RSL10 Hardware Reference

M-20829-016
November 2019

ON Semiconductor®

Table of Contents

Page

1. Introduction . . . 17
   1.1 Purpose . . . 17
   1.2 Intended Audience . . . 17
   1.3 Conventions . . . 17
   1.4 Further Reading . . . 17
2. Overview . . . 19
   2.1 System Architecture . . . 19
   2.2 Radio System Architecture . . . 20
   2.3 Power . . . 21
   2.4 Clocking . . . 22
   2.5 Memory . . . 22
   2.6 Interfaces . . . 23
   2.7 Peripherals . . . 24
   2.8 Audio Components . . . 24
   2.9 SoC Identification . . . 24
      2.9.1 Device Identification Register . . . 25
         2.9.1.1 AHBREGS_CHIP_ID_NUM . . . 25
3. Arm Cortex-M3 Processor . . . 26
   3.1 Introduction . . . 26
      3.1.1 Arm Cortex-M3 Processor Loop Cache Register . . . 26
         3.1.1.1 SYSCTRL_CSS_LOOP_CACHE_CFG . . . 27
   3.2 Debug Port . . . 27
   3.3 IP Protection . . . 28
      3.3.1 Using IP Protection with Segger J-LInk . . . 29
      3.3.2 IP Protection Registers . . . 30
         3.3.2.1 SYSCTRL_DBG_LOCK . . . 30
         3.3.2.2 SYSCTRL_DBG_LOCK_KEY . . . 31
         3.3.2.3 SYSCTRL_DBG_UNLOCK_KEY . . . 31
   3.4 Activity Counters . . . 31
      3.4.1 Registers . . . 31
         3.4.1.1 SYSCTRL_CNT_CTRL . . . 32
         3.4.1.2 SYSCTRL_SYSCLK_CNT . . . 32
         3.4.1.3 SYSCTRL_CM3_CNT . . . 32
         3.4.1.4 SYSCTRL_LPDSP32_CNT . . . 32
         3.4.1.5 SYSCTRL_FLASH_READ_CNT . . . 32
4. LPDSP32 Processor . . . 33
   4.1 Overview . . . 33
   4.2 System Integration . . . 33
   4.3 Memory Support . . . 33
   4.4 Interrupt Handling . . . 33
   4.5 LPDSP32 Debug Port. . . . 34
   4.6 Registers . . . 35
      4.6.1 SYSCTRL_DSS_CTRL . . . 35
      4.6.2 SYSCTRL_DSS_CMD . . . 35
      4.6.3 SYSCTRL_DSS_LOOP_CACHE_CFG . . . 36

4.6.4 SYSCTRL_LPDSP32_DEBUG_CFG . . . 36
5. Power . . . 37
5.1 Power Supply Overview . . . 37
5.1.1 Power Management Unit . . . 38
5.2 Power Supply Inputs . . . 38
5.2.1 Battery Supply Voltage (VBAT) . . . 38
5.2.2 Digital Output Supply Voltage (VDDO) . . . 39
5.3 Internal Power Supply Voltages . . . 39
5.3.1 System Supply Voltage (VCC) . . . 39
5.3.1.1 VCC and DC-DC Converter Registers . . . 41
5.3.1.1.1 ACS_VCC_CTRL . . . 41
5.3.2 Internal Band Gap Reference Voltage . . . 42
5.3.2.1 Bandgap Converter Registers . . . 43
5.3.2.1.1 ACS_BG_CTRL . . . 43
5.3.3 RF Supply Voltage . . . 43
5.3.3.1 RF Block Configuration and Control Registers . . . 44
5.3.3.1.1 ACS_VDDRF_CTRL . . . 44
5.3.3.1.2 ACS_VDDPA_CTRL . . . 45
5.3.4 Digital Supply Voltages . . . 46
5.3.4.1 Digital Supply Configuration / Control Registers . . . 47
5.3.4.1.1 ACS_VDDC_CTRL . . . 47
5.3.4.1.2 ACS_VDDM_CTRL . . . 48
5.3.4.1.3 ACS_VDDRET_CTRL . . . 49
5.3.5 Analog Supply Voltage (VDDA) . . . 49
5.3.5.1 Analog Voltage Configuration and Control Registers . . . 50
5.3.5.1.1 ACS_VDDA_CP_CTRL . . . 50
5.4 Power Modes . . . 50
5.4.1 Overview . . . 50
5.4.2 Measuring Power Mode Current Consumption . . . 51
5.4.3 Run Mode . . . 51
5.4.4 Standby Mode . . . 51
5.4.4.1 Wakeup Sources . . . 52
5.4.4.2 ACS_PWR_MODES_CTRL . . . 52
5.4.4.3 ACS_WAKEUP_CFG . . . 53
5.4.4.4 ACS_WAKEUP_STATE. . . . 54
5.4.5 Sleep Mode . . . 54
5.4.5.1 Wakeup from Retention Memory in Sleep Mode . . . 56
5.4.6 ACS_WAKEUP_CTRL . . . 58
5.4.7 SYSCTRL_MEM_ACCESS_CFG . . . 60
5.4.8 SYSCTRL_WAKEUP_ADDR . . . 62
5.4.9 SYSCTRL_WAKEUP_PAD . . . 62
5.4.10 ACS_WAKEUP_GP_DATA . . . 63
5.5 Resets . . . 63
5.5.1 Reset Status Register . . . 65
5.5.1.1 DIG_RESET_STATUS . . . 65

5.5.2 ACS_RESET_STATUS 66
5.5.3 The nRESET Pad 68
5.6 Analog Test Signals 68
5.6.1 Analog Output Configuration Register 69
5.6.2 ACS_AOUT_CTRL 69
6. Clocking 72
6.1 Overview 72
6.2 Clock Generation 74
6.2.1 RC Oscillator 74
6.2.2 48 MHz Crystal Oscillator 74
6.2.3 Standby RC Oscillator 74
6.2.4 32 kHz Crystal Oscillator 75
6.2.5 External Clock Input (EXTCLK) 75
6.2.6 Debug Port Clock 76
6.2.7 Clock Generation Registers 76
6.2.7.1 ACS_RCOSC_CTRL 76
6.2.7.2 ACS_XTAL32K_CTRL 77
6.3 Clock Distribution 78
6.3.1 System Clock (SYSCLK) 78
6.3.2 Standby Clock (STANDBYCLK) 79
6.3.3 Slow Clock (SLOWCLK) 80
6.3.4 Baseband Clock (BBCLK) and Other Clocks for the Bluetooth Low Energy Baseband 81
6.3.5 Real-Time Clock (RTC). 81
6.3.6 User Clock (USRCLK) 81
6.3.7 Power Supply Clocks 82
6.3.8 Interface Clocks 82
6.3.9 Clock Distribution Registers 83
6.3.9.1 CLK_SYS_CFG 83
6.3.9.2 CLK_DIV_CFG0 84
6.3.9.3 CLK_DIV_CFG1 85
6.3.9.4 CLK_DIV_CFG2 86
6.3.9.5 ACS_RTC_CFG 87
6.3.9.6 ACS_RTC_COUNT 88
6.3.9.7 ACS_RTC_CTRL 88
6.4 Clock Detection 89
6.4.1 Clock Detector and System Monitor 89
6.4.2 External Clock Detector 90
6.4.3 Clock Detector Registers 91
6.4.3.1 CLK_DET_CFG 91
6.4.3.2 CLK_DET_STATUS 92
6.4.3.3 ACS_CLK_DET_CTRL 92
7. Memory 93
7.1 Memory Architecture 93
7.1.1 Memory Instances 93

7.1.2 Memory Buses 94
7.1.3 Memory Arbitration 94
7.2 Memory Map and Usage 95
7.2.1 Arm Cortex-M3 Processor Memory Usage 95
7.2.2 Flash Memory 97
7.2.2.1 Non-Volatile Record (NVR) Sectors 97
7.2.2.2 Redundancy Sectors 98
7.2.2.3 Error-Correction Coding 99
7.2.2.4 Flash Delay Timings 99
7.2.3 RAM 100
7.2.3.1 Program RAM 100
7.2.3.2 Data RAM 101
7.2.3.3 Shared RAM Instances 101
7.2.4 Other Memory Mapped Areas 101
7.2.4.1 Peripherals and Interfaces 101
7.2.4.2 Private Peripherals 102
7.2.5 LPDSP32 DSP Memory Usage 102
7.2.5.1 Program Memory 102
7.2.5.2 Data Memory 103
7.2.6 Bluetooth Low Energy Baseband (BB) Memory Usage 105
7.2.6.1 Exchange Memory 105
7.3 Flash Memory Operations 105
7.3.1 Reading and Writing Flash Memory 105
7.3.2 Low-Power Read Mode 107
7.3.3 Reading and Programming Flash Memory Procedures 107
7.3.3.1 Non-Sequential Programming 107
7.3.3.2 Sequential Programming 108
7.3.4 Locking / Unlocking Mechanism 108
7.3.4.1 Locking Mechanism 108
7.3.4.2 Unlocking Mechanism 108
7.4 Flash Copier 108
7.4.1 Block Characteristics 109
7.5 Memory Registers 110
7.5.1 SYSCTRL_FLASH_OVERLAY_CFG 110
7.5.2 SYSCTRL_FLASH_READ_CNT 111
7.5.3 SYSCTRL_MEM_ERROR 111
7.5.4 SYSCTRL_MEM_POWER_CFG 112
7.5.5 SYSCTRL_MEM_ACCESS_CFG 114
7.5.6 SYSCTRL_MEM_RETENTION_CFG 116
7.5.7 SYSCTRL_MEM_ARBITER_CFG 117
7.5.8 SYSCTRL_MEM_TIMING_CFG 120
7.6 Flash Memory Registers 120
7.6.1 FLASH_IF_CTRL 121
7.6.2 FLASH_MAIN_WRITE_UNLOCK 122
7.6.3 FLASH_MAIN_CTRL 122

7.6.4 FLASH_DELAY_CTRL . . . 123
7.6.5 FLASH_CMD_CTRL . . . 124
7.6.6 FLASH_IF_STATUS . . . 124
7.6.7 FLASH_ADDR . . . 126
7.6.8 FLASH_DATA . . . 126
7.6.9 FLASH_NVR_WRITE_UNLOCK . . . 126
7.6.10 FLASH_NVR_CTRL . . . 127
7.6.11 FLASH_PATCH_ADDR . . . 127
7.6.12 FLASH_COPY_CFG . . . 127
7.6.13 FLASH_COPY_CTRL . . . 128
7.6.14 FLASH_COPY_SRC_ADDR_PTR . . . 128
7.6.15 FLASH_COPY_DST_ADDR_PTR . . . 129
7.6.16 FLASH_COPY_WORD_CNT . . . 129
7.6.17 FLASH_ECC_CTRL . . . 129
7.6.18 FLASH_ECC_STATUS . . . 130
7.6.19 FLASH_ECC_ERROR_ADDR . . . 130
7.6.20 FLASH_ECC_UNCOR_ERROR_CNT . . . 130
7.6.21 FLASH_ECC_COR_ERROR_CNT . . . 130

8. RF Front-End . . . 131
8.1 Overview . . . 131
8.1.1 RF Front-End Registers . . . 132
8.1.2 SYSCTRL_RF_POWER_CFG . . . 133
8.1.3 SYSCTRL_RF_ACCESS_CFG . . . 133
8.2 System Integration . . . 134
8.2.1 48 MHz Crystal Oscillator . . . 135
8.3 Packet Handling . . . 135
8.3.1 Packet Format . . . 137
8.3.1.1 Preamble . . . 137
8.3.1.2 Pattern . . . 137
8.3.1.3 Packet Length . . . 137
8.3.1.4 Address . . . 138
8.3.1.5 Multi-Frame . . . 138
8.3.2 CRC . . . 139
8.3.3 Accessing the FIFOs . . . 140
8.3.4 FIFO Status . . . 140
8.4 Modulator and Radio Configuration . . . 140
8.4.1 Radio Configuration . . . 141
8.4.1.1 Data Rate . . . 141
8.4.1.1.1 Fractional Data Rate . . . 142
8.4.1.2 Central Frequency . . . 142
8.4.1.3 Channels . . . 143
8.4.1.4 On/Off Timing . . . 143
8.4.1.4.1 Tx On/Off Timing . . . 143

8.4.1.4.2 Rx On/Off Timing 144
8.4.1.4.3 Power Amplifier (PA) Power-Up 144
8.4.2 Tx Specific Configuration 145
8.4.2.1 Pulse Shape 145
8.4.2.2 Modulation Index 145
8.4.2.3 Interpolator 146
8.4.2.4 Power Amplifier (PA) Source 146
8.4.3 Rx Specific Configuration 147
8.4.3.1 Channel Filter Configuration 147
8.4.3.2 Phase and RSSI Fractional Decimation 147
8.4.3.3 Carrier Recovery 149
8.4.3.3.1 Rough Carrier Recovery .149
8.4.3.3.2 Fine Carrier Recovery 149
8.4.3.3.3 Carrier Recovery Boundaries 150
8.4.3.3.4 RSSI Detection. 151
8.4.3.3.5 Delay Line Synchronization. 153
8.4.3.4 Matched Filtering 153
8.4.3.5 Clock and Data-Rate Recovery. 154
8.4.3.6 Decision 155
8.4.3.6.1 Viterbi Algorithm 156
8.4.3.7 RSSI Filtering and AGC 156
8.4.3.7.1 Peak Detector 157
8.4.3.7.2 RSSI and Peak-Detector Combined AGC Strategy 157
8.4.4 Continuous Wave (CW) Configuration 158
8.4.5 Direct Test Mode (DTM). 158
8.5 RF Front-End Registers 159
8.5.1 RF_REG00 160
8.5.2 RF_REG01 162
8.5.3 RF_REG02 163
8.5.4 RF_REG03 163
8.5.5 RF_REG04 164
8.5.6 RF_REG05 165
8.5.7 RF_CENTER_FREQ 166
8.5.8 RF_REG07 166
8.5.9 RF_REG08 167
8.5.10 RF_REG09 168
8.5.11 RF_REG0A 169
8.5.12 RF_SYNC_PATTERN 169
8.5.13 RF_REG0C 170
8.5.14 RF_CRC_POLYNOMIAL 171
8.5.15 RF_CRC_RST 171
8.5.16 RF_REG0F 171
8.5.17 RF_REG10 172
8.5.18 RF_TX_PULSE0. 173

8.5.19 RF_TX_PULSE1 . . . 173
8.5.20 RF_TX_PULSE2 . . . 174
8.5.21 RF_TX_PULSE3 . . . 174
8.5.22 RF_RX_PULSE. . . . 175
8.5.23 RF_REG16 . . . 175
8.5.24 RF_REG17 . . . 176
8.5.25 RF_REG18 . . . 177
8.5.26 RF_REG19 . . . 178
8.5.27 RF_REG1A . . . 179
8.5.28 RF_REG1B . . . 179
8.5.29 RF_AGC_LUT1 . . . 180
8.5.30 RF_AGC_LUT2 . . . 181
8.5.31 RF_AGC_LUT3 . . . 181
8.5.32 RF_AGC_LUT4 . . . 182
8.5.33 RF_REG20 . . . 182
8.5.34 RF_AGC_ATT1. . . . 183
8.5.35 RF_REG22 . . . 184
8.5.36 RF_REG23 . . . 184
8.5.37 RF_REG24 . . . 185
8.5.38 RF_REG25 . . . 186
8.5.39 RF_REG26 . . . 186
8.5.40 RF_REG27 . . . 187
8.5.41 RF_REG28 . . . 187
8.5.42 RF_PLL_CTRL . . . 189
8.5.43 RF_REG2A . . . 191
8.5.44 RF_XTAL_CTRL . . . 191
8.5.45 RF_REG2C . . . 193
8.5.46 RF_REG2D . . . 194
8.5.47 RF_REG2E . . . 195
8.5.48 RF_REG2F . . . 196
8.5.49 RF_REG30 . . . 197
8.5.50 RF_REG31 . . . 198
8.5.51 RF_REG32 . . . 198
8.5.52 RF_TXFIFO . . . 198
8.5.53 RF_RXFIFO . . . 198
8.5.54 RF_DESER_STATUS . . . 199
8.5.55 RF_IRQ_STATUS . . . 199
8.5.56 RF_REG37 . . . 199
8.5.57 RF_REG38 . . . 199
8.5.58 RF_REG39 . . . 199
8.5.59 RF_REVISION . . . 200

9. Bluetooth Low Energy Baseband 201
9.1 Overview 201
9.1.1 Bluetooth Baseband Error Handling 202
9.1.2 Support Interfaces 204
9.2 Baseband Registers and Memory Usage 204
9.2.1 Baseband Abstraction Layer Primitives 205
9.2.2 Control Structures 205
9.3 Baseband Timing 206
9.3.1 Clock Structures 206
9.3.1.1 ACS_BB_TIMER_CTRL 207
9.3.2 Slot Timing 207
9.3.3 Timing and Event-Related Interrupts 208
9.3.3.1 Schedule Interrupt 209
9.3.3.2 End of Event Interrupt 210
9.3.3.3 Reception Interrupt 211
9.3.3.4 Wakeup Interrupt 212
9.3.3.5 Software Interrupt 212
9.3.3.6 Encryption Interrupt 212
9.4 Baseband Registers 213
9.4.1 BBIF_CTRL 215
9.4.2 BBIF_STATUS 216
9.4.3 BBIF_COEX_CTRL 217
9.4.4 BBIF_COEX_STATUS 217
9.4.5 BBIF_COEX_INT_CFG 218
9.4.6 BBIF_COEX_INT_STATUS 218
9.4.7 BBIF_SYNC_CFG 219
9.4.8 BB_RWBBCNTL 219
9.4.9 BB_VERSION 221
9.4.10 BB_RWBLEBCONF 222
9.4.11 BB_INTCNTL 223
9.4.12 BB_INTSTAT 224
9.4.13 BB_INTRAWSTAT 225
9.4.14 BB_INTACK 227
9.4.15 BB_BASETIMECNT 228
9.4.16 BB_FINETIMECNT 229
9.4.17 BB_BDADDRL 229
9.4.18 BB_BDADDRU 229
9.4.19 BB_ET_CURRENTRXDESCPTR 229
9.4.20 BB_DEEPSLCNTL 230
9.4.21 BB_DEEPSLWKUP 230
9.4.22 BB_DEEPSLSTAT 231
9.4.23 BB_ENBPRESET 231
9.4.24 BB_FINECNTCORR 231
9.4.25 BB_BASETIMECNTCORR 231
9.4.26 BB_DIAGCNTL 232

9.4.27 BB_DIAGSTAT . . . 232
9.4.28 BB_DEBUGADDMAX . . . 233
9.4.29 BB_DEBUGADDMIN . . . 233
9.4.30 BB_ERRORTYPESTAT . . . 233
9.4.31 BB_SWPROFILING . . . 235
9.4.32 BB_RADIOCNTL0 . . . 236
9.4.33 BB_RADIOCNTL1 . . . 236
9.4.34 BB_RADIOCNTL2 . . . 237
9.4.35 BB_RADIOPWRUPDN0 . . . 237
9.4.36 BB_RADIOPWRUPDN1 . . . 238
9.4.37 BB_RADIOTXRXTIM0 . . . 238
9.4.38 BB_RADIOTXRXTIM1 . . . 238
9.4.39 BB_SPIPTRCNTL0 . . . 239
9.4.40 BB_SPIPTRCNTL1 . . . 239
9.4.41 BB_SPIPTRCNTL2 . . . 239
9.4.42 BB_ADVCHMAP . . . 239
9.4.43 BB_ADVTIM . . . 240
9.4.44 BB_ACTSCANSTAT . . . 240
9.4.45 BB_WLPUBADDPTR . . . 240
9.4.46 BB_WLPRIVADDPTR . . . 240
9.4.47 BB_WLNBDEV . . . 240
9.4.48 BB_AESCNTL . . . 241
9.4.49 BB_AESKEY31_0 . . . 241
9.4.50 BB_AESKEY63_32 . . . 241
9.4.51 BB_AESKEY95_64 . . . 241
9.4.52 BB_AESKEY127_96 . . . 242
9.4.53 BB_AESPTR . . . 242
9.4.54 BB_TXMICVAL . . . 242
9.4.55 BB_RXMICVAL . . . 242
9.4.56 BB_RFTESTCNTL . . . 243
9.4.57 BB_RFTESTTXSTAT . . . 243
9.4.58 BB_RFTESTRXSTAT . . . 244
9.4.59 BB_TIMGENCNTL . . . 244
9.4.60 BB_GROSSTIMTGT . . . 244
9.4.61 BB_FINETIMTGT . . . 244
9.4.62 BB_COEXIFCNTL0 . . . 245
9.4.63 BB_COEXIFCNTL1 . . . 247
9.4.64 BB_COEXIFCNTL2 . . . 248
9.4.65 BB_BBMPRIO0 . . . 248
9.4.66 BB_BBMPRIO1 . . . 249
9.4.67 BB_RALPTR . . . 249
9.4.68 BB_RALNBDEV . . . 249

9.4.69 BB_RAL_LOCAL_RND . . . 250
9.4.70 BB_RAL_PEER_RND . . . 250
9.4.71 BB_ISOCHANCNTL0 . . . 250
9.4.72 BB_ISOMUTECNTL0 . . . 251
9.4.73 BB_ISOCURRENTTXPTR0 . . . 251
9.4.74 BB_ISOCURRENTRXPTR0 . . . 252
9.4.75 BB_ISOTRCNL0 . . . 252
9.4.76 BB_ISOEVTCNTLOFFSETL0 . . . 252
9.4.77 BB_ISOEVTCNTLOFFSETU0 . . . 252
9.4.78 BB_ISOCHANCNTL1 . . . 253
9.4.79 BB_ISOMUTECNTL1 . . . 253
9.4.80 BB_ISOCURRENTTXPTR1 . . . 254
9.4.81 BB_ISOCURRENTRXPTR1 . . . 254
9.4.82 BB_ISOTRCNL1 . . . 254
9.4.83 BB_ISOEVTCNTLOFFSETL1 . . . 255
9.4.84 BB_ISOEVTCNTLOFFSETU1 . . . 255
9.4.85 BB_ISOCHANCNTL2 . . . 255
9.4.86 BB_ISOMUTECNTL2 . . . 256
9.4.87 BB_ISOCURRENTTXPTR2 . . . 256
9.4.88 BB_ISOCURRENTRXPTR2 . . . 257
9.4.89 BB_ISOTRCNL2 . . . 257
9.4.90 BB_ISOEVTCNTLOFFSETL2 . . . 257
9.4.91 BB_ISOEVTCNTLOFFSETU2 . . . 257
9.4.92 BB_BBPRIOSCHARB . . . 258

10. Digital Input/Output . . . 259
10.1 Overview . . . 259
10.2 Functional Configuration . . . 259
10.2.1 Special Functional Configurations . . . 262
10.3 Physical Configuration . . . 263
10.4 DIO Registers . . . 264
10.4.1 DIO_CFG . . . 265
10.4.2 DIO_DATA . . . 268
10.4.3 DIO_DIR . . . 271
10.4.4 DIO_MODE . . . 274
10.4.5 DIO_INT_CFG . . . 275
10.4.6 DIO_INT_DEBOUNCE . . . 276
10.4.7 DIO_PCM_SRC . . . 276
10.4.8 DIO_SPI_SRC . . . 279
10.4.9 DIO_UART_SRC . . . 281
10.4.10 DIO_I2C_SRC . . . 281
10.4.11 DIO_AUDIOSINK_SRC . . . 283
10.4.12 DIO_NMI_SRC. . . . 283
10.4.13 DIO_BB_RX_SRC. . . . 284
10.4.14 DIO_BB_SPI_SRC . . . 286

10.4.15 DIO_RF_SPI_SRC . . . 287
10.4.16 DIO_RF_GPIO03_SRC . . . 289
10.4.17 DIO_RF_GPIO47_SRC . . . 292
10.4.18 DIO_RF_GPIO89_SRC . . . 294
10.4.19 DIO_DMIC_SRC . . . 296
10.4.20 DIO_LPDSP32_JTAG_SRC . . . 297
10.4.21 DIO_JTAG_SW_PAD_CFG . . . 299
10.4.22 DIO_EXTCLK_CFG. . . . 300
10.4.23 DIO_PAD_CFG . . . 300
11. External Digital Interfaces . . . 301
11.1 Introduction . . . 301
11.2 Analog-to-Digital Converters (ADCs) . . . 301
11.2.1 ADC Input Configuration . . . 302
11.2.2 ADC Sampling Configuration . . . 302
11.2.3 ADC Output Data . . . 304
11.2.4 Power Supply Monitoring. . . . 304
11.2.5 ADC and Power Supply Monitoring Interrupt . . . 304
11.2.6 ADC Registers . . . 305
11.2.6.1 ADC_DATA_TRIM_CH . . . 305
11.2.6.2 ADC_DATA_AUDIO_CH . . . 305
11.2.6.3 ADC_INPUT_SEL. . . . 305
11.2.6.4 ADC_CFG . . . 306
11.2.6.5 ADC_OFFSET . . . 307
11.2.6.6 ADC_BATMON_CFG . . . 307
11.2.6.7 ADC_BATMON_INT_ENABLE . . . 308
11.2.6.8 ADC_BATMON_COUNT_VAL . . . 309
11.2.6.9 ADC_BATMON_STATUS . . . 309
11.3 General-Purpose I/O (GPIO) Interface . . . 309
11.3.1 GPIO Interrupts . . . 310
11.4 I2C Interface . . . 310
11.4.1 Slave Mode Specific Configuration. . . . 314
11.4.2 Master Mode Specific Configuration . . . 315
11.4.3 I2C Interrupts . . . 316
11.4.3.1 Operation Using Manual Acknowledgement . . . 316
11.4.3.2 Operation Using Auto Acknowledgement . . . 317
11.4.4 I2C Registers. . . . 317
11.4.4.1 I2C_CTRL0 . . . 318
11.4.4.2 I2C_CTRL1 . . . 321
11.4.4.3 I2C_DATA . . . 321
11.4.4.4 I2C_DATA_M . . . 321
11.4.4.5 I2C_ADDR_START . . . 322
11.4.4.6 I2C_STATUS . . . 322
11.5 Pulse Code Modulation (PCM) Interface . . . 324
11.5.1 PCM Signal Configuration . . . 325

11.5.1.1 Frame Signal Configuration and Timing 325
11.5.1.2 Data Serial Input and Output Configuration 328
11.5.2 PCM Interrupt Configuration 328
11.5.3 I2S Configuration and Usage 328
11.5.4 PCM Registers 331
11.5.4.1 PCM_CTRL 331
11.5.4.2 PCM_TX_DATA. 332
11.5.4.3 PCM_RX_DATA. 332
11.5.4.4 PCM_STATUS 332
11.6 Pulse Width Modulation (PWM) 333
11.6.1 PWM Registers 334
11.6.1.1 PWM_CFG 334
11.6.1.2 PWM_CTRL 334
11.7 Serial Peripheral Interfaces (SPI) 335
11.7.1 SPI Data Transfers 335
11.7.2 SPI Interrupts 337
11.7.3 SPI DMA Control 337
11.7.4 SPI Registers 338
11.7.4.1 SPI0_CTRL0 338
11.7.4.2 SPI0_CTRL1 340
11.7.4.3 SPI0_TX_DATA 340
11.7.4.4 SPI0_RX_DATA 340
11.7.4.5 SPI0_STATUS 341
11.7.4.6 SPI1_CTRL0 341
11.7.4.7 SPI1_CTRL1 342
11.7.4.8 SPI1_TX_DATA 343
11.7.4.9 SPI1_RX_DATA 343
11.7.4.10 SPI1_STATUS 343
11.8 Universal Asynchronous Receiver-Transmitter (UART) Interfaces 344
11.8.1 UART Interrupts 345
11.8.2 UART Interface Registers 345
11.8.2.1 UART_CFG 346
11.8.2.2 UART_TX_DATA 346
11.8.2.3 UART_RX_DATA 346
11.8.2.4 UART_STATUS 346
11.9 Support Interfaces 346
12. Peripherals 350
12.1 Cyclic Redundancy Check (CRC) Generator 350
12.1.1 CRC Registers 351
12.1.1.1 CRC_CTRL 351
12.1.1.2 CRC_VALUE. 352
12.1.1.3 CRC_ADD_1 352
12.1.1.4 CRC_ADD_8 352
12.1.1.5 CRC_ADD_16 352
12.1.1.6 CRC_ADD_24 353
12.1.1.7 CRC_ADD_32 353

12.1.1.8 CRC_FINAL . . . 353
12.2 Direct Memory Access (DMA) Controller . . . 353
12.2.1 Introduction . . . 353
12.2.2 DMA Channel Configuration . . . 353
12.2.3 Word Size, Data Packing and Transfer Length . . . 356
12.2.3.1 Word Size . . . 356
12.2.3.2 Data Packing . . . 356
12.2.3.3 Transfer Length . . . 359
12.2.4 DMA Transfer Types . . . 359
12.2.4.1 Memory-to-Memory (MM) . . . 361
12.2.4.2 Memory-to-Peripheral . . . 361
12.2.4.3 Peripheral-to-Memory (PM) . . . 362
12.2.4.4 Peripheral-to-Peripheral (PP) . . . 362
12.2.5 DMA Interrupt Configuration . . . 363
12.2.6 Channel Priority . . . 364
12.2.7 Data Memory Usage by the DMA and ARM Cortex-M3 Processor . . . 364
12.2.8 DMA Registers . . . 365
12.2.8.1 DMA_CTRL0 . . . 367
12.2.8.2 DMA_SRC_BASE_ADDR . . . 370
12.2.8.3 DMA_DEST_BASE_ADDR . . . 370
12.2.8.4 DMA_CTRL1 . . . 370
12.2.8.5 DMA_NEXT_SRC_ADDR . . . 371
12.2.8.6 DMA_NEXT_DEST_ADDR . . . 371
12.2.8.7 DMA_WORD_CNT . . . 371
12.2.8.8 DMA_STATUS . . . 371
12.3 Timers . . . 372
12.3.1 Starting or Stopping Timers . . . 373
12.3.2 Mode Selection . . . 373
12.3.3 Timer Registers . . . 373
12.3.3.1 TIMER_CFG . . . 374
12.3.3.2 TIMER_CTRL . . . 375
12.3.3.3 TIMER_VAL . . . 375
12.4 Watchdog Timer . . . 375
12.4.1 Watchdog Registers . . . 376
12.4.1.1 WATCHDOG_CFG . . . 376
12.4.1.2 WATCHDOG_CTRL . . . 377
13. Audio . . . 378
13.1 Digital Microphone (DMIC) Inputs . . . 378
13.1.1 Digital Microphone and Shared Digital Microphone/Output Driver Registers . . . 380
13.1.1.1 AUDIO_CFG . . . 380
13.1.1.2 AUDIO_STATUS . . . 382
13.1.1.3 AUDIO_DMIC_CFG . . . 383
13.1.1.4 AUDIO_DMIC0_GAIN . . . 385
13.1.1.5 AUDIO_DMIC1_GAIN . . . 385
13.1.1.6 AUDIO_DMIC_DATA . . . 386

13.1.1.7 AUDIO_DMIC0_DATA . . . 386
13.1.1.8 AUDIO_DMIC1_DATA . . . 386
13.2 Output Driver . . . 386
13.2.1 Output Driver Registers. . . . 388
13.2.1.1 AUDIO_OD_CFG . . . 388
13.2.1.2 AUDIO_OD_GAIN . . . 389
13.2.1.3 AUDIO_OD_DATA. . . . 390
13.2.1.4 AUDIO_SDM_CFG . . . 390
13.3 Audio Sink Clock Counters . . . 390
13.3.1 Audio Sink Registers . . . 392
13.3.1.1 AUDIOSINK_CTRL . . . 393
13.3.1.2 AUDIOSINK_CFG . . . 394
13.3.1.3 AUDIOSINK_CNT . . . 394
13.3.1.4 AUDIOSINK_PHASE_CNT . . . 394
13.3.1.5 AUDIOSINK_PERIOD_CNT . . . 394
13.4 Asynchronous Sample Rate Converter (ASRC) . . . 394
13.4.1 ASRC Registers . . . 396
13.4.1.1 ASRC_CTRL . . . 397
13.4.1.2 ASRC_INT_ENABLE . . . 398
13.4.1.3 ASRC_OUT . . . 398
13.4.1.4 ASRC_IN . . . 398
13.4.1.5 ASRC_CFG . . . 398
13.4.1.6 ASRC_OUTPUT_CNT. . . . 399
13.4.1.7 ASRC_PHASE_INC . . . 399
13.4.1.8 ASRC_PHASE_CNT . . . 399
13.4.1.9 ASRC_STATE_MEM . . . 399

14. Private Peripherals . . . 400
14.1 Nested Vectored Interrupt Controller (NVIC) . . . 400
14.1.1 Interrupt Controller Type Register . . . 404
14.1.2 Interrupt Set Enable and Clear Enable Registers . . . 404
14.1.3 Interrupt Set-Pending Registers and Interrupt Clear-Pending Registers . . . 405
14.1.4 Active Bit Register . . . 406
14.1.5 Interrupt Priority Registers . . . 406
14.1.6 Registers Described by Arm Documentation . . . 407
14.2 SysTick . . . 408
14.2.1 SysTick Control and Configuration Registers . . . 408
14.2.1.1 SysTick_CTRL . . . 409
14.2.1.2 SysTick_LOAD . . . 409
14.2.1.3 SysTick_VAL . . . 409
14.2.1.4 SysTick_CALIB . . . 409
14.3 Debug Controller . . . 410
14.3.1 Halting Debug Configuration and Status . . . 410
14.3.2 Debug Monitor Configuration. . . . 411
14.3.3 Arm Cortex-M3 Processor Core Register Access . . . 411
14.3.4 Debug Fault Status Register . . . 411
14.3.5 Arm Cortex-M3 Processor Debug Port Specific Control and Configuration Registers . . . 412

14.3.5.1 SCB_DFSR Settings . . . 412
14.3.5.2 CoreDebug_DHCSR Settings . . . 413
14.3.5.3 CoreDebug_DCRSR Settings . . . 413
14.3.5.4 Debug Exception and Monitor Control Register. . . . 413
A. Control and Configuration Registers . . . 415
A.1 Chip Identification . . . 416
A.2 System Control . . . 417
A.3 Clock Generation . . . 422
A.4 Reset . . . 423
A.5 Watchdog Timer . . . 424
A.6 General-Purpose Timers 0, 1, 2 and 3 . . . 425
A.7 Flash Interface Configuration and Control . . . 426
A.8 DMA Controller Configuration and Control . . . 429
A.9 DIO Interface and Digital Pad control . . . 431
A.10 SPI Interface Configuration and Control . . . 433
A.11 SPI Interface Configuration and Control . . . 434
A.12 PCM Interface Configuration and Control . . . 435
A.13 I2C Interface Configuration and Control . . . 436
A.14 UART Interface Configuration and Control . . . 438
A.15 PWM 0 and 1 Configuration and Control . . . 439
A.16 DMIC Input and Output Driver Configuration and Control . . . 440
A.17 CRC Generator Control . . . 443
A.18 Audio Sink Clock Counters . . . 444
A.19 ASRC Configuration and Control . . . 445
A.20 Analog-to-Digital Converter and Battery Monitoring . . . 446
A.21 ACS domain (Analog Bridge Access) . . . 447
A.22 Baseband Controller Interface . . . 452
A.23 Baseband Controller . . . 453
A.24 RF Front-End 2.4 GHz . . . 466
A.25 SYSTICK Timer . . . 493
A.26 System Control and ID register not in the SCB . . . 494
A.27 Nested Vector Interrupt Controller . . . 495
A.28 System Control Block . . . 512
A.29 Debug Controller . . . 515
B. Glossary . . . 518

CHAPTER 1

Introduction

1.1 PURPOSE

This manual provides a reference to the system hardware for application developers working with RSL10. This manual describes all of the components belonging to the RSL10 System-on-Chip (SoC), including:

- Bluetooth® and generic RF support
- Data processing and control components
- Power supply and clocking components
- Memory components
- Peripherals
- External interfaces

The data processing and control component information provided by this manual complements the Arm® Cortex®-M3 core description in the *The Definitive Guide to the ARM Cortex-M3* and other third-party documentation for the Arm Cortex-M3 processor.

