORR, EOR, and BIC

Logical AND, OR, Exclusive OR, and Bit Clear.

#### 24.3.5.2.1. Syntax

ANDS {*Rd*, } *Rn*, *Rm*

ORRS {*Rd*, } *Rn*, *Rm*

EORS {*Rd*, } *Rn*, *Rm*

BICS {*Rd*, } *Rn*, *Rm*

where:

*Rd* is the destination register.

*Rn* is the register holding the first operand and is the same as the destination register.

*Rm* second register.

#### 24.3.5.2.2. Operation

The AND, EOR, and ORR instructions perform bitwise AND, exclusive OR, and inclusive OR operations on the values in *Rn* and *Rm*.

The BIC instruction performs an AND operation on the bits in *Rn* with the logical negation of the corresponding bits in the value of *Rm*.

The condition code flags are updated on the result of the operation, see Condition flags in chapter 24.3.4.6.4 on page 315.

#### 24.3.5.2.3. Restrictions

In these instructions, *Rd*, *Rn*, and *Rm* must only specify R0-R7.

#### 24.3.5.2.4. Condition flags

These instructions:

- update the N and Z flags according to the result
- do not affect the C or V flag.

#### 24.3.5.2.5. Examples

ANDS R2, R2, R1

ORRS R2, R2, R5

ANDS R5, R5, R8

EORS R7, R7, R6

BICS R0, R0, R1

### 24.3.5.3. ASR, LSL, LSR, and ROR

Arithmetic Shift Right, Logical Shift Left, Logical Shift Right, and Rotate Right.

#### 24.3.5.3.1. Syntax

ASRS {*Rd*, } *Rm*, *Rs*

ASRS {*Rd*, } *Rm*, #*imm*

LSLS {*Rd*, } *Rm*, *Rs*

LSLS {*Rd*, } *Rm*, #*imm*

LSRS {*Rd*, } *Rm*, *Rs*

LSRS {*Rd*, } *Rm*, #*imm*

RORS {*Rd*, } *Rm*, *Rs*

where:

*Rd* is the destination register. If *Rd* is omitted, it is assumed to take the same value as *Rm*.

*Rm* is the register holding the value to be shifted.

*Rs* is the register holding the shift length to apply to the value in *Rm*.

*imm* is the shift length. The range of shift length depends on the instruction:

ASR shift length from 1 to 32

LSL shift length from 0 to 31

LSR shift length from 1 to 32.

#### Note

MOVS *Rd*, *Rm* is a pseudonym for LSLS *Rd*, *Rm*, #0.

#### 24.3.5.3.2. Operation

ASR, LSL, LSR, and ROR perform an arithmetic-shift-left, logical-shift-left, logical-shift-right or a right-rotation of the bits in the register *Rm* by the number of places specified by the immediate *imm* or the value in the least-significant byte of the register specified by *Rs*.

For details on what result is generated by the different instructions, see Shift Operations in chapter 24.3.3.3 on page 304.

### 24.3.5.3.3. Restrictions

In these instructions, *Rd*, *Rm*, and *Rs* must only specify R0-R7. For non-immediate instructions, *Rd* and *Rm* must specify the same register.

### 24.3.5.3.4. Condition flags

These instructions update the N and Z flags according to the result.

The C flag is updated to the last bit shifted out, except when the shift length is 0, see Shift Operations in chapter 24.3.3.3 on page 304. The V flag is left unmodified.

### 24.3.5.3.5. Examples

```
ASRS R7, R5, #9      ; Arithmetic shift right by 9 bits
LSLS R1, R2, #3      ; Logical shift left by 3 bits with flag update
LSRS R4, R5, #6      ; Logical shift right by 6 bits
RORS R4, R4, R6      ; Rotate right by the value in the bottom byte of R6.
```

### 24.3.5.4. CMP and CMN

Compare and Compare Negative.

#### 24.3.5.4.1. Syntax

```
CMN Rn, Rm
```

```
CMP Rn, #imm
```

```
CMP Rn, Rm
```

where:

*Rn* is the register holding the first operand.

*Rm* is the register to compare with.

*imm* is the immediate value to compare with.

#### 24.3.5.4.2. Operation

These instructions compare the value in a register with either the value in another register or an immediate value. They update the condition flags on the result, but do not write the result to a register.

The `CMP` instruction subtracts either the value in the register specified by *Rm*, or the immediate *imm* from the value in *Rn* and updates the flags. This is the same as a `SUBS` instruction, except that the result is discarded.

The `CMN` instruction adds the value of *Rm* to the value in *Rn* and updates the flags. This is the same as an `ADDS`

instruction, except that the result is discarded.

#### 24.3.5.4.3. Restrictions

For the:

- CMN instruction *Rn*, and *Rm* must only specify R0-R7.
- CMP instruction:
  - *Rn* and *Rm* can specify R0-R14
  - immediate must be in the range 0-255.

#### 24.3.5.4.4. Condition flags

These instructions update the N, Z, C and V flags according to the result

#### 24.3.5.4.5. Examples

CMP R2, R9

CMN R0, R2

#### 24.3.5.5. MOV and MVN

Move and Move NOT.

#### 24.3.5.5.1. Syntax

MOV{S} *Rd*, *Rm*

MOVS *Rd*, #*imm*

MVNS *Rd*, *Rm*

where:

*S* is an optional suffix. If *S* is specified, the condition code flags are updated on the result of the operation, see Conditional execution in chapter 24.3.3.6 on page 307.

*Rd* is the destination register.

*Rm* is a register.

*imm* is any value in the range 0-255.

### 24.3.5.5.2. Operation

The `MOV` instruction copies the value of *Rm* into *Rd*.

The `MOVS` instruction performs the same operation as the `MOV` instruction, but also updates the N and Z flags.

The `MVNS` instruction takes the value of *Rm*, performs a bitwise logical negate operation on the value, and places the result into *Rd*.

### 24.3.5.5.3. Restrictions

In these instructions, *Rd*, and *Rm* must only specify R0-R7.

When *Rd* is the PC in a `MOV` instruction:

- Bit[0] of the result is discarded.
- A branch occurs to the address created by forcing bit[0] of the result to 0. The T-bit remains unmodified.
- 

#### Note

Though it is possible to use `MOV` as a branch instruction, ARM strongly recommends the use of a `BX` or `BLX` instruction to branch for software portability.

### 24.3.5.5.4. Condition flags

If S is specified, these instructions:

- update the N and Z flags according to the result
- do not affect the C or V flags.

### 24.3.5.5.5. Example

```
MOVS R0, #0x000B      ; Write value of 0x000B to R0, flags get updated
MOVS R1, #0x0         ; Write value of zero to R1, flags are updated
MOV R10, R12          ; Write value in R12 to R10, flags are not updated
MOVS R3, #23         ; Write value of 23 to R3
MOV R8, SP            ; Write value of stack pointer to R8
MVNS R2, R0          ; Write inverse of R0 to the R2 and update flags
```

### 24.3.5.6. MULS

Multiply using 32-bit operands, and producing a 32-bit result.

### 24.3.5.6.1. Syntax

`MULS Rd, Rn, Rm`

where:

*Rd* is the destination register.

*Rn, Rm* are registers holding the values to be multiplied.

### 24.3.5.6.2. Operation

The `MUL` instruction multiplies the values in the registers specified by *Rn* and *Rm*, and places the least significant 32 bits of the result in *Rd*. The condition code flags are updated on the result of the operation, see Conditional execution in chapter 24.3.3.6 on page 307.

The results of this instruction does not depend on whether the operands are signed or unsigned.

### 24.3.5.6.3. Restrictions

In this instruction:

- *Rd, Rn, and Rm* must only specify R0-R7
- *Rd* must be the same as *Rm*.

### 24.3.5.6.4. Condition flags

This instruction:

- updates the N and Z flags according to the result
- does not affect the C or V flags.