The *RSL10 Hardware Reference* further describes how each component of the RSL10 SoC can be used in the implementation of an application, and provides information about the configuration of the various system components. This manual is a part of the RSL10 Evaluation and Development Kit (RSL10 EDK).

1.2 INTENDED AUDIENCE

This manual is primarily intended for engineers and other individuals who are responsible for developing and/or maintaining Bluetooth and other RF-based communication applications that make use of an RSL10 SoC. People who are developing local interfaces between an external device and RSL10, as well as those interested in the details of the audio, power supply and clocking components, will find this manual particularly helpful as it focuses on the configuration and use of system components.

This manual assumes that readers are familiar with C-level programming, RF application concepts, and Bluetooth low energy technology.

1.3 CONVENTIONS

The following conventions are used in this manual to signify particular types of information:

- Control and configuration registers are shown in a `special font`.
- Angle brackets “<” and “>” identify an optional parameter or indicate a placeholder for specific information. To use an optional parameter or replace a placeholder, specify the information within the brackets; do not include the brackets themselves.
- Default settings for registers and bit fields are marked with an asterisk (*).
- All sample rates specified are the final decimated sample rates, unless stated otherwise.
- In general, numbers are presented in decimal notation. In cases where hexadecimal or binary notation is more convenient, these numbers are identified by the prefixes “0x” and “0b” respectively. For example, the decimal number 123456 can also be represented as 0x1E240 or 0b11110001001000000.

1.4 FURTHER READING

For more information, refer to the following documents:

- *ARMv7M Architecture Reference Manual*

- *Bluetooth Core Specification version 5.0*, available from https://www.bluetooth.com/specifications/adopted-specifications
- *RSL10 Firmware Reference*

CHAPTER 2

Overview

2.1 System Architecture

RSL10 is an ultra-low-power, highly flexible multi-protocol 2.4 GHz SoC specifically designed for use in high−performance wearable and medical applications, or any other applications that can benefit from low-power wireless connectivity. With its Arm Cortex-M3 processor and LPDSP32 DSP core, RSL10 supports Bluetooth low energy technology and any 2.4 GHz proprietary protocol stacks, without sacrificing power consumption.

Figure 1 illustrates a high-level component diagram describing RSL10.

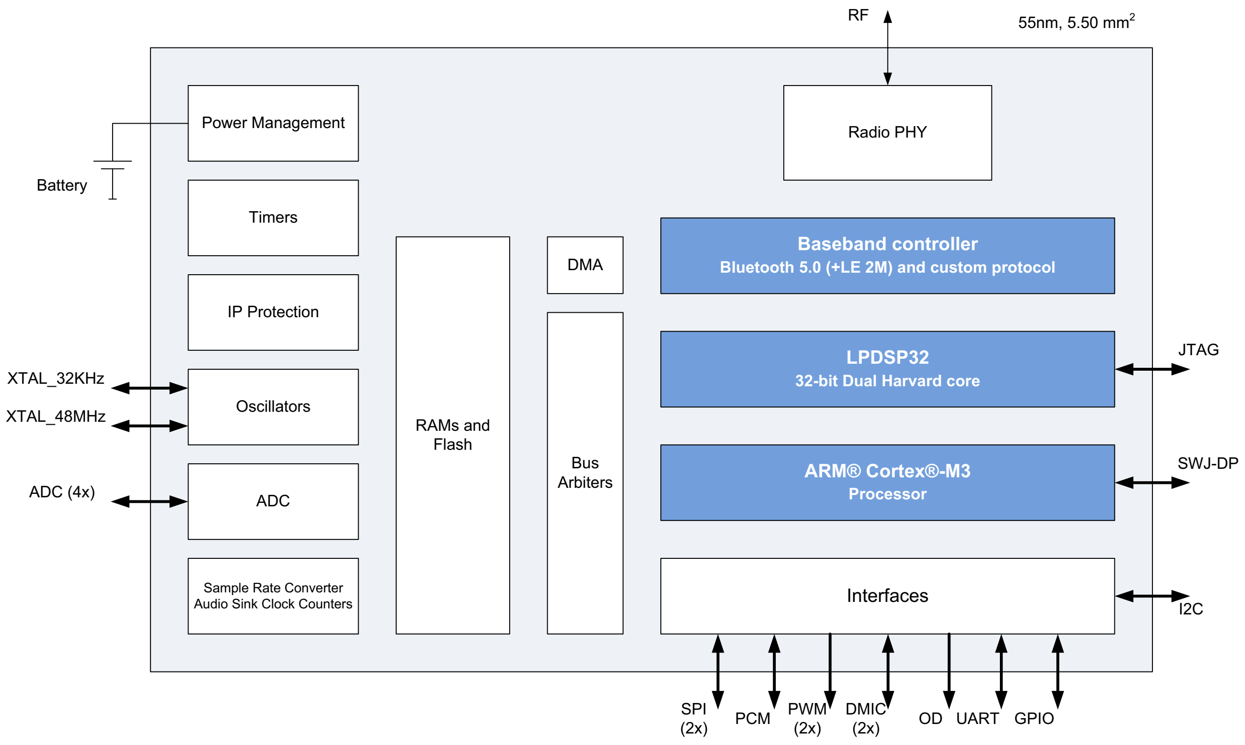


Figure 1. Top Level Component Diagram

RSL10 is based around four key components:

1. **The** Arm Cortex-M3 processor**:** A 32-bit core for real-time applications, specifically developed to enable high-performance low-cost platforms for a broad range of low-power applications.

   This processor provides general-purpose processing that is used to configure and control all of the components of the RSL10 system including the LPDSP32 DSP, the RF front-end, and the Bluetooth baseband.

2. LPDSP32 **Digital Signal Processor (DSP):** A 32-bit Dual Harvard DSP core that efficiently supports audio codecs required for wireless audio communication. If not used for audio applications, this core can be used for other signal processing tasks for advanced developments that need additional processing power. To enable development using the LPDSP32, contact your ON Semiconductor representative.

3. **Radio Frequency Front-End:** Based on a 2.4 GHz RF transceiver, the RFFE implements the physical layer of the Bluetooth low energy technology standard and other proprietary or custom protocols.
4. **Bluetooth Protocol Baseband Hardware:** Bluetooth low energy technology compliant, including support for the LE 2 Mb PHY feature first defined in the Bluetooth core 5.0 technology release, and all optional features of Bluetooth low energy technology that were also defined in earlier core releases. The RSL10 baseband stack is supplemented by support structures that enable implementation of ON Semiconductor and customer designed custom protocols.

This dual-core architecture is complemented by high-efficiency power management units, oscillators, flash and RAM memories, a DMA controller, along with a full complement of peripherals and interfaces.

2.2 RADIO SYSTEM ARCHITECTURE

The RSL10 SoC is built around an radio system architecture that supports the implementation of Bluetooth and proprietary protocol stacks.

The most common use case for RSL10 devices is in applications that use Bluetooth technology. Accordingly, the RSL10 architecture supports the application structure shown in Figure 2.

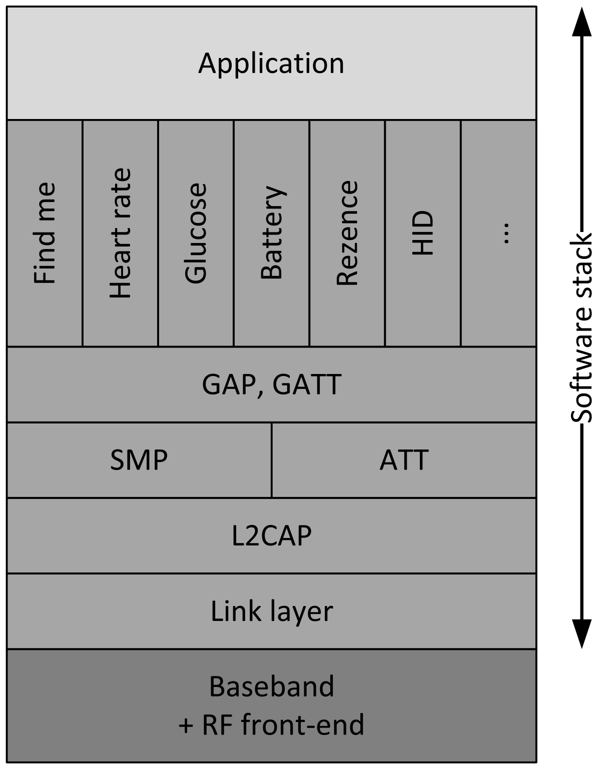


Figure 2. Application Structure if Using a Bluetooth Stack

The radio system architecture for this use case consists of:

- The RF front-end described in Chapter 8, “RF Front-End” on page 131.
- The Bluetooth protocol stack hardware described in Chapter 9, “Bluetooth Low Energy Baseband” on page 201.

- A Bluetooth library that uses the protocol stack hardware to implement a Bluetooth stack (host and controller), as described in the *RSL10 Firmware Reference*'s "Bluetooth Stack and Profiles" chapter.
- A set of Bluetooth profile libraries that use the Bluetooth low energy stack to provide client and/or server functionality to the user application, as described in the *RSL10 Firmware Reference*'s "Bluetooth Stack and Profiles" chapter.
- An event kernel that works within a user application to drive events, as described in the *RSL10 Firmware Reference*'s "Event Kernel" chapter.

For applications that use a proprietary or custom protocol instead of Bluetooth technology or in addition to it, the radio system architecture accesses the RF front-end and the RF front-end's integrated simple baseband directly as shown in Figure 3.

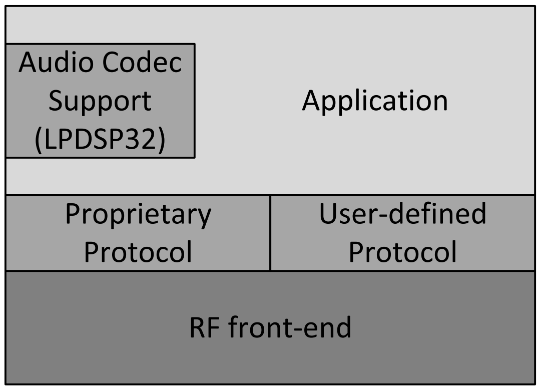


Figure 3. Application Structure if Using a Proprietary or Custom Protocol

The radio system architecture for this use case consists of:

- The RF front-end described in Chapter 8, "RF Front-End" on page 131.
- User-defined (or sample defined) protocol libraries.
- For audio-based applications, support codecs are implemented on the LPDSP32, which is described in Chapter 4, "LPDSP32 Processor" on page 33.
- An event kernel that works within a user application to drive events, as described in the *RSL10 Firmware Reference*'s "Event Kernel" chapter.

2.3 POWER

The RSL10 SoC is supplied from the VBAT pin, with a supply voltage in the range from 1.1 to 3.6 V in typical operating conditions. This supply is regulated by a DC-DC converter consisting of a low dropout voltage regulator (LDO) and a buck converter (can be used for VBAT > 1.4 V). It produces VCC, which is a filtered supply voltage in the range from 1.0 to 1.32 V that is used as the supply for the rest of the system.

Other power supplies include:

VDDRF	Supply voltage for the RF analog blocks

VDDPA	Optional boosted supply voltage for the RF transmitter's power amplifier. We recommend using this supply voltage only if the voltage level required to achieve a specified TX power exceeds the VDDRF supply voltage.
VDDA	Supply voltage for the non-RF analog blocks and flash memory
VDDC	Supply voltage for the digital logic, and for the RF front-end
VDDM	Supply voltage for the memories and memory interfaces
VDDO	Input supply for the digital I/O pads, including the debug port (SWJ-DP)

In addition to the various power supplies, the power components include supply monitoring, analog test points, and reset circuitry that is used to ensure continued correct operation of the overall RSL10 SoC.

NOTE: A variety of power supplies, and clocking or power supply related signals can be routed to the analog test point *AOUT*. This is useful for monitoring or accessing a variety of elements within a system where most circuit elements do not have test points, or are otherwise inaccessible.

For more information, see Chapter 5, "Power" on page 37.

2.4 Clocking

The RSL10 SoC clock trees are based around two independent clocks that can be generated from a variety of sources.

The system clock (SYSCLK) is the main clock for the system, and acts as the source for most components of the system. This includes all of the interfaces, the cores, the power supplies, and all timers. SYSCLK can be derived from the following clock sources:

- Startup RC oscillator
- 48 MHz crystal oscillator
- External Clock pad
- SWJ-DP JCLK pad
- RTC

The real-time clock (RTC) is used to provide a time reference to the system, and to provide a low-frequency option for SYSCLK when operating in Standby Mode. The RTC can be sourced from the following clock sources:

- 32 kHz RC oscillator
- 32 kHz crystal oscillator
- Digital I/O Sources

For more information on clocks and clock sources, see Chapter 6, "Clocking" on page 72.

2.5 Memory

All aspects of the RSL10 SoC, including all memory instances, registers and other components, are accessible from the Arm Cortex-M3 processor through one or more of the processor's standard buses. This structure supports the control and configuration of all of the components of an RSL10 SoC in any system, and simplifies control of the LPDSP32, the RF front-end and Bluetooth protocol baseband hardware.

The memory space of the RSL10 SoC is subdivided into four main segments:

1. The program memory used for storing and/or executing code on the Arm Cortex-M3 processor and/or the LPDSP32. This segment of the RSL10 memory map contains:
   - A 4 KB ROM instance
   - A 384 KB flash instance
   - Three 2 KB flash sectors supporting non-volatile records
   - One 1 KB redundant flash sector supporting configuration information in a non-volatile record from the RSL10 manufacturing test
   - Four 8 KB program RAM instances dedicated to the Arm Cortex-M3 processor
   - Four 10 KB program RAM instances accessible to either the LPDSP32 DSP or the Arm Cortex-M3 processor

2. The data memory used for storing data and intermediate variables of the Arm Cortex-M3 processor, the LPDSP32, and/or the Bluetooth protocol baseband hardware. This segment of the RSL10 memory map contains:
   - Three 8 KB data RAM instances dedicated to the Arm Cortex-M3 processor
   - Six 8 KB data RAM instances accessible to either the LPDSP32 DSP or the Arm Cortex-M3 processor
   - Two 8 KB baseband data RAM instances acting as exchange memory between the Bluetooth protocol baseband hardware and the Arm Cortex-M3 processor

3. The peripheral bus accessible memory-mapped control and configuration registers
4. The private peripheral bus accessible Arm Cortex-M3 processor system registers

A number of elements support these memory components including:

- A direct memory access (DMA) controller module which allows background transfers between peripherals and memory without core intervention
- A flash copier module that can be used to efficiently transfer data from flash to other memories and peripherals in the system
- Several memory arbiters
- Several redundant flash sectors

For more information about the memory structures available and the use of memory in an RSL10-based system, see Chapter 7, “Memory” on page 93.

2.6 Interfaces

The RSL10 SoC supports a number of interfaces that can be used to communicate with external devices including other SoCs, support components such as external non-volatile storage, and sensors.

The RSL10 system uses a DIO (Digital Input/Output) concept, where any DIO pad can be configured in software at any time to connect with any interface input or output. The only exception is the debug SWD-JTAG interface, which has dedicated pads for the serial-wire inputs. A maximum of 16 DIOs are available depending on the package used, with four DIOs (DIO0 to DIO3) supporting use as analog-to-digital converter inputs, and three DIOs supporting use as the additional pads needed to use the SWJ-DP interface in 4-wire JTAG mode (DIO14, DIO15) or 5-wire JTAG mode (DIO13 to DIO15). The DIO concept allows for full flexibility because PCB routing and layout can be optimized for size, cost, and complexity.

By default, all the DIOs are configured to be disabled. In the different power modes, you can enable a DIO pad’s state retention, where the pad configuration is retained even though the digital core is powered down.

For more information about DIO configuration, see Chapter 10, “Digital Input/Output” on page 259.

The interfaces that are supported by the DIOs are:

- One I^2C interface
- Four external inputs to the analog-to-digital converters (ADC)
- One PCM interface
- Two PWM drivers
- Two SPI interfaces
- One UART interface
- Support interfaces that can be used to monitor control of the RF front-end and Bluetooth baseband

In addition to these interface options, each DIO can act as a general-purpose I/O, where the processor can read the input or set the output at any time.

For more information about these interfaces, see Chapter 11, “External Digital Interfaces” on page 301.

2.7 PERIPHERALS

The Arm Cortex-M3 processor has a number of peripherals to support auxiliary non-RF tasks that it needs to perform. The Arm Cortex-M3 processor peripherals include:

- A standard nested-vectored interrupt controller (NVIC)
- A system timer (SYSTICK)
- A watchdog timer
- Four general-purpose timers
- A CRC calculation unit
- A direct-memory access (DMA) unit

For more information about the peripherals, see Chapter 12, “Peripherals” on page 350 and Chapter 14, “Private Peripherals” on page 400.

2.8 AUDIO COMPONENTS

The RSL10 SoC provides a set of interfaces and peripherals to assist in applications that have an audio component. This includes:

- An Asynchronous Sample Rate Converter (ASRC)
- A set of audio sink clock counters (used to measure the timing of samples received from an audio source)
- Two digital microphone (DMIC) input channels
- One mono output driver

For more information about the audio support components, see Chapter 13, “Audio” on page 378.

2.9 SOC IDENTIFICATION

To distinguish between different RSL10 revisions, a 32-bit register at address 0x1FFFFFFC contains a static value that can be used to provide version information about the system. The value in this register (called `AHBREGS_CHIP_ID_NUM`), can also be used to verify compatibility with application software and external applications. A list of the bit fields in the device identification register, along with a brief description of each, is provided in Table 1.

Table 1. Microcontroller Identification Register

Field Name	Description	Byte
AHBREGS_CHIP_ID_NUM_CHIP_FAMILY	The chip technology family to which the microcontroller belongs.	3
AHBREGS_CHIP_ID_NUM_CHIP_VERSION	Version number for the microcontroller. Used to indicate major updates that might not be backwards compatible.	2
AHBREGS_CHIP_ID_NUM_CHIP_MAJOR_REVISION	An update of the major revision number indicates updates that could affect source code or binary objects, and thus could require firmware library or software updates. When the major revision number is updated, the minor revision number is reset.	1
AHBREGS_CHIP_ID_NUM_CHIP_MINOR_REVISION	An update of the minor revision number indicates minor backwards-compatible or non-functional updates to the system that require no update to source code or binary objects.	0, bits 7:3
-	Unused feature bit (reserved for future use).	0, bit 2
-	Unused feature bit (reserved for future use).	0, bit 1
AHBREGS_CHIP_ID_NUM_AOBLE_FEATURE	Feature identification bits indicating the available features of the device.	0, bit 0

The chip version is commonly written as X.YY.ZZ (for example, 1.02.03), where X is the chip version number, YY is the major revision number, and ZZ is the minor revision number and features.

2.9.1 Device Identification Register

Register Name	Register Description	Address
AHBREGS_CHIP_ID_NUM	Chip ID number	0x1FFFFFFC

2.9.1.1 AHBREGS_CHIP_ID_NUM

Bit Field	Field Name	Description
31:24	CHIP_FAMILY	Chip Family number
23:16	CHIP_VERSION	Chip Version number
15:8	CHIP_MAJOR_REVISION	Chip Major Revision number
7:3	CHIP_MINOR_REVISION	Chip Minor Revision number
2	-	Unused feature bit (reserved for future use).
1	-	Unused feature bit (reserved for future use).
0	AOBLE_FEATURE	Hardware needed to support AOBLE feature

Field Name	Value Symbol	Value Description	Hex Value
AOBLE_FEATURE	CHIP_ID_AOBLE_NOT_PRESENT	Device does not have the Audio over Bluetooth low energy feature	0x0
	CHIP_ID_AOBLE_PRESENT	Device has the Audio over Bluetooth low energy feature	0x1

CHAPTER 3

Arm Cortex-M3 Processor

3.1 INTRODUCTION

The Arm Cortex-M3 processor plays a role as the central controller for the RSL10 microcontroller system.The Arm Cortex-M3 processor provides users with an interface for configuring and controlling all of the other system components. Following a power-on reset (POR), the system starts executing the Boot ROM on the Arm Cortex-M3 processor, and uses this ROM application to validate and initialize a system application. It is also closely coupled to many of the memory components, and as a result, has access to all of the data that is being processed by the system. For more information about the Boot ROM code, see the *RSL10 Firmware Reference*.

The Arm Cortex-M3 processor implements the Armv7-M architecture. It has the following main features:

- Thumb-2 (ISA) subset consisting of all base Thumb-2 instructions, both 16-bit and 32-bit
- Harvard processor architecture enabling simultaneous instruction fetches with data load or store
- Three-stage pipeline
- Single cycle 32-bit multiply
- Hardware divide
- Thumb and debug states
- Handler and thread modes
- Low latency interrupt subroutine (ISR) entry and exit
- Processor state saving and restoration, with no instruction fetch overhead; the exception vector is fetched from memory in parallel with the state saving, enabling faster ISR entry
- Support for late-arriving interrupts
- Tightly coupled interface to interrupt controller, enabling efficient processing of late-arriving interrupts
- Tail-chaining of interrupts, enabling back-to-back interrupt processing without the overhead of state saving and restoration between interrupts
- Interruptible-continued LDM/STM, and PUSH/POP
- Armv6 style BE8/LE support
- Armv6 unaligned
- Registers:
    - 13 general-purpose 32-bit registers
    - Link Register (LR)
    - Program Counter (PC)
    - Program Status Register (xPSR)
    - Two banked SP registers

To reduce the current consumption from fetching instructions, the Arm Cortex-M3 processor is supported by a 32-instruction loop cache that can be used to cover a continuous block of reads. To enable this cache, set the `CSS_LOOP_CACHE_ENABLE` bit from the `SYSCTRL_CSS_LOOP_CACHE_CFG` register.

3.1.1 Arm Cortex-M3 Processor Loop Cache Register

Register Name	Register Description	Address
`SYSCTRL_CSS_LOOP_CACHE_CFG`	CSS Loop Cache Configuration	0x4000000C

3.1.1.1 SYSCTRL_CSS_LOOP_CACHE_CFG

Bit Field	Field Name	Description
0	`CSS_LOOP_CACHE_ENABLE`	CSS loop cache enable

Field Name	Value Symbol	Value Description	Hex Value
`CSS_LOOP_CACHE_ENABLE`	`CSS_LOOP_CACHE_DISABLE`	CSS loop cache disabled	0x0*
	`CSS_LOOP_CACHE_ENABLE`	CSS loop cache enabled	0x1

3.2 Debug Port

All applications executing on the Arm Cortex-M3 processor can be debugged through the SWJ-DP which can be configured to either serial wire or JTAG debug port communications. The SWJ-DP provides access to all of the Arm Cortex-M3 processor registers, and to the memory available through the memory buses attached to the Arm Cortex-M3 processor. This transparent view of the system enables a user to review the current state of the system components through their associated memory-mapped control registers, and use this information to troubleshoot most issues in an application.

By default, the SWJ-DP is accessed using the JTAG connection that uses DIOs 13 to 15, in addition to two dedicated pads (`JTCK`, `JTMS`), to form a 5-wire JTAG interface. If a user prefers to use this interface in 2-pin serial wire (SW) mode instead, the dedicated pads can be used to re-initialize the interface to be the needed SWCLK and SWDIO pads, and DIOs 13 to 15 can be reassigned to other functionality. For more information about the functional configuration of DIOs for this mode, see Section 10.2.1, “Special Functional Configurations” on page 262.

When using the SWJ-DP in serial wire mode, data is transferred using a single clock and single data signal. This interface mode is useful when the number of debug port connections needs to be limited, but it is only half-duplex and has a relatively high overhead to transfer data. When using the SWJ-DP in JTAG mode, data is transferred using a data control line, a pair of data lines, a clock, and an optional reset signal. This interface mode is useful when the number of debug port connections is not limited, and a higher throughput is required, as JTAG mode uses less data overhead than SW mode, full-duplex transfers.

To configure the DIOs to support the SWJ-DP interface in JTAG mode, make the following configurations before configuring the interface for JTAG mode:

- The JTAG test-reset signal (JNTRST) can be connected to DIO 13 by setting the `CM3_JTAG_TRST_EN` bit from the `DIO_JTAG_SW_PAD_CFG` register; this is only required if using the SWJ-DP as a 5-pin JTAG interface.
- The JTAG data signals can be connected to DIOs 14 (JTDI) and 15 (JTDO) by setting the `CM3_JTAG_DATA_EN` bit from the `DIO_JTAG_SW_PAD_CFG` register; this is required if you are using the SWJ-DP as a 4-pin or 5-pin JTAG interface.

To switch the SWJ-DP into JTAG mode, follow this initialization sequence:

1. Switch the debug port into serial wire mode using the previous sequence.
2. Use SWCLK to send 0xE73C (transmitted LSB first) using SWDIO. This initialization pattern switches the debug port to JTAG mode.
3. Send at least 5 SWCLK cycles with SWDIO HIGH. This ensures that the JTAG interface enters its test-logic reset state.

To switch the SWJ-DP into serial wire mode, follow this initialization sequence:

1. Send at least 50 clock cycles on SWCLK with the SWDIO pad held high. This ensures that the current interface is in its reset state.
2. Use SWCLK to send 0xE79E (transmitted LSB first) using SWDIO. This initialization pattern switches the debug port to serial wire mode.
3. Send at least 50 SWCLK cycles with SWDIO HIGH. This ensures that the serial wire interface enters its line reset state.

The pads that form the SWJ-DP interface support a limited amount of physical configuration. The output pads support a configurable drive strength, and input pads support configurable pull-up resistances. Configuration of these interface pads uses the `DIO_JTAG_SW_PAD_CFG` register. The physical configuration parameters are configured as follows:

- JTMS (SWDIO) pad configuration:
  - A pull-up or pull-down resistor can be configured and applied to the pad using the `DIO_JTAG_SW_PAD_CFG_JTMS_PULL` bit-field
  - The drive strength can be configured using the `DIO_JTAG_SW_PAD_CFG_JTMS_DRIVE` bit-field
  - A low-pass filter can be enabled or disabled using the `DIO_JTAG_SW_PAD_CFG_JTMS_LPF` bit

- JTCK (SWCLK) pad configuration:
  - A pull-up or pull-down resistor can be configured and applied to the pad using the `DIO_JTAG_SW_PAD_CFG_JTCK_PULL` bit-field
  - A low-pass filter can be enabled or disabled using the `DIO_JTAG_SW_PAD_CFG_JTCK_LPF` bit

- Physical configuration of the DIOs are used even when those pads are configured for use as part of the JTAG interface

The debug port provides external access to the standard Arm Cortex-M3 core debug controller that is on the private peripheral bus. For more information about the debug controller, see Section 14.3, “Debug Controller” on page 410.

3.3 IP PROTECTION

The Arm Cortex-M3 debug port includes a restricted access mode that provides protection for intellectual property which takes the form of application source code. When the debug port is in restricted access mode, communications using the Arm Cortex-M3 debug port are limited to requests and instructions that do not access any of the system memory buses. The only exception to this rule are the `SYSCTRL_DBG_UNLOCK_KEY` registers that can be written (but not read) by the debug port while it is in a locked state.

To restrict accesses through the debug port, the `SYSCTRL_DBG_LOCK` register can be written with the debug port lock key (0x4C6F634B). At startup and whenever the RSL10 system is reset, the debug port defaults to this locked status. After start-up and system initialization, the Program ROM then loads the value from the word stored to LOCK_INFO_SETTING in the flash NVR3 sector into the `SYSCTRL_DBG_LOCK` register. After loading this value:

- If the LOCK_INFO_SETTING in the flash contains the debug port lock key, the debug port continues to operate in restricted mode unless explicitly unlocked by a user application.
- If the LOCK_INFO_SETTING in the flash does not contain the debug port lock key, the debug port is switched to operate in Normal Mode and the SWJ-DP debug port responds to all requests. In this mode, the

debug port has access to all system resources that are typically accessible in an Arm Cortex-M3 core debug environment.

CAUTION: When the Program ROM reads the LOCK_INFO_SETTING, if the value read back is 0x00000000 or 0xFFFFFFFF it will temporarily boost the VDDM supply voltage up to a maximum of 1.25 V to ensure that the value read is correct. This can cause the system not to boot for low VBAT levels. To avoid this failure under these conditions, set the `LOCK_INFO_SETTING` to a non-zero, non-one value (typically `DBG_ACCESS_LOCK` or `DBG_ACCESS_UNLOCK`).

In addition to the `SYSCTRL_DBG_LOCK` register, the Arm Cortex-M3 debug port can be unlocked using an application-defined 128-bit user key. This key is stored in the four `SYSCTRL_DBG_LOCK_KEY` registers. The contents of the `SYSCTRL_DBG_LOCK_KEY` registers are compared against the values stored to the `SYSCTRL_DBG_UNLOCK_KEY` registers, and the `SYSCTRL_DBG_LOCK` register is cleared to unlock the debug port if the `SYSCTRL_DBG_LOCK_KEY` is valid and the two keys match.

3.3.1 Using IP Protection with Segger J-LInk

As an example of how to use the IP protection feature of RSL10, the following steps show to use the debug port lock/unlock mechanism via Segger J-Link Commander commands.

There are three operations to using the debug port lock feature:

1. Writing a key
2. Locking
3. Unlocking

Normally once the application firmware is loaded, steps 1 and 2 should be executed to lock the debug port.

Step 3 can be used to unlock the device for debugging or loading new firmware. Note that RSL10 will re-lock on next reset unless the `LOCK_INFO_KEY` field is also erased from flash.This also means that erasing the `LOCK_INFO_KEY` field disables the lock feature.

The corresponding J-Link Commander commands for these operations are described below. In these examples, **`<wordX>`** are each 32-bit words of the lock key in sequential order.

1. Writing a key:

```
// Write key to LOCK_INFO_KEY in flash
w4 0x00081044 <word0> <word1> <word2> <word3>
```

2. Locking:

```
// Write lock word to LOCK_INFO_SETTING in flash
w4 0x00081040 0x4C6F634B
```

3. Unlocking:

```
// Initiate SWD interface
SWDSelect
// Write to DP[2] to set APSEL to 0 (AHB-AP), and AHBBANKSEL to 0
SWDWriteDP 2 0
// Write to AP[0] (CSW) to set size to 32 bit and enable auto-increment
SWDWriteAP 0 0x23000052
```

```
// Set AP[1] (TAR) = 0x400000F0 (SYSCTRL_DBG_UNLOCK_KEY[0])
SWDWriteAP 1 0x400000F0
// Write key word 0 to AP[3] (DRW)
SWDWriteAP 3 <word0>
// Write key word 1 to AP[3] (DRW)
SWDWriteAP 3 <word1>
// Write key word 2 to AP[3] (DRW)
SWDWriteAP 3 <word2>
// Write key word 3 to AP[3] (DRW)
SWDWriteAP 3 <word3>
```

IMPORTANT: Sometimes a user wants to reprogram a device without exposing the application IP. To clear all flash and RAM memory before unrestricting the debug port using the `SYSCTRL_DBG_LOCK` register, the user application can use the Program ROM `Sys_ProgramROM_UnlockDebug()` function which erases all flash memory except for the NVR3 and NVR4 sectors, and clears all RAM memory instances before unlocking the debug port. For more information, see the *RSL10 Firmware Reference*.

3.3.2 IP Protection Registers

Register Name	Register Description	Address
SYSCTRL_DBG_LOCK	Debug Port Access Configuration	0x400000DC
SYSCTRL_DBG_LOCK_KEY_0	Debug Port Lock Key Part 0	0x400000E0
SYSCTRL_DBG_LOCK_KEY_1	Debug Port Lock Key Part 1	0x400000E4
SYSCTRL_DBG_LOCK_KEY_2	Debug Port Lock Key Part 2	0x400000E8
SYSCTRL_DBG_LOCK_KEY_3	Debug Port Lock Key Part 3	0x400000EC
SYSCTRL_DBG_UNLOCK_KEY_0	Debug Port Unlock Key Part 0	0x400000F0
SYSCTRL_DBG_UNLOCK_KEY_1	Debug Port Unlock Key Part 1	0x400000F4
SYSCTRL_DBG_UNLOCK_KEY_2	Debug Port Unlock Key Part 2	0x400000F8
SYSCTRL_DBG_UNLOCK_KEY_3	Debug Port Unlock Key Part 3	0x400000FC

3.3.2.1 SYSCTRL_DBG_LOCK

Bit Field	Field Name	Description
31:0	DBG_LOCK_WR	Debug port access lock/unlock
0	DBG_LOCK_RD	Debug port access state

Field Name	Value Symbol	Value Description	Hex Value
DBG_LOCK_WR	DBG_ACCESS_UNLOCK	Unlock debug port access	0xB3909CB5
	DBG_ACCESS_LOCK	Lock debug port access	0x4C6F634B
DBG_LOCK_RD	DBG_ACCESS_UNLOCKED	Debug port access is unlocked	0x0
	DBG_ACCESS_LOCKED	Debug port access is locked	0x1*

3.3.2.2 SYSCTRL_DBG_LOCK_KEY

Bit Field	Field Name	Description
31:0	`DBG_LOCK_KEY`	Debug port lock key

3.3.2.3 SYSCTRL_DBG_UNLOCK_KEY

Bit Field	Field Name	Description
31:0	`DBG_UNLOCK_KEY`	Debug port unlock key

3.4 Activity Counters

The activity counters help to analyze how long the system has been running and how actively the Arm Cortex-M3 processor, the LPDSP32 DSP, and the flash memory have been used by the user application. This information is useful to estimate and optimize the power consumption of the application.

To gauge how many cycles have elapsed, a reference counter that counts SYSCLK cycles is accessed through the `SYSCTRL_SYSCLK_CNT` register. This cycle count can be compared with cycle counts for three other critical system elements:

- Execution on the Arm Cortex-M3 processor is recorded in the `SYSCTRL_CM3_CNT` register. This counter only counts when the Arm Cortex-M3 processor is running, and does not increment when the core is sleeping.
- Execution on the LPDSP32 processor is recorded in the `SYSCTRL_LPDSP32_CNT` register. This counter only counts when the LPDSP32 processor is running, and does not increment when the core is paused or sleeping.
- Read accesses from the flash memory are tracked in the `SYSCTRL_FLASH_READ_CNT` register.

The activity counters are configured and controlled using the `SYSCTRL_CNT_CTRL` register. These counters are:

- Cleared by setting the `SYSCTRL_CNT_CTRL_CNT_CLEAR` bit
- Started by setting the `SYSCTRL_CNT_CTRL_CNT_START` bit
- Stopped by setting the `SYSCTRL_CNT_CTRL_CNT_STOP` bit

When the counters are tracking activity, the `SYSCTRL_CNT_CTRL_CNT_STATUS` bit is set to `CNT_RUNNING`.

3.4.1 Registers

Register Name	Register Description	Address
`SYSCTRL_CNT_CTRL`	Activity Counters Control	0x40000030
`SYSCTRL_SYSCLK_CNT`	System Clock Counter Value	0x40000034
`SYSCTRL_CM3_CNT`	Arm Cortex-M3 core Activity Counter Value	0x40000038
`SYSCTRL_LPDSP32_CNT`	LPDSP32 Activity Counter Value	0x4000003C
`SYSCTRL_FLASH_READ_CNT`	Flash Read Access Counter Value	0x40000040

3.4.1.1 SYSCTRL_CNT_CTRL

Bit Field	Field Name	Description
3	CNT_STATUS	Activity counters status bit
2	CNT_CLEAR	Clear activity counters
1	CNT_STOP	Stop activity counters
0	CNT_START	Start activity counters

Field Name	Value Symbol	Value Description	Hex Value
CNT_STATUS	CNT_STOPPED	Activity counters stopped	0x0*
	CNT_RUNNING	Activity counters running	0x1
CNT_CLEAR	CNT_CLEAR	Clear activity counters	0x1
CNT_STOP	CNT_STOP	Stop activity counters	0x1
CNT_START	CNT_START	Start activity counters	0x1

3.4.1.2 SYSCTRL_SYSCLK_CNT

Bit Field	Field Name	Description
31:0	SYSCLK_CNT	System clock counter value

3.4.1.3 SYSCTRL_CM3_CNT

Bit Field	Field Name	Description
31:0	CM3_CNT	Arm Cortex-M3 core activity counter value

3.4.1.4 SYSCTRL_LPDSP32_CNT

Bit Field	Field Name	Description
31:0	LPDSP32_CNT	LPDSP32 activity counter value

3.4.1.5 SYSCTRL_FLASH_READ_CNT

Bit Field	Field Name	Description
31:0	FLASH_READ_CNT	Flash read access counter value

CHAPTER 4

LPDSP32 Processor

4.1 Overview

The LPDSP32 DSP is a low power, programmable, pipelined DSP that uses a dual-Harvard, dual-MAC architecture to efficiently process 32-bit signal data. This processor is included in the design to efficiently support digital signal processing tasks, such as audio codecs that might be required for wireless audio communication tasks and other advanced developments requiring the additional processing power that this core provides.

To reset the LPDSP32 core, its interrupt handler, and the LPDSP32 loop cache, write `DSS_RESET` to the `SYSCTRL_DSS_CTRL_DSS_RESET` bit from the `SYSCTRL_DSS_CTRL` register.

4.2 System Integration

The LPDSP32 does not have a boot ROM. Instead it relies on the Arm Cortex-M3 processor to initialize its memories and peripherals.

To pause and resume the LPDSP32 processing, set the `SYSCTRL_DSS_CTRL_LPDSP32_PAUSE` and `SYSCTRL_DSS_CTRL_LPDSP32_RESUME` bits respectively in the `SYSCTRL_DSS_CTRL` register. The `SYSCTRL_DSS_CTRL_LPDSP32_RUNNING` bit from `SYSCTRL_DSS_CTRL` indicates the current state (running or paused) of the LPDSP32.

When the LPDSP32 core is paused, the clock provided to the core is stopped, which prevents the execution of any functions and all responses to interrupts generated by the DMA or Arm Cortex-M3 processor. All incomplete accesses to any memory by the LPDSP32 core are put into a non-blocking wait state, and these accesses are completed once processing resumes on the LPDSP32 core. While the LPDSP32 core is paused, the Arm Cortex-M3 core can access shared memory resources at any address without the LPDSP32 introducing additional delays.

4.3 Memory Support

The LPDSP32 is supported by data memory and program memory, as defined in Section 7.2.5, “LPDSP32 DSP Memory Usage” on page 102.

The LPDSP32 program memory uses 40-bit words, and is located in RAM. When loading the DSP’s program memories from flash memory, use the flash copier to efficiently handle the transfer between 32-bit and 40-bit words (see Section 7.4, “Flash Copier” on page 108).

If the LPDSP32 is not using one or more of its assigned memory instances, the unused memory instances can be reassigned in the firmware and the linker configuration scripts to be used by the Arm Cortex-M3 processor and employed elsewhere in the user’s application.

To reduce the current consumption from fetching instructions, the LPDSP32 is supported by a 32-word loop cache that can be used to cover a continuous block of reads. To enable this cache, set the `DSS_LOOP_CACHE_ENABLE` bit from the `SYSCTRL_DSS_LOOP_CACHE_CFG` register.

4.4 Interrupt Handling

To coordinate execution of data-processing functions on the LPDSP32 processor with data that is available to be processed, the LPDSP32 is supported by two sets of interrupts that can trigger execution on the LPDSP32:

1. Each DMA channel can issue one application-defined command, triggered by the DMA channel's enabled interrupts. To clear all pending DMA triggered interrupts to the LPDSP32, set the `SYSCTRL_DSS_CTRL_DSS_DMA_INT_RESET` bit from the `SYSCTRL_DSS_CTRL` register.
2. The Arm Cortex-M3 processor can issue one of seven application-defined commands by setting the appropriate bit in the `SYSCTRL_DSS_CMD` register. To clear all pending Arm Cortex-M3 core triggered interrupts to the LPDSP32, set the `SYSCTRL_DSS_CTRL_DSS_CSS_INT_RESET` bit from the `SYSCTRL_DSS_CTRL` register.

The LPDSP32 interrupt vector table is provided in Table 2. The LPDSP32 does not support pre-emption of interrupt handler execution. If more than one interrupt is pending when execution completes on a previous interrupt handler, the handler for the lowest numbered pending interrupt is selected for execution.

Table 2. LPDSP32 Interrupt Vectors

Interrupt Number	Description
0	DMA channel 0 interrupt
1	DMA channel 1 interrupt
2	DMA channel 2 interrupt
3	DMA channel 3 interrupt
4	DMA channel 4 interrupt
5	DMA channel 5 interrupt
6	DMA channel 6 interrupt
7	DMA channel 7 interrupt
8	Arm Cortex-M3 core triggered interrupt 0
9	Arm Cortex-M3 core triggered interrupt 1
10	Arm Cortex-M3 core triggered interrupt 2
11	Arm Cortex-M3 core triggered interrupt 3
12	Arm Cortex-M3 core triggered interrupt 4
13	Arm Cortex-M3 core triggered interrupt 5
14	Arm Cortex-M3 core triggered interrupt 6

When any function completes, or at any other logical intermediate point, the LPDSP32 processor can send the Arm Cortex-M3 processor an interrupt through one of eight LPDSP32 interrupts (`DSP0_IRQn`). These interrupts are intended to notify the Arm Cortex-M3 core when the data processing on the LPDSP32 is complete, or to signal when the LPDSP32 is ready to receive requests from the Arm Cortex-M3 core for additional processing.

4.5 LPDSP32 DEBUG PORT

LPDSP32 is supported by a 4-wire JTAG debug port that is mapped onto the DIOs. For information on assigning DIOs to be used as the LPDSP32 debug port, refer to Chapter 10, "Digital Input/Output" on page 259. For proper operation of this debug port, all four signals must be mapped to DIOs.

To enable the LPDSP32 debug port, set the `SYSCTRL_LPDSP32_DEBUG_CFG_LPDSP32_DEBUG_ENABLE` bit from the `SYSCTRL_LPDSP32_DEBUG_CFG` register. To allow the LPDSP32 debug port to force the core into an enabled state when halted over the debug port, set the `SYSCTRL_LPDSP32_DEBUG_CFG_LPDSP32_EXIT_POWERDOWN_WHEN_HALTED` from the `SYSCTRL_LPDSP32_DEBUG_CFG` register.

4.6 Registers

Register Name	Register Description	Address
SYSCTRL_DSS_CTRL	DSS Control	0x40000000
SYSCTRL_DSS_CMD	DSS Commands	0x40000004
SYSCTRL_DSS_LOOP_CACHE_CFG	DSS Loop Cache Configuration	0x40000010
SYSCTRL_LPDSP32_DEBUG_CFG	LPDSP32 Debug Port Configuration	0x4000004C

4.6.1 SYSCTRL_DSS_CTRL

Bit Field	Field Name	Description
5	DSS_CSS_INT_RESET	Write a 1 to reset pending CSS interrupts in the DSS interrupt controller
4	DSS_DMA_INT_RESET	Write a 1 to reset pending DMA interrupts in the DSS interrupt controller
3	DSS_RESET	Write a 1 to reset DSS
2	LPDSP32_PAUSE	Write a 1 to pause LPDSP32
1	LPDSP32_RESUME	Write a 1 to run LPDSP32
0	LPDSP32_RUNNING	LPDSP32 running status

Field Name	Value Symbol	Value Description	Hex Value
DSS_CSS_INT_RESET	DSS_CSS_INT_RESET	Reset CSS interrupts in the DSS interrupt controller	0x1
DSS_DMA_INT_RESET	DSS_DMA_INT_RESET	Reset DMA interrupts in the DSS interrupt controller	0x1
DSS_RESET	DSS_RESET	Reset DSS	0x1
LPDSP32_PAUSE	DSS_LPDSP32_PAUSE	Pause LPDSP32	0x1
LPDSP32_RESUME	DSS_LPDSP32_RESUME	Resume LPDSP32	0x1
LPDSP32_RUNNING	DSS_LPDSP32_STATE_PAUSE	LPDSP32 paused	0x0*
	DSS_LPDSP32_STATE_RUN	LPDSP32 running	0x1

4.6.2 SYSCTRL_DSS_CMD

Bit Field	Field Name	Description
6	DSS_CMD_6	Write a 1 to issue DSS command 6
5	DSS_CMD_5	Write a 1 to issue DSS command 5
4	DSS_CMD_4	Write a 1 to issue DSS command 4
3	DSS_CMD_3	Write a 1 to issue DSS command 3
2	DSS_CMD_2	Write a 1 to issue DSS command 2
1	DSS_CMD_1	Write a 1 to issue DSS command 1
0	DSS_CMD_0	Write a 1 to issue DSS command 0

Field Name	Value Symbol	Value Description	Hex Value
DSS_CMD_6	DSS_CMD_6	Issue DSS command 6	0x1
DSS_CMD_5	DSS_CMD_5	Issue DSS command 5	0x1
DSS_CMD_4	DSS_CMD_4	Issue DSS command 4	0x1
DSS_CMD_3	DSS_CMD_3	Issue DSS command 3	0x1
DSS_CMD_2	DSS_CMD_2	Issue DSS command 2	0x1
DSS_CMD_1	DSS_CMD_1	Issue DSS command 1	0x1
DSS_CMD_0	DSS_CMD_0	Issue DSS command 0	0x1

4.6.3 SYSCTRL_DSS_LOOP_CACHE_CFG

Bit Field	Field Name	Description
0	DSS_LOOP_CACHE_ENABLE	DSS loop cache enable

Field Name	Value Symbol	Value Description	Hex Value
DSS_LOOP_CACHE_ENABLE	DSS_LOOP_CACHE_DISABLE	DSS loop cache disabled	0x0
	DSS_LOOP_CACHE_ENABLE	DSS loop cache enabled	0x1*

4.6.4 SYSCTRL_LPDSP32_DEBUG_CFG

Bit Field	Field Name	Description
1	LPDSP32_EXIT_POWERDOWN_WHEN_HALTED	LPDSP32 exit power down mode configuration when halted
0	LPDSP32_DEBUG_ENABLE	LPDSP32 debug port enable

Field Name	Value Symbol	Value Description	Hex Value
LPDSP32_EXIT_POWERDOWN_WHEN_HALTED	LPDSP32_EXIT_POWERDOWN_WHEN_HALTED_DISABLED	LPDSP32 exit power down when halted disabled	0x0*
	LPDSP32_EXIT_POWERDOWN_WHEN_HALTED_ENABLED	LPDSP32 exit power down when halted enabled	0x1
LPDSP32_DEBUG_ENABLE	LPDSP32_DEBUG_DISABLED	LPDSP32 debug port disabled	0x0*
	LPDSP32_DEBUG_ENABLED	LPDSP32 debug port enabled	0x1

CHAPTER 5

Power

5.1 POWER SUPPLY OVERVIEW

The power supply is a critical component of the RSL10 system. Supplied power has significant effects on both RF and other types of system performance.

The components that make up the power supply can be divided into two types of supply voltages:

1. Power supply input voltages, described further in Section 5.2, “Power Supply Inputs”
2. Internal power supply voltages, described further in Section 5.3, “Internal Power Supply Voltages”

The power supply tree is powered by the system supply voltage (`VCC`), which is sourced from one of two supplies:

- Directly from the battery supply voltage (`VBAT`)
- Indirectly from the battery supply voltage, through the DC-DC buck converter or the internal LDO regulator

The system supply voltage is used as the source for a number of internal supply voltages, including:

- A regulated, low-noise voltage bandgap reference supply for the analog components
- Two configurable regulated supplies for digital components (`VDDC`, `VDDM`)
- Two configurable regulated Retention Mode supplies for retaining the digital component state in Sleep Power Mode (`VDDC_RET`, `VDDM_RET`)
- A configurable regulated supply for the RF front-end (`VDDRF`)
- An on-chip charge pump for the RF front-end power amplifier, for cases where the TX power requirements exceed the voltage that can be supplied using `VDDRF` (`VDDPA`)
- An on-chip charge pump for the other analog system components (`VDDA`)

Figure 4 on page 38 illustrates the RSL10 power supply and related components at a high level.

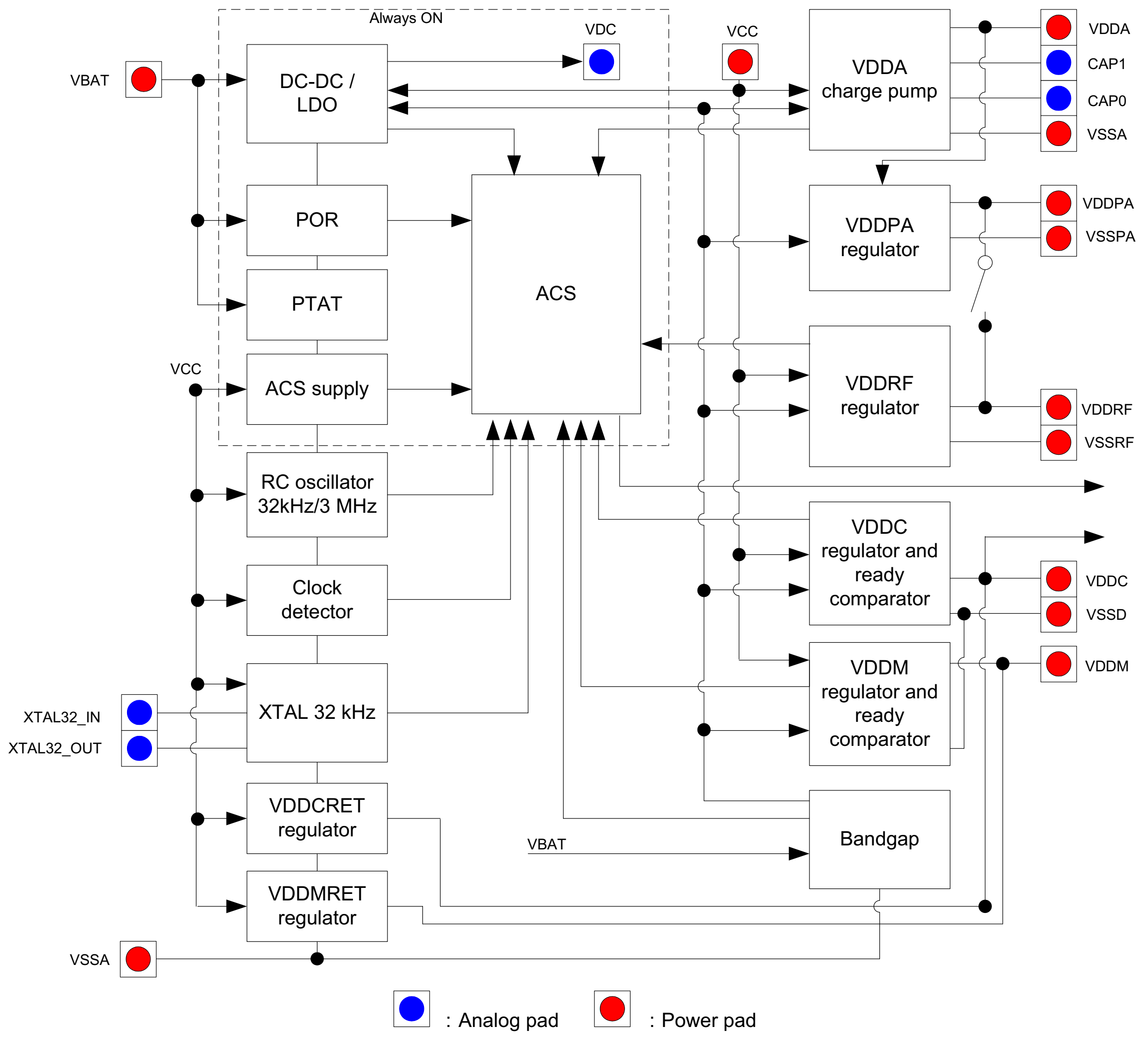


Figure 4. Power Supply and POR Top Level Block Diagram

5.1.1 Power Management Unit

The power management unit contains:

- The power supplies, as described in Section 5.1, “Power Supply Overview”
- The power supervisory blocks and reset blocks, as described in Section 5.5, “Resets” on page 63
- Blocks that support configuring the power supplies, and supervisory blocks for use in different power modes and under different wakeup conditions, as described in Section 5.4, “Power Modes” on page 50

5.2 Power Supply Inputs

5.2.1 Battery Supply Voltage (VBAT)

The primary voltage supplied to an RSL10 SoC is the battery supply voltage. This supply is used as the source for:

- The VCC supply (see Section 5.3.1, “System Supply Voltage (VCC)”), which in turn is used as the source supply for most other power supply components
- The Power-On Reset (POR) block (see Section 5.5, “Resets” on page 63)

IMPORTANT: When laying out the RSL10 device as part of a system care should be taken to place the VBAT decoupling capacitors close to the VBAT input, and to place the VBAT and DCDC capacitors so that their grounds are relatively close together and well connected, ideally on the surface, as this minimizes return paths.

IMPORTANT: To ensure proper system behavior and IP security over a wide range of conditions, the Program ROM uses a read of the `LOCK_INFO_SETTING` (0x00081040) to determine if the VDDM supply voltage is sufficient for all memory reads. If this setting contains 0x00000000 or 0xFFFFFFFF, this results in VDDM being temporarily raised to 1.25 V during boot, which will fail for low VBAT conditions. To avoid this failure under these conditions, set the `LOCK_INFO_SETTING` to a non-zero, non-one value (typically `DBG_ACCESS_LOCK` or `DBG_ACCESS_UNLOCK`).

5.2.2 Digital Output Supply Voltage (VDDO)

The digital output supply voltage is attached to the digital I/O pads on an RSL10 SoC. This includes:

- All DIO pads, as described in Chapter 10, “Digital Input/Output” on page 259
- The JTCK and JTMS pads for the SWJ-DP debug port. For more information, see Section 3.2, “Debug Port” on page 27.
- The EXTCLK input pad, described in Section 6.2, “Clock Generation” on page 74
- The WAKEUP pad, described in Section 5.4, “Power Modes” on page 50
- The NRESET pad, described in Section 5.5, “Resets” on page 63

The internal system digital logic is attached to the pads through internal level translators. They shift the voltage level from VDDC to the correct VDDO voltage for digital outputs, and translate input digital signals from the correct VDDO voltage to VDDC for digital inputs.

The VDDO inputs are usually connected externally to VBAT or VDDA, based on the desired voltages of digital communications using the associated digital I/O pads.

5.3 Internal Power Supply Voltages

5.3.1 System Supply Voltage (VCC)

The System Supply Voltage (VCC) is used as the source for all of the internally generated supply voltages in the RSL10 SoC, and is supplied from VBAT. This power supply is used to reduce the battery voltage from a high voltage range (from 1.1 to 3.6 V) down to a supply voltage in the range from 1.0 to 1.32 V.

The voltages supplied by the VCC are configured using the `ACS_VCC_CTRL_VTRIM` bit-field from the `ACS_VCC_CTRL` register. This trim setting defines a VCC target, with VCC being supplied directly from the battery, or from either the internal LDO or DC-DC converters. Conditions for each of the different VCC supply configurations can be found in Table 3:

Table 3. VCC Supply Configuration

VCC Trim Configuration	VBAT Supply and Mode	VCC Supply Level	Description
VCC target ≥ VBAT	-	VCC = VBAT	VCC supplied from VBAT
VCC target < VBAT	VBAT < 1.4 V	VCC = VCC target	VCC supplied through a low drop out regulator (LDO) from VBAT
VCC target < VBAT	VBAT > 1.4 V (LDO mode)	VCC = VCC target	VCC supplied through a low drop out regulator (LDO) from VBAT
VCC target < VBAT	VBAT > 1.4 V (DC-DC mode)	VCC = VCC target	VCC supplied through the DC-DC buck converter from VBAT

The internal LDO is used to reduce VCC to the specified target when VBAT exceeds the target VCC. For low noise operation (no switching DC-DC operation), or for low VBAT voltages where it make no sense to use a switching converter, configure VCC to LDO Mode by clearing the `ACS_VCC_CTRL_BUCK_ENABLE` bit from the `ACS_VCC_CTRL` register. In this mode, the RSL10 SoC only uses VBAT or the internal LDO to supply VCC. Use of this mode requires an external VCC filtering capacitor, but does not require an external DC-DC converter inductor.

The DC-DC converter is a buck converter used to reduce the battery voltage from a high value (from 1.4 to 3.6 V) to a lower VCC voltage value (from 1.0 to 1.32 V) with high efficiency. The DC-DC converter is only enabled if the supplied VBAT exceeds 1.4 V, and if the VCC is configured to select DC-DC Mode by setting the `ACS_VCC_CTRL_BUCK_ENABLE` bit from the `ACS_VCC_CTRL` register. Use of the DC-DC converter requires both an external VCC filtering capacitor and the DC-DC converter's charge transferring inductor.

The DC-DC buck converter periodically refreshes the flow of current through the external inductor to maintain the supply output. The refresh frequency for the buck converter is divided from `SYSCLK` (see Section 6.3.1, "System Clock (SYSCLK)" on page 78) using the `CLK_DIV_CFG2_DCCLK_PRESCALE` bit field from the `CLK_DIV_CFG2` register. This prescaler provides a division of between 1 and 64 from SYSCLK, with a frequency defined by the following equation:

$$f_{DCCLK} = \frac{f_{SYSCLK}}{(CLK\_DIV\_CFG2\_DCCLK\_PRESCALE + 1)}$$

This clock should be configured for an update frequency between 4 and 12 MHz, based on the expected level of VBAT as shown in Table 4 to minimize the supply ripple present on the VCC supply voltage. For VBAT levels not specified, configure the DCCLK frequency based on the closest VBAT level provided.

NOTE: DCCLK frequencies that approach the SYSCLK frequency can negatively impact RF receive sensitivity. To limit this impact, a compromise setting of 4 MHz is recommended for the DCCLK frequency in RSL10 applications using an 8 MHz SYSCLK setting.

Table 4. DCCLK Frequency verses VBAT Level

Approximate VBAT Level (V)	Recommended DCCLK Frequency (MHz)
1.5	4
1.8	6
2.5	8
3.3	12

When the DC-DC converter is disabled, the `CLK_DIV_CFG2_DCCLK_ENABLE` bit-field from the `CLK_DIV_CFG2` register can be used to disable DCCLK as well.

Other configurations provided by the `ACS_VCC_CTRL` register for VCC include:

- A charge control mode setting (`ACS_VCC_CTRL_CHARGE_CTRL`) that selects between:
  a. A constant current mode where the output current is defined by the supplied VBAT voltage and trimmed VCC voltage. In this configuration, any output ripple on the VCC supply from the buck converter remains stable across VBAT input voltages, reducing the overall noise in the RSL10 system. This configuration can only be used if VBAT exceeds VCC by 0.2 V.
  b. A maximum current mode where the peak current transferred by the buck converter's inductor is defined by a trim configuration (`ACS_VCC_CTRL_ICH_TRIM`) bit-field. If this mode is used, the maximum output from VCC is nominally limited to 1/2 of the current defined by the selected setting for an ideal inductor. In this mode, the output ripple increases as VBAT decreases towards VCC - which might cause additional noise within the RSL10 system. To limit charge current in the DC-DC converter, the default setting of 80 mA (maximum output of 40 mA) is recommended. If the power supervisory circuit is resetting the system when using the DC-DC converter in the user circuit and under the user application's operating conditions, we recommend:
     - Using a higher setting than the default (up to the highest setting of 256 mA), or
     - Increasing the DCCLK frequency to reduce the amount of charge transfer that is required in each charge cycle.

  This recommendation is intended to improve the system stability, handling the kind of stability issues that can arise from high current consumption cases when combined with non-ideal inductor and board series resistance values.

- A pulse control (`ACS_VCC_CTRL_PULSE_CTRL`) bit that can be used to enable a self-clocking mode. By default, the DC-DC converter is set to Single-Pulse Mode, where it is clocked with each pulse of a divided clock that is pre-scaled from SYSCLK using the `CLK_DIV_CFG2_DCCLK_PRESCALE` bit-field from the `CLK_DIV_CFG2` register. In Multi-Pulse Mode, the DC-DC converter operates in a self-clocking mode that continuously charges and discharges the DC-DC inductor until VCC reaches its trimmed level. Multi-Pulse Mode is recommended only for cases where SYSCLK is below the desired DCCLK frequency, as this will result in increased DC-DC current consumption.
- To support high-current use cases, a continuous conduction mode has been provided that can be enabled using the `ACS_VCC_CTRL_CCM_ENABLE` bit. In this mode, the DC-DC converter continually operates to allow support for higher current loads. As use of this mode increases the power consumption of the RSL10 system, we recommend not using this mode for most use cases.