### 24.3.5.6.5. Examples

```
MULS R0, R2, R0 ; Multiply with flag update, R0 = R0 x R2
```

### 24.3.5.7. REV, REV16, and REVSH

Reverse bytes.

### 24.3.5.7.1. Syntax

`REV Rd, Rn`

`REV16 Rd, Rn`



REVSH *Rd, Rn*

where:

*Rd* is the destination register.

*Rn* is the source register.

### 24.3.5.7.2. Operation

Use these instructions to change endianness of data:

REV converts 32-bit big-endian data into little-endian data or 32-bit little-endian data into big-endian data.

REV16 converts two packed 16-bit big-endian data into little-endian data or two packed 16-bit little-endian data into big-endian data.

REVSH converts 16-bit signed big-endian data into 32-bit signed little-endian data or 16-bit signed little-endian data into 32-bit signed big-endian data.

### 24.3.5.7.3. Restrictions

In these instructions, *Rd*, and *Rn* must only specify R0-R7.

### 24.3.5.7.4. Condition flags

These instructions do not change the flags.

### 24.3.5.7.5. Examples

```
REV R3, R7           ; Reverse byte order of value in R7 and write it to R3
REV16 R0, R0         ; Reverse byte order of each 16-bit halfword in R0
REVSH R0, R5         ; Reverse signed halfword
```

### 24.3.5.8. SXT and UXT

Sign extend and Zero extend.

#### 24.3.5.8.1. Syntax

SXTB *Rd, Rm*

SXTH *Rd, Rm*

UXTB *Rd, Rm*

UXTH *Rd*, *Rm*

where:

*Rd* is the destination register.

*Rm* is the register holding the value to be extended.

### 24.3.5.8.2. Operation

These instructions extract bits from the resulting value:

- SXTB extracts bits[7:0] and sign extends to 32 bits
- UXTB extracts bits[7:0] and zero extends to 32 bits
- SXTH extracts bits[15:0] and sign extends to 32 bits
- UXTH extracts bits[15:0] and zero extends to 32 bits.

### 24.3.5.8.3. Restrictions

In these instructions, *Rd* and *Rm* must only specify R0-R7.

### 24.3.5.8.4. Condition flags

These instructions do not affect the flags.

### 24.3.5.8.5. Examples

```
SXTH R4, R6           ; Obtain the lower halfword of the
                       ; value in R6 and then sign extend to
                       ; 32 bits and write the result to R4.
UXTB R3, R1           ; Extract lowest byte of the value in R10 and zero
                       ; extend it, and write the result to R3
```

### 24.3.5.9. TST

Test bits.

### 24.3.5.9.1. Syntax

TST *Rn*, *Rm*

where:

*Rn* is the register holding the first operand.

*Rm* the register to test against.

### 24.3.5.9.2. Operation

This instruction tests the value in a register against another register. It updates the condition flags based on the result, but does not write the result to a register.

The `TST` instruction performs a bitwise AND operation on the value in *Rn* and the value in *Rm*. This is the same as the `ANDS` instruction, except that it discards the result.

To test whether a bit of *Rn* is 0 or 1, use the `TST` instruction with a register that has that bit set to 1 and all other bits cleared to 0.

### 24.3.5.9.3. Restrictions

In these instructions, *Rn* and *Rm* must only specify R0-R7.

### 24.3.5.9.4. Condition flags

This instruction:

- updates the N and Z flags according to the result
- does not affect the C or V flags.

### 24.3.5.9.5. Examples

```
TST R0, R1           ; Perform bitwise AND of R0 value and R1 value,
                    ; condition code flags are updated but result is discarded
```

## 24.3.6. Branch and control instructions

Table 24-19. Branch and Control instructions on page 327 shows the branch and control instructions:

**Table 24-19. Branch and Control instructions**

MNEMONIC	BRIEF DESCRIPTION	SEE
B{CC}	Branch {conditionally}	B, BL, BX, and BLX in chapter 24.3.6.1 on page 328
BL	Branch with Link	B, BL, BX, and BLX in chapter 24.3.6.1 on page 328
BLX	Branch indirect with Link	B, BL, BX, and BLX in chapter 24.3.6.1 on page 328
BX	Branch indirect	B, BL, BX, and BLX in chapter 24.3.6.1 on page 328

### 24.3.6.1. B, BL, BX, and BLX

Branch instructions.

#### 24.3.6.1.1. Syntax

$B\{cond\}$  *label*

BL *label*

BX *Rm*

BLX *Rm*

where:

*cond* is an optional condition code, see Conditional execution in chapter 24.3.3.6 on page 307.

*label* is a PC-relative expression. See PC-relative expressions in chapter 24.3.3.5 on page 306.

*Rm* is a register providing the address to branch to.

#### 24.3.6.1.2. Operation

All these instructions cause a branch to the address indicated by label or contained in the register specified by *Rm*. In addition:

- The BL and BLX instructions write the address of the next instruction to LR, the link register R14.
- The BX and BLX instructions result in a HardFault exception if bit[0] of *Rm* is 0.

BL and BLX instructions also set bit[0] of the LR to 1. This ensures that the value is suitable for use by a subsequent POP {PC} or BX instruction to perform a successful return branch.

Table 24-20. Branch ranges on page 328 shows the ranges for the various branch instructions.

**Table 24-20. Branch ranges**

INSTRUCTION	BRANCH RANGE
B label	-2KB to + 2KB
Bcond label	-256bytes to +254 bytes
BL label	-16MB to +16MB
BX <i>Rm</i>	Any value in register
BLX <i>Rm</i>	Any value in register

#### 24.3.6.1.3. Restrictions

In these instructions:

- Do not use SP or PC in the BX or BLX instruction.
- For BX and BLX, bit[0] of *Rm* must be 1 for correct execution. Bit[0] is used to update the EPSR T-bit

and is discarded from the target address.

•

### Note

BCOND is the only conditional instruction on the Cortex-M0 processor.

### Condition flags

These instructions do not change the flags.

### Examples

```

B loopA                ; Branch to loopA
BL funcC               ; Branch with link (Call) to function funcC, return address
                       ; stored in LR
BX LR                 ; Return from function call
BLX R0                ; Branch with link and exchange (Call) to an address stored
                       ; in R0
BEQ labelD            ; Conditionally branch to labelD if last flag setting
                       ; instruction set the Z flag, else do not branch.
  
```

### 24.3.7. Miscellaneous instructions

Table 24-21. Miscellaneous instructions on page 329 shows the remaining Cortex-M0 instructions:

**Table 24-21. Miscellaneous instructions**

MNEMONIC	BRIEF DESCRIPTION	SEE
BKPT	Breakpoint	BKPT in chapter 24.3.7.1 on page 330
CPSID	Change Processor State, Diable Interrupts	CPS in chapter 24.3.7.2 on page 330
CPSIE	Change Processor State, Enable Interrupts	CPS in chapter 24.3.7.2 on page 330
DMB	Data Memory Barrier	DMB in chapter 24.3.7.3 on page 331
DSB	Data Synchronization Barrier	DSB in chapter 24.3.7.4 on page 332
ISB	Instruction Synchronization Barrier	ISB in chapter 24.3.7.5 on page 332
MRS	Move from special register to register	MRS in chapter 24.3.7.6 on page 333
MSR	Move form register to special register	MSR in chapter 24.3.7.7 on page 334
NOP	No operation	NOP in chapter 24.3.7.8 on page 335
SEV	Send Event	SEV in chapter 24.3.7.9 on page 335
SVC	Supervisor Call	SVC in chapter 24.3.7.10 on page 336
WFE	Wait for Event	WFE in chapter 24.3.7.11 on page 337
WFI	Wait for interrupt	WFI in chapter 24.3.7.12 on page 338

#### 24.3.7.1. BKPT

Breakpoint.

##### 24.3.7.1.1. Syntax

BKPT #*imm*

where:

*imm* is an integer in the range 0-255.

##### 24.3.7.1.2. Operation

The BKPT instruction causes the processor to enter Debug state. Debug tools can use this to investigate system state when the instruction at a particular address is reached.

*imm* is ignored by the processor. If required, a debugger can use it to store additional information about the breakpoint.

The processor might also produce a HardFault or go in to lockup if a debugger is not attached when a BKPT instruction is executed. See Lockup in chapter 24.2.4.1 on page 297 for more information.

##### 24.3.7.1.3. Restrictions

There are no restrictions.

##### 24.3.7.1.4. Condition flags

This instruction does not change the flags.

##### 24.3.7.1.5. Examples

```
BKPT #0 ; Breakpoint with immediate value set to 0x0.
```

#### 24.3.7.2. CPS

Change Processor State.

##### 24.3.7.2.1. Syntax

CPSID *i*

CPSIE *i*

#### 24.3.7.2.2. Operation

CPS changes the PRIMASK special register values. CPSID causes interrupts to be disabled by setting PRIMASK. CPSIE cause interrupts to be enabled by clearing PRIMASK. See Exception mask register in chapter 24.2.1.3.10 on page 284 for more information about these registers.

#### 24.3.7.2.3. Restrictions

There are no restrictions.

#### 24.3.7.2.4. Condition flags

This instruction does not change the condition flags.

#### 24.3.7.2.5. Examples

```
CPSID i           ; Disable all interrupts except NMI (set PRIMASK)
CPSIE i           ; Enable interrupts (clear PRIMASK)
```

#### 24.3.7.3. DMB

Data Memory Barrier.

##### 24.3.7.3.1. Syntax

DMB

##### 24.3.7.3.2. Operation

DMB acts as a data memory barrier. It ensures that all explicit memory accesses that appear in program order before the DMB instruction are observed before any explicit memory accesses that appear in program order after the DMB instruction. DMB does not affect the ordering of instructions that do not access memory.

##### 24.3.7.3.3. Restrictions

There are no restrictions.

#### 24.3.7.3.4. Condition flags

This instruction does not change the flags.

#### 24.3.7.3.5. Examples

DMB ; Data Memory Barrier

#### 24.3.7.4. DSB

Data Synchronization Barrier.

#### 24.3.7.4.1. Syntax

DSB

#### 24.3.7.4.2. Operation

DSB acts as a special data synchronization memory barrier. Instructions that come after the DSB, in program order, do not execute until the DSB instruction completes. The DSB instruction completes when all explicit memory accesses before it complete.

#### 24.3.7.4.3. Restrictions

There are no restrictions.

#### 24.3.7.4.4. Condition flags

This instruction does not change the flags.

#### 24.3.7.4.5. Examples

DSB ; Data Synchronisation Barrier

#### 24.3.7.5. ISB

Instruction Synchronization Barrier.



### 24.3.7.5.1. Syntax

ISB

### 24.3.7.5.2. Operation

ISB acts as an instruction synchronization barrier. It flushes the pipeline of the processor, so that all instructions following the ISB are fetched from cache or memory again, after the ISB instruction has been completed.

### 24.3.7.5.3. Restrictions

There are no restrictions.

### 24.3.7.5.4. Condition flags

This instruction does not change the flags.

### 24.3.7.5.5. Examples

```
ISB ; Instruction Synchronisation Barrier
```

### 24.3.7.6. MRS

Move the contents of a special register to a general-purpose register.

#### 24.3.7.6.1. Syntax

```
MRS Rd, spec_reg
```

where:

*Rd* is the general-purpose destination register.

*spec\_reg* is one of the special-purpose registers: APSR, IPSR, EPSR, IEPSR, IAPSR, EAPSR, PSR, MSP, PSP, PRIMASK, or CONTROL.

#### 24.3.7.6.2. Operation

MRS stores the contents of a special-purpose register to a general-purpose register. The MRS instruction can be combined with the MSR instruction to produce read-modify-write sequences, which are suitable for modifying a specific flag in the PSR.

See MSR in chapter 24.3.7.7 on page 334.

### 24.3.7.6.3. Restrictions

In this instruction, *Rd* must not be SP or PC.

### 24.3.7.6.4. Condition flags

This instruction does not change the flags.

### 24.3.7.6.5. Examples

```
MRS R0, PRIMASK ; Read PRIMASK value and write it to R0
```

### 24.3.7.7. MSR

Move the contents of a general-purpose register into the specified special register.

#### 24.3.7.7.1. Syntax

```
MSR spec_reg, Rn
```

where:

*Rn* is the general-purpose source register.

*spec\_reg* is the special-purpose destination register: APSR, IPSR, EPSR, IEPSR, IAPSR, EAPSR, PSR, MSP, PSP, PRIMASK, or CONTROL.

#### 24.3.7.7.2. Operation

MSR updates one of the special registers with the value from the register specified by *Rn*.

See MRSON 24.3.7.6 on page 333.

#### 24.3.7.7.3. Restrictions

In this instruction, *Rn* must not be SP and must not be PC.

#### 24.3.7.7.4. Condition flags

This instruction updates the flags explicitly based on the value in *Rn*.

### 24.3.7.7.5. Examples

`MSR CONTROL, R1` ; Read R1 value and write it to the CONTROL register

### 24.3.7.8. NOP

No Operation.

#### 24.3.7.8.1. Syntax

`NOP`

#### 24.3.7.8.2. Operation

`NOP` performs no operation and is not guaranteed to be time consuming. The processor might remove it from the pipeline before it reaches the execution stage. Use `NOP` for padding, for example to place the subsequent instructions on a 64-bit boundary.

#### 24.3.7.8.3. Restrictions

There are no restrictions.

#### 24.3.7.8.4. Condition flags

This instruction does not change the flags.

#### 24.3.7.8.5. Examples

`NOP` ; No operation

### 24.3.7.9. SEV

Send Event.

#### 24.3.7.9.1. Syntax

`SEV`

#### 24.3.7.9.2. Operation

`SEV` causes an event to be signaled to all processors within a multiprocessor system. It also sets the local event register, see Power management in chapter 24.2.5 on page 298.

See also `WFE` in in chapter 24.3.7.11 on page 337.

#### 24.3.7.9.3. Restrictions

There are no restrictions.

#### 24.3.7.9.4. Condition flags

This instruction does not change the flags.

#### 24.3.7.9.5. Examples

```
SEV                                ; Send Event
```

#### 24.3.7.10. SVC

Supervisor Call.

##### 24.3.7.10.1. Syntax

```
SVC #imm
```

where:

*imm* is an integer in the range 0-255.

##### 24.3.7.10.2. Operation

The `svc` instruction causes the SVC exception.

*imm* is ignored by the processor. If required, it can be retrieved by the exception handler to determine what service is being requested.

##### 24.3.7.10.3. Restrictions

There are no restrictions.

#### 24.3.7.10.4. Condition flags

This instruction does not change the flags.

#### 24.3.7.10.5. Examples

```
SVC #0x32          ; Supervisor Call (SVC handler can extract the immediate value  
                  ; by locating it via the stacked PC)
```

#### 24.3.7.11. WFE

Wait For Event.

#### 24.3.7.11.1. Syntax

```
WFE
```

#### 24.3.7.11.2. Operation

If the event register is 0, `WFE` suspends execution until one of the following events occurs:

- an exception, unless masked by the exception mask registers or the current priority level
- an exception enters the Pending state, if `SEVONPEND` in the System Control Register is set
- a Debug Entry request, if debug is enabled
- an event signaled by a peripheral or another processor in a multiprocessor system using the `SEV` instruction.

If the event register is 1, `WFE` clears it to 0 and completes immediately.

For more information see see Power management in chapter 24.2.5 on page 298.

#### Note

`WFE` is intended for power saving only. When writing software assume that `WFE` might behave as `NOP`.

#### 24.3.7.11.3. Restrictions

There are no restrictions.

#### 24.3.7.11.4. Condition flags

This instruction does not change the flags.