5.3.1.1 VCC and DC-DC Converter Registers

Register Name	Register Description	Address
`ACS_VCC_CTRL`	DC-DC / LDO Supply Configuration / Control register	0x40001304

5.3.1.1.1 ACS_VCC_CTRL

Bit Field	Field Name	Description
19:16	`ICH_TRIM`	Inductor charge current trimming
11	`CCM_ENABLE`	Enable CCM Mode
10	`PULSE_CTRL`	Pulse Mode control

Bit Field	Field Name	Description
9	CHARGE_CTRL	Charge Mode control
8	BUCK_ENABLE	Enable Buck Converter Mode
4:0	VTRIM	Output voltage trimming configuration in 10 mV steps

Field Name	Value Symbol	Value Description	Hex Value
ICH_TRIM	VCC_ICHTRIM_16MA	Charge pump max current to 16 mA	0x0
	VCC_ICHTRIM_32MA	Charge pump max current to 32 mA	0x1
	VCC_ICHTRIM_64MA	Charge pump max current to 64 mA	0x3
	VCC_ICHTRIM_80MA	Charge pump max current to 80 mA	0x4*
	VCC_ICHTRIM_256MA	Charge pump max current to 256 mA	0xF
CCM_ENABLE	VCC_DCM_MODE	Discontinuous current mode	0x0*
	VCC_CCM_MODE	Continuous current mode enabled	0x1
PULSE_CTRL	VCC_SINGLE_PULSE	Single pulse per clock cycle	0x0*
	VCC_MULTI_PULSE	Multi pulses enabled (until VCC > VCC_TRIM)	0x1
CHARGE_CTRL	VCC_CONSTANT_CHARGE	Constant charge transfer (valid for VBAT > VCC + 0.2 V)	0x0
	VCC_CONSTANT_IMAX	Constant inductor maximum charge current	0x1*
BUCK_ENABLE	VCC_LDO	Linear Mode	0x0*
	VCC_BUCK	Buck Converter Mode enabled	0x1
VTRIM	VCC_TRIM_1P00V	Output voltage 1.00V	0x0
	VCC_TRIM_1P05V	Output voltage 1.05V	0x5
	VCC_TRIM_1P10V	Output voltage 1.10V	0xA*
	VCC_TRIM_1P15V	Output voltage 1.15V	0xF
	VCC_TRIM_1P20V	Output voltage 1.20V	0x14
	VCC_TRIM_1P25V	Output voltage 1.25V	0x19
	VCC_TRIM_1P31V	Output voltage 1.31V	0x1F

5.3.2 Internal Band Gap Reference Voltage

The bandgap block provides a 0.75 V reference voltage, stabilized over temperature and process variations by the regulators. This voltage is soft programmable in steps of 2.5 mV from 0.67 to 0.825 V. This block also provides the bias current for all analog blocks, except for the digital supply and POR blocks. This reference voltage is calibrated during production, and use of this calibrated setting is recommended for all use cases.

ACS_BG_CTRL is the bandgap configuration and control register, whose bits can be set for various functions. The ACS_BG_CTRL_SLOPE_TRIM bit field controls whether trimming depends on the temperature coefficient, and the ACS_BG_CTRL_VTRIM bit field configures reference voltage trimming in 2.5 mV steps.

5.3.2.1 Bandgap Converter Registers

Register Name	Register Description	Address
`ACS_BG_CTRL`	Bandgap Configuration / Control register	0x40001300

5.3.2.1.1 ACS_BG_CTRL

Bit Field	Field Name	Description
12:8	`SLOPE_TRIM`	Temperature coefficient trimming
5:0	`VTRIM`	Reference voltage trimming (2.5 mV steps)

Field Name	Value Symbol	Value Description	Hex Value
`SLOPE_TRIM`	`BG_SLOPE_TRIM_VALUE`	Temperature dependency 0 ppm/C	0xB*
`VTRIM`	`BG_TRIM_0P675V`	0.6750 V	0x0
	`BG_TRIM_0P678V`	0.6775 V	0x1
	`BG_TRIM_0P748V`	0.7475 V	0x1D
	`BG_TRIM_0P750V`	0.7500 V	0x1E*
	`BG_TRIM_0P753V`	0.7525 V	0x1F
	`BG_TRIM_0P830V`	0.8300 V	0x3E
	`BG_TRIM_0P833V`	0.8325 V	0x3F

5.3.3 RF Supply Voltage

The RF front-end is supplied by the RF supply voltage (VDDRF). This supply voltage can be supplemented by the RF power amplifier supply voltage (VDDPA), if the TX power required by a user application exceeds the available TX power levels possible from VDDRF.

The VDDRF block is used to provide a regulated voltage, trimmable from 0.75 to 1.38 V in 10 mV steps, from the VCC supply. This voltage is used to supply the radio front-end, which requires a high current.

NOTE: The VDDRF pin can be driven by an external voltage regulator when the regulator is disabled. To disable VDDRF, clear the `ACS_VDDRF_CTRL_ENABLE` bit from the `ACS_VDDRF_CTRL` register.

This supply is typically trimmed to a level that supplies the TX power amplifier with appropriate TX power for the user application’s use case. If the TX power amplifier requires a supply at a level exceeding VCC, the VDDPA separately powers the TX power amplifier, and the VDDRF supply needs to be trimmed to the lowest available calibrated setting that provides the desired RX sensitivity (see Table 5).

In the `ACS_VDDRF_CTRL` register, the `ACS_VDDRF_CTRL_READY` bit configures the whether the supply voltage is in Ready Mode. The `ACS_VDDRF_CTRL_CLAMP` bit controls the output in Disable Mode—it can be used to send the output HIZ or clamp the output to ground. The `ACS_VDDRF_CTRL_ENABLE` bit enables or disables the VDDRF regulator. The `ACS_VDDRF_CTRL_VTRIM` bit field controls configuration of the output voltage trimming in 10 mV steps.

VDDPA is an optional RF TX power amplifier supply voltage. This block is used to provide a regulated voltage, trimmable from 1.1 V to 1.7 V in 10 mV steps, from the VDDA voltage (charge pump). This voltage is used to supply the TX power amplifier block of the radio, whenever this block requires a supply voltage that exceeds the level of VCC to achieve the desired TX output power. To enable VDDPA, set the `ACS_VDDPA_CTRL_ENABLE` bit from the `ACS_VDDPA_CTRL` register.

NOTE: The VDDPA pin can be driven by an external voltage regulator when the regulator is disabled.

`ACS_VDDPA_CTRL_VDDPA_SW_CTRL`, in the `ACS_VDDRF_CTRL` register, is the bit that controls the power amplifier supply, setting the output to HIZ in disable mode, and connecting the switched output to the VDDRF regulator (the Enable bit must be reset in this case). The `ACS_VDDPA_CTRL_ENABLE_ISENSE` bit is used to enable or disable the VDDPA regulator current sensing circuit. The `ACS_VDDPA_CTRL_ENABLE` bit enables or disables the VDDPA regulator. And `ACS_VDDPA_CTRL_VTRIM` controls configuration of the output voltage trimming in 10 mV steps.

Table 5. Target Voltages for TX Power Sources

TX Power (dBm)	Source	Target Voltage (V)
0	VDDRF	1.07
1	VDDRF or VDDPA (low VCC cases)	1.13
2	VDDRF or VDDPA (low VCC cases)	1.20
3	VDDPA	1.26
4	VDDPA	1.35
5	VDDPA	1.45
6	VDDPA	1.65

The VDDRF supply can be broken down into its two internal supply voltages that can be independently over driven at the VDDRF_SW and VDDSYN_SW pads. The VDDRF_SW and VDDSYN_SW pads supply inputs are only intended for test purposes, and as such can be left floating separately or shorted together externally, but not otherwise connected.

5.3.3.1 RF Block Configuration and Control Registers

Register Name	Register Description	Address
`ACS_VDDRF_CTRL`	RF Block Regulator Configuration / Control register	0x40001314
`ACS_VDDPA_CTRL`	RF Block Regulator Configuration / Control register	0x40001318

5.3.3.1.1 ACS_VDDRF_CTRL

Bit Field	Field Name	Description
24	`READY`	Supply ready
12	`CLAMP`	Disable mode clamp control
8	`ENABLE`	Enable control
5:0	`VTRIM`	Output voltage trimming configuration in 10 mV steps

Field Name	Value Symbol	Value Description	Hex Value
`READY`	`VDDRF_NOT_READY`	Supply voltage not ready	0x0*
	`VDDRF_READY`	Supply voltage ready	0x1

Field Name	Value Symbol	Value Description	Hex Value
CLAMP	VDDRF_DISABLE_HIZ	Set the output HIZ (floating) in disable mode	0x0*
	VDDRF_DISABLE_GND	Clamp output to ground in disable mode	0x1
ENABLE	VDDRF_DISABLE	Disable the VDDRF regulator	0x0*
	VDDRF_ENABLE	Enable the VDDRF regulator	0x1
VTRIM	VDDRF_TRIM_0P75V	0.75 V	0x0
	VDDRF_TRIM_0P76V	0.76 V	0x1
	VDDRF_TRIM_1P00V	1.0 V	0x19
	VDDRF_TRIM_1P08V	1.08 V	0x21
	VDDRF_TRIM_1P10V	1.1 V	0x23*
	VDDRF_TRIM_1P15V	1.15 V	0x28
	VDDRF_TRIM_1P20V	1.2 V	0x2D
	VDDRF_TRIM_1P25V	1.25 V	0x32
	VDDRF_TRIM_1P32V	1.32 V	0x39
	VDDRF_TRIM_1P38V	1.38 V	0x3F

5.3.3.1.2 ACS_VDDPA_CTRL

Bit Field	Field Name	Description
12	VDDPA_SW_CTRL	Power amplifier supply control
9	ENABLE_ISENSE	Enable current sensing circuit
8	ENABLE	Enable control
5:0	VTRIM	Output voltage trimming configuration in 10 mV steps

Field Name	Value Symbol	Value Description	Hex Value
VDDPA_SW_CTRL	VDDPA_SW_HIZ	Set the output HIZ (floating) in disable mode	0x0*
	VDDPA_SW_VDDRF	Connect switched output to VDDRF regulator (ENABLE bit must be reset)	0x1
ENABLE_ISENSE	VDDPA_ISENSE_DISABLE	Disable the VDDPA regulator current sensing circuit	0x0*
	VDDPA_ISENSE_ENABLE	Enable the VDDPA regulator current sensing circuit	0x1
ENABLE	VDDPA_DISABLE	Disable the VDDPA regulator	0x0*
	VDDPA_ENABLE	Enable the VDDPA regulator	0x1

Field Name	Value Symbol	Value Description	Hex Value
VTRIM	VDDPA_TRIM_1P05V	1.05 V	0x0
	VDDPA_TRIM_1P06V	1.06 V	0x1
	VDDPA_TRIM_1P59V	1.59 V	0x36
	VDDPA_TRIM_1P60V	1.60 V	0x37*
	VDDPA_TRIM_1P61V	1.61 V	0x38
	VDDPA_TRIM_1P68V	1.68 V	0x3F

5.3.4 Digital Supply Voltages

The RSL10 SoC includes internally regulated digital supply voltages, for which the calibrated settings are strongly recommended:

- VDDC is the core digital voltage that is used for most of the RSL10 system's digital components.
- VDDCRET replaces VDDC in power modes that use state retention of the RSL10 system's digital components.
- VDDM is the memory digital voltage that is used for memories and memory-mapped elements of the RSL10 system.
- VDDMRET replaces VDDM in power modes that use state retention of memories and memory-mapped elements of the RSL10 system.
- VDDTRET is a retention regulator that complements the VDDCRET regulator to maintain the baseband timer execution.

This block is used twice to provide two regulated voltages derived from the VCC supply. These supplies are trimmable from 0.75 V to 1.38 V in 10 mV steps. The default voltage at startup is controlled by the POR block and Program ROM, to ensure safe operation with an untrimmed bandgap.

NOTE: The VDDC and VDDM supplies can also be driven by an external voltage regulator when the regulators are disabled.

In Run Mode, both VDDC and VDDACS (Analog Control Subsystem) regulators' outputs are shorted together. If VDDC is trimmed below 1.0 V for low frequency operating use cases, the VDDACS must also be trimmed lower. If this is not done, the VDDC level saturates to the VDDACS voltage.

The digital retention supply regulator is designed to consume less power and to guarantee the retention of the state of digital blocks (IVDDCRET) and the contents of memory (VDDMRET) to the extended supply limit.

The `ACS_VDDC_CTRL` and `ACS_VDDM_CTRL` registers contain bits with identical names and identical functions. The `STANDBY_VTRIM` bit controls the VDDC standby voltage trimming in 10 mV steps. The `ENABLE_LOW_BIAS` bit is used to specify whether the regulator biasing is normal or low. The `SLEEP_CLAMP` bit sets the output to HIZ or clamps the output to ground, in Sleep Mode. The `VTRIM` bit configures output voltage trimming in 10 mV steps in both `ACS_VDDC_CTRL` and `CS_VDDM_CTRL`.

The `ACS_VDDRET_CTRL` has three different bit fields used to trim each of the retention regulators. These are:

- `ACS_VDDRET_CTRL_VDDCRET_VTRIM` bit controls the VDDCRET retention regulator voltage trimming value, while `ACS_VDDRET_CTRL_VDDCRET_ENABLE` enables or disables the VDDCRET retention regulator.
- `ACS_VDDRET_CTRL_VDDMRET_VTRIM` bit controls the VDDMRET retention regulator voltage trimming value. `ACS_VDDRET_CTRL_VDDMRET_ENABLE` enables or disables the VDDMRET retention regulator.

- `ACS_VDDRET_CTRL_VDDTRET_VTRIM` bit controls the VDDTRET retention regulator voltage trimming value. `ACS_VDDRET_CTRL_VDDTRET_ENABLE` enables or disables the VDDTRET retention regulator.

5.3.4.1 Digital Supply Configuration / Control Registers

Register Name	Register Description	Address
ACS_VDDC_CTRL	Digital Core Voltage Regulator Configuration / Control register	0x4000130C
ACS_VDDM_CTRL	Memories Voltage Configuration / Control register	0x40001310
ACS_VDDRET_CTRL	Retention Regulator Configuration / Control register	0x4000131C

5.3.4.1.1 ACS_VDDC_CTRL

Bit Field	Field Name	Description
21:16	STANDBY_VTRIM	VDDC standby voltage trimming (10 mV steps)
13	ENABLE_LOW_BIAS	Low power mode control
12	SLEEP_CLAMP	Sleep mode clamp control
5:0	VTRIM	Output voltage trimming configuration in 10 mV steps

Field Name	Value Symbol	Value Description	Hex Value
STANDBY_VTRIM	VDDC_STANDBY_TRIM_0P75V	0.75 V	0x0
	VDDC_STANDBY_TRIM_0P76V	0.76 V	0x1
	VDDC_STANDBY_TRIM_1P00V	1.0 V	0x19
	VDDC_STANDBY_TRIM_1P08V	1.08 V	0x21
	VDDC_STANDBY_TRIM_1P10V	1.1 V	0x23*
	VDDC_STANDBY_TRIM_1P15V	1.15 V	0x28
	VDDC_STANDBY_TRIM_1P20V	1.2 V	0x2D
	VDDC_STANDBY_TRIM_1P25V	1.25 V	0x32
	VDDC_STANDBY_TRIM_1P32V	1.32 V	0x39
	VDDC_STANDBY_TRIM_1P38V	1.38 V	0x3F
ENABLE_LOW_BIAS	VDDC_NOMINAL_BIAS	Nominal regulator biasing	0x0*
	VDDC_LOW_BIAS	Low regulator biasing	0x1
SLEEP_CLAMP	VDDC_SLEEP_HIZ	Set the output HIZ (floating) in Sleep Mode	0x0*
	VDDC_SLEEP_GND	Clamp output to ground in Sleep Mode	0x1

Field Name	Value Symbol	Value Description	Hex Value
VTRIM	VDDC_TRIM_0P75V	0.75 V	0x0
	VDDC_TRIM_0P76V	0.76 V	0x1
	VDDC_TRIM_1P00V	1.0 V	0x19
	VDDC_TRIM_1P08V	1.08 V	0x21
	VDDC_TRIM_1P10V	1.1 V	0x23*
	VDDC_TRIM_1P15V	1.15 V	0x28
	VDDC_TRIM_1P20V	1.2 V	0x2D
	VDDC_TRIM_1P25V	1.25 V	0x32
	VDDC_TRIM_1P32V	1.32 V	0x39
	VDDC_TRIM_1P38V	1.38 V	0x3F

5.3.4.1.2 ACS_VDDM_CTRL

Bit Field	Field Name	Description
21:16	STANDBY_VTRIM	VDDM standby voltage trimming (10 mV steps)
13	ENABLE_LOW_BIAS	Low power mode control
12	SLEEP_CLAMP	Sleep mode clamp control
5:0	VTRIM	Output voltage trimming configuration in 10 mV steps

Field Name	Value Symbol	Value Description	Hex Value
STANDBY_VTRIM	VDDM_STANDBY_TRIM_0P75V	0.75 V	0x0
	VDDM_STANDBY_TRIM_0P76V	0.76 V	0x1
	VDDM_STANDBY_TRIM_1P00V	1.0 V	0x19
	VDDM_STANDBY_TRIM_1P08V	1.08 V	0x21
	VDDM_STANDBY_TRIM_1P10V	1.1 V	0x23*
	VDDM_STANDBY_TRIM_1P15V	1.15 V	0x28
	VDDM_STANDBY_TRIM_1P20V	1.2 V	0x2D
	VDDM_STANDBY_TRIM_1P25V	1.25 V	0x32
	VDDM_STANDBY_TRIM_1P32V	1.32 V	0x39
	VDDM_STANDBY_TRIM_1P38V	1.38 V	0x3F
ENABLE_LOW_BIAS	VDDM_NOMINAL_BIAS	Nominal regulator biasing	0x0*
	VDDM_LOW_BIAS	Low regulator biasing	0x1
SLEEP_CLAMP	VDDM_SLEEP_HIZ	Set the output HIZ (floating) in Sleep Mode	0x0*
	VDDM_SLEEP_GND	Clamp output to ground in Sleep Mode	0x1

Field Name	Value Symbol	Value Description	Hex Value
VTRIM	VDDM_TRIM_0P75V	0.75 V	0x0
	VDDM_TRIM_0P76V	0.76 V	0x1
	VDDM_TRIM_1P00V	1.0 V	0x19
	VDDM_TRIM_1P08V	1.08 V	0x21
	VDDM_TRIM_1P10V	1.1 V	0x23*
	VDDM_TRIM_1P15V	1.15 V	0x28
	VDDM_TRIM_1P20V	1.2 V	0x2D
	VDDM_TRIM_1P25V	1.25 V	0x32
	VDDM_TRIM_1P32V	1.32 V	0x39
	VDDM_TRIM_1P38V	1.38 V	0x3F

5.3.4.1.3 ACS_VDDRET_CTRL

Bit Field	Field Name	Description
18:17	VDDMRET_VTRIM	VDDMRET retention regulator voltage trimming
16	VDDMRET_ENABLE	Enable/Disable the VDDMRET retention regulator
10:9	VDDTRET_VTRIM	VDDTRET retention regulator voltage trimming
8	VDDTRET_ENABLE	Enable/Disable the VDDTRET retention regulator
2:1	VDDCRET_VTRIM	VDDCRET retention regulator voltage trimming
0	VDDCRET_ENABLE	Enable/Disable the VDDCRET retention regulator

Field Name	Value Symbol	Value Description	Hex Value
VDDMRET_VTRIM	VDDMRET_TRIM_VALUE	VDDMRET trimming value	0x3*
VDDMRET_ENABLE	VDDMRET_DISABLE	The VDDMRET retention regulator is disabled	0x0*
	VDDMRET_ENABLE	The VDDMRET retention regulator is enabled	0x1
VDDTRET_VTRIM	VDDTRET_TRIM_VALUE	VDDTRET trimming value	0x3*
VDDTRET_ENABLE	VDDTRET_DISABLE	The VDDTRET retention regulator is disabled	0x0*
	VDDTRET_ENABLE	The VDDTRET retention regulator is enabled	0x1
VDDCRET_VTRIM	VDDCRET_TRIM_VALUE	VDDCRET trimming value	0x3*
VDDCRET_ENABLE	VDDCRET_DISABLE	The VDDCRET retention regulator is disabled	0x0*
	VDDCRET_ENABLE	The VDDCRET retention regulator is enabled	0x1

5.3.5 Analog Supply Voltage (VDDA)

The analog supply voltage makes use of an internal charge pump to generate a configurable regulated supply voltage. This voltage is used for all of the non-RF analog components and the flash memory. This supply uses a boost converter with an external capacitance between the CAP0 and CAP1 pads as part of its charge pump circuit, to effectively increase the system supply (VCC) voltage as required by these blocks.

The charge pump has four different output power trimming modes to allow a better balance between consumption and power delivery. The default maximum current draw of the output power is 4 mA. If an application requires a heavy current draw on VDDA, the `PTRIM` bit (1:0) of the `ACS_VDDA_CP_CTRL` register can be configured to 0x3, to receive a maximum current of 16 mA.

The VDDA charge pump periodically refreshes its external capacitance to maintain the supply output. The refresh frequency for the charge pump is divided from `SLOWCLK` (see Section 6.3.3, "Slow Clock (SLOWCLK)" on page 80) using the `CLK_DIV_CFG2_CPCLK_PRESCALE` bit field from the `CLK_DIV_CFG2` register. This prescaler provides a division of between 1 and 64 from SLOWCLK, with a frequency defined by the following equation:

$$f_{CPCLK} = \frac{f_{SLOWCLK}}{(CLK\_DIV\_CFG2\_CPCLK\_PRESCALE + 1)}$$

This clock needs to be configured for an update frequency between 10 kHz and 400 kHz, with no restrictions on the duty cycle of the source clock. When VDDA is disabled, the `CLK_DIV_CFG2_CPCLK_ENABLE` bit-field from the `CLK_DIV_CFG2` register can be used to disable the charge pump clock as well.

The analog supply voltage is accessible at the VDDA pad for capacitive filtering.

5.3.5.1 Analog Voltage Configuration and Control Registers

Register Name	Register Description	Address
`ACS_VDDA_CP_CTRL`	Analog Voltage and Flash Charge Pump Configuration / Control register	0x40001308

5.3.5.1.1 ACS_VDDA_CP_CTRL

Bit Field	Field Name	Description
1:0	`PTRIM`	Output power trimming

Field Name	Value Symbol	Value Description	Hex Value
`PTRIM`	`VDDA_PTRIM_4MA`	Charge pump max current to 4 mA	0x0*
	`VDDA_PTRIM_8MA`	Charge pump max current to 8 mA	0x1
	`VDDA_PTRIM_12MA`	Charge pump max current to 12 mA	0x2
	`VDDA_PTRIM_16MA`	Charge pump max current to 16 mA	0x3

5.4 POWER MODES

5.4.1 Overview

The available power modes in RSL10 consist of Standby Mode, Sleep Mode, and Run Mode. Before entering Sleep Mode, RSL10 can be configured by the user to wake up from retention memory. RSL10 effectively uses power modes between RF events, while maintaining a Bluetooth low energy connection, and minimizing power consumption while in Standby Mode for duty cycled applications.

The POWER_MODE bit of the ACS_PWR_MODES_CTRL register holds a 32-bit key which specifies whether RSL10 enters Run Mode, Standby Mode, or Sleep Mode.

5.4.2 Measuring Power Mode Current Consumption

To minimize the power consumption for measurements, here are a few guidelines:

- Disable pad retention
- Disconnect the JTAG cable
- Disconnect all probes on GPIO pins and power supplies
- Disable the VDDT/VDDC/VDDM retention regulators during sleep when possible

The 32 kHz crystal oscillator power consumption can be reduced (possibly as low as the level of the RC oscillator consumption) by enabling the following settings in the ACS_XTAL32K_CTRL register:

- EN_AMPL_CTRL = 1 (this regulates the oscillation amplitude to minimize power consumption)
- Set the CLOAD_TRIM register to bring the frequency as close as possible to 32,768 Hz. This register adjusts the crystal load capacitors.
- Leave the other settings at their default values.

5.4.3 Run Mode

In Run Mode (default functional mode), all the circuitry is powered on. Most of the blocks can be enabled/disabled individually using memory-mapped registers (refer to Chapter 7, “Memory” on page 93).

5.4.4 Standby Mode

Standby Mode can be used to reduce the average power consumption for inactive times, which typically range from a few ms to a few hundreds of ms. In this state, the logic and memories are not clocked and are powered at a reduced voltage to minimize the leakage current.

The ACS (Analog Control System), bandgap, DC-DC converter, charge pump, and digital regulator are active. The RF block can be disconnected from its supply through the ACS_VDDRF_CTRL register.

The reduced voltage level can be programmed in the STANDBY_VTRIM field of the ACS_VDDC_CTRL register.

Entering the Standby Mode by writing the standby key in the ACS_PWR_MODES_CTRL register (refer to Section 5.4.4.2, “ACS_PWR_MODES_CTRL”) starts the following sequence:

1. The system clock is stopped.
2. All memories (FLASH, PROM, RAM) are isolated from the core (AND gates).
3. ROM and Flash are powered off, and used RAMs are placed in Retention Mode.
4. The VDDC and VDDM regulator output voltages are set to their standby voltages.

IMPORTANT: For an RSL10 **SoC in standby, the 48 MHz crystal oscillator, RF block and STANDBY_VTRIM settings contribute significantly to the current of the battery (IBAT).**

To minimize the power consumption, if the 48 MHz crystal oscillator and RF block are not required, both should be turned off and the STANDBY_VTRIM setting should be lowered.

At wake-up (see Section 5.4.4.1, “Wakeup Sources” on page 52 for sources), the following sequence restarts the system:

1. The RTC counter’s 8 LSBs are captured in a register to record the wakeup time. This value can be read from the `RTC_VALUE` bit field in the `ACS_WAKEUP_STATE` register (see Section 5.4.4.4, “ACS_WAKEUP_STATE” on page 54).
2. The VDDC and VDDM regulator output voltages are set to their normal voltages.
3. Memories are powered back on.
4. The wakeup DELAY is applied (see `ACS_WAKEUP_CFG` register, refer to Section 5.4.4.3, “ACS_WAKEUP_CFG”).
5. Memory isolation is removed.
6. Clock is enabled and system execution is resumed.

5.4.4.1 Wakeup Sources

The following are the sources through which a wakeup event can occur:

- DC-DC overload
- Baseband timer
- Wakeup pad
- One of DIO [3:0]

`ACS_WAKEUP_CFG_DCDC_OVERLOAD_EN` enables or disables the DC-DC overload flag’s wakeup functionality. The `ACS_WAKEUP_CFG_WAKEUP_PAD_POL` bit controls whether wakeup occurs on the wakeup pad’s rising edge, enabling a pull-down, or on the falling edge, with a pull-up enabled. The `ACS_WAKEUP_CFG_DIO*_POL` bit (where * is 0 to 3) controls the wakeup polarity on DIO pads 0 to 3. Bit `ACS_WAKEUP_CFG_DIO*_EN` (where * is 0 to 3) enables or disables wakeup functionality on the corresponding DIO pad.

The `ACS_WAKEUP_CFG_DELAY` bit in the `ACS_WAKEUP_CFG` register controls the number of clock cycles (in powers of 2, between 1 and 28) that elapse after VDDC wakeup.

For the `ACS_WAKEUP_STATE` register, the `ACS_WAKEUP_STATE_WAKEUP_SRC` bit indicates the source of the last wakeup, while the `ACS_WAKEUP_STATE_RCT_VALUE` bit contains the value of the RTC counter captured at the last wakeup event.

NOTE: A maximum sleep duration baseband timer can be enabled by software. See the *RSL10 Firmware Reference* for how to configure this baseband timer.

5.4.4.2 ACS_PWR_MODES_CTRL

Bit Field	Field Name	Description
31:0	`POWER_MODE`	32-bit key to enter RUN, STANDBY or SLEEP mode

Field Name	Value Symbol	Value Description	Hex Value
`POWER_MODE`	`PWR_RUN_MODE`	Keep the system in normal RUN mode	0x0*
	`PWR_STANDBY_MODE`	Enter STANDBY mode	0x9B1D79A0
	`PWR_SLEEP_MODE`	Enter SLEEP mode	0xE0045650

5.4.4.3 ACS_WAKEUP_CFG

Bit Field	Field Name	Description
18:16	DELAY	Delay from VDDC ready to digital clock enable (power of 2)
9	DCDC_OVERLOAD_EN	Enable / Disable the wakeup functionality on the DC-DC overload flag
8	WAKEUP_PAD_POL	Wakeup polarity on the WAKEUP pad
7	DIO3_POL	Wakeup polarity on the DIO3 pad
6	DIO2_POL	Wakeup polarity on the DIO2 pad
5	DIO1_POL	Wakeup polarity on the DIO1 pad
4	DIO0_POL	Wakeup polarity on the DIO0 pad
3	DIO3_EN	Enable / Disable the wakeup functionality on the DIO3 pad
2	DIO2_EN	Enable / Disable the wakeup functionality on the DIO2 pad
1	DIO1_EN	Enable / Disable the wakeup functionality on the DIO1 pad
0	DIO0_EN	Enable / Disable the wakeup functionality on the DIO0 pad

Field Name	Value Symbol	Value Description	Hex Value
DELAY	WAKEUP_DELAY_1	Wait for 1 clock cycle	0x0
	WAKEUP_DELAY_2	Wait for 2 clock cycles	0x1
	WAKEUP_DELAY_4	Wait for 4 clock cycles	0x2
	WAKEUP_DELAY_8	Wait for 8 clock cycles	0x3
	WAKEUP_DELAY_16	Wait for 16 clock cycles	0x4
	WAKEUP_DELAY_32	Wait for 32 clock cycles (typ.10 us)	0x5*
	WAKEUP_DELAY_64	Wait for 64 clock cycles	0x6
	WAKEUP_DELAY_128	Wait for 128 clock cycles	0x7
DCDC_OVERLOAD_EN	WAKEUP_DCDC_OVERLOAD_DISABLE	Disable the wakeup functionality on the DC-DC overload flag	0x0*
	WAKEUP_DCDC_OVERLOAD_ENABLE	Enable the wakeup functionality on the DC-DC overload flag	0x1
WAKEUP_PAD_POL	WAKEUP_WAKEUP_PAD_RISING	Wake up on the WAKEUP pad rising edge and enable pull-down	0x0*
	WAKEUP_WAKEUP_PAD_FALLING	Wake up on the WAKEUP pad falling edge and enable pull-up	0x1
DIO3_POL	WAKEUP_DIO3_RISING	Wake up on the DIO3 rising edge	0x0*
	WAKEUP_DIO3_FALLING	Wake up on the DIO3 falling edge	0x1
DIO2_POL	WAKEUP_DIO2_RISING	Wake up on the DIO2 rising edge	0x0*
	WAKEUP_DIO2_FALLING	Wake up on the DIO2 falling edge	0x1
DIO1_POL	WAKEUP_DIO1_RISING	Wake up on the DIO1 rising edge	0x0*
	WAKEUP_DIO1_FALLING	Wake up on the DIO1 falling edge	0x1
DIO0_POL	WAKEUP_DIO0_RISING	Wake up on the DIO0 rising edge	0x0*
	WAKEUP_DIO0_FALLING	Wake up on the DIO0 falling edge	0x1

Field Name	Value Symbol	Value Description	Hex Value
DIO3_EN	WAKEUP_DIO3_DISABLE	Disable the Wakeup functionality on the DIO3 pad	0x0*
	WAKEUP_DIO3_ENABLE	Enable the Wakeup functionality on the DIO3 pad	0x1
DIO2_EN	WAKEUP_DIO2_DISABLE	Disable the Wakeup functionality on the DIO2 pad	0x0*
	WAKEUP_DIO2_ENABLE	Enable the Wakeup functionality on the DIO2 pad	0x1
DIO1_EN	WAKEUP_DIO1_DISABLE	Disable the Wakeup functionality on the DIO1 pad	0x0*
	WAKEUP_DIO1_ENABLE	Enable the Wakeup functionality on the DIO1 pad	0x1
DIO0_EN	WAKEUP_DIO0_DISABLE	Disable the Wakeup functionality on the DIO0 pad	0x0*
	WAKEUP_DIO0_ENABLE	Enable the Wakeup functionality on the DIO0 pad	0x1

5.4.4.4 ACS_WAKEUP_STATE

Bit Field	Field Name	Description
18:16	WAKEUP_SRC	Status register indicates the last wakeup source
7:0	RTC_VALUE	RTC counter value captured at wakeup event (only 8 LSBs, corresponds to 7.8 ms)

Field Name	Value Symbol	Value Description	Hex Value
WAKEUP_SRC	WAKEUP_DUE_TO_DCDC_OVERLOAD	The last wakeup was due to the DC-DC overload	0x7
	WAKEUP_DUE_TO_WAKEUP_PAD	The last wakeup was due to the WAKEUP pad	0x6
	WAKEUP_DUE_TO_RTC_ALARM	The last wakeup was due to the RTC Timer alarm	0x5
	WAKEUP_DUE_TO_BB_TIMER	The last wakeup was due to the baseband timer alarm	0x4
	WAKEUP_DUE_TO_DIO3	The last wakeup was due to the DIO3 pad	0x3
	WAKEUP_DUE_TO_DIO2	The last wakeup was due to the DIO2 pad	0x2
	WAKEUP_DUE_TO_DIO1	The last wakeup was due to the DIO1 pad	0x1
	WAKEUP_DUE_TO_DIO0	The last wakeup was due to the DIO0 pad	0x0*

5.4.5 Sleep Mode

When operating in Sleep Mode, RSL10 will exhibit the lowest current consumption. Only the wakeup logic (see Section 5.4.4.1, “Wakeup Sources” on page 52 for sources of wakeup) is kept powered. The bandgap, regulators, RF block, etc. are disabled. The digital core and the memories can optionally be powered at low voltage.

When Sleep Mode is entered, a Power-On Reset (POR) is generated, which sets all registers in the digital core to their default values. Only the registers inferred in the ACS keep their values.

The PADS_RETENTION_EN bit in the ACS_WAKEUP_CTRL register (refer to Section 5.4.6, "ACS_WAKEUP_CTRL") has to be set prior to entering Sleep Mode, as the core is not powered. Setting this bit makes sure the pads keep configuration (direction, state, etc.) during sleep time. Upon wakeup, the boot PROM code is executed. The initial pad configuration (the one used before going into Sleep Mode) needs to be restored by the software before resetting the PADS_RETENTION_EN bit, to avoid toggling the pads.

The following sub-modes are typically selected using the configuration registers of the ACS:

1. Wakeup through an external event on the wakeup pad or the DIO[3:0] pads (ACS_WAKEUP_CTRL_DIO*_WAKEUP bits). This is the minimum power consumption sub-mode.
2. Wakeup through the RTC (the ACS_WAKEUP_CTRL_RTC_ALARM_WAKEUP bit), clocked either by the internal RC oscillator or by the 32 kHz crystal oscillator. The sleep time can be programmed using the RTC configuration registers of the ACS (see Section 6.3.5, "Real-Time Clock (RTC)").
3. Wakeup through the baseband timer (the ACS_WAKEUP_CTRL_BB_TIMER_WAKEUP bit).
4. ACS_WAKEUP_CTRL_BOOT_FLASH_APP_REBOOT determines whether the reboot mode flag is set.
5. ACS_WAKEUP_CTRL_RC_CLOCK_MULT controls whether the startup RC oscillator is at 3 or 12 MHz.
6. ACS_WAKEUP_CTRL_RC_FTRIM_FLAG bit configures whether or not the oscillators are treated as calibrated.
7. ACS_WAKEUP_CTRL_BOOT_SELECT bit-field controls whether the system attempts to boot directly from flash or from a custom location in memory. This field also configures whether or not the RF crystal oscillator will be started.

The status bits in the ACS_WAKEUP_CTRL register that indicate whether a wakeup event has been triggered at least once for specific causes since last being cleared:

- DCDC_OVERLOAD_WAKEUP means that the wakeup event was triggered by a DC-DC overload
- WAKEUP_PAD_WAKEUP means that the wakeup event was triggered by the wakeup pad
- RTC_ALARM_WAKEUP means that the wakeup event was triggered by the RTC reaching the alarm value
- BB_TIMER_WAKEUP means that the wakeup event was triggered by the baseband timer reaching the specified timeout
- DIO*_WAKEUP means that the wakeup event was triggered by the specified DIO

Bits that reset sticky flags:

- DCDC_OVERLOAD_CLEAR clears the DC-DC overload wakeup status bit
- WAKEUP_PAD_WAKEUP_CLEAR clears the wakeup pad wakeup status bit
- RTC_ALARM_WAKEUP_CLEAR clears the RTC alarm wakeup status bit
- BB_TIMER_WAKEUP_CLEAR clears the baseband timer wakeup status bit
- DIO*_WAKEUP_CLEAR clears the DIO wakeup status bit indicated by *

Entering Sleep Mode by writing the sleep key in the ACS_PWR_MODES_CTRL register starts the following sequence:

1. The system clock is stopped.
2. Reset is asserted unless the VDDC retention regulator is enabled.
3. All memories (flash, PROM, RAM) are isolated from the core (AND gates).
4. Memories are powered off. RAMs which are enabled in the memory enable retention latches are put into Retention Mode, if the VDDM retention regulator is enabled.
5. The logic is disconnected from its supply unless the VDDC retention regulator is enabled.
6. The baseband timer is disconnected from its supply unless the VDDT retention regulator is enabled.

7. The RF block is disconnected from its supplies (VDDRF and VDDPA). Note that the RF block needs to be isolated manually if the VDDC retention regulator is enabled.
8. The VDDA, VDDC, VDDM, VDDRF and VDDPA regulators are disabled.
9. The VCC regulator/DC-DC converter is disabled.
10. The bandgap is disabled.

At wakeup (through RTC or a pad), the following sequence restarts the system:

1. The RTC 8 LSBs are captured into a register to record the wakeup time
2. The bandgap is enabled.
3. The VCC regulator/DC-DC converter is enabled when the bandgap voltage is ready.
4. The other regulators are set according to the configuration registers when the VCC is ready.
5. Memories are powered back on when VDDC, VDDM and VDDA are ready.
6. The wakeup `DELAY` is applied (see `ACS_WAKEUP_CFG` register).
7. Memory isolation is removed.
8. Clock is enabled.
9. The digital reset is released, enabling boot PROM execution unless the VDDC retention regulator is enabled.

To use the RTC timer as wakeup source, refer to Section 6.3.5, “Real-Time Clock (RTC)”

5.4.5.1 Wakeup from Retention Memory in Sleep Mode

When going to Sleep Mode for wakeup from RAM, the application must do the following:

1. Configure all memories to be powered up and accessible when waking up from Sleep Mode, in the `SYSCTRL_MEM_ACCESS_CFG` register:
   - The `SYSCTRL_MEM_ACCES_CFG_WAKEUP_ADDR_PACKED` bit field gives the wakeup restore address in packed 7-bit format.
   - The `SYSCTRL_MEM_ACCES_CFG_DSP_DRAM*_ACCESS`, `SYSCTRL_MEM_ACCESS_CFG_DSP_PRAM*`, `SYSCTRL_MEM_ACCESS_CFG_BB_DRAM_ACCESS*`, `SYSCTRL_MEM_ACCESS_CFG_DRAM_ACCESS*` and `SYSCTRL_MEM_ACCESS_CFG_PRAM_ACCESS*` bits control enabling or disabling access to the corresponding DSP DRAM, DSP PRAM, baseband DRAM, DRAM and PRAM, respectively.
   - The `SYSCTRL_MEM_ACCES_CFG_FLASH_ACCESS` bit controls the flash memory access.
   - The `SYSCTRL_MEM_ACCES_CFG_PROM_ACCESS` bit configures PROM access.

   See Section 5.4.7, “SYSCTRL_MEM_ACCESS_CFG” on page 60 for details.

2. Write the wakeup restore address to the `SYSCTRL_WAKEUP_ADDR` register, which contains the wakeup restore address in unpacked 32-bit format (see Section 5.4.8, “SYSCTRL_WAKEUP_ADDR” on page 62 for more information).

NOTE: The wakeup restore address must be either the first address of a RAM instance or the last address minus 20. Table 6 on page 57 lists all possible addresses.

3. Read the `SYSCTRL_MEM_ACCESS_CFG` register contents. If some of the memories that must be powered up and accessible when waking up from Sleep Mode are not set, because their power bit is not set or because their Retention Mode bit is set, then enable these bits in the read value (by using a logical OR operation). Write this value to the `ACS_WAKEUP_GP_DATA` register (see Section 5.4.10, “ACS_WAKEUP_GP_DATA” on page 63).
4. Copy the following six words to memory starting from the wakeup restore address:
   - `SYSCTRL_DBG_LOCK` register contents
   - `SYSCTRL_DBG_LOCK_KEY_0` register contents
   - `SYSCTRL_DBG_LOCK_KEY_1` register contents

- `SYSCTRL_DBG_LOCK_KEY_2` register contents
- `SYSCTRL_DBG_LOCK_KEY_3` register contents
- Application start address to be used after waking up

5. In the `SYSCTRL_MEM_POWER_CFG` register, configure all the memories that are to be kept in Retention Mode even while the device is in Sleep Mode.
6. Set the `BOOT_SELECT` bit-field in the `ACS_WAKEUP_CTRL` register to `BOOT_CUSTOM`.
7. Enable the required retention regulators and set their trimming values as needed.
8. Enter Sleep Mode.

NOTE: To verify the wakeup pad value, read from the `SYSCTRL_WAKEUP_PAD_WAKEUP_PAD_VALUE` bit in the `SYSCTRL_WAKEUP_PAD` register (see Section 5.4.9, “SYSCTRL_WAKEUP_PAD” on page 62).

IMPORTANT: To configure the memory retention, the following operations are required:

- **`ACS_VDDRET_CTRL->VDDMRET_VTRIM` = 1**
- **`ACS_VDDRET_CTRL->VDDMRET_ENABLE` = 1**
- **Enable memory instance(s) to be retained in `SYSCTRL_MEM_POWER_CFG`**
- **`SYSCTRL_MEM_ACCESS_CFG` can be ignored**

Table 6. Possible Wakeup Restore Addresses

Location	Address
PRAM0	0x00200000
PRAM1	0x00202000
PRAM2	0x00204000
PRAM3	0x00206000
DSP_PRAM3	0x00208000
DSP_PRAM2	0x0020a000
DSP_PRAM1	0x0020c000
DSP_PRAM0	0x0020e000
DRAM0	0x20000000
DRAM1	0x20002000
DRAM2	0x20004000
DSP_DRAM0	0x20006000
DSP_DRAM1	0x20008000
DSP_DRAM2	0x2000a000
DSP_DRAM3	0x2000c000
DSP_DRAM4	0x2000e000
DSP_DRAM5	0x20010000
BB_DRAM0	0x20012000
BB_DRAM1	0x20014000
PRAM0_END	0x00201FE8
PRAM1_END	0x00203FE8
PRAM2_END	0x00205FE8
PRAM3_END	0x00207FE8

Table 6. Possible Wakeup Restore Addresses

Location	Address
DSP_PRAM3_END	0x00209FE8
DSP_PRAM2_END	0x0020BFE8
DSP_PRAM1_END	0x0020DFE8
DSP_PRAM0_END	0x0020FFE8
DRAM0_END	0x20001FE8
DRAM1_END	0x20003FE8
DRAM2_END	0x20005FE8
DSP_DRAM0_END	0x20007FE8
DSP_DRAM1_END	0x20009FE8
DSP_DRAM2_END	0x2000BFE8
DSP_DRAM3_END	0x2000DFE8
DSP_DRAM4_END	0x2000FFE8
DSP_DRAM5_END	0x20011FE8
BB_DRAM0_END	0x20013FE8
BB_DRAM1_END	0x20015FE8

5.4.6 ACS_WAKEUP_CTRL

Bit Field	Field Name	Description
24	`PADS_RETENTION_EN`	Enable / Disable the Retention Mode of the pads
20	`BOOT_FLASH_APP_REBOOT`	Boot mode flag
19	`RC_CLOCK_MULT`	RC oscillator clock multiplier read only flag (mirror of CLOCK_MULT of ACS_RCOSC_CTRL register)
18	`RC_FTRIM_FLAG`	RC oscillator trimming read only flag (mirror of FTRIM_FLAG of ACS_RCOSC_CTRL register
17:16	`BOOT_SELECT`	Boot selection bit to indicate boot source
15	`DCDC_OVERLOAD_WAKEUP`	Status bit indicating that the wakeup was triggered by DC-DC overload
14	`WAKEUP_PAD_WAKEUP`	Status bit indicating that the wakeup was triggered by the wakeup pad
13	`RTC_ALARM_WAKEUP`	Status bit indicating that the wakeup was triggered by the RTC alarm
12	`BB_TIMER_WAKEUP`	Status bit indicating that the wakeup was triggered by the baseband timer
11	`DIO3_WAKEUP`	Status bit indicating that the wakeup was triggered by DIO3
10	`DIO2_WAKEUP`	Status bit indicating that the wakeup was triggered by DIO2
9	`DIO1_WAKEUP`	Status bit indicating that the wakeup was triggered by DIO1
8	`DIO0_WAKEUP`	Status bit indicating that the wakeup was triggered by DIO0
7	`DCDC_OVERLOAD_CLEAR`	Clear the DC-DC overload wakeup status bit
6	`WAKEUP_PAD_WAKEUP_CLEAR`	Clear the wakeup pad wakeup status bit
5	`RTC_ALARM_WAKEUP_CLEAR`	Clear the RTC alarm wakeup status bit
4	`BB_TIMER_WAKEUP_CLEAR`	Clear the baseband timer wakeup status bit
3	`DIO3_WAKEUP_CLEAR`	Clear the DIO3 wakeup status bit
2	`DIO2_WAKEUP_CLEAR`	Clear the DIO2 wakeup status bit

Bit Field	Field Name	Description
1	DIO1_WAKEUP_CLEAR	Clear the DIO1 wakeup status bit
0	DIO0_WAKEUP_CLEAR	Clear the DIO0 wakeup status bit

Field Name	Value Symbol	Value Description	Hex Value
PADS_RETENTION_EN	PADS_RETENTION_DISABLE	Disable the pad Retention Mode	0x0*
	PADS_RETENTION_ENABLE	Enable the pad Retention Mode	0x1
BOOT_FLASH_APP_REBOOT	BOOT_FLASH_APP_REBOOT_DISABLE	The reboot mode flag is not set	0x0*
	BOOT_FLASH_APP_REBOOT_ENABLE	The reboot mode flag is set (ROM will not read the calibration values from flash and will directly execute the application)	0x1
RC_CLOCK_MULT	RC_START_OSC_STATUS_3MHZ	The startup RC Oscillator is at 3 MHz	0x0*
	RC_START_OSC_STATUS_12MHZ	The startup RC Oscillator is at 12 MHz	0x1
RC_FTRIM_FLAG	RC_OSC_STATUS_UNCALIBRATED	The oscillators are not calibrated	0x0*
	RC_OSC_STATUS_CALIBRATED	The oscillators are calibrated	0x1
BOOT_SELECT	BOOT_FLASH_XTAL_DISABLE	The Arm Cortex-M3 executes code from the flash and the XTAL will not be started at boot	0x0*
	BOOT_CUSTOM	The Arm Cortex-M3 core executed code from the address specified in the wakeup information in retention RAM and the XTAL will not be started at boot	0x1
	BOOT_FLASH_XTAL_DEFAULT_TRIM	The Arm Cortex-M3 core executes code from the flash and the XTAL will be started at boot with the default trim	0x2
	BOOT_FLASH_XTAL_CUSTOM_TRIM	The Arm Cortex-M3 core executes code from the flash and the XTAL will be started at boot with trim from ACS_WAKEUP_GP_DATA	0x3
DCDC_OVERLOAD_WAKEUP	WAKEUP_DCDC_OVERLOAD_NOT_SET	DC-DC overload has not triggered a wakeup event	0x0*
	WAKEUP_DCDC_OVERLOAD_SET	DC-DC overload has triggered a wakeup event at least once	0x1
WAKEUP_PAD_WAKEUP	WAKEUP_PAD_EVENT_NOT_SET	Wakeup pad has not triggered a wakeup event	0x0*
	WAKEUP_PAD_EVENT_SET	Wakeup pad has triggered a wakeup event at least once	0x1
RTC_ALARM_WAKEUP	WAKEUP_RTC_ALARM_EVENT_NOT_SET	RTC alarm has not triggered a wakeup event	0x0*
	WAKEUP_RTC_ALARM_EVENT_SET	RTC alarm has triggered a wakeup event at least once	0x1
BB_TIMER_WAKEUP	WAKEUP_BB_TIMER_EVENT_NOT_SET	BB timer has not triggered a wakeup event	0x0*
	WAKEUP_BB_TIMER_EVENT_SET	BB timer has triggered a wakeup event at least once	0x1

Field Name	Value Symbol	Value Description	Hex Value
DIO3_WAKEUP	WAKEUP_DIO3_EVENT_NOT_SET	DIO3 has not triggered a wakeup event	0x0*
	WAKEUP_DIO3_EVENT_SET	DIO3 has triggered a wakeup event at least once	0x1
DIO2_WAKEUP	WAKEUP_DIO2_EVENT_NOT_SET	DIO2 has not triggered a wakeup event	0x0*
	WAKEUP_DIO2_EVENT_SET	DIO2 has triggered a wakeup event at least once	0x1
DIO1_WAKEUP	WAKEUP_DIO1_EVENT_NOT_SET	DIO1 has not triggered a wakeup event	0x0*
	WAKEUP_DIO1_EVENT_SET	DIO1 has triggered a wakeup event at least once	0x1
DIO0_WAKEUP	WAKEUP_DIO0_EVENT_NOT_SET	DIO0 has not triggered a wakeup event	0x0*
	WAKEUP_DIO0_EVENT_SET	DIO0 has triggered a wakeup event at least once	0x1
DCDC_OVERLOAD_CLEAR	WAKEUP_DCDC_OVERLOAD_CLEAR	Reset the sticky WAKEUP_DCDC_OVERLOAD flag	0x1
WAKEUP_PAD_WAKEUP_CLEAR	WAKEUP_PAD_EVENT_CLEAR	Reset the sticky WAKEUP_PAD_WAKEUP flag	0x1
RTC_ALARM_WAKEUP_CLEAR	WAKEUP_RTC_ALARM_CLEAR	Reset the sticky WAKEUP_RTC_ALARM flag	0x1
BB_TIMER_WAKEUP_CLEAR	WAKEUP_BB_TIMER_CLEAR	Reset the sticky WAKEUP_BB_TIMER flag	0x1
DIO3_WAKEUP_CLEAR	WAKEUP_DIO3_EVENT_CLEAR	Reset the sticky WAKEUP_DIO3_EVENT flag	0x1
DIO2_WAKEUP_CLEAR	WAKEUP_DIO2_EVENT_CLEAR	Reset the sticky WAKEUP_DIO2_EVENT flag	0x1
DIO1_WAKEUP_CLEAR	WAKEUP_DIO1_EVENT_CLEAR	Reset the sticky WAKEUP_DIO1_EVENT flag	0x1
DIO0_WAKEUP_CLEAR	WAKEUP_DIO0_EVENT_CLEAR	Reset the sticky WAKEUP_DIO0_EVENT flag	0x1

5.4.7 SYSCTRL_MEM_ACCESS_CFG

Bit Field	Field Name	Description
30:24	WAKEUP_ADDR_PACKED	Wakeup restore address in packed 7-bit format. When written, SYSCTRL_WAKEUP_ADDR is updated. This field reads back as zero when SYSCTRL_WAKEUP_ADDR does not point to an enabled RAM instance.
21	DSP_DRAM5_ACCESS	DSP PRAM5 access configuration
20	DSP_DRAM4_ACCESS	DSP PRAM4 access configuration
19	DSP_DRAM3_ACCESS	DSP PRAM3 access configuration
18	DSP_DRAM2_ACCESS	DSP PRAM2 access configuration
17	DSP_DRAM1_ACCESS	DSP PRAM1 access configuration
16	DSP_DRAM0_ACCESS	DSP PRAM0 access configuration
15	DSP_PRAM3_ACCESS	DSP PRAM3 access configuration

Bit Field	Field Name	Description
14	DSP_PRAM2_ACCESS	DSP PRAM2 access configuration
13	DSP_PRAM1_ACCESS	DSP PRAM1 access configuration
12	DSP_PRAM0_ACCESS	DSP PRAM0 access configuration
11	BB_DRAM1_ACCESS	Baseband DRAM1 access configuration
10	BB_DRAM0_ACCESS	Baseband DRAM0 access configuration
8	DRAM2_ACCESS	DRAM2 access configuration
7	DRAM1_ACCESS	DRAM1 access configuration
6	DRAM0_ACCESS	DRAM0 access configuration
5	PRAM3_ACCESS	PRAM3 access configuration
4	PRAM2_ACCESS	PRAM2 access configuration
3	PRAM1_ACCESS	PRAM1 access configuration
2	PRAM0_ACCESS	PRAM0 access configuration
1	FLASH_ACCESS	Flash access configuration
0	PROM_ACCESS	PROM access configuration

Field Name	Value Symbol	Value Description	Hex Value
DSP_DRAM5_ACCESS	DSP_DRAM5_ACCESS_DISABLE	DSP DRAM5 access disabled	0x0*
	DSP_DRAM5_ACCESS_ENABLE	DSP DRAM5 access enabled	0x1
DSP_DRAM4_ACCESS	DSP_DRAM4_ACCESS_DISABLE	DSP DRAM4 access disabled	0x0*
	DSP_DRAM4_ACCESS_ENABLE	DSP DRAM4 access enabled	0x1
DSP_DRAM3_ACCESS	DSP_DRAM3_ACCESS_DISABLE	DSP DRAM3 access disabled	0x0*
	DSP_DRAM3_ACCESS_ENABLE	DSP DRAM3 access enabled	0x1
DSP_DRAM2_ACCESS	DSP_DRAM2_ACCESS_DISABLE	DSP DRAM2 access disabled	0x0*
	DSP_DRAM2_ACCESS_ENABLE	DSP DRAM2 access enabled	0x1
DSP_DRAM1_ACCESS	DSP_DRAM1_ACCESS_DISABLE	DSP DRAM1 access disabled	0x0*
	DSP_DRAM1_ACCESS_ENABLE	DSP DRAM1 access enabled	0x1
DSP_DRAM0_ACCESS	DSP_DRAM0_ACCESS_DISABLE	DSP DRAM0 access disabled	0x0*
	DSP_DRAM0_ACCESS_ENABLE	DSP DRAM0 access enabled	0x1
DSP_PRAM3_ACCESS	DSP_PRAM3_ACCESS_DISABLE	DSP PRAM3 access disabled	0x0*
	DSP_PRAM3_ACCESS_ENABLE	DSP PRAM3 access enabled	0x1
DSP_PRAM2_ACCESS	DSP_PRAM2_ACCESS_DISABLE	DSP PRAM2 access disabled	0x0*
	DSP_PRAM2_ACCESS_ENABLE	DSP PRAM2 access enabled	0x1
DSP_PRAM1_ACCESS	DSP_PRAM1_ACCESS_DISABLE	DSP PRAM1 access disabled	0x0*
	DSP_PRAM1_ACCESS_ENABLE	DSP PRAM1 access enabled	0x1
DSP_PRAM0_ACCESS	DSP_PRAM0_ACCESS_DISABLE	DSP PRAM0 access disabled	0x0*
	DSP_PRAM0_ACCESS_ENABLE	DSP PRAM0 access enabled	0x1
BB_DRAM1_ACCESS	BB_DRAM1_ACCESS_DISABLE	Baseband DRAM1 access disabled	0x0*
	BB_DRAM1_ACCESS_ENABLE	Baseband DRAM1 access enabled	0x1

Field Name	Value Symbol	Value Description	Hex Value
BB_DRAM0_ACCESS	BB_DRAM0_ACCESS_DISABLE	Baseband DRAM0 access disabled	0x0*
	BB_DRAM0_ACCESS_ENABLE	Baseband DRAM0 access enabled	0x1
DRAM2_ACCESS	DRAM2_ACCESS_DISABLE	DRAM2 access disabled	0x0*
	DRAM2_ACCESS_ENABLE	DRAM2 access enabled	0x1
DRAM1_ACCESS	DRAM1_ACCESS_DISABLE	DRAM1 access disabled	0x0*
	DRAM1_ACCESS_ENABLE	DRAM1 access enabled	0x1
DRAM0_ACCESS	DRAM0_ACCESS_DISABLE	DRAM0 access disabled	0x0
	DRAM0_ACCESS_ENABLE	DRAM0 access enabled	0x1*
PRAM3_ACCESS	PRAM3_ACCESS_DISABLE	PRAM3 access disabled	0x0*
	PRAM3_ACCESS_ENABLE	PRAM3 access enabled	0x1
PRAM2_ACCESS	PRAM2_ACCESS_DISABLE	PRAM2 access disabled	0x0*
	PRAM2_ACCESS_ENABLE	PRAM2 access enabled	0x1
PRAM1_ACCESS	PRAM1_ACCESS_DISABLE	PRAM1 access disabled	0x0*
	PRAM1_ACCESS_ENABLE	PRAM1 access enabled	0x1
PRAM0_ACCESS	PRAM0_ACCESS_DISABLE	PRAM0 access disabled	0x0*
	PRAM0_ACCESS_ENABLE	PRAM0 access enabled	0x1
FLASH_ACCESS	FLASH_ACCESS_DISABLE	Flash access disabled	0x0*
	FLASH_ACCESS_ENABLE	Flash access enabled	0x1
PROM_ACCESS	PROM_ACCESS_DISABLE	PROM access disabled	0x0
	PROM_ACCESS_ENABLE	PROM access enabled	0x1*

5.4.8 SYSCTRL_WAKEUP_ADDR

Bit Field	Field Name	Description
31:0	WAKEUP_ADDR	Wakeup restore address in unpacked 32-bit format. When written, the WAKEUP_ADDR_PACKED field of SYSCTRL_MEM_ACCESS_CFG is updated. Bits 0-12 must be 0x0000 or 0x1FE8 (top or bottom of memory instance). Bits 17-20, 22-28 and 30-31 must be zero. When the WAKEUP_ADDR_PACKED field does not point to memory that is currently accessible, then SYSCTRL_WAKEUP_ADDR reads back as all zeros.

5.4.9 SYSCTRL_WAKEUP_PAD

Bit Field	Field Name	Description
0	WAKEUP_PAD_VALUE	WAKEUP pad value

Field Name	Value Symbol	Value Description	Hex Value
WAKEUP_PAD_VALUE	WAKEUP_PAD_LOW	WAKEUP pad value equal to '0'	0x0*
	WAKEUP_PAD_HIGH	WAKEUP pad value equal to '1'	0x1

5.4.10 ACS_WAKEUP_GP_DATA

Bit Field	Field Name	Description
31:0	`GP_DATA`	32-bit General-Purpose RW Data

5.5 Resets

The RSL10 SoC contains a variety of reset sources that can be used to reset the entire RSL10 system, or a set of its system components. A system reset causes the system to restart, and status bits to be set for each of the relevant reset causes. These reset status bits exist in the `ACS_RESET_STATUS` and `DIG_RESET_STATUS` registers. The reset bits and their encoding can be seen in Figure 5, which also shows the ordering of reset flags. These flags remain set until cleared by writing to their associated clear flags.

IMPORTANT: To clear the status bits that indicate the source of a reset, the `DIG_RESET_STATUS` register must be cleared before the `ACS_RESET_STATUS` register.

We recommend clearing all reset status flags at the start of application execution (after the reset source has been determined), to allow future executions to determine the cause of a reset or resets.

ACS_RESET_STATUS							DIG_RESET_STATUS				
POR_RESET_FLAG	PAD_RESET_FLAG	VDDC_RESET_FLAG	VDDM_RESET_FLAG	VDDA_RESET_FLAG	CLK_DET_RESET_FLAG	TIMEOUT_RESET_FLAG	ACS_RESET_FLAG	CM3_SW_RESET_FLAG	WATCHDOG_RESET_FLAG	LOCKUP_FLAG	Flags / Reset sources
1	-	-	-	-	-	-	1	-	-	-	Reset due to PMU-POR
0	1	-	-	-	-	-	1	-	-	-	Reset due to the NRESET pad
0	-	1	-	-	-	-	1	-	-	-	Reset due to VDDC regulator
0	-	-	1	-	-	-	1	-	-	-	Reset due to VDDM regulator
0	-	-	-	1	-	-	1	-	-	-	Reset due to VDDA charge pump
0	-	-	-	-	1	-	1	-	-	-	Reset due to system clock detector
0	-	-	-	-	-	1	1	-	-	-	Reset due to power state machine timeout
0	0	0	0	0	0	0	1	-	-	-	Reset due to wake-up from sleep mode
-	-	-	-	-	-	-	-	1	-	-	Reset due to the ARM Cortex-M3 processor software system reset
-	-	-	-	-	-	-	-	-	1	0	Reset due to the watchdog with the ARM Cortex-M3 processor not in lockup state
-	-	-	-	-	-	-	-	-	1	1	Reset due to the watchdog with the ARM Cortex-M3 processor in lockup state

Figure 5. Reset Flag Decoding

NOTE: For a given reset cause, any bit with an explicit value will be set or cleared as appropriate. Other bit values that are not explicit are marked as unknown, as they could be set or cleared based on reset causes that could coexist with the given reset cause.

Exiting from any full system reset triggers the POR sequence. For all full system resets, the `ACS_RESET_FLAG` in the `DIG_RESET_STATUS` register is set, indicating that an asynchronous reset has occurred, including a full reset of the analog system resources. For a wakeup from sleep, no other asynchronous status bits will be set; but for all other asynchronous reset sources, other status bits will be set by:

- The power supervisory:
  - A POR occurs on power-on for the device and the `POR_RESET_FLAG` bit is set.
  - If VDDC falls below 90% of the configured VDDC target value, the system will reset and the `VDDC_RESET_FLAG` bit will be set.
  - If VDDM falls below 90% of the configured VDDM target value, the system will reset and the `VDDM_RESET_FLAG` bit will be set.
  - If VDDA falls below 1.7 V, the system will reset and the `VDDA_RESET_FLAG` bit will be set.

  - If a failure occurs within the power supply during power up, a timeout will occur, and the `TIMEOUT_RESET_FLAG` bit will be set. This can happen if the startup sequence (bandgap startup, VCC startup, VDDA, VDDC and VDDM startup) is too long (greater than 1 ms unexpected delay) due to a configuration problem.

- The clock detection circuits (see Section 6.4.1, "Clock Detector and System Monitor" on page 89): if the clock detection circuit causes a reset, the `CLK_DETECT_RESET_FLAG` bit will be set.
- The dedicated NRESET pad: if a reset using this pad occurs, the `PAD_RESET_FLAG` bit will be set.

Partial resets supported by the RSL10 system include:

- The watchdog timer (see Section 12.4, "Watchdog Timer" on page 375):
  - If the Arm Cortex-M3 processor is still running when the watchdog reset occurs, the `WATCHDOG_REFRESH_FLAG` bit will be set.
  - If the Arm Cortex-M3 processor is locked up (for example, due to a fault that occurred while handling other faults), the `LOCKUP_FLAG` bit will be set.

  All watchdog timer resets are designed to reset the digital system and the analog control system's registers, but not the underlying analog control circuits.

- The Arm Cortex-M3 processor resets, which provide:
  - A reset of the processor, excluding debug logic, but including other digital components due to the Arm Cortex-M3 software processor system reset. This reset is triggered whenever the a software system reset is requested by setting bit 2 of the `SCB_AIRCR` internal Arm Cortex-M3 processor register. When this reset is requested, the digital system also sets the `DIG_RESET_STATUS_CM3_SW_RESET_FLAG` bit.
  - A reset of the processor only, excluding debug logic. This reset is triggered when bit 0 of the `SCB_AIRCR` register is set, and does not set a reset status bit.
  - A reset for the JTAG controller in the JTAG debug port, synchronized with JTCK (see Section 3.2, "Debug Port" on page 27 for more information). This reset source does not set a reset status bit.

- An LPDSP32 DSP reset (see Chapter 4, "LPDSP32 Processor" on page 33): resetting this block resets the LPDSP32 DSP and its related components, and does not set a reset status bit.
- Resets for individual interfaces or peripherals only reset the associated components.