#### 24.3.7.11.5. Examples

```
WFE                ; Wait for event
```

#### 24.3.7.12. WFI

Wait for Interrupt.

#### 24.3.7.12.1. Syntax

```
WFI
```

#### 24.3.7.12.2. Operation

WFI suspends execution until one of the following events occurs:

- an exception
- an interrupt becomes pending which would preempt if PRIMASK was clear
- a Debug Entry request, regardless of whether debug is enabled.
- 

#### Note

WFI is intended for power saving only. When writing software assume that WFI might behave as a NOP operation.

#### 24.3.7.12.3. Restrictions

There are no restrictions.

#### 24.3.7.12.4. Condition flags

This instruction does not change the flags.

#### 24.3.7.12.5. Examples

```
WFI                ; Wait for interrupt
```

## 24.4. Cortex-M0 Peripherals

The following sections are the reference material for the ARM Cortex-M0 core peripherals descriptions in a User Guide:

- About the Cortex-M0 peripherals in chapter 24.4.1 on page 339
- Nested Vectored Interrupt Controller in chapter 24.4.2 on page 339
- System Control Block in chapter 24.4.3 on page 345
- System timer, SysTick in chapter 24.4.4 on page 352

### 24.4.1. About the Cortex-M0 peripherals

The address map of the Private peripheral bus (PPB) is:

**Table 24-22. Core peripheral register regions**

ADDRESS	CORE PERIPHERAL	DESCRIPTION
0xE000 E008 – 0xE000 E00F	System Control Block	Table 24-31. Summary of the SCB register on page 345
0xE000 E010 – 0xE000 E01F	System Timer	Table 24-40. System timer register summary on page 352
0xE000 E100 – 0xE000 E4EF	Nested Vectored Interrupt Controller	Table 24-23. NVIC register summary on page 339
0xE000 ED00 – 0xE000 ED3F	System Control Block	Table 24-31. Summary of the SCB register on page 345
0xE000 EF00 – 0xE000 EF03	Nested Vectored Interrupt Controller	Table 24-23. NVIC register summary on page 339

In register descriptions, the register type is described as follows:

RW Read and write.

RO Read-only.

WO Write-only.

### 24.4.2. Nested Vectored Interrupt Controller

This section describes the Nested Vectored Interrupt Controller (NVIC) and the registers it uses. The NVIC supports:

- 1 to 32 interrupts.
- A programmable priority level of 0-192 in steps of 64 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Level and pulse detection of interrupt signals.
- Interrupt tail-chaining.
- An external Non-maskable interrupt (NMI).

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead. This provides low latency exception handling. The hardware implementation of the NVIC registers is:

**Table 24-23. NVIC register summary**

ADDRESS	NAME	TYPE	RESET VALUE	DESCRIPTION
0xE000 E100	ISER	RW	0x0000 0000	Interrupt Set-enable Register in chapter 24.4.2.2 on page 340
0xE000 E180	ICER	RW	0x0000 0000	Interrupt Clear-enable Register in chapter 24.4.2.3 on page 341
0xE000 E200	ISPR	RW	0x0000 0000	Interrupt Set-pending Register in chapter 24.4.2.4 on page 341
0xE000 E280	ICPR	RW	0x0000 0000	Interrupt Clear-pending Register in chapter 24.4.2.5 on page 342
0xE000 E400 – 0xE000 E41C	IPR0-7	RW	0x0000 0000	Interrupt Priority Registers in chapter 24.4.2.6 on page 342

#### 24.4.2.1. Accessing the Cortex-M0 NVIC registers using CMSIS

CMSIS functions enable software portability between different Cortex-M profile processors.

To access the NVIC registers when using CMSIS, use the following functions:

**Table 24-24. CMSIS access NVIC functions**

CMSIS FUNCTION	DESCRIPTION
<code>void NVIC_EnableIRQ(IRQn_Type IRQn) *</code>	Enables an interrupt or exception
<code>void NVIC_DisableIRQ(IRQn_Type IRQn) *</code>	Disables an interrupt or exception
<code>void NVIC_SetPendingIRQ(IRQn_Type IRQn) *</code>	Set the pending status of interrupt or exception to 1
<code>void NVIC_ClearPendingIRQ(IRQn_Type IRQn) *</code>	Clears the pending status of interrupt or exception to 0
<code>uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn) *</code>	Reads the pending status of interrupt or exception. This function returns non-zero value if the pending status is set to 1.
<code>void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority) *</code>	Sets the priority of an interrupt or exception with configurable priority level to 1
<code>uint32_t NVIC_GetPriority(IRQn_Type IRQn) *</code>	Reads the priority of an interrupt or exception with configurable priority level. This function returns the current priority level

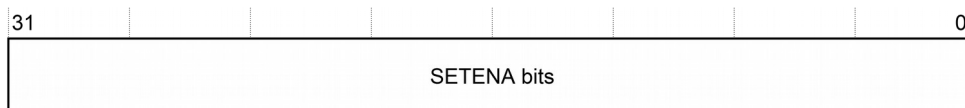
\* The input parameter IRQn is the IRQ number see Table 24-10. Properties of the different exception types on page 292 for more information

#### 24.4.2.2. Interrupt Set-enable Register

The ISER enables interrupts, and shows which interrupts are enabled. See the register summary in Table 24-23. NVIC register summary on page 339 for the register attributes.

The bit assignments are:

**Figure 24-15. ISER**



**Table 24-25. ISER bit assignments**



BIT	NAME	FUNCTION
31:0	<b>SETENA</b>	Interrupt set-enable bits. Write: 1: enable interrupt 0: no effect Read: 1: interrupt enabled 0: interrupt disabled

If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority.

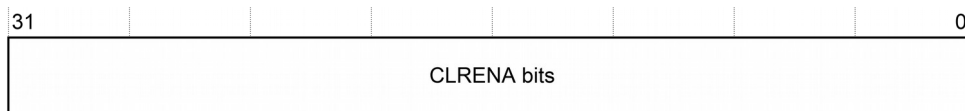
If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.

#### 24.4.2.3. Interrupt Clear-enable Register

The ICER disables interrupts, and show which interrupts are enabled. See the register summary in Table 24-23. NVIC register summary on page 339 for the register attributes.

The bit assignments are:

**Figure 24-16. ICER**



**Table 24-26. ICER bit assignments**

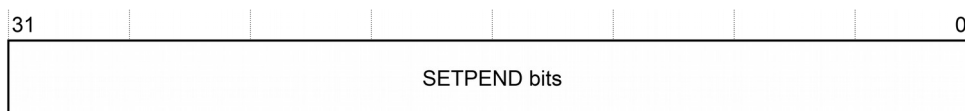
BIT	NAME	FUNCTION
31:0	<b>CLEARENA</b>	Interrupt clear-enable bits. Write: 1: disable interrupt 0: no effect Read: 1: interrupt enabled 0: interrupt disabled

#### 24.4.2.4. Interrupt Set-pending Register

The ISPR forces interrupts into the pending state, and shows which interrupts are pending. See the register summary in Table 24-23. NVIC register summary on page 339 for the register attributes.

The bit assignments are:

**Figure 24-17. ISPR**



**Table 24-27. ISPR bit assignments**

BIT	NAME	FUNCTION
31:0	<b>SETPEND</b>	Interrupt set-pending bits. Write: 1: changes interrupt state to pending 0: no effect Read: 1: changes interrupt state to pending 0: interrupt not pending

**Note**

Writing 1 to the ISPR bit corresponding to:

- an interrupt that is pending has no effect
- a disabled interrupt sets the state of that interrupt to pending.

*24.4.2.5. Interrupt Clear-pending Register*

The ICPR removes the pending state from interrupts, and shows which interrupts are pending. See the register summary in Table 24-23. NVIC register summary on page 339 for the register attributes.