5.5.1 Reset Status Register

Register Name	Register Description	Address
DIG_RESET_STATUS	Reset status register	0x40000200
ACS_RESET_STATUS	ACS reset source status registers	0x40001354

5.5.1.1 DIG_RESET_STATUS

Bit Field	Field Name	Description
7	`LOCKUP_RESET_FLAG_CLEAR`	Reset the sticky LOCKUP flag
6	`WATCHDOG_RESET_FLAG_CLEAR`	Reset the sticky Watchdog time-out reset flag
5	`CM3_SW_RESET_FLAG_CLEAR`	Reset the sticky Arm Cortex-M3 processor software reset flag
4	`ACS_RESET_FLAG_CLEAR`	Reset the sticky ACS reset flag
3	`LOCKUP_FLAG`	Sticky flag that detects that a LOCKUP occurred

Bit Field	Field Name	Description
2	WATCHDOG_RESET_FLAG	Sticky flag that detects that a Watchdog time-out reset occurred
1	CM3_SW_RESET_FLAG	Sticky flag that detects that an Arm Cortex-M3 processor software reset occurred
0	ACS_RESET_FLAG	Sticky flag that detects that an ACS reset occurred

Field Name	Value Symbol	Value Description	Hex Value
LOCKUP_RESET_FLAG_CLEAR	LOCKUP_FLAG_CLEAR	Reset the sticky LOCKUP flag	0x1
WATCHDOG_RESET_FLAG_CLEAR	WATCHDOG_RESET_FLAG_CLEAR	Reset the sticky Watchdog time-out reset flag	0x1
CM3_SW_RESET_FLAG_CLEAR	CM3_SW_RESET_FLAG_CLEAR	Reset the sticky Arm Cortex-M3 processor software reset flag	0x1
ACS_RESET_FLAG_CLEAR	ACS_RESET_FLAG_CLEAR	Reset the sticky ACS reset flag	0x1
LOCKUP_FLAG	LOCKUP_NOT_SET	The LOCKUP has not triggered at least once	0x0*
	LOCKUP_SET	The LOCKUP was triggered at least once	0x1
WATCHDOG_RESET_FLAG	WATCHDOG_RESET_NOT_SET	The Watchdog time-out reset has not triggered at least once	0x0*
	WATCHDOG_RESET_SET	The Watchdog time-out reset was triggered at least once since this status bit was last cleared	0x1
CM3_SW_RESET_FLAG	CM3_SW_RESET_NOT_SET	The Arm Cortex-M3 processor software system reset has not triggered at least once	0x0*
	CM3_SW_RESET_SET	The Arm Cortex-M3 processor software system reset was triggered at least once since this status bit was last cleared	0x1
ACS_RESET_FLAG	ACS_RESET_NOT_SET	The ACS reset has not triggered at least once	0x0
	ACS_RESET_SET	The ACS reset was triggered at least once since this status bit was last cleared	0x1*

5.5.2 ACS_RESET_STATUS

Bit Field	Field Name	Description
14	TIMEOUT_RESET_FLAG	Sticky flag that detects that a timeout in the power up sequence occurred
13	CLK_DET_RESET_FLAG	Sticky flag that detects that a clock detector reset occurred
12	VDDA_RESET_FLAG	Sticky flag that detects that a VDDA reset occurred (triggered by vdda_ready = 0)
11	VDDM_RESET_FLAG	Sticky flag that detects that a VDDM reset occurred (triggered by vddc_ready = 0)
10	VDDC_RESET_FLAG	Sticky flag that detects that a VDDC reset occurred (triggered by vddc_ready = 0)

Bit Field	Field Name	Description
9	PAD_RESET_FLAG	Sticky flag that detects that a reset occurred due to pad NRESET
8	POR_RESET_FLAG	Sticky flag that detects that a POR reset occurred
6	TIMEOUT_RESET_FLAG_CLEAR	Reset the sticky TIMEOUT_RESET flag.
5	CLK_DET_RESET_FLAG_CLEAR	Reset the sticky CLK_DET_RESET flag.
4	VDDA_RESET_FLAG_CLEAR	Reset the sticky VDDA_RESET flag.
3	VDDM_RESET_FLAG_CLEAR	Reset the sticky VDDM_RESET flag.
2	VDDC_RESET_FLAG_CLEAR	Reset the sticky VDDC_RESET flag.
1	PAD_RESET_FLAG_CLEAR	Reset the sticky PAD_RESET flag.
0	POR_RESET_FLAG_CLEAR	Reset the sticky POR_RESET flag.

Field Name	Value Symbol	Value Description	Hex Value
TIMEOUT_RESET_FLAG	TIMEOUT_RESET_FLAG_NOT_SET	The timeout reset has not triggered at least once	0x0*
	TIMEOUT_RESET_FLAG_SET	The timeout reset was triggered at least once since this status bit was last cleared	0x1
CLK_DET_RESET_FLAG	CLK_DET_RESET_FLAG_NOT_SET	The clock detector reset has not triggered at least once	0x0*
	CLK_DET_RESET_FLAG_SET	The clock detector reset was triggered at least once since this status bit was last cleared	0x1
VDDA_RESET_FLAG	VDDA_RESET_FLAG_NOT_SET	The VDDA reset has not triggered at least once	0x0
	VDDA_RESET_FLAG_SET	The VDDA reset was triggered at least once since this status bit was last cleared	0x1*
VDDM_RESET_FLAG	VDDM_RESET_FLAG_NOT_SET	The VDDM reset has not triggered at least once	0x0
	VDDM_RESET_FLAG_SET	The VDDM reset was triggered at least once since this status bit was last cleared	0x1*
VDDC_RESET_FLAG	VDDC_RESET_FLAG_NOT_SET	The VDDC reset has not triggered at least once	0x0
	VDDC_RESET_FLAG_SET	The VDDC reset was triggered at least once since this status bit was last cleared	0x1*
PAD_RESET_FLAG	PAD_RESET_FLAG_NOT_SET	The NRESET pad reset has not triggered at least once	0x0*
	PAD_RESET_FLAG_SET	The NRESET pad reset was triggered at least once since this status bit was last cleared	0x1
POR_RESET_FLAG	POR_RESET_FLAG_NOT_SET	The POR reset has not triggered at least once	0x0
	POR_RESET_FLAG_SET	The POR reset was triggered at least once since this status bit was last cleared	0x1*

Field Name	Value Symbol	Value Description	Hex Value
`TIMEOUT_RESET_FLAG_CLEAR`	`TIMEOUT_RESET_FLAG_CLEAR`	Reset the sticky TIMEOUT_RESET flag.	0x1
`CLK_DET_RESET_FLAG_CLEAR`	`CLK_DET_RESET_FLAG_CLEAR`	Reset the sticky CLK_DET_RESET flag.	0x1
`VDDA_RESET_FLAG_CLEAR`	`VDDA_RESET_FLAG_CLEAR`	Reset the sticky VDDA_RESET flag.	0x1
`VDDM_RESET_FLAG_CLEAR`	`VDDM_RESET_FLAG_CLEAR`	Reset the sticky VDDM_RESET flag.	0x1
`VDDC_RESET_FLAG_CLEAR`	`VDDC_RESET_FLAG_CLEAR`	Reset the sticky VDDC_RESET flag.	0x1
`PAD_RESET_FLAG_CLEAR`	`PAD_RESET_FLAG_CLEAR`	Reset the sticky PAD_RESET flag.	0x1
`POR_RESET_FLAG_CLEAR`	`POR_RESET_FLAG_CLEAR`	Reset the sticky POR_RESET flag.	0x1

5.5.3 The nRESET Pad

The nRESET pad contains a pull-up resistor that cannot be disabled by the user. During the boot process, the pull-up resistor is 100 kΩ; after the boot process is complete, it automatically switches to 200 kΩ without any user intervention.

5.6 ANALOG TEST SIGNALS

The AOUT pad can have the following functions:

1. Analog test bus is used to monitor internal analog signals for characterization and debug.
   - Bandgap regulated supply voltage
   - VDDRF
   - Baseband timer supply voltage
   - VDDC
   - VDDA
   - VDDM
   - VDDPA

2. Digital test bus is used to monitor internal digital signals for characterization and debug.
   - Bandgap ready
   - VCC ready
   - DC-DC overload / activated
   - VDDRF ready
   - VDDC ready
   - VDDM ready
   - VDDA ready
   - 32 kHz crystal oscillator clock
   - 32 kHz RC oscillator clock

DIO0 can be used to output the RTC clock to control an external device. This mode is configured using the `ACS_AOUT_CTRL` register, which has the following configurable bit-fields:

1. `RTC_CLOCK_DIO0_START` configures the RTC clock to be output to DIO0 starting at the defined interval, between 125 ms and 8 s.
2. `RTC_CLOCK_DIO0_STOP_SRC` is used to select what DIO (from DIO0 to DIO3) will stop the output of the 32 kHz RTC clock.

3. `RTC_CLOCK_DIO0_STOP_EDGE` is used to select whether this clock output is stopped on a rising or a falling edge on the selected DIO. For a clean (glitchless) signal on DIO0, this event needs to occur synchronously with the clock falling edge, as the detected event *gates* (sets to 0) the signal on DIO0.

The control register in the ACS configures which signal is brought out on the AOUT and DIO0 pads. The `ACS_AOUT_CTRL_TEST_AOUT` bit field provides a selection from 32 test signals.

5.6.1 Analog Output Configuration Register

Register Name	Register Description	Address
ACS_AOUT_CTRL	Analog output configuration register	0x40001358

5.6.2 ACS_AOUT_CTRL

Bit Field	Field Name	Description
13	`RTC_CLOCK_DIO0_STOP_EDGE`	Stop edge for RTC clock output on AOUT
12:11	`RTC_CLOCK_DIO0_STOP_SRC`	Stop source for RTC clock output on AOUT
10:8	`RTC_CLOCK_DIO0_START`	Start event for RTC clock output on AOUT (RTC prescaler and counter need to be enabled)
4:0	`TEST_AOUT`	AOUT test signal selection

Field Name	Value Symbol	Value Description	Hex Value
`RTC_CLOCK_DIO0_STOP_EDGE`	`DIO0_RTC_CLK_STOP_RISING`	Stop to output RTC clock on rising edge	0x0*
	`DIO0_RTC_CLK_STOP_FALLING`	Stop to output RTC clock on falling edge	0x1
`RTC_CLOCK_DIO0_STOP_SRC`	`DIO0_RTC_CLK_STOP_DIO0`	Stop to output RTC clock on DIO0 event	0x0*
	`DIO0_RTC_CLK_STOP_DIO1`	Stop to output RTC clock on DIO1 event	0x1
	`DIO0_RTC_CLK_STOP_DIO2`	Stop to output RTC clock on DIO2 event	0x2
	`DIO0_RTC_CLK_STOP_DIO3`	Stop to output RTC clock on DIO3 event	0x3
`RTC_CLOCK_DIO0_START`	`DIO0_RTC_CLK_DISABLE`	No start event (DIO0 not driven)	0x0*
	`DIO0_RTC_CLK_125MS`	Start to output RTC clock every 125 ms	0x1
	`DIO0_RTC_CLK_250MS`	Start to output RTC clock every 250 ms	0x2
	`DIO0_RTC_CLK_500MS`	Start to output RTC clock every 500 ms	0x3
	`DIO0_RTC_CLK_1S`	Start to output RTC clock every 1 s	0x4
	`DIO0_RTC_CLK_2S`	Start to output RTC clock every 2 s	0x5
	`DIO0_RTC_CLK_4S`	Start to output RTC clock every 4 s	0x6
	`DIO0_RTC_CLK_8S`	Start to output RTC clock every 8 s	0x7

Field Name	Value Symbol	Value Description	Hex Value
TEST_AOUT	AOUT_VSSA	AOUT grounded	0x0*
	AOUT_VCC_SENSE	AOUT high / VCC connected on AOUT (can be sensed for 4 wires measurement of the load regulation)	0x1
	AOUT_VREF_0P75V_OUTPUT	Bandgap reference voltage 0p75V connected on AOUT	0x2
	AOUT_VREF_0P67V_OUTPUT	Bandgap reference voltage 0p67V connected on AOUT	0x3
	AOUT_IREF_50N_OUTPUT	Bandgap iref current source connected on AOUT	0x4
	AOUT_IREF_1N_OUTPUT	PTAT iref current source connected on AOUT	0x5
	AOUT_VDDACS_OUTPUT	vddacs voltage connected on AOUT	0x6
	AOUT_VREF_0P75V_BUF_OUTPUT	Bandgap buffered reference voltage 0p75V connected on AOUT	0x7
	AOUT_VREG_BG	Bandgap regulated supply voltage	0x8
	AOUT_VDDRF_SW	vddrf_sw voltage connected on AOUT	0x9
	AOUT_VDDRF_SENSE	VDDRF connected on AOUT (can be sensed for 4 wires measurement of the load regulation)	0xA
	AOUT_VDDT	Baseband timer supply voltage	0xB
	AOUT_VDDC_SENSE	VDDC connected on AOUT (can be sensed for 4 wires measurement of the load regulation)	0xC
	AOUT_VDDA_SW	vdda_sw voltage connected on AOUT	0xD
	AOUT_VDDA_SENSE	VDDA connected on AOUT (can be sensed for 4 wires measurement of the load regulation)	0xE
	AOUT_VDDM_SENSE	VDDM connected on AOUT (can be sensed for 4 wires measurement of the load regulation)	0xF
	AOUT_NC	AOUT floating (for pad leakage measurement)	0x10
	AOUT_VDDPA_SENSE	VDDPA connected on AOUT (can be sensed for 4 wires measurement of the load regulation)	0x11
	AOUT_VDDPA_ISENSE	VDDPA current sensing circuit connected to AOUT	0x12

Field Name	Value Symbol	Value Description	Hex Value
TEST_AOUT (continued)	AOUT_TM0	Flash TM0 connected to AOUT	0x13
	AOUT_BG_READY	Bandgap ready on AOUT (digital signal using VSSA and VCC states)	0x14
	AOUT_VCC_READY	vcc_ready on AOUT (digital signal using VSSA and VCC states)	0x15
	AOUT_DCDC_OVERLOAD	dcdc_overload on AOUT (digital signal using VSSA and VCC states)	0x16
	AOUT_DCDC_ACTIVATED	dcdc_activated on AOUT (digital signal using VSSA and VCC states)	0x17
	AOUT_VDDRF_READY	vddrf_ready on AOUT (digital signal using VSSA and VCC states)	0x18
	AOUT_VDDC_READY	vddc_ready on AOUT (digital signal using VSSA and VCC states)	0x19
	AOUT_VDDM_READY	vddm_ready on AOUT (digital signal using VSSA and VCC states)	0x1A
	AOUT_VDDA_READY	vdda_ready on AOUT (digital signal using VSSA and VCC states)	0x1B
	AOUT_CLK_PRESENT	Clock present from clock detector on AOUT (digital signal using VSSA and VCC states)	0x1C
	AOUT_XTAL_OK	XTAL ok on AOUT (digital signal using VSSA and VCC states)	0x1D
	AOUT_XTAL_CLK	XTAL clock on AOUT (digital signal using VSSA and VCC states)	0x1E
	AOUT_CLK_32K	32 kHz RC oscillator clock on AOUT (digital signal using VSSA and VCC states)	0x1F

CHAPTER 6

Clocking

6.1 Overview

All clocks and clock domains in the RSL10 system are derived from the system clock (SYSCLK) or the standby clock (STANDBYCLK).

SYSCLK can be generated from one of five different sources for maximum flexibility. Available sources for SYSCLK include:

1. The internal RC oscillator (discussed in Section 6.2.1, "RC Oscillator" on page 74)
2. The RF clock provided by the 48 MHz crystal oscillator (discussed in Section 6.2.2, "48 MHz Crystal Oscillator" on page 74)
3. STANDBYCLK
4. The external clock pad (discussed in Section 6.2.5, "External Clock Input (EXTCLK)" on page 75)
5. The SWCLK pad from the SWJ-DP (discussed in Section 6.2.6, "Debug Port Clock" on page 76)

For more information about configuring the system clock, see Section 6.3.1, "System Clock (SYSCLK)" on page 78.

Similarly, the STANDBYCLK can be generated from one of six different sources, including:

1. The internal standby oscillator (discussed in Section 6.2.3, "Standby RC Oscillator" on page 74)
2. The 32 kHz crystal oscillator (discussed in Section 6.2.4, "32 kHz Crystal Oscillator" on page 75)
3. A DIO source from one of DIOs 0 to 3

For more information about configuring STANDBYCLK, see Section 6.3.2, "Standby Clock (STANDBYCLK)" on page 79.

A top-level clock diagram showing the clock generation and distribution of clocks within the RSL10 system is provided in Figure 6.

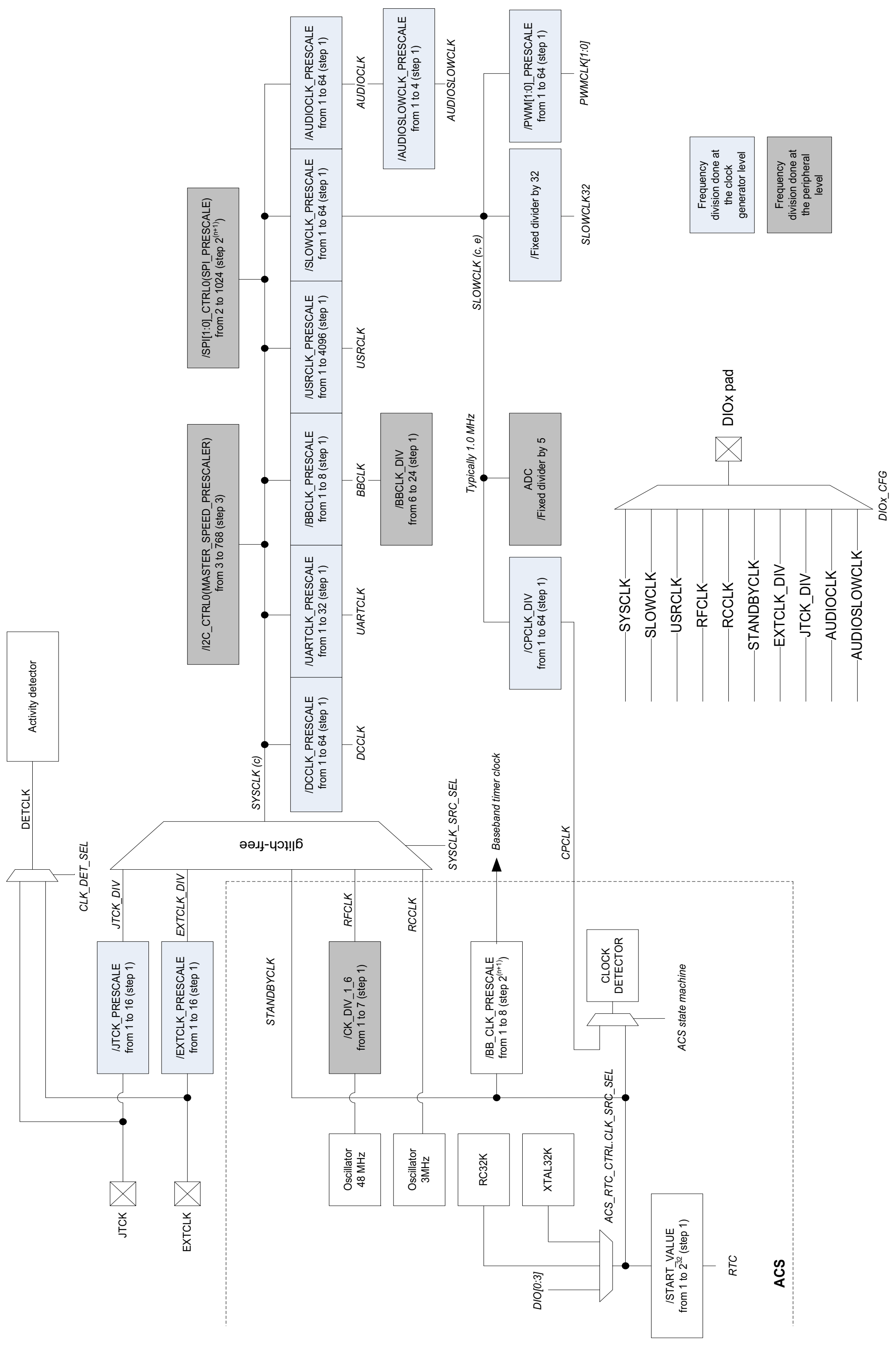

JTCK
EXTCLK
/JTCK_PRESCALE from 1 to 16 (step 1)
/EXTCLK_PRESCALE from 1 to 16 (step 1)
JTCK_DIV
EXTCLK_DIV
CLK_DET_SEL
DETCLK
Activity detector
glitch-free
SYSCLK_SRC_SEL
SYSCLK (c)
STANDBYCLK
RFCLK
RCCLK
/CK_DIV_1_6 from 1 to 7 (step 1)
Oscillator 48 MHz
Oscillator 3MHz
RC32K
XTAL32K
DIO[0:3]
ACS_RTC_CTRL.CLK_SRC_SEL
/START_VALUE from 1 to 2^32 (step 1)
RTC
ACS
/BB_CLK_PRESCALE from 1 to 8 (step 2^(n+1))
Baseband timer clock
CLOCK DETECTOR
ACS state machine
CPCLK
/DCCLK_PRESCALE from 1 to 64 (step 1)
DCCLK
/UARTCLK_PRESCALE from 1 to 32 (step 1)
UARTCLK
/BBCLK_PRESCALE from 1 to 8 (step 1)
BBCLK
/BBCLK_DIV from 6 to 24 (step 1)
/USRCLK_PRESCALE from 1 to 4096 (step 1)
USRCLK
/SLOWCLK_PRESCALE from 1 to 64 (step 1)
/AUDIOCLK_PRESCALE from 1 to 64 (step 1)
AUDIOCLK
/AUDIOSLOWCLK_PRESCALE from 1 to 4 (step 1)
AUDIOSLOWCLK
/I2C_CTRL0(MASTER_SPEED_PRESCALER) from 3 to 768 (step 3)
/SPI[1:0]_CTRL0(SPI_PRESCALE) from 2 to 1024 (step 2^(n+1))
/CPCLK_DIV from 1 to 64 (step 1)
Typically 1.0 MHz
ADC /Fixed divider by 5
SLOWCLK (c, e)
/Fixed divider by 32
SLOWCLK32
/PWM[1:0]_PRESCALE from 1 to 64 (step 1)
PWMCLK[1:0]
DIOx pad
DIOx_CFG
SYSCLK
SLOWCLK
USRCLK
RFCLK
RCCLK
STANDBYCLK
EXTCLK_DIV
JTCK_DIV
AUDIOCLK
AUDIOSLOWCLK
Frequency division done at the clock generator level
Frequency division done at the peripheral level

Figure 6. Clock Distribution

6.2 Clock Generation

6.2.1 RC Oscillator

The RC oscillator is a ring oscillator that produces a trimmable output clock (RCCLK) that is used by the RSL10 system as SYSCLK at startup. It can be used while operating in Run Mode without RF traffic to minimize current consumption and to maximize the amount of processing that can be completed while waiting for the 48 MHz oscillator when it is started.

The frequency of the RC oscillator is trimmed using the `ACS_RCOSC_CTRL_FTRIM_START` bit-field from the `ACS_RCOSC_CTRL` register, providing a default output frequency from this clock source of 3 MHz. If this bit-field has been written, the `ACS_RCOSC_CTRL_FTRIM_FLAG` bit will indicate that the RC oscillator has been trimmed. This trimming is supplemented by a frequency multiplier, enabled using the `ASC_RCOSC_CTRL_CLOCK_MULT` bit from the `ASC_RCOSC_CTRL` register that multiplies the output of this RC oscillator by a factor of 4.

CAUTION: When enabled, the frequency multiplier applied to the RC oscillator modifies the effective RC constant of the ring oscillator. As such, this configuration bit provides a nominal multiplication by 4, not an absolute multiplication by 4, and separate configuration trim settings must be kept for both un-multiplied and multiplied settings.

Calibrated values for both the un-multiplied and multiplied trim settings are provided as part of the manufacturing records in NVR4. For more information, see the *RSL10 Firmware Reference*.

RCCLK can be output through one or more DIO pads using the DIO components. For more information about the DIO configuration, see Section 10.2, “Functional Configuration” on page 259.

6.2.2 48 MHz Crystal Oscillator

The RF front-end for the RSL10 system includes a 48 MHz crystal oscillator. To use this crystal oscillator, the RF front-end must be powered with access enabled, as described in Section 8.1, “Overview” on page 131.

To enable the 48 MHz crystal oscillator, set the `XTAL_CTRL_XO_EN_B_REG` bit from the RF front-end `XTAL_CTRL` register. When enabled, the 48 MHz crystal oscillator takes some time before it is ready for use by the rest of the system. When this clock is ready, the `ANALOG_INFO_CLK_DIG_READY` bit from the `ANALOG_INFO` RF front-end register will be set. When the PLL based on this oscillator is ready, the `ANALOG_INFO_CLK_PLL_READY` bit from the `ANALOG_INFO` RF front-end register will also be set. Information about further configuration of this oscillator can be found in Section 8.2.1, “48 MHz Crystal Oscillator” on page 135.

NOTE: When processing RF traffic, the RF front-end is always directly clocked from the 48 MHz crystal oscillator, with the analog components of the RF front-end using a frequency synthesizer to produce an appropriate carrier for the RF traffic in the 2.4 GHz RF band.

The 48 MHz crystal oscillator is divided using the 3-bit prescaler defined in the `CK_DIV_1_6_CK_DIV_1_6` bit-field from the `CK_DIV_1_6` RF front-end register to produce RFCLK. This clock, which divides the 48 MHz clock source by a factor between 1 and 7, can be used as the source for SYSCLK, or output through one or more DIO pads using the DIO components. For more information about the DIO configuration, see Section 10.2, “Functional Configuration” on page 259.

6.2.3 Standby RC Oscillator

The standby RC oscillator is a ring oscillator that produces a trimmable output clock that can be used by the RSL10 system as a source for STANDBYCLK, and hence as a source for the RTC. This oscillator produces a nominal output frequency of 32 kHz. Enable the standby RC oscillator by setting the `ACS_RCOSC_CTRL_RC_OSC_EN` bit from the `ACS_RCOSC_CTRL` register.

The frequency of the standby RC oscillator is trimmed using the `ACS_RCOSC_CTRL_FTRIM_32K` bit-field from the `ACS_RCOSC_CTRL` register. The trimming range for this oscillator can be shifted down by approximately 25% (producing a clock with a nominal output frequency of 24 kHz) by setting the `ACS_RCOSC_CTRL_FTRIM_32K_ADJ` bit from the `ACS_RCOSC_CTRL` register.

IMPORTANT: If the standby RC oscillator is used as a source of timing for RF traffic, this oscillator should be measured using the 48 MHz crystal oscillator and the audio sink counters. Appropriate adjustments should be made to the Bluetooth baseband timer driven counters and RTC starting countdown setting stored to the `ACS_RTC_CFG_START_VALUE` bit-field from the `ACS_RTC_CFG` register.

For more information about configuring the RTC, see Section 6.3.5, “Real-Time Clock (RTC)” on page 81. For more information about measuring the standby RC oscillator using the audio sink, see Section 13.3, “Audio Sink Clock Counters” on page 390.

6.2.4 32 kHz Crystal Oscillator

The 32 kHz crystal oscillator provides a very low-power, accurate reference clock that can be used as the source for the baseband and RTC when timing RF traffic and other elements where a high-accuracy clock is required. The 32 kHz crystal oscillator is a Pierce oscillator that provides a 32768 Hz reference clock. Configure it by using the `ACS_XTAL32K_CTRL` register. Configuration and status options for this oscillator include:

- The oscillator can be enabled or disabled by configuring the `ACS_XTAL32K_CTRL_ENABLE` bit.
- The `ACS_XTAL32K_CTRL_READY` bit indicates when the oscillator output is available for use. This status can be forced using the `ACS_XTAL32K_CTRL_FORCE_ENABLE` bit; however using this option is not recommended.
- The trim parameters for interacting with the external 32 kHz crystals can be trimmed to:
  - Provide a variety of different startup current levels using the `ACS_XTAL32K_CTRL_ITRIM` bit-field (sets the nominal startup current levels) and the `ACS_XTAL32K_CTRL_IBOOST` bit (which boosts the startup currents by an approximate factor of 4)
  - Provide an appropriate capacitive load, configured using the `ACS_XTAL32K_CTRL_CLOAD_TRIM` bit-field
- The output from the crystal can be configured to:
  - Enable or disable regulation of the amplitude of the crystal output using the `ACS_XTAL32K_CTRL_EN_AMPL_CTRL` bit
  - Include or bypass the serial output cap for the oscillator, configured using the `ACS_XTAL32K_CTRL_XIN_CAP_BYPASS_EN` bit. If the external crystal selected does not need this buffering capacitor, removing this capacitor can reduce the leakage of the 32 kHz crystal oscillator.

6.2.5 External Clock Input (EXTCLK)

An input signal from the EXTCLK input pad can be used as an external input clock source that supplies SYSCLK. Prior to use in clocking the system, this clock is prescaled using the `CLK_SYS_CFG_EXTCLK_PRESCALE` bit-field from the `CLK_SYS_CFG` register. This produces a divided EXTCLK clock input that is prescaled by between 1 and 16 to produce a potential SYSCLK frequency defined by:

$$f_{EXTCLK\_DIV} = \frac{f_{EXTCLK}}{CLK\_SYS\_CFG\_EXTCLK\_PRESCALE + 1}$$

The EXTCLK input can be configured for low-pass filtering and pull-up or pull-down resistor configuration using the `DIO_EXTCLK_CFG` register. This configuration is identical to the input physical configuration applied to the DIO inputs. For more information about the physical pad configuration of EXTCLK, see Section 10.3, “Physical Configuration” on page 263. The divided EXTCLK input can be output through one or more DIO pads using the DIO components. For more information about the DIO output configuration, see Section 10.2, “Functional Configuration” on page 259.

A clock detection circuit can be used to monitor the divided EXTCLK input. For more information, see Section 6.4.2, “External Clock Detector” on page 90.

6.2.6 Debug Port Clock

The JTCK signal from the SWJ-DP interface in the RSL10 system can be used as an external input clock source that supplies SYSCLK. Prior to use in clocking the system, this clock is prescaled using the `CLK_SYS_CFG_JTCK_PRESCALE` bit-field from the `CLK_SYS_CFG` register. This produces a divided JTCK clock output that is prescaled by between 1 and 16 to produce a potential SYSCLK frequency defined by:

$$f_{JTCK\_DIV} = \frac{f_{JTCK}}{CLK\_SYS\_CFG\_JTCK\_PRESCALE + 1}$$

IMPORTANT: Only use the JTCK pad as an input clock source if the SWJ-DP interface is configured for JTAG mode or is not used. For more information about debug port configuration, see Section 3.2, “Debug Port” on page 27.

The divided JTCK input can be output through one or more DIO pads using the DIO components. For more information about the DIO configuration, see Section 10.2, “Functional Configuration” on page 259.

A clock detection circuit can be used to monitor the divided JTCK input. For more information, see Section 6.4.2, “External Clock Detector” on page 90.

6.2.7 Clock Generation Registers

Register Name	Register Description	Address
`ACS_RCOSC_CTRL`	RC Oscillator Configuration / Control register	0x40001320
`ACS_XTAL32K_CTRL`	XTAL 32 kHz configuration register	0x40001324

6.2.7.1 ACS_RCOSC_CTRL

Bit Field	Field Name	Description
18	`CLOCK_MULT`	Enable 12 MHz mode of startup oscillator
16	`RC_OSC_EN`	Enable/Disable the 32 kHz RC Oscillator
15	`FTRIM_FLAG`	Trimming flag
13:8	`FTRIM_START`	Start RC oscillator frequency trimming
6	`FTRIM_32K_ADJ`	Adjust 32 kHz oscillator frequency range
5:0	`FTRIM_32K`	32 kHz RC oscillator frequency trimming

Field Name	Value Symbol	Value Description	Hex Value
CLOCK_MULT	RC_START_OSC_3MHZ	The startup RC Oscillator is at 3 MHz	0x0*
	RC_START_OSC_12MHZ	The startup RC Oscillator is at 12 MHz	0x1
RC_OSC_EN	RC_OSC_DISABLE	The 32kHz RC Oscillator is disabled	0x0*
	RC_OSC_ENABLE	The 32kHz RC Oscillator is enabled	0x1
FTRIM_FLAG	RC_OSC_UNCALIBRATED	The oscillators are not calibrated	0x0*
	RC_OSC_CALIBRATED	The oscillators are calibrated	0x1
FTRIM_START	RC_START_OSC_M48	-48% trimming	0x0
	RC_START_OSC_M46P5	-46.5% trimming	0x1
	RC_START_OSC_NOM	Nominal frequency	0x20*
	RC_START_OSC_P46P5	+46.5% trimming	0x3F
FTRIM_32K_ADJ	RC_OSC_RANGE_NOM	The 32 kHz RC Oscillator frequency range is nominal	0x0*
	RC_OSC_RANGE_M25	The 32 kHz RC Oscillator frequency range is lowered by 25%	0x1
FTRIM_32K	RC_OSC_M48	-48% trimming	0x0
	RC_OSC_M46P5	-46.5% trimming	0x1
	RC_OSC_NOM	Nominal frequency	0x20*
	RC_OSC_P46P5	+46.5% trimming	0x3F

6.2.7.2 ACS_XTAL32K_CTRL

Bit Field	Field Name	Description
24	READY	XTAL ready status
18	XIN_CAP_BYPASS_EN	Switch to bypass the added XIN serial cap to reduce the leakage
17	EN_AMPL_CTRL	XTAL enable amplitude control (regulation)
16	FORCE_READY	XTAL bypass the ready detector
13:8	CLOAD_TRIM	XTAL load capacitance configuration
7:4	ITRIM	XTAL current trimming
1	IBOOST	XTAL current boosting (4x)
0	ENABLE	Enable the XTAL 32 kHz oscillator

Field Name	Value Symbol	Value Description	Hex Value
READY	XTAL32K_NOT_OK	XTAL 32K not available	0x0*
	XTAL32K_OK	XTAL 32K is OK	0x1
XIN_CAP_BYPASS_EN	XTAL32K_XIN_CAP_BYPASS_DISABLE	Disable the XTAL bypass switch of the XIN serial cap	0x0*
	XTAL32K_XIN_CAP_BYPASS_ENABLE	Enable the XTAL bypass switch of the XIN serial cap	0x1
EN_AMPL_CTRL	XTAL32K_AMPL_CTRL_DISABLE	XTAL 32K amplitude control disabled	0x0*
	XTAL32K_AMPL_CTRL_ENABLE	XTAL 32K amplitude control enabled	0x1

Field Name	Value Symbol	Value Description	Hex Value
FORCE_READY	XTAL32K_NOT_FORCE_READY	XTAL 32K ready not forced	0x0*
	XTAL32K_FORCE_READY	XTAL 32K ready forced	0x1
CLOAD_TRIM	XTAL32K_CTRIM_0P0PF	0 pF internal capacitor	0x0
	XTAL32K_CTRIM_0P4PF	0.4 pF internal capacitor	0x1
	XTAL32K_CTRIM_3P6PF	3.6 pF internal capacitor	0x9*
	XTAL32K_CTRIM_25P2PF	25.2 pF internal capacitor	0x3F
ITRIM	XTAL32K_ITRIM_20NA	20 nA startup current	0x0
	XTAL32K_ITRIM_80NA	80 nA startup current	0x3
	XTAL32K_ITRIM_160NA	160 nA startup current	0x7*
	XTAL32K_ITRIM_320NA	320 nA startup current	0xF
IBOOST	XTAL32K_IBOOST_DISABLE	Disable the XTAL 32 kHz current boosting mode	0x0*
	XTAL32K_IBOOST_ENABLE	Enable the XTAL 32 kHz current boosting mode (4x itrim currents)	0x1
ENABLE	XTAL32K_DISABLE	Disable the XTAL 32 kHz oscillator	0x0*
	XTAL32K_ENABLE	Enable the XTAL 32 kHz oscillator	0x1

6.3 Clock Distribution

6.3.1 System Clock (SYSCLK)

The system clock (SYSCLK) is the primary clock for the RSL10 system and all other clocks except STANDBYCLK. The internal clock structures for the RF front-end are derived from SYSCLK.

The `CLK_SYS_CFG_SYSCLK_SRC_SEL` bit field from the `CLK_SYS_CFG` register is used to configure the source for this clock. The sources of this clock can be the following:

- The RC oscillator output (RCCLK; default configuration)
- The standby RC oscillator (through standby clock)
- The 32 kHz crystal oscillator (through standby clock)
- The divided 48 MHz crystal oscillator output (RFCLK)
- The EXTCLK input signal
- The JTCK (SWCLK) input signal

SYSCLK will typically be sourced only through STANDBYCLK when operating in Standby Mode where it is inefficient to go to Sleep Mode, but there is a period of time when the system does not need to process RF traffic or other data. If the clock source for SYSCLK is routed through STANDBYCLK, the following divided forms of SYSCLK are sourced directly from SYSCLK:

- SLOWCLK and divided forms of SLOWCLK (SLOWCLK_DIV2, SLOWCLK_DIV32)
- DCCLK
- CPCLK

If the clock source for SYSCLK is the EXTCLK or SWCLK/JTCK inputs, the frequency of the input clock must be controlled to limit the SYSCLK frequency to a valid frequency.

NOTE: The EXTCLK cannot be used as a source for SYSCLK with RF activity, even if its frequency value is 48 MHz. It can only be used for SYSCLK without RF activity.

SYSCLK can be output through one or more DIO pads using the DIO components. For more information about the DIO configuration, see Section 10.2, “Functional Configuration” on page 259.

IMPORTANT: When the RSL10 system is processing RF traffic that uses the Bluetooth low energy baseband, SYSCLK must be sourced appropriately to provide the necessary BBCLK frequencies. For more information about limitations on BBCLK, see Section 9.3.1, “Clock Structures” on page 206.

NOTE: For optimal system power performance when configuring the RSL10 system to execute from a SYSCLK frequency of 48 MHz, contact your ON Semiconductor Customer Service Representative. For other SYSCLK configurations, use the standard calibration configurations of the power supplies provided for the device, as described in the Manufacturing Records section of the *RSL10 Firmware Reference*.

6.3.2 Standby Clock (STANDBYCLK)

The RSL10 system includes a standby clock (STANDBYCLK) that is used as the source for the RTC (see Section 6.3.5, “Real-Time Clock (RTC)”), and can be used as the source for SYSCLK in standby operating modes.

The `ACS_RTC_CTRL_CLK_SRC_SEL` bit field from the `ACS_RTC_CTRL` register is used to configure the source for this clock. The sources for this clock can be the following:

- The standby RC oscillator
- The 32 kHz crystal oscillator
- DIO0, DIO1, DIO2, or DIO3

STANDBYCLK can be enabled or disabled by configuring the `ACS_RTC_CTRL_RTC_ENABLE` bit from the `ACS_RTC_CTRL` register.

CAUTION: Switching between STANDBYCLK sources is not guaranteed to be glitch-free. For this reason, changing the selected STANDBYCLK clock source should only be done when SYSCLK is sourced from another clock.

Use of the 32 kHz crystal oscillator is recommended over use of the standby RC oscillator due to:

- Improved clock accuracy
- Simplification of system designs
- Lower power consumption

Typically, the standby RC oscillator should only be used as STANDBYCLK in cases where the external 32 kHz crystal is not available.

Typically, STANDBYCLK is supplied with a 32 kHz clock source. If STANDBYCLK is supplied at a frequency other than 32 kHz, a correction factor can be applied to the timers that drive RF traffic to support input clock frequencies from the DIOs in the range from 25 to 100 kHz.

IMPORTANT: In typical configurations, the RSL10 Bluetooth stack is provided with a 32 kHz crystal oscillator source that provides 32768 Hz source with a variation of up to 500 ppm. If STANDBYCLK meets these operating assumptions, use the `LowPowerClock_Source_Set(0)` API function from the Bluetooth stack library to inform the Bluetooth stack.

If STANDBYCLK does not meet these operating conditions, the stack must be informed using the `LowPowerClock_Source_Set(1)` API function call. In these cases, the RSL10 **Bluetooth stack must also be provided with the actual frequency of STANDBYCLK by setting the RTC clock period using the `RTCCLK_Period_Value_Set()` API function from the Bluetooth stack library.**

- **If the clock source is stable with a variation of less than 500 ppm, the period can be set once during initialization and used throughout the execution of an application.**
- **For all other clock sources with a higher error rate or that might vary over time due to changes in environmental conditions, the application should periodically measure and update the RTC clock period. For more information on measuring the RTC clock period, see Section 13.3, "Audio Sink Clock Counters" on page 390.**

STANDBYCLK can be output through one or more DIO pads using the DIO components. For more information about the DIO configuration, see Section 10.2, "Functional Configuration" on page 259.

6.3.3 Slow Clock (SLOWCLK)

Slow clock (SLOWCLK) is a prescaled form of SYSCLK that is used as an intermediate divided clock for system components that need a lower maximum clock frequency. This clock is typically set to 1 MHz, and is used as the clock source for:

- CPCLK (see Section 6.3.7, "Power Supply Clocks")
- ADC (see Section 11.2.2, "ADC Sampling Configuration" on page 302)
- PWM (see Section 11.6, "Pulse Width Modulation (PWM)" on page 333)
- Further divided forms of SLOWCLK (SLOWCLK_DIV2, SLOWCLK_DIV32), which are used as the source for:
  - The general purpose timers, which select between SLOWCLK_DIV2 and SLOWCLK_DIV32 (see Section 12.3, "Timers" on page 372)
  - The watchdog timer, which uses SLOWCLK_DIV32 (see Section 12.4, "Watchdog Timer" on page 375)

SLOWCLK is derived from SYSCLK through a 6-bit integer division by the `CLK_DIV_CFG0_SLOWCLK_PRESCALE` bit field in the `CLK_DIV_CFG0` register. This prescaler provides a clock prescaled from SYSCLK by 1 to 64, and results in a SLOWCLK with a frequency defined by the following equation:

$$f_{\mathrm{SLOWCLK}} = \frac{f_{\mathrm{SYSCLK}}}{(\mathrm{CLK\_DIV\_CFG0\_SLOWCLK\_PRESCALE} + 1)}$$

SLOWCLK can be output through one or more DIO pads using the DIO components. For more information about the DIO configuration, see Section 10.2, "Functional Configuration" on page 259.

6.3.4 Baseband Clock (BBCLK) and Other Clocks for the Bluetooth Low Energy Baseband

The Bluetooth low energy baseband is clocked using three clocks:

1. Baseband clock (BBCLK)
2. Divided baseband clock (BBCLK_DIV)
3. Baseband timer clock

For more information about these clocks and timing in the Bluetooth low energy baseband, see Section 9.3.1, “Clock Structures” on page 206.

6.3.5 Real-Time Clock (RTC)

The real-time clock (RTC) is supported by the RTC timer, which is clocked from the configured STANDBYCLK. For information about configuring the clock source used by the RTC and RTC timer, see Section 6.3.2, “Standby Clock (STANDBYCLK)”.

The RTC timer is a 32-bit free-running countdown timer that counts down from the value specified in the `ACS_RTC_CFG_START_VALUE` bit-field of the `ACS_RTC_CFG` register. The current RTC counter value is available through the `ACS_RTC_COUNT` register, and the current count can be reset by setting the `ACS_RTC_CTRL_RESET` bit from the `ACS_RTC_CTRL` register. When the RTC timer reaches 0, the start value is loaded to the current count and the RTC timer continues.

The RTC timer triggers an RTC clock (`RTC_CLOCK_IRQ`) interrupt when a rising edge is detected on bit 14 of the RTC timer. For a typical STANDBYCLK configuration of 32,768 Hz, this produces an RTC clock interrupt at one-second intervals.

The RTC timer also triggers an RTC alarm (`RTC_ALARM_IRQ`) interrupt when the RTC timer encounters an alarm event, as configured using the `ACS_RTC_CTRL_ALARM_CFG` bit-field from the `ACS_RTC_CTRL` register. This bit-field specifies one of the following:

- The RTC alarm is disabled (configured to 0x0)
- The RTC alarm is triggered when a rising edge is detected on a specified bit between 7 and 20 of the RTC timer (configured from 0x1 to 0xE, respectively)
- The RTC alarm is triggered when the RTC timer reaches 0 and reloads the start value (configured to 0xF)

6.3.6 User Clock (USRCLK)

The user clock is an output clock that you can use as a clock source for the PCM interfaces or for any external components. This clock is not used internally by the RSL10 system, so its usage can depend entirely on the outside needs of the larger system containing RSL10.

USRCLK is derived from SYSCLK through a 12-bit integer division by the `CLK_DIV_CFG0_USRCLK_PRESCALE` bit field in the `CLK_DIV_CFG0` register. This prescaler provides a clock prescaled from SYSCLK by 1 to 4096, and results in a USRCLK with a frequency defined by the following equation:

$$f_{\mathrm{USRCLK}} = \frac{f_{\mathrm{SYSCLK}}}{(\mathrm{CLK\_DIV\_CFG0\_USRCLK\_PRESCALE} + 1)}$$

USRCLK can be output through one or more DIO pads using the DIO components. For more information about the DIO configuration, see Section 10.2, “Functional Configuration” on page 259.

6.3.7 Power Supply Clocks

The following power supply components each have their own clock sources:

- The VDDA charge pump is clocked using CPCLK, which is divided from SLOWCLK. Configuration and restrictions on this clock are described in Section 5.3.5, “Analog Supply Voltage (VDDA)” on page 49.
- The DC-DC buck converter is clocked using the DCCLK, which is divided from SYSCLK. Configuration and recommendations for this clock are described in Section 5.3.1, “System Supply Voltage (VCC)” on page 39.

6.3.8 Interface Clocks

The following interface components each have a clock divided from SYSCLK that is used to clock the interface:

DMIC and the Output Driver (AUDIOCLK)

The digital microphone inputs (Section 13.1, “Digital Microphone (DMIC) Inputs” on page 378) and output driver (Section 13.2, “Output Driver” on page 386) share a pair of divided clock sources.

The audio clock (AUDIOCLK) is divided from SYSCLK using the `CLK_DIV_CFG1_AUDIOCLK_PRESCALE` bit-field from the `CLK_DIV_CFG1` register. This prescaler provides a clock prescaled from SYSCLK by 1 to 64, and results in an AUDIOCLK with a frequency defined by the following equation:

$$f_{\mathrm{AUDIOCLK}} = \frac{f_{\mathrm{SYSCLK}}}{(\mathrm{CLK\_DIV\_CFG1\_AUDIOCLK\_PRESCALE} + 1)}$$

Audio slow clock is further divided from AUDIOCLK using the `CLK_DIV_CFG1_AUDIOSLOWCLK_PRESCALE` bit-field from the `CLK_DIV_CFG1` register. This prescaler provides a clock prescaled from AUDIOCLK by 1 to 4, and results in an AUDIOSLOWCLK with a frequency defined by the following equation:

$$f_{\mathrm{AUDIOSLOWCLK}} = \frac{f_{\mathrm{AUDIOCLK}}}{(\mathrm{CLK\_DIV\_CFG1\_AUDIOSLOWCLK\_PRESCALE} + 1)}$$

NOTE: For correct operation of the DMIC inputs and output driver in the same system, configuration of AUDIOCLK, AUDIOSLOWCLK, the DMIC input decimation filters, and the output driver interpolation filters must result in the same decimated sampling frequency for both the DMIC inputs and the output driver outputs.

Both AUDIOCLK and AUDIOSLOWCLK can be output through one or more DIO pads using the DIO components. For more information about the DIO configuration, see Section 10.2, “Functional Configuration” on page 259.

I^{2}C

The I^{2}C clock is only used by the I^{2}C interface when the interface is configured for master mode. For information about configuring the I^{2}C clock, see Section 11.4.2, “Master Mode Specific Configuration” on page 315.

PWM

For more information about this interface and its clocks, see Section 11.6, “Pulse Width Modulation (PWM)” on page 333.

SPI

For more information about these interfaces and their clocks, see Section 11.7, “Serial Peripheral Interfaces (SPI)” on page 335.

UART

The UART interface indirectly divides SYSCLK to achieve the baud rate for UART communications. For the UART TX output, this baud rate is applied directly; for the UART RX input, this baud rate is used in the asynchronous recovery of data from this pad. For more information about this interface and its clock, see Section 11.8, “Universal Asynchronous Receiver-Transmitter (UART) Interfaces” on page 344.

The ADCs use either SLOWCLK or SLOWCLK divided by a fixed divisor of 5 to sample analog signals. Information on the ADC interface clocking and sample configuration can be found in Section 11.2.2, “ADC Sampling Configuration” on page 302.

The PCM interface does not have its own divided clock, but is asynchronously clocked relative to the clock input provided at the PCM clock input pad. This clock source can be provided by the RSL10 system by routing a clock output to the same DIO that is acting as the PCM clock input (see Section 10.2, “Functional Configuration” on page 259 for more information about configuring a DIO as both a clock output and a PCM clock input; see Section 11.5, “Pulse Code Modulation (PCM) Interface” on page 324 for more information about the PCM interface).

6.3.9 Clock Distribution Registers

Register Name	Register Description	Address
`CLK_SYS_CFG`	System Clock Configuration Register	0x40000100
`CLK_DIV_CFG0`	Prescale register for SLOWCLK, BBCLK and USRCLK clocks	0x40000104
`CLK_DIV_CFG1`	Prescale register for PWM clock, UART and DMIC clocks	0x40000108
`CLK_DIV_CFG2`	Prescale register for DC-DC converter and charge pump clocks	0x4000010C
`ACS_RTC_CFG`	RTC Timer Counter Preload	0x40001330
`ACS_RTC_COUNT`	RTC Timer Counter Current Value (only reset by pmu reset or by writing 1 at ACS_RTC_CTRL.RESET)	0x40001334
`ACS_RTC_CTRL`	RTC Control Register	0x40001338

6.3.9.1 CLK_SYS_CFG

Bit Field	Field Name	Description
19:16	`JTCK_PRESCALE`	Prescale value for the input clock from pad JTCK (1 to 16 in steps of 1)
11:8	`EXTCLK_PRESCALE`	Prescale value for the input clock from pad EXTCLK (1 to 16 in steps of 1)
2:0	`SYSCLK_SRC_SEL`	Controls the source of the system clock: JTCK, RFCLK, RCCLK, EXTCLK or STANDBYCLK

Field Name	Value Symbol	Value Description	Hex Value
JTCK_PRESCALE	JTCK_PRESCALE_1	Divide by 1	0x0*
	JTCK_PRESCALE_2	Divide by 2	0x1
	JTCK_PRESCALE_15	Divide by 15	0xE
	JTCK_PRESCALE_16	Divide by 16	0xF
EXTCLK_PRESCALE	EXTCLK_PRESCALE_1	Divide by 1	0x0*
	EXTCLK_PRESCALE_2	Divide by 2	0x1
	EXTCLK_PRESCALE_15	Divide by 15	0xE
	EXTCLK_PRESCALE_16	Divide by 16	0xF
SYSCLK_SRC_SEL	SYSCLK_CLKSRC_RCCLK	Select the RCCLK clock as SYSCLK clock source	0x0*
	SYSCLK_CLKSRC_STANDBYCLK	Select the STANDBYCLK clock as SYSCLK clock source	0x1
	SYSCLK_CLKSRC_RFCLK	Select the RFCLK clock as SYSCLK clock source	0x2
	SYSCLK_CLKSRC_EXTCLK	Select the EXTCLK clock as SYSCLK clock source	0x3
	SYSCLK_CLKSRC_JTCK	Select the JTCK clock as SYSCLK clock source	0x4

6.3.9.2 CLK_DIV_CFG0

Bit Field	Field Name	Description
27:16	USRCLK_PRESCALE	Prescale value for the USR clock (1 to 4096 in steps of 1)
10:8	BBCLK_PRESCALE	Prescale value for the Baseband peripheral clock (1 to 8 in steps of 1)
5:0	SLOWCLK_PRESCALE	Prescale value for the SLOWCLK clock (1 to 64 in steps of 1)

Field Name	Value Symbol	Value Description	Hex Value
USRCLK_PRESCALE	USRCLK_PRESCALE_1	Divide by 1	0x0*
	USRCLK_PRESCALE_2	Divide by 2	0x1
	USRCLK_PRESCALE_3	Divide by 3	0x2
	USRCLK_PRESCALE_4095	Divide by 4095	0xFFE
	USRCLK_PRESCALE_4096	Divide by 4096	0xFFF
BBCLK_PRESCALE	BBCLK_PRESCALE_1	Divide by 1	0x0*
	BBCLK_PRESCALE_2	Divide by 2	0x1
	BBCLK_PRESCALE_3	Divide by 3	0x2
	BBCLK_PRESCALE_4	Divide by 4	0x3
	BBCLK_PRESCALE_5	Divide by 5	0x4
	BBCLK_PRESCALE_6	Divide by 6	0x5
	BBCLK_PRESCALE_7	Divide by 7	0x6
	BBCLK_PRESCALE_8	Divide by 8	0x7

Field Name	Value Symbol	Value Description	Hex Value
SLOWCLK_PRESCALE	SLOWCLK_PRESCALE_1	Divide by 1	0x0
	SLOWCLK_PRESCALE_2	Divide by 2	0x1
	SLOWCLK_PRESCALE_3	Divide by 3	0x2*
	SLOWCLK_PRESCALE_4	Divide by 4	0x3
	SLOWCLK_PRESCALE_8	Divide by 8	0x7
	SLOWCLK_PRESCALE_10	Divide by 10	0x9
	SLOWCLK_PRESCALE_12	Divide by 12	0xB
	SLOWCLK_PRESCALE_16	Divide by 16	0xF
	SLOWCLK_PRESCALE_24	Divide by 24	0x17
	SLOWCLK_PRESCALE_48	Divide by 48	0x2F
	SLOWCLK_PRESCALE_63	Divide by 63	0x3E
	SLOWCLK_PRESCALE_64	Divide by 64	0x3F

6.3.9.3 CLK_DIV_CFG1

Bit Field	Field Name	Description
31:30	AUDIOSLOWCLK_PRESCALE	Prescale value for the slow audio clock down from the fast audio clock (1 to 4 in steps of 1)
29:24	AUDIOCLK_PRESCALE	Prescale value for the fast audio clock (1 to 64 in steps of 1)
20:16	UARTCLK_PRESCALE	Prescale value for the UART peripheral clock (1 to 32 in steps of 1)
13:8	PWM1CLK_PRESCALE	Prescale value for the PWM1 peripheral clock (1 to 64 in steps of 1)
5:0	PWM0CLK_PRESCALE	Prescale value for the PWM0 peripheral clock (1 to 64 in steps of 1)

Field Name	Value Symbol	Value Description	Hex Value
AUDIOSLOWCLK_PRESCALE	AUDIOSLOWCLK_PRESCALE_1	Divide by 1	0x0*
	AUDIOSLOWCLK_PRESCALE_2	Divide by 2	0x1
	AUDIOSLOWCLK_PRESCALE_3	Divide by 3	0x2
	AUDIOSLOWCLK_PRESCALE_4	Divide by 4	0x3

Field Name	Value Symbol	Value Description	Hex Value
AUDIOCLK_PRESCALE	AUDIOCLK_PRESCALE_1	Divide by 1	0x0*
	AUDIOCLK_PRESCALE_2	Divide by 2	0x1
	AUDIOCLK_PRESCALE_3	Divide by 3	0x2
	AUDIOCLK_PRESCALE_4	Divide by 4	0x3
	AUDIOCLK_PRESCALE_5	Divide by 5	0x4
	AUDIOCLK_PRESCALE_6	Divide by 6	0x5
	AUDIOCLK_PRESCALE_7	Divide by 7	0x6
	AUDIOCLK_PRESCALE_8	Divide by 8	0x7
	AUDIOCLK_PRESCALE_9	Divide by 9	0x8
	AUDIOCLK_PRESCALE_10	Divide by 10	0x9
	AUDIOCLK_PRESCALE_11	Divide by 11	0xA
	AUDIOCLK_PRESCALE_12	Divide by 12	0xB
	AUDIOCLK_PRESCALE_13	Divide by 13	0xC
	AUDIOCLK_PRESCALE_14	Divide by 14	0xD
	AUDIOCLK_PRESCALE_15	Divide by 15	0xE
	AUDIOCLK_PRESCALE_16	Divide by 16	0xF
	AUDIOCLK_PRESCALE_63	Divide by 63	0x3E
	AUDIOCLK_PRESCALE_64	Divide by 64	0x3F
UARTCLK_PRESCALE	UARTCLK_PRESCALE_1	Divide by 1	0x0*
	UARTCLK_PRESCALE_2	Divide by 2	0x1
	UARTCLK_PRESCALE_31	Divide by 31	0x1E
	UARTCLK_PRESCALE_32	Divide by 32	0x1F
PWM1CLK_PRESCALE	PWM1CLK_PRESCALE_1	Divide by 1	0x0*
	PWM1CLK_PRESCALE_2	Divide by 2	0x1
	PWM1CLK_PRESCALE_63	Divide by 63	0x3E
	PWM1CLK_PRESCALE_64	Divide by 64	0x3F
PWM0CLK_PRESCALE	PWM0CLK_PRESCALE_1	Divide by 1	0x0*
	PWM0CLK_PRESCALE_2	Divide by 2	0x1
	PWM0CLK_PRESCALE_63	Divide by 63	0x3E
	PWM0CLK_PRESCALE_64	Divide by 64	0x3F

6.3.9.4 CLK_DIV_CFG2

Bit Field	Field Name	Description
15	CPCLK_DISABLE	Charge pump clock disable
13:8	CPCLK_PRESCALE	Prescale value for the charge pump clock from the SLOWCLK clock (1 to 64 in steps of 1)
7	DCCLK_DISABLE	DC-DC converter clock disable
5:0	DCCLK_PRESCALE	Prescale value for the DC-DC converter clock (1 to 64 in steps of 1)

Field Name	Value Symbol	Value Description	Hex Value
CPCLK_DISABLE	CPCLK_ENABLE	Charge pump clock enabled	0x0*
	CPCLK_DISABLE	Charge pump clock disabled	0x1
CPCLK_PRESCALE	CPCLK_PRESCALE_1	Divide by 1	0x0
	CPCLK_PRESCALE_2	Divide by 2	0x1
	CPCLK_PRESCALE_3	Divide by 3	0x2
	CPCLK_PRESCALE_4	Divide by 4	0x3
	CPCLK_PRESCALE_5	Divide by 5	0x4
	CPCLK_PRESCALE_6	Divide by 6	0x5
	CPCLK_PRESCALE_7	Divide by 7	0x6
	CPCLK_PRESCALE_8	Divide by 8	0x7*
	CPCLK_PRESCALE_9	Divide by 9	0x8
	CPCLK_PRESCALE_10	Divide by 10	0x9
	CPCLK_PRESCALE_63	Divide by 63	0x3E
	CPCLK_PRESCALE_64	Divide by 64	0x3F
DCCLK_DISABLE	DCCLK_ENABLE	DC-DC converter clock enabled	0x0*
	DCCLK_DISABLE	DC-DC converter clock disabled	0x1
DCCLK_PRESCALE	DCCLK_PRESCALE_1	Divide by 1	0x0*
	DCCLK_PRESCALE_2	Divide by 2	0x1
	DCCLK_PRESCALE_3	Divide by 3	0x2
	DCCLK_PRESCALE_4	Divide by 4	0x3
	DCCLK_PRESCALE_5	Divide by 5	0x4
	DCCLK_PRESCALE_6	Divide by 6	0x5
	DCCLK_PRESCALE_7	Divide by 7	0x6
	DCCLK_PRESCALE_8	Divide by 8	0x7
	DCCLK_PRESCALE_9	Divide by 9	0x8
	DCCLK_PRESCALE_10	Divide by 10	0x9
	DCCLK_PRESCALE_63	Divide by 63	0x3E
	DCCLK_PRESCALE_64	Divide by 64	0x3F

6.3.9.5 ACS_RTC_CFG

Bit Field	Field Name	Description
31:0	START_VALUE	Start value for the RTC timer counter (counts from start_value down to 0)

Field Name	Value Symbol	Value Description	Hex Value
START_VALUE	RTC_CNT_START_0	Divide by 1	0x0
	RTC_CNT_START_1	Divide by 2	0x1
	RTC_CNT_START_32767	Divide by 32768	0x7FFF*
	RTC_CNT_START_4294967295	Divide by 2^{32}	0xFFFFFFFF

6.3.9.6 ACS_RTC_COUNT

Bit Field	Field Name	Description
31:0	VALUE	RTC timer current value

6.3.9.7 ACS_RTC_CTRL

Bit Field	Field Name	Description
25	FORCE_CLOCK	Force a clock on RTC timer (Test Purpose)
24	RESET	Reset the RTC timer
7:4	ALARM_CFG	Configure RTC timer alarm
3:1	CLK_SRC_SEL	Select the RTC, standby and bb timer clock source
0	ENABLE	Enable counter and RTC interrupt

Field Name	Value Symbol	Value Description	Hex Value
FORCE_CLOCK	RTC_FORCE_CLOCK	Clock the RTC timer (has an effect only if the source clock is low)	0x1
RESET	RTC_RESET	The RTC timer is reset	0x1

Field Name	Value Symbol	Value Description	Hex Value
ALARM_CFG	RTC_ALARM_DISABLE	RTC alarm is disabled	0x0*
	RTC_ALARM_7P8125MS	RTC alarm on counter bit 7 rising edge (7.8125 ms)	0x1
	RTC_ALARM_15P625MS	RTC alarm on counter bit 8 rising edge (15.625 ms)	0x2
	RTC_ALARM_31P25MS	RTC alarm on counter bit 9 rising edge (31.25 ms)	0x3
	RTC_ALARM_62P5MS	RTC alarm on counter bit 10 rising edge (62.5 ms)	0x4
	RTC_ALARM_125MS	RTC alarm on counter bit 11 rising edge (125 ms)	0x5
	RTC_ALARM_250MS	RTC alarm on counter bit 12 rising edge (250 ms)	0x6
	RTC_ALARM_500MS	RTC alarm on counter bit 13 rising edge (500 ms)	0x7
	RTC_ALARM_1S	RTC alarm on counter bit 14 rising edge (1 s)	0x8
	RTC_ALARM_2S	RTC alarm on counter bit 15 rising edge (2 s)	0x9
	RTC_ALARM_4S	RTC alarm on counter bit 16 rising edge (4 s)	0xA
	RTC_ALARM_8S	RTC alarm on counter bit 17 rising edge (8 s)	0xB
	RTC_ALARM_16S	RTC alarm on counter bit 18 rising edge (16 s)	0xC
	RTC_ALARM_32S	RTC alarm on counter bit 19 rising edge (32 s)	0xD
	RTC_ALARM_64S	RTC alarm on counter bit 20 rising edge (64 s)	0xE
	RTC_ALARM_ZERO	RTC alarm on (down) counter reaching zero (up to 36.4 hours)	0xF
CLK_SRC_SEL	RTC_CLK_SRC_RC_OSC	Select the internal RC Oscillator clock	0x0*
	RTC_CLK_SRC_XTAL32K	Select the internal XTAL 32 kHz clock	0x1
	RTC_CLK_SRC_DIO0	Select DIO0 as a clock source	0x4
	RTC_CLK_SRC_DIO1	Select DIO1 as a clock source	0x5
	RTC_CLK_SRC_DIO2	Select DIO2 as a clock source	0x6
	RTC_CLK_SRC_DIO3	Select DIO3 as a clock source	0x7
ENABLE	RTC_DISABLE	The RTC is disabled	0x0*
	RTC_ENABLE	The RTC is enabled	0x1

6.4 Clock Detection

6.4.1 Clock Detector and System Monitor

The clock detector is used to monitor the clocks that are crucial to proper system execution. This circuit can be used to detect the presence of these key clock signals. If required, and if the clock is missing or not toggling, this block can be configured to reset the digital portions of the device, as described in Section 5.5, “Resets” on page 63.

The clock source monitored by the clock detector depends on the power mode (see Section 5.4, “Power Modes” on page 50), and the state of the system power supplies in that mode. The clock detector will indicate that a clock is present in the system whenever the monitored clock is at or above a minimum frequency of 4 kHz. This clock selection is automatically controlled by the underlying power-supply state machines of the RSL10 SoC, selecting the following clock sources for each mode:

System startup

RC oscillator (see Section 6.2.1, “RC Oscillator” on page 74)

System shutdown

No clock monitored as the system is already being reset or being held in a reset state pending recovery of a supplied voltage above the monitored minimum thresholds configured for proper system execution.

Run Mode

CPCLK (see Section 6.3.7, “Power Supply Clocks” on page 82)

Sleep or Standby Mode

RTC clock (see Section 6.3.5, “Real-Time Clock (RTC)” on page 81)

To enable resets using the clock detector, use the `ACS_CLK_DET_CTRL` register to:

1. Enable the clock detector by setting the `ACS_CLK_DET_CTRL_ENABLE` bit
2. Monitor the `ACS_CLK_DET_CTRL_CLOCK_PRESENT` bit, waiting for this flag to go high (indicating that the monitored clock is present)
3. Clear the `ACS_CLK_DET_CTRL_RESET_IGNORE` bit

To disable resets using the clock detector, or to disable the clock detector itself, use the `ACS_CLK_DET_CTRL` register to:

1. Set the `ACS_CLK_DET_CTRL_RESET_IGNORE` bit to prevent reset signals
2. Disable the clock detector by clearing the `ACS_CLK_DET_CTRL_ENABLE` bit

The `ACS_RESET_STATUS_CLK_DET_RESET_FLAG` bit from the `ACS_RESET_STATUS` register is used to indicate if the clock detector triggered a reset.

6.4.2 External Clock Detector

The external clock sources that can be used to supply SYSCLK can be monitored using the external clock detector. This clock detector selects between monitoring the external clock input (see Section 6.2.5, “External Clock Input (EXTCLK)” on page 75) and monitoring the JTCK clock from the SWJ-DP debug interface (see Section 6.2.6, “Debug Port Clock” on page 76), as configured using the `CLK_DET_CFG_CLK_DET_SEL` bit from the `CLK_DET_CFG` register.

This clock detector is disabled by default, and can be enabled by setting the `CLK_DET_CFG_CLK_DET_ENABLE` bit from the `CLK_DET_CFG` register.

When enabled, the external clock detector will monitor the specified clock using a divided form of SLOWCLK as configured using the `CLK_DET_CFG_CLK_DET_DIV` bit field from the `CLK_DET_CFG` register. The external clock detector will indicate that the monitored clock is present by setting the `CLK_DET_STATUS_CLK_DET_STATUS` bit from the `CLK_DET_STATUS` register if the frequency of the monitored clock is at least 54% of the monitoring clock source.

When enabled, the clock detector can be used to trigger the CLKDET_IRQ interrupt. Configuration of this interrupt uses the CLK_DET_CFG_CLK_DET_INT_SEL bit field from the CLK_DET_CFG register, which can be configured to cause an interrupt if:

- The monitored clock source becomes active
- The monitored clock source becomes inactive
- The state of the monitored clock source changes (becomes active or becomes inactive)

If an external clock detector interrupt has been triggered, the CLK_DET_INT_STATUS_CLK_DET_STATUS bit from the CLK_DET_STATUS register will be set until the CLK_DET_STATUS register is read.