The bit assignments are:

**Figure 24-18. ICPR**



**Table 24-28. ICPR bit assignments**

BIT	NAME	FUNCTION
31:0	<b>CLRPEND</b>	Interrupt clear-pending bits. Write: 1: removes pending state an interrupt 0: no effect Read: 1: interrupt is pending 0: interrupt not pending

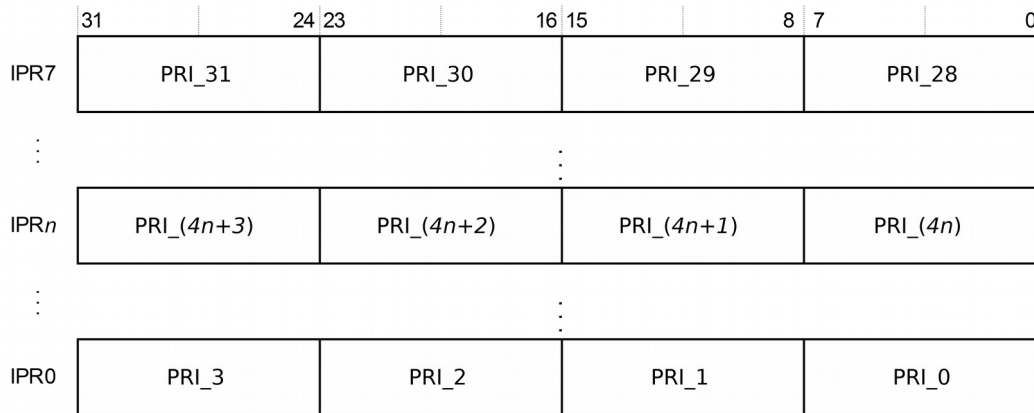
**Note**

Writing 1 to an ICPR bit does not affect the active state of the corresponding interrupt

*24.4.2.6. Interrupt Priority Registers*

The IPR0-IPR7 registers provide an 8-bit priority field for each interrupt. These registers are only word-accessible. See the register summary in Table 24-23. NVIC register summary on page 339 for their attributes. Each register holds four priority fields as shown:

**Figure 24-19. IPR**



**Table 24-29. IPR bit assignments**

BIT	NAME	FUNCTION
31:24	Priority, byte offset 3	Each priority field holds a priority value, 0-192. The lower the value, the greater the priority of the corresponding interrupt. The processor implements only bits[7:6] of each field, bits [5:0] read as zero and ignore writes. This means writing 255 to a priority register saves value 192 to the register
23:16	Priority, byte offset 2	
15:8	Priority, byte offset 1	
7:0	Priority, byte offset 3	

See Accessing the Cortex-M0 NVIC registers using CMSIS in chapter 24.4.2.1 on page 340 for more information about the access to the interrupt priority array, which provides the software view of the interrupt priorities.

Find the IPR number and byte offset for interrupt M as follows:

- the corresponding IPR number, N, is given by  $N = N \text{ DIV } 4$
- the byte offset of the required Priority field in this register is  $M \text{ MOD } 4$ , where:
  - byte offset 0 refers to register bits[7:0]
  - byte offset 1 refers to register bits[15:8]
  - byte offset 2 refers to register bits[23:16]
  - byte offset 3 refers to register bits[31:24]

#### 24.4.2.7. Level-sensitive and pulse interrupts

The processor supports both level-sensitive and pulse interrupts. Pulse interrupts are also described as edge-triggered interrupts.

A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Typically this happens because the ISR accesses the peripheral, causing it to clear the interrupt request. A pulse interrupt is an interrupt signal sampled synchronously on the rising edge of the processor clock. To ensure the NVIC detects the interrupt, the peripheral must assert the interrupt signal for at least one clock cycle, during which the NVIC

detects the pulse and latches the interrupt.

When the processor enters the ISR, it automatically removes the pending state from the interrupt, see Hardware and software control of interrupts in chapter 24.4.2.7.1 on page 344. For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. This means that the peripheral can hold the interrupt signal asserted until it no longer needs servicing.

#### **24.4.2.7.1. Hardware and software control of interrupts**

The Cortex-M0 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

- the NVIC detects that the interrupt signal is active and the corresponding interrupt is not active
- the NVIC detects a rising edge on the interrupt signal
- software writes to the corresponding interrupt set-pending register bit, see Interrupt Set-pending Register in chapter 24.4.2.4 on page 341

A pending interrupt remains pending until one of the following:

- The processor enters the ISR for the interrupt. This changes the state of the interrupt from pending to active. Then:
  - For a level-sensitive interrupt, when the processor returns from the ISR, the NVIC samples the interrupt signal. If the signal is asserted, the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.
  - For a pulse interrupt, the NVIC continues to monitor the interrupt signal, and if this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. If the interrupt signal is not pulsed while the processor is in the ISR, when the processor returns from the ISR the state of the interrupt changes to inactive.
- Software writes to the corresponding interrupt clear-pending register bit. For a level-sensitive interrupt, if the interrupt signal is still asserted, the state of the interrupt does not change. Otherwise, the state of the interrupt changes to inactive. For a pulse interrupt, state of the interrupt changes to:
  - inactive, if the state was pending
  - active, if the state was active and pending.

#### *24.4.2.8. NVIC usage hints and tips*

Ensure software uses correctly aligned register accesses. The processor does not support unaligned accesses to NVIC registers.

An interrupt can enter pending state even if it is disabled. Disabling an interrupt only prevents the processor from taking that interrupt.

#### **24.4.2.8.1. NVIC programming hints**

Software uses the CPSIE *i* and CPSID *i* instructions to enable and disable interrupts. The CMSIS provides the following intrinsic functions for these instructions:

```
void __disable_irq(void) // Disable Interrupts
void __enable_irq(void) // Enable Interrupts
```

In addition, the CMSIS provides a number of functions for NVIC control, including:

**Table 24-30. CMSIS access NVIC functions**

CMSIS FUNCTION	DESCRIPTION
void NVIC_EnableIRQ(IRQn_Type IRQn)	Enable IRQn
void NVIC_DisableIRQ(IRQn_Type IRQn)	Disable IRQn
uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)	Return true (1) if IRQn is pending
void NVIC_SetPendingIRQ(IRQn_Type IRQn)	Set IRQn pending
void NVIC_ClearPendingIRQ(IRQn_Type IRQn)	Clear IRQn pending state
void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)	Set priority for IRQn
uint32_t NVIC_GetPriority(IRQn_Type IRQn)	Read priority of IRQn
void NVIC_SystemReset (void)	Reset the system

The input parameter IRQn is the IRQ number, see Table 24-10. Properties of the different exception types on page 292 more information. For more information about these functions, see the CMSIS documentation.

### 24.4.3. System Control Block

The System Control Block (SCB) provides system implementation information, and system control. This includes configuration, control, and reporting of the system exceptions. The SCB registers are:

**Table 24-31. Summary of the SCB register**

ADDRESS	NAME	TYPE	RESET VALUE	DESCRIPTION
0xE000 ED00	CPUID	RO	0x410C C200	CPUID Register in chapter 24.4.3.2 on page 346
0xE000 ED04	ICSR	RW*	0x0000 0000	Interrupt Control and State Register in chapter 24.4.3.3 on page 346
0xE00 ED0C	AIRCR	RW*	0xFA05 0000	Application Interrupt and Reset Control Register in chapter 24.4.3.4 on page 348
0xE000 ED10	SCR	RW	0x0000 0000	System Control Register in chapter 24.4.3.5 on page 349
0xE000 ED14	CCR	RW	0x0000 0204	Configuration and Control register in chapter 24.4.3.6 on page 350
0xE000 ED1C	SHPR2	RW	0x0000 0000	System Handler Priority register in chapter 24.4.3.7 on page 350
0xE000 ED20	SHPR3	RW	0x0000 0000	System Handler Priority register in chapter 24.4.3.7 on page 350

\* See the register description for more information

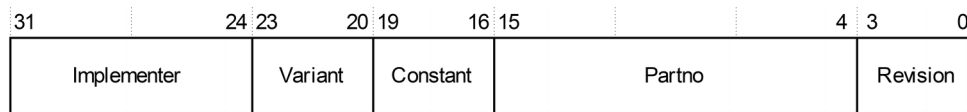
#### 24.4.3.1. The CMSIS mapping of the Cortex-M0 SCB registers

To improve software efficiency, the CMSIS simplifies the SCB register presentation. In the CMSIS, the array SHP[1] corresponds to the registers SHPR2-SHPR3

### 24.4.3.2. CPUID Register

The CPUID register contains the processor part number, version, and implementation information. See the register summary in Table 24-32. CPUID register bit assignments on page 346 for its attributes. The bit assignments are:

**Figure 24-20. CPUID**



**Table 24-32. CPUID register bit assignments**

BIT	NAME	FUNCTION
31:24	<b>Implementer</b>	Implementer code: 0x41 = ARM
23:20	<b>Variant</b>	Variant number, the r value in the rnpn product revision identifier: 0x0 = Revision 0
19:16	<b>Constant</b>	Constant that defines the architecture of the processor:, reads as 0xC = ARMv6-M architecture
15:4	<b>Partno</b>	Part number of the processor: 0xC20 = Cortex-M0
3:0	<b>Revision</b>	Revision number, the p value in the rnpn product revision identifier: 0x0 = Patch 0

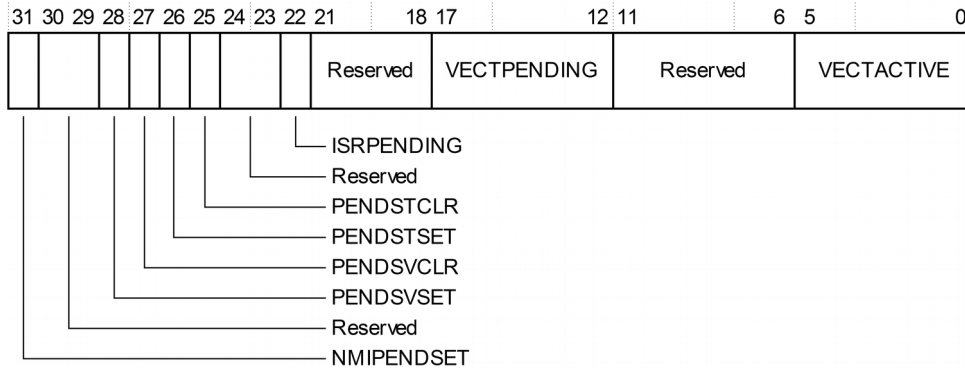
### 24.4.3.3. Interrupt Control and State Register

The ICSR:

- provides:
  - a set-pending bit for the Non-Maskable Interrupt (NMI) exception
  - set-pending and clear-pending bits for the PendSV and SysTick exceptions
- indicates:
  - the exception number of the exception being processed
  - whether there are preempted active exceptions
  - the exception number of the highest priority pending exception
  - whether any interrupts are pending.

See the register summary in Table 24-33. ICSR register bit assignments on page 347 for the ICSR attributes. The bit assignments are:

**Figure 24-21. ICSR**



**Table 24-33. ICSR register bit assignments**

BIT	NAME	TYPE	FUNCTION
31	<b>NMIPENDSET</b>	RW	NMI set-pending bit Write: 1: changes NMI exception state to pending 0: no effect Read: 1: NMI exception is pending 0: NMI exception is not pending Because NMI is the highest-priority exception, normally the processor enters the NMI exception handler as soon as it detects a write of 1 to this bit. Entering the handler then clears this bit to 0. This means a read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.
30:29	<b>Reserved</b>	RW	Reserved
28	<b>PENDSVSET</b>	RW	PendSV set-pending bit Write: 1: changes PendSV exception state to pending 0: no effect Read: 1: PendSV exception is pending 0: PendSV exception is not pending Writing 1 to this bit is the only way to set the PendSV state to pending
27	<b>PENDSVCLR</b>	WO	PendSV clear-pending bit Write: 1: removes the pending state from the PendSV exception 0: no effect
26	<b>PENDSTSET</b>	RW	SysTick exception set-pending bit. Write: 1: changes SysTick exception state to pending 0: no effect Read: 1: SysTick exception is pending 0: SysTick exception is not pending
25	<b>PENDSTCLR</b>	WO	SysTick exception clear-pending bit. Write: 1: removes the pending state from the SysTick exception This bit is WO. On a register read it's value is unknown
24:23	<b>Reserved</b>	RW	Reserved
22	<b>ISRPENDING</b>	RO	Interrupt pending flag, excluding NMI and Faults: 1: Interrupt pending 0: Interrupt not pending

BIT	NAME	TYPE	FUNCTION
21:18	<b>Reserved</b>	RW	Reserved
17:12	<b>VECTPENDING</b>	RO	Indicates the exception number of the highest priority pending enabled exception: Nonzero: the exception number of the highest priority pending enabled exception 0: no pending exceptions
11:6	<b>Reserved</b>	RW	Reserved
5:0	<b>VECTACTIVE*</b>	RO	Contains the active exception number: Nonzero: The exception number* of the currently active exception 0: Thread mode <b>Note:</b> Subtract 16 from this value to obtain the CMSIS IRQ number that identifies the corresponding bit in the Interrupt Clear-Enable, Set-Enable, Clear-Pending, Set-pending, and Priority Register, see Table 24-5. IPSR bit assignments on page 282

\* This is the same value as IPSR bits[5:0], see Table 24-5. IPSR bit assignments on page 282

When you write to the ICSR, the effect is Unpredictable if you:

- write 1 to the PENDSVSET bit and write 1 to the PENDSVCLR bit
- write 1 to the PENDSTSET bit and write 1 to the PENDSTCLR bit.

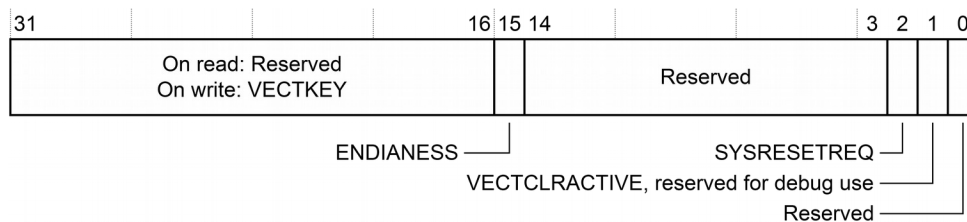
#### 24.4.3.4. Application Interrupt and Reset Control Register

The AIRCR provides endian status for data accesses and reset control of the system. See the register summary in Table 24-31. Summary of the SCB register on page 345 and Table 24-34. AIRCR register bit assignments on page 348 for its attributes.

To write to this register, you must write 0x05FA to the VECTKEY field, otherwise the processor ignores the write.

The bit assignments are:

**Figure 24-22. AIRCR**



**Table 24-34. AIRCR register bit assignments**

BIT	NAME	TYPE	FUNCTION
31:16	<b>VECTKEY</b>	RW	Register key Read: unknown Write: write 0x05FA to VECTKEY, otherwise the write is ignored
15	<b>ENDIANESS</b>	RO	Data endianess implemented 0x0: Little Endian

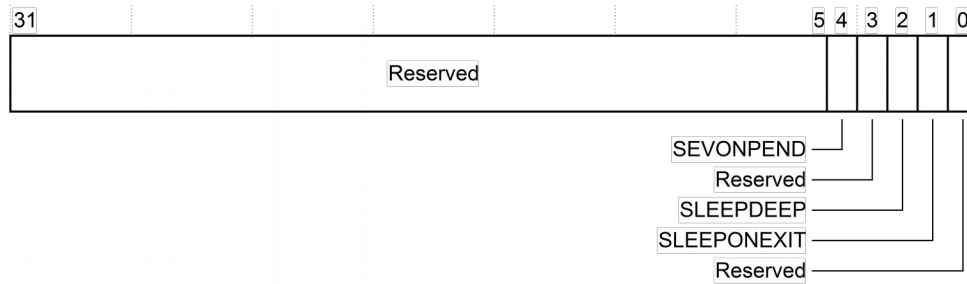


BIT	NAME	TYPE	FUNCTION
14:3	Reserved	RW	Reserved
2	<b>SYSRESETREQ</b>	WO	System reset request: Read: This bit reads as 0. Write: 1: requests a system level reset. 0: no effect
1	<b>VECTCLRACTIVE</b>	WO	Reserved for debug use. This bit reads as 0. When writing to the register you must write 0 to this bit, otherwise behavior is Unpredictable.
0	Reserved	RW	Reserved

#### 24.4.3.5. System Control Register

The SCR controls features of entry to and exit from low power state. See the register summary in Table 24-35. SCR register bit assignments on page 349 for its attributes. The bit assignments are:

**Figure 24-23. SCR**



**Table 24-35. SCR register bit assignments**

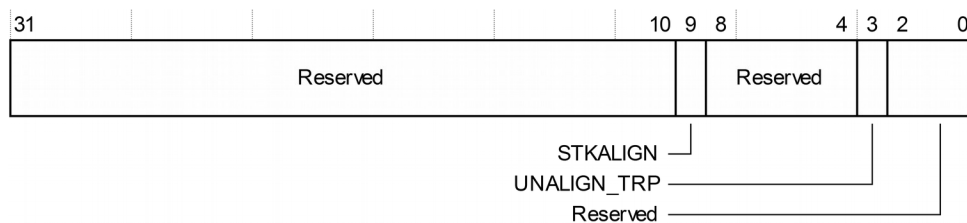
BIT	NAME	TYPE	FUNCTION
31:5	Reserved	R	Reserved
4	<b>SEVONPEND</b>	RW	Send Event on Pending bit: 1: enabled events and all interrupts, including disabled interrupts, can wakeup the processor 0: only enabled interrupts or events can wakeup the processor ,disabled interrupts are excluded When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event ,the event is registered and affects the next WFE. The processor also wakes up on execution of an SEV instruction or external event
3	Reserved	R	Reserved
2	<b>SLEEPDEEP</b>	RW	Controls whether the processor uses sleep or deep sleep as its low power mode: 1: deep sleep 0: sleep

BIT	NAME	TYPE	FUNCTION
1	<b>SLEEPONEXIT</b>	RW	Indicates sleep-on-exit when returning from Handler mode to Thread mode: 1: enter sleep, or deep sleep, on return from an ISR to Thread mode. 0: do not sleep when returning to Thread mode. Setting this bit to 1 enables an interrupt driven application to avoid returning to an empty main application
0	<b>Reserved</b>	R	Reserved

#### 24.4.3.6. Configuration and Control Register

The CCR is a read-only register and indicates some aspects of the behavior of the Cortex-M0 processor. See the register summary in Table 24-36. CCR register bit assignments in page 350 for the CCR attributes. The bit assignments are:

**Figure 24-24. CCR**



**Table 24-36. CCR register bit assignments**

BIT	NAME	TYPE	FUNCTION
31:10	<b>Reserved</b>	R	Reserved
9	<b>STKALIGN</b>	RW	Always reads as one, indicates 8-byte stack alignment on exception entry. On exception entry, the processor uses bit[9] of the stacked PSR to indicate the stack alignment. On return from the exception it uses this stacked bit to restore the correct stack alignment.
8:4	<b>Reserved</b>	R	Reserved
3	<b>UNALIGN_TRP</b>	RW	Always reads as one, indicates that all unaligned accesses generate a HardFault.
0	<b>Reserved</b>	R	Reserved

#### 24.4.3.7. System Handler Priority Registers

The SHPR2-SHPR3 registers set the priority level, 0 to 192, of the exception handlers that have configurable priority.

SHPR2-SHPR3 are word accessible. See the register summary in Table 24-31. Summary of the SCB register on page 345 for their attributes.

To access to the system exception priority level using CMSIS, use the following CMSIS functions:

- `uint32_t NVIC_GetPriority(IRQn_Type IRQn)`
- `void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)`

The input parameter IRQn is the IRQ number, see Table 24-10. Properties of the different exception types on

page 292 for more information.

The system fault handlers, and the priority field and register for each handler are:

**Table 24-37. System fault handler priority fields**

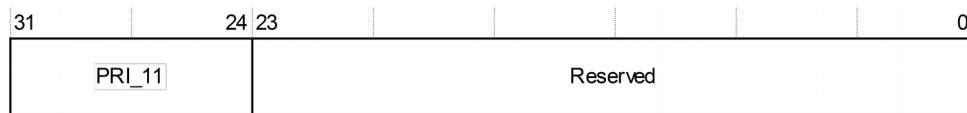
HANDLER	FIELD	REGISTER DESCRIPTION
SVCall	<b>PRI_11</b>	System handler Priority Register 2 on page
PendSV	<b>PRI_14</b>	System Handler Priority register 3 on page
SysTick	<b>PRI_15</b>	System Handler Priority register 3 on page

Each PRI\_N field is 8 bits wide, but the processor implements only bits[7:6] of each field, and bits[5:0] read as zero and ignore writes.

### 24.4.3.7.1. System Handler Priority Register 2

The bit assignments are:

**Figure 24-25. SHPR2**



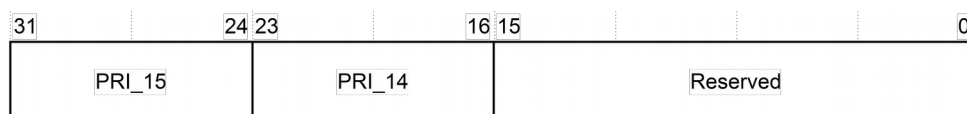
**Table 24-38. SHPR2 register bit assignments**

BIT	NAME	TYPE	FUNCTION
31:24	<b>PRI_11</b>	RW	Priority of system Handler 11, SVCall
23:0	<b>Reserved</b>	R	Reserved

### 24.4.3.7.2. System Handler Priority Register 3

The bit assignments are:

**Figure 24-26. SHPR3**



**Table 24-39. SHPR3 register bit assignments**

BIT	NAME	TYPE	FUNCTION
31:24	<b>PRI_15</b>	RW	Priority of system Handler 15, SysTick exception

BIT	NAME	TYPE	FUNCTION
23:16	<b>PRI_14</b>	RW	Priority of system Handler 14, PendSV
15:0	<b>Reserved</b>	R	Reserved

#### 24.4.3.8. SCB usage hints and tips

Ensure software uses aligned 32-bit word size transactions to access all the SCB registers.

#### 24.4.4. System timer, SysTick

When enabled, the timer counts down from the reload value to zero, reloads (wraps to) the value in the SYST\_RVR on the next clock cycle, then decrements on subsequent clock cycles. Writing a value of zero to the SYST\_RVR disables the counter on the next wrap. When the counter transitions to zero, the COUNTFLAG status bit is set to 1. Reading SYST\_CSR clears the COUNTFLAG bit to 0.

Writing to the SYST\_CVR clears the register and the COUNTFLAG status bit to 0. The write does not trigger the SysTick exception logic. Reading the register returns its value at the time it is accessed.

#### Note

When the processor is halted for debugging the counter does not decrement. The system timer registers are:

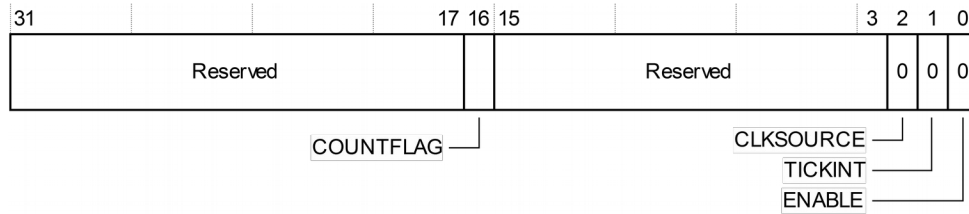
**Table 24-40. System timer register summary**

ADDRESS	NAME	TYPE	RESET VALUE	DESCRIPTION
0xE000 E010	SYST_CSR	RW	0x0000 0000	Systick Control and Status Register in chapter 24.4.4.1 on page 352
0xE000 E014	SYST_RVR	RW	Unknown	Systick Reload Value Register in chapter 24.4.4.2 on page 353
0xE000 E018	SYST_CVR	RW	Unknown	Systick Current Register in chapter 24.4.4.3 on page 354
0xE000 E01C	SYST_CALIB	RO	0x0000 0000	Systick Calibration value Register in chapter 24.4.4.4 on page 354

#### 24.4.4.1. SysTick Control and Status Register

The SYST\_CSR enables the SysTick features. See the register summary in Table 24-41. SYST\_CSR register bit assignments on page 353 for its attributes. The bit assignments are:

**Figure 24-27. SYST\_CSR**



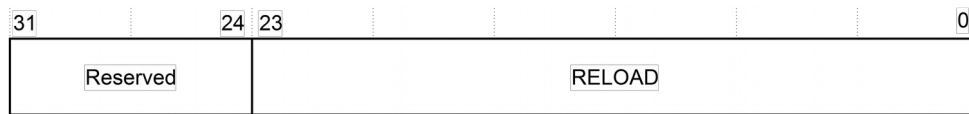
**Table 24-41. SYST\_CSR register bit assignments**

BIT	NAME	TYPE	FUNCTION
31:17	<b>Reserved</b>	R	Reserved
16	<b>COUNTFLAG</b>	RW	Returns 1 if timer counted to 0 since the last read of this register
15:3	<b>Reserved</b>	R	Reserved
2	<b>CLKSOURCE</b>	RW	Selects the SysTick timer clock source 1: HCLK 0: FCLK / 3
1	<b>TICKINT</b>	RW	Enables SysTick exception request 1: counting down to zero to assert the SysTick exception request 0: counting down to zero does not assert the SysTick exception request
0	<b>ENABLE</b>	RW	Enables the counter 1: counter enabled 0: counter disabled

#### 24.4.4.2. SysTick Reload Value Register

The SYST\_RVR specifies the start value to load into the SYST\_CVR. See the register summary in Table 24-42. SYST\_RVR register bit assignments on page 353 for its attributes. The bit assignments are:

**Figure 24-28. SYST\_RVR**



**Table 24-42. SYST\_RVR register bit assignments**

BIT	NAME	TYPE	FUNCTION
31:24	<b>Reserved</b>	R	Reserved
23:0	<b>RELOAD</b>	RW	Value to load into the SYST_CVR when the counter is enabled and when it reaches 0, see Calculating the RELOAD value.

#### 24.4.4.2.1. Calculating the RELOAD value

The RELOAD value can be any value in the range 0x000 00001 – 0x00FF FFFF. You can program a value of 0,

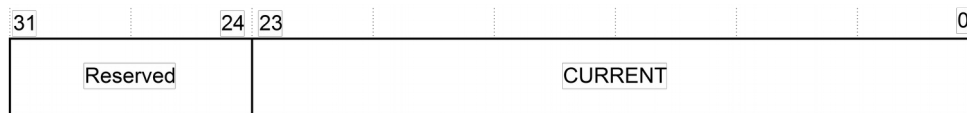
but this has no effect because the SysTick exception request and COUNTFLAG are activated when counting from 1 to 0.

To generate a multi-shot timer with a period of N processor clock cycles, use a RELOAD value of N-1. For example, if the SysTick interrupt is required every 100 clock pulses, set RELOAD to 99.

#### 24.4.4.3. SysTick Current Value Register

The SYST\_CVR contains the current value of the SysTick counter. See the register summary in Table 24-43. SYST\_CVR register bit assignments on page 354 for its attributes. The bit assignments are:

**Figure 24-29. SYST\_CVR**



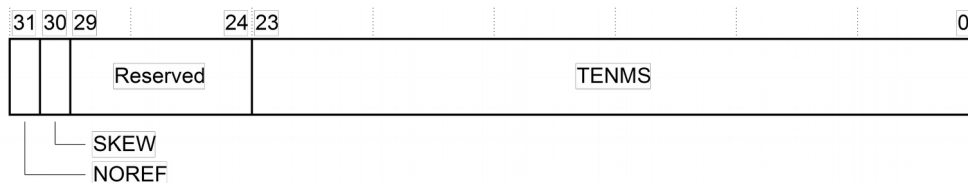
**Table 24-43. SYST\_CVR register bit assignments**

BIT	NAME	TYPE	FUNCTION
31:24	<b>Reserved</b>	R	Reserved
23:0	<b>CURRENT</b>	RW	Reads return the current value of the SysTick counter. A write of any value clears the field to 0, and also clears the SYST_CSR.COUNTFLAG bit to 0.

#### 24.4.4.4. SysTick Calibration Value Register

The SYST\_CALIB register indicates the SysTick calibration properties. See the register summary in Table 24-44. SYST\_CALIB register bit assignments on page 354 for its attributes. The bit assignments are:

**Figure 24-30. SYST\_CALIB**



**Table 24-44. SYST\_CALIB register bit assignments**

BIT	NAME	TYPE	FUNCTION
31	<b>NOREF</b>	R	Reads as one. Indicates no separate reference clock is provided
30	<b>SKEW</b>	R	Reads as one. Calibration value for the 10ms inexact timing is not known because TENMS is not known. This can affect the suitability of SysTick as a software real time clock.

BIT	NAME	TYPE	FUNCTION
29:24	<b>Reserved</b>	R	Reserved
23:0	<b>TENMS</b>	RW	Reads as zero. Indicates calibration value is not known

If calibration information is not known, calculate the calibration value required from the frequency of the processor clock or external clock.

#### 24.4.4.5. *SysTick usage hints and tips*

The interrupt controller clock updates the SysTick counter.

Ensure software uses word accesses to access the SysTick registers.

If the SysTick counter reload and current value are undefined at reset, the correct initialization sequence for the SysTick counter is:

1. Program reload value.
2. Clear current value.
3. Program Control and Status register.

## 25. LEGAL INFORMATION

Copyright © 2013 Active-Semi, Inc. All rights reserved.

All information provided in this document is subject to legal disclaimers.

Active-Semi reserves the right to modify its products, circuitry or product specifications without notice. Active-Semi products are not intended, designed, warranted or authorized for use as critical components in life-support, life-critical or safety-critical devices, systems, or equipment, nor in applications where failure or malfunction of any Active-Semi product can reasonably be expected to result in personal injury, death or severe property or environmental damage. Active-Semi accepts no liability for inclusion and/or use of its products in such equipment or applications. Active-Semi does not assume any liability arising out of the use of any product, circuit, or any information described in this document. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of Active-Semi or others. Active-Semi assumes no liability for any infringement of the intellectual property rights or other rights of third parties which would result from the use of information contained herein. Customers should evaluate each product to make sure that it is suitable for their applications. Customers are responsible for the design, testing, and operation of their applications and products using Active-Semi products. Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. All products are sold subject to Active-Semi's terms and conditions of sale supplied at the time of order acknowledgment. Exportation of any Active-Semi product may be subject to export control laws.

Active-Semi™, Active-Semi logo, Solutions for Sustainability™, Power Application Controller™, Micro Application Controller™, Multi-Mode Power Manager™, Configurable Analog Front End™, and Application Specific Power Drivers™ are trademarks of Active-Semi, Inc.

ARM® is a registered trademark and Cortex™ is a trademark of ARM Limited. All referenced brands and trademarks are the property of their respective owners.

For more information on this and other products, contact [sales@active-semi.com](mailto:sales@active-semi.com) or visit [www.active-semi.com](http://www.active-semi.com).