6.4.3 Clock Detector Registers

Register Name	Register Description	Address
CLK_DET_CFG	External clock detector configuration register (including interrupt)	0x40000110
CLK_DET_STATUS	External clock detector status register	0x40000114
ACS_CLK_DET_CTRL	Clock Detector configuration register	0x4000132C

6.4.3.1 CLK_DET_CFG

Bit Field	Field Name	Description
5	CLK_DET_SEL	Clock detector source selection
4:3	CLK_DET_INT_SEL	Clock detector interrupt configuration
2:1	CLK_DET_DIV	Clock detector configuration - Not used when running on standby clock
0	CLK_DET_ENABLE	Clock detector enable/disable

Field Name	Value Symbol	Value Description	Hex Value
CLK_DET_SEL	CLK_DET_SEL_EXTCLK	Select EXTCLK source	0x0*
	CLK_DET_SEL_JTCK	Select JTCK source	0x1
CLK_DET_INT_SEL	CLK_DET_INT_DISABLE	Clock detector interrupt disabled	0x0*
	CLK_DET_INT_ACTIVATED	If the clock source becomes active an interrupt is created	0x1
	CLK_DET_INT_DEACTIVATED	If the clock source becomes inactive an interrupt is created	0x2
	CLK_DET_INT_ACTIVITY_CHANGE	Any the clock source activity change will create an interrupt	0x3
CLK_DET_DIV	CLK_DET_SLOWCLK_DIV32	EXTCLK or JTCK detector runs on SLOWCLK divided by 32	0x0*
	CLK_DET_SLOWCLK_DIV64	EXTCLK or JTCK detector runs on SLOWCLK divided by 64	0x1
	CLK_DET_SLOWCLK_DIV96	EXTCLK or JTCK detector runs on SLOWCLK divided by 96	0x2
	CLK_DET_SLOWCLK_DIV128	EXTCLK or JTCK detector runs on SLOWCLK divided by 128	0x3

Field Name	Value Symbol	Value Description	Hex Value
CLK_DET_ENABLE	CLK_DET_DISABLE	No clock detected	0x0*
	CLK_DET_ENABLE	Clock detected	0x1

6.4.3.2 CLK_DET_STATUS

Bit Field	Field Name	Description
1	CLK_DET_INT_STATUS	Clock detector interrupt status (cleared when read)
0	CLK_DET_STATUS	Clock detector status

Field Name	Value Symbol	Value Description	Hex Value
CLK_DET_INT_STATUS	CLK_DET_INT_FALSE	Clock detector interrupt not triggered	0x0*
	CLK_DET_INT_TRUE	Clock detector interrupt triggered	0x1
CLK_DET_STATUS	CLK_NOT_ACTIVE	No clock detected	0x0*
	CLK_ACTIVE	Clock detected	0x1

6.4.3.3 ACS_CLK_DET_CTRL

Bit Field	Field Name	Description
8	CLOCK_PRESENT	Clock present flag
1	RESET_IGNORE	Clock detector reset condition ignore
0	ENABLE	Clock detector enable

Field Name	Value Symbol	Value Description	Hex Value
CLOCK_PRESENT	ACS_CLK_DET_NO_CLOCK	No clock detected	0x0
	ACS_CLK_DET_CLOCK_PRESENT	Clock detected	0x1*
RESET_IGNORE	ACS_CLK_DET_RESET_DISABLE	Clock detector reset condition ignore	0x1
	ACS_CLK_DET_RESET_ENABLE	Clock detector reset condition accept	0x0*
ENABLE	ACS_CLK_DET_ENABLE	Clock detector enable	0x1*
	ACS_CLK_DET_DISABLE	Clock detector disable	0x0

CHAPTER 7

Memory

7.1 MEMORY ARCHITECTURE

The RSL10 system uses a memory architecture based on the predefined memory map of the Arm Cortex-M3 processor — a state-of-the-art 32-bit core with embedded multiplier and ALU for handling typical control functions as well as dedicated LPDSP32 functions.

The implementation of the memory architecture uses a number of single-port memories and memory-mapped registers interconnected with memory buses and support elements. All memories are accessible through the Arm Cortex-M3 processor, although some interfaces and peripherals provide additional access paths to specific memory elements. The connections to the components that make up the memory for the RSL10 system are shown in Figure 7.

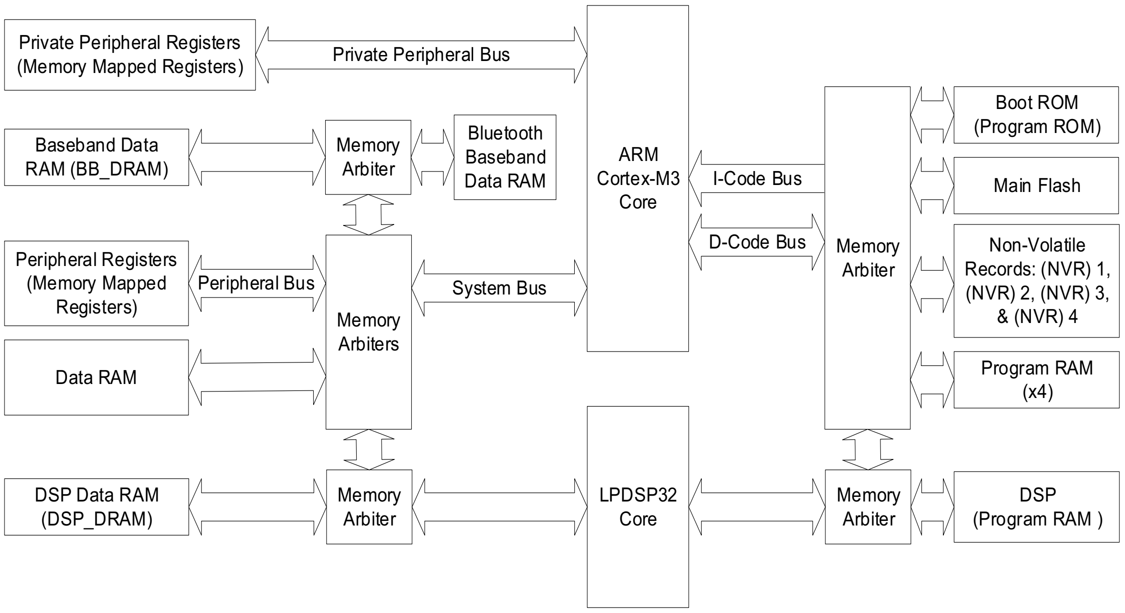


Figure 7. System Memory Architecture

7.1.1 Memory Instances

The memory architecture for RSL10, including the memory instances, registers, and other components, are accessible from the Arm Cortex-M3 processor through one or more of the processor's standard buses. All the memory instances are shown in Table 7.

Table 7. RSL10 Memory Instances

Instance Name	Size (bytes)	Type	Start Address
Boot ROM (Program ROM)	4096	ROM	0x00000000
Main Flash	393216	Flash	0x00100000
Non-Volatile Record (NVR) 1	2048	Flash	0x00080000
Non-Volatile Record (NVR) 2	2048	Flash	0x00080800

Table 7. RSL10 Memory Instances (Continued)

Instance Name	Size (bytes)	Type	Start Address
Non-Volatile Record (NVR) 3	2048	Flash	0x00081000
Non-Volatile Record (NVR) 4 (Manufacturing Test)	1024	Flash	0x00081800
Program RAM (x4)	4 x 8192	RAM	0x00200000
DSP (Program RAM)	4 x 10240	RAM	0x00220000
Data RAM	3 x 8192	RAM	0x20000000
LPDSP32 / Arm Cortex-M3 processor Shared Data RAM	6 x 8192	RAM	0x20006000
Bluetooth Baseband Data RAM	2 x 8192	RAM	0x20012000

7.1.2 Memory Buses

Buses connected to the Arm Cortex-M3 processor implement the standard Arm Cortex-M3 core memory map. These buses, which can also be seen in Figure 7 on page 93, are as follows:

I-Code Bus (I-Bus)

Allows the Arm Cortex-M3 processor to fetch instruction information from the ROM, flash, PRAM, and DSP_PRAM memory instances.

D-Code Bus (D-Bus)

Allows the Arm Cortex-M3 processor to fetch data information from the ROM, flash, PRAM, and DSP_PRAM memory instances.

System Bus (S-Bus)

Allows the Arm Cortex-M3 processor to fetch instructions and data information from the DRAM, DSP_DRAM, and BB_DRAM memory instances. This bus also provides access to the peripheral bus.

Peripheral Bus (P-Bus)

Allows the Arm Cortex-M3 processor to access memory-mapped peripherals and external memory instances.

Private Peripheral Bus

Allows the Arm Cortex-M3 processor to access standard memory-mapped peripherals. This includes the NVIC, SysTick timer and the Arm Cortex-M3 processor debug port controller.

7.1.3 Memory Arbitration

In front of each memory instance, an arbiter manages the simultaneous accesses between the masters - the Arm Cortex-M3 processor, the LPDSP32 or the baseband controller (BB), and the DMA.

The arbitration scheme is configurable per memory instance, or per memory instance group, in the `SYSCTRL_MEM_ARBITER_CFG` register.

The following arbitration schemes are available:

Fixed priority mode:

Arm Cortex-M3 processor > LPDSP32/BB > DMA

Fixed priority mode:

LPDSP32/BB > Arm Cortex-M3 processor > DMA

Round-robin mode:

Depending on the `ROUND_ROBIN_TOKEN` configuration bit, the following arbitration scheme is used:

- *Real-time DMA mode:*

  In this mode the priority order is normally Arm Cortex-M3 processor > LPDSP32/BB > DMA, but if a real-time DMA channel memory access has been blocked for 7 consecutive cycles, then the priority order becomes DMA > Arm Cortex-M3 processor > LPDSP32/BB. A real-time DMA channel is defined as a DMA channel with the MSB of its priority level equal to 1.

- *Continuous round-robin mode:*

  In this mode, the priorities are rotated between the following three-memory priorities during each SYSCLK cycle:

  - Arm Cortex-M3 processor > LPDSP32/BB > DMA
  - LPDSP32/BB > Arm Cortex-M3 processor > DMA
  - DMA > Arm Cortex-M3 processor > LPDSP32/BB

Smart mode:

This mode is only available for BB memories. In this mode, the priority order is normally Arm Cortex-M3 processor > DMA > BB, but if the baseband divided clock is active, then the priority order becomes BB > Arm Cortex-M3 processor > DMA.

The arbitration between the flash memory copier and DMA depends on the **`ROUND_ROBIN_TOKEN`** configuration bit, but in either mode, the DMA only has priority over the flash memory copier when the DMA has the (round-robin) priority token.

IMPORTANT: We recommend that the memories used by the baseband controller not be configured in the fixed Arm Cortex-M3 processor priority mode or the round-robin mode, as in these modes the functionality of the baseband controller cannot be guaranteed.

7.2 Memory Map and Usage

7.2.1 Arm Cortex-M3 Processor Memory Usage

The memory provided on the RSL10 SoC is divided into five distinct areas with distinct uses. These areas are mapped into the RSL10 memory map, as shown in Figure 8 on page 96.

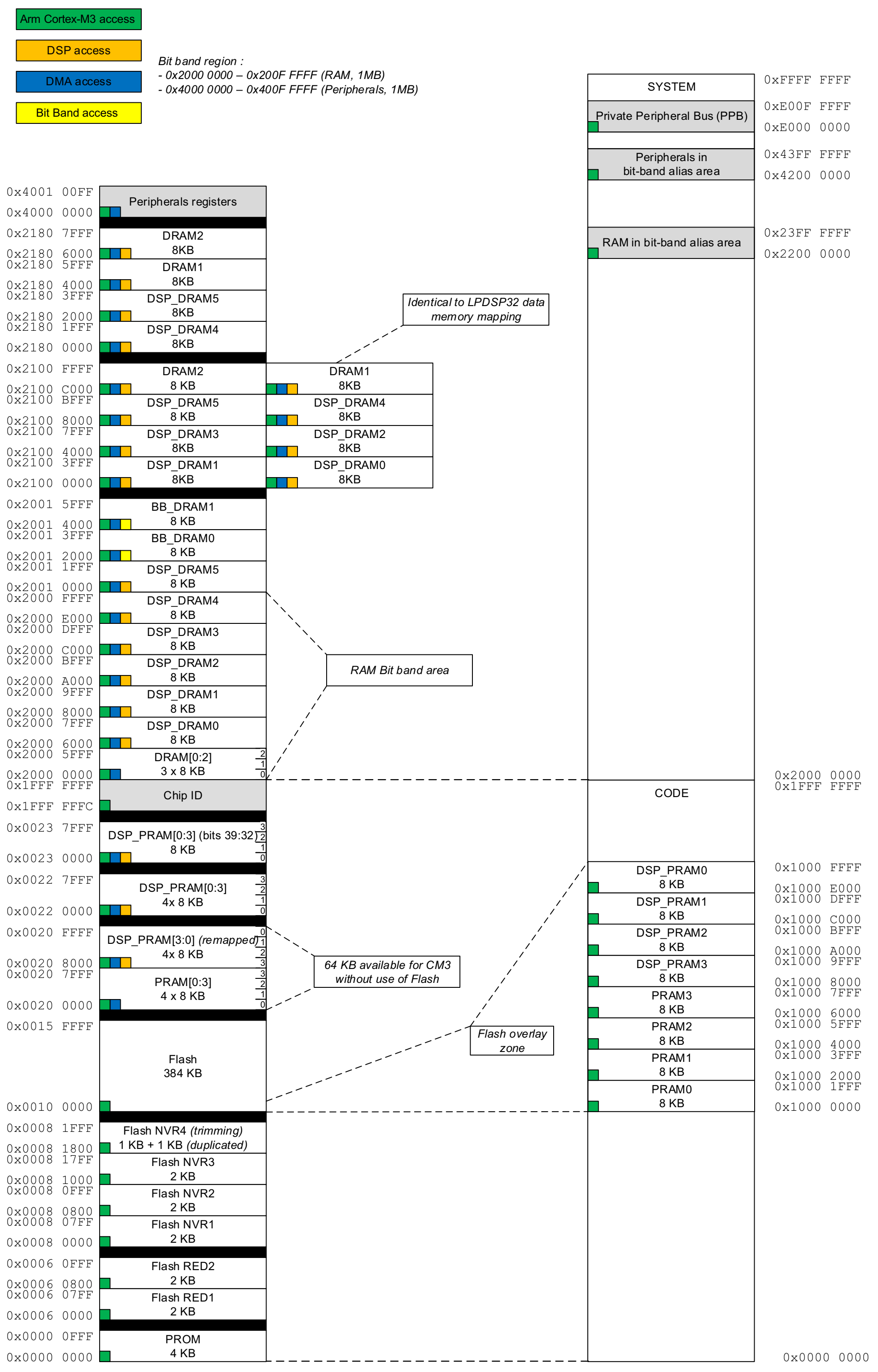


Figure 8. RSL10 Memory Map

The address ranges are in hexadecimal format. The following subsections provide a brief description and usage of each area. These areas are as follows:

1. Program ROM, as described in the Program ROM Chapter of the *RSL10 Firmware Reference*
2. Flash Memory
3. Program RAM
4. Data RAM
5. Shared RAM Instances
6. Other Memory Mapped Areas

These sections are mapped into the RSL10 memory map, as shown in Figure 8 on page 96. The address ranges are in hexadecimal format. The following subsections provide a brief description and usage of each.

7.2.2 Flash Memory

The Arm Cortex-M3 processor address space includes 384 KB of flash memory as non-volatile memory. It can only be written through the flash memory interface. The flash memory is used for storing:

- The user application and data
- The Bluetooth stack software
- The Bluetooth profiles
- The LPDSP32 application

The flash memory is organized into sectors, each containing 2048 bytes of flash memory.

7.2.2.1 Non-Volatile Record (NVR) Sectors

The non-volatile record (NVR) sections of flash that are used to hold system information include the following:

- Application specific information (NVR1)
- Address and key information for bonded devices (NVR2)
- Device configuration information (NVR3):
    - The local device's Bluetooth address information
    - IP protection configuration
    - An initialization function that is verified, and if valid, called by the ROM to initialize the system by loading calibrated settings to their desired registers, or to perform an alternate boot routine.

- Manufacturing information (NVR4):
    - Calibration settings for power supplies and clocks
    - The delays needed to write to the local flash instance
    - Manufacturing and test information

For more information about the use of these sectors, refer to the Hardware Definitions Chapter of the *RSL10 Firmware Reference*.

NVR1, NVR2 and NVR3 each consist of one 2048-byte sector of flash memory. NVR4 consists of four 256-byte redundant pages gathered into one sector of flash memory.

When the NVR4 is read, the `RECALL` bit in `FLASH_IF_CTRL` must be set. While this bit is set, accessing the main flash or redundancy sectors is not possible. NVR4 is only programmed during the production test and cannot be written by users. When reading from NVR4, bit 8 of FLASH_ADDR is not used in the address decoding, as both pages for this bit are expected to be identical.

NOTE: The system library provides the `Sys_ReadNVR4()` function to support reading the NVR4 sector. See the *RSL10 Firmware Reference* manual for more information.

IMPORTANT: NVR 1-2-3 sectors are only guaranteed for 1,000 program and erase cycles versus 100,000 programming cycles for all other memory sectors.

7.2.2.2 Redundancy Sectors

The flash memory contains four redundancy sectors; two redundancy sectors to improve yield, and two redundancy sectors that can be used by customers to patch (replace) sectors that have become damaged (either through overuse or through violation of the flash timing parameters) or are otherwise identified as defective sectors during the lifetime of the product. These redundancy sectors are used to patch any sector in main flash, NVR1, and NVR2. NVR3 cannot be patched.

The addresses of the sectors that the redundancy sectors are replacing can be read through the `FLASH_PATCH_ADDR[3:0]` registers. These registers contain the first address of the defective sector that will be patched. The `FLASH_PATCH_ADDR` registers are loaded from the `MANU_FLASH_RR*` locations in NVR4 using the flash memory's `CMD_LOAD_TRIM` operation (see Table 9 on page 105), which is typically executed during boot as part of the `CMD_WAKE_UP` operation.

NOTE: NVR4 cannot be patched, since it contains the patch configuration. NVR3 cannot be patched, as the manufacturing initialization function (`MANU_INFO_INIT`) implementation enables recall mode to read from NVR4 rather than using the `Sys_ReadNVR4()` function, as described in the *RSL10 Firmware Reference*.

The two customer-configurable redundancy sectors are the last two sectors of the main flash memory area. Use of these sectors is configured using the `CMD_WRITE_USER_RED*` commands to set to the NVR4 `MANU_FLASH_RR*` configurations that are loaded to the `FLASH_PATCH_ADDR` registers.

CAUTION: If a redundancy sector is used, it should not be used as a regular sector in the main flash area as this configuration results in this sector being used for two addressable locations in the flash memory area. To ensure that the customer-configurable redundancy sectors are not overwritten, the flashloader does not support writing to either of the potentially reassigned customer-configurable redundancy sectors.

Table 8. User Redundancy Sectors

Sector	Address	Configuration Register	NVR4 Configuration Address	Low-level Configuration Command
USER_RED1	0x0015F800	FLASH_PATCH_ADDR[2]	MANU_FLASH_RR2	CMD_WRITE_USER_RED1
USER_RED2	0x0015F000	FLASH_PATCH_ADDR[3]	MANU_FLASH_RR3	CMD_WRITE_USER_RED2

NOTE: After production testing, the `MANU_FLASH_RR*` locations in NVR4 that define the user redundancy sectors remain erased. This means a user patch address can be stored in NVR4 without erasing the sector.

CAUTION: The NVR4 `MANU_FLASH_RR*` configurations can only be written once using the `CMD_WRITE_USER_RED*` commands. If either of the MANU_FLASH_RR* configurations is overwritten with a second target, the redundancy sector configuration stored to that MANU_FLASH_RR* is likely to fail its ECC check (causing the redundancy sector to not be used) or to select an unexpected sector to be patched.

In case multiple patch configuration registers contain the same address, the following priority is observed (from highest to lowest): **`USER_RED2`**, **`USER_RED1`**, **`RED2`**, **`RED1`**. This allows a user application to patch a sector that has already been patched for yield regions in the same way as other sectors would be patched.

7.2.2.3 Error-Correction Coding

In order to prevent possible issues inherent in the flash memory technology, flash memory is organized into 64-bit double words that are protected by an Error Correcting Code (ECC) that ensures the integrity of the flash memory content as follows:

1. When writing to the flash memory, the ECC bits are automatically generated by the flash memory interface and appended to the data.
2. When reading from the flash memory, the error detection and/or correction is applied automatically.

The algorithm relies on the (72, 64) extended Hamming code, where 64 data bits from a pair of words of flash memory are extended by seven parity bits plus one overall parity bit, to form a 72-bit double word. This is a Single Error Correcting, Double Error Detecting (SECDED) code, and allows correcting a single bit error or detecting two-bit errors. The flash memory ECC generation, error detection, and error correction are performed by a dedicated hardware block with no incurred latency on flash read/write operations.

IMPORTANT: Since the flash memory is organized into 64-bit double words, flash memory is written two words at a time. Writes of individual words (or an odd number of words) are not possible, without corrupting the ECC bits, and as such, are not supported by the flash write library described in the *RSL10 Firmware Reference*.

While we recommend always keeping the error correcting code enabled, it is possible to disable the ECC when reading (or writing) from the flash, by clearing the appropriate bits in the `FLASH_ECC_CTRL` register:

- `The FLASH_ECC_CTRL_IDBUS_ECC_CTRL` bit disables ECC for accesses over the memory buses.
- The `FLASH_ECC_CTRL_CMD_ECC_CTRL` bit disables ECC for accesses using the flash memory's command interface.
- The `FLASH_ECC_CTRL_COPY_ECC_CTRL` bit disables ECC for accesses using the flash copier.

If the error correcting code detects an uncorrectable error, a `FLASH_ECC_IRQn` interrupt will be triggered. This interrupt can also be configured to trigger after correcting a certain number of detected correctable errors. This interrupt condition is configured by writing the number of corrected bit errors that should be allowed to the `FLASH_ECC_CTRL_ECC_COR_CNT_INT_THRESHOLD` bit-field in the `FLASH_ECC_CTRL` register.

7.2.2.4 Flash Delay Timings

All read, erase, and program cycles that access flash memory require adherence to the timing requirements of the flash memory instance. To set the flash delay timing for a given system clock frequency, set the

`FLASH_DELAY_CTRL_SYSCLK_FREQ` bit-field from the `FLASH_DELAY_CTRL` register to match your system clock frequency.

- Read operations are limited to minimum timing delays. To guarantee timing of all flash reads, set the `FLASH_DELAY_CTRL_SYSCLK_FREQ` bit-field to indicate a value greater than or equal to the `SYSCLK` frequency.

  If `SYSCLK` is known precisely, then the `FLASH_DELAY_CTRL_READ_MARGIN` bit-field from the `FLASH_DELAY_CTRL` register can be set to support fast read margins using the `FAST_READ_MARGIN` bit setting.

- Erase and program operations have both minimum and maximum timing delay limitations. Due to these limitations, the `FLASH_DELAY_CTRL_SYSCLK_FREQ` bit-field must be set to within ±10% of the actual SYSCLK frequency.

CAUTION: Erasing and programming flash memory is not allowed when using the RC oscillator with the multiplier enabled as the source for SYSCLK, because the variation of this clock source over temperature (specified at ±25% maximum) is more than the maximum allowed flash erase and program timing variation of ±10% .

The firmware directly supports setting the required delays when updating the `SYSCLK` configuration, with the flash delay registers reset whenever the `SystemCoreClock` global variable is updated and the `SystemCoreClockUpdate()` function from the CMSIS library is executed (for more information see the *RSL10 Firmware Reference).*

IMPORTANT: A minimum SYSCLK frequency of 1 MHz is required to safely complete a flash memory operation.

CAUTION: When the command CMD_PROGRAM_SEQ is in execution, any access through the I/D buses generates a bus fault.

7.2.3 RAM

7.2.3.1 Program RAM

PRAM0 to PRAM3 memory is used to store program code or data needed for user applications. 32 KB of the Program RAM is distributed into 4 instances of 2048x32 bits, and acts as flash memory overlay for executing the Bluetooth low energy software stack and the user application. These memory instances can serve a purpose similar to that of a program cache, where commonly used functions or routines are placed. The memories are primarily used for mirroring the real-time functions of the Bluetooth stack and the application stored in the flash memory. The advantage of using PRAM rather than flash memory is that it minimizes the number of flash memory accesses in order to lower the overall power consumption.

The DSP PRAM consists of 4 instances of 2048 x 40 bits (DSP_PRAM0 - DSP_PRAM3) that can be independently accessed by the Arm Cortex-M3 processor or the LPDSP32 DSP system via dynamic arbitration.

- When used as program memory or flash memory overlay by the Arm Cortex-M3 processor, the DSP PRAM is seen as 32-bit words, and appears in reversed order on the Arm Cortex-M3 processor memory map. Since the memory is 40-bit native, the upper byte in each word is not used. The main purpose of the DSP PRAM is to mirror the frequently used functions of the software stack and the application stored in the flash memory. The objective is to minimize the number of flash memory accesses in order to lower the overall power consumption.

- When used by the LPDSP32, the DSP PRAM is seen as 40-bit data, and appears in normal order on the Arm Cortex-M3 processor memory map, as viewed by the LPDSP32. The bits 39:32 of each word are mapped as bits 7:0 at a different address. When this different address is read, bits 31:8 return zero. This address is mainly used for storing or observing the LPDSP32 program code.

The PRAM0 to PRAM3 and DSP_PRAM0 to DSP_PRAM3 (when not used by the LPDSP32) can be made to operate in default or overlay mode, by configuring the corresponding bit of the `SYSCTRL_FLASH_OVERLAY_CFG` register. When the bit is cleared, the memories are only mapped to the default addresses range given in Figure 8. When this bit is set, the memories are also mapped to the flash memory read-only addressing range as follows:

- PRAM0 overlays addresses [0x100000; 0x101FFF].
- PRAM1 overlays addresses [0x102000; 0x103FFF].
- PRAM2 overlays addresses [0x104000; 0x105FFF].
- PRAM3 overlays addresses [0x106000; 0x107FFF].
- DSP_PRAM3 overlays addresses [0x108000; 0x109FFF].
- DSP_PRAM2 overlays addresses [0x10A000; 0x10BFFF].
- DSP_PRAM1 overlays addresses [0x10C000; 0x10DFFF].
- DSP_PRAM0 overlays addresses [0x10E000; 0x10FFFF].

7.2.3.2 Data RAM

The 32-bit data memory is distributed into DRAM, DSP DRAM and BB DRAM memory instances.

- 24 KB of DRAM are shared between the Bluetooth stack and the user application. The DRAM is subdivided into 3 instances of 2048 words (DRAM0 - DRAM2) that are used to store any type of data needed for user applications.
- 48 KB of DSP DRAM are shared between the CSS and the DSS. The DSP DRAM is subdivided into 6 instances of 2048 words (DSP_DRAM0 - DSP_DRAM5) that are independently attributed to the CSS or the DSS via dynamic arbitration.
- 16 KB of BB DRAM act as the exchange memory between the Arm Cortex-M3 processor and the baseband controller. The BB DRAM is subdivided into 2 instances of 2048 words (BB_DRAM0 and BB_DRAM1) that are directly accessible by the Arm Cortex-M3 processor and the DMA, parallel to the baseband controller. A configurable arbiter manages the simultaneous accesses between the Arm Cortex-M3 processor, the DMA, and the baseband controller. The arbiter can be configured through the `SYSCTRL_MEM_ARBITER_CFG` register.

7.2.3.3 Shared RAM Instances

- 24 KB of DRAM are shared between the Bluetooth stack software and the user application. The DRAM is subdivided into 3 instances of 2048 words (DRAM0 - DRAM2).
- 48 KB of DSP DRAM are shared between the CSS and the DSS. The DSP DRAM is subdivided into 6 instances of 2048 words (DSP_DRAM0 - DSP_DRAM5) that are independently attributed to the CSS or the DSS via dynamic arbitration.

7.2.4 Other Memory Mapped Areas

7.2.4.1 Peripherals and Interfaces

Memory-mapped registers on the peripheral bus are addressed between 0x4000 0000 and 0x400F FFFF (1 MB). This region contains the registers that are used to control various peripherals, interfaces, and other system components. This entire region also supports bit-band memory accesses.

7.2.4.2 Private Peripherals

Memory-mapped registers on the private peripheral bus are addressed between 0xE000 0000 and 0xE00F FFFF. This region contains registers related to the Arm Cortex-M3 processor.

7.2.5 LPDSP32 DSP Memory Usage

7.2.5.1 Program Memory

The LPDSP32’s program memory space uses 24-bit addressing relative to 20-bit words. Two 20-bit words form a 40-bit memory word in a big endian manner. 40 KB of DSP PRAM are shared with the DSS. This DSP PRAM is subdivided into 4 instances of 2048 x 40 bits (DSP_PRAM0 - DSP_PRAM3) that are independently accessed by the CSS or the DSS via dynamic arbitration.

- When used as program memory or flash memory overlay by the Arm Cortex-M3 processor, the DSP PRAM is seen as 32-bit words and appears in reversed order on the Arm Cortex-M3 processor memory map. Since the memory is 40-bit native, the upper byte of each word is not used. The main purpose of the DSP PRAM is to mirror the frequently used functions of the software stack and the application stored in the flash memory. The objective is to minimize the number of flash memory accesses in order to lower the overall power consumption.
- When used by the LPDSP32, the DSP PRAM is seen as 40-bit data, and appears in normal order on the Arm Cortex-M3 processor memory map as viewed by the LPDSP32. The bits 39:32 of each word are mapped as bits 7:0 at a different address. When this different address is read, bits 31:8 return zero. It is mainly used for storing or observing the LPDSP32 program code.

The LPDSP32 has the following program memory mapping (refer to Figure 9 on page 102):

LPDSP32-PM: 0x000000-0x003FFF (40 KB)

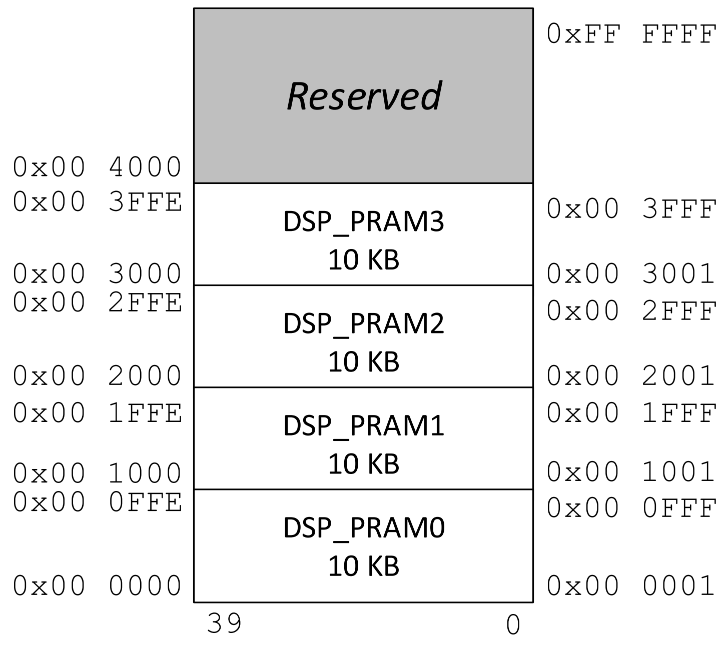


Figure 9. LPDSP32 Program Memory Mapping

7.2.5.2 Data Memory

The LPDSP32's data memory space uses 24-bit addressing relative to 8-bit words. Four 8-bit words form a 32-bit memory word in a little endian manner. 48 KB of DSP DRAM are shared between the CSS and the DSS. The DSP DRAM is subdivided into 6 instances of 2048 words (DSP_DRAM0 - DSP_DRAM5) that are independently attributed to the CSS or the DSS via dynamic arbitration.

The LPDSP32 has 3 usable data memory mappings (refer to Figure 10 on page 104), which are accessible through the LPDSP32 data memory A (DMA) and data memory B (DMB) buses:

1. LPDSP32-DMA: 0x000000-0x00FFFF (64 KB), LPDSP32-DMB: N/A
2. LPDSP32-DMA: 0x000000-0x00BFFF (48 KB), LPDSP32-DMB: 0x804000-0x807FFF (16 KB)
3. LPDSP32-DMA: 0x000000-0x007FFF (32 KB), LPDSP32-DMB: 0x800000-0x807FFF (32 KB)

IMPORTANT: In case simultaneous accesses of LPDSP32-DMA and LPDSP32-DMB are performed on the same memory instance, then the LPDSP32-DMA access takes priority.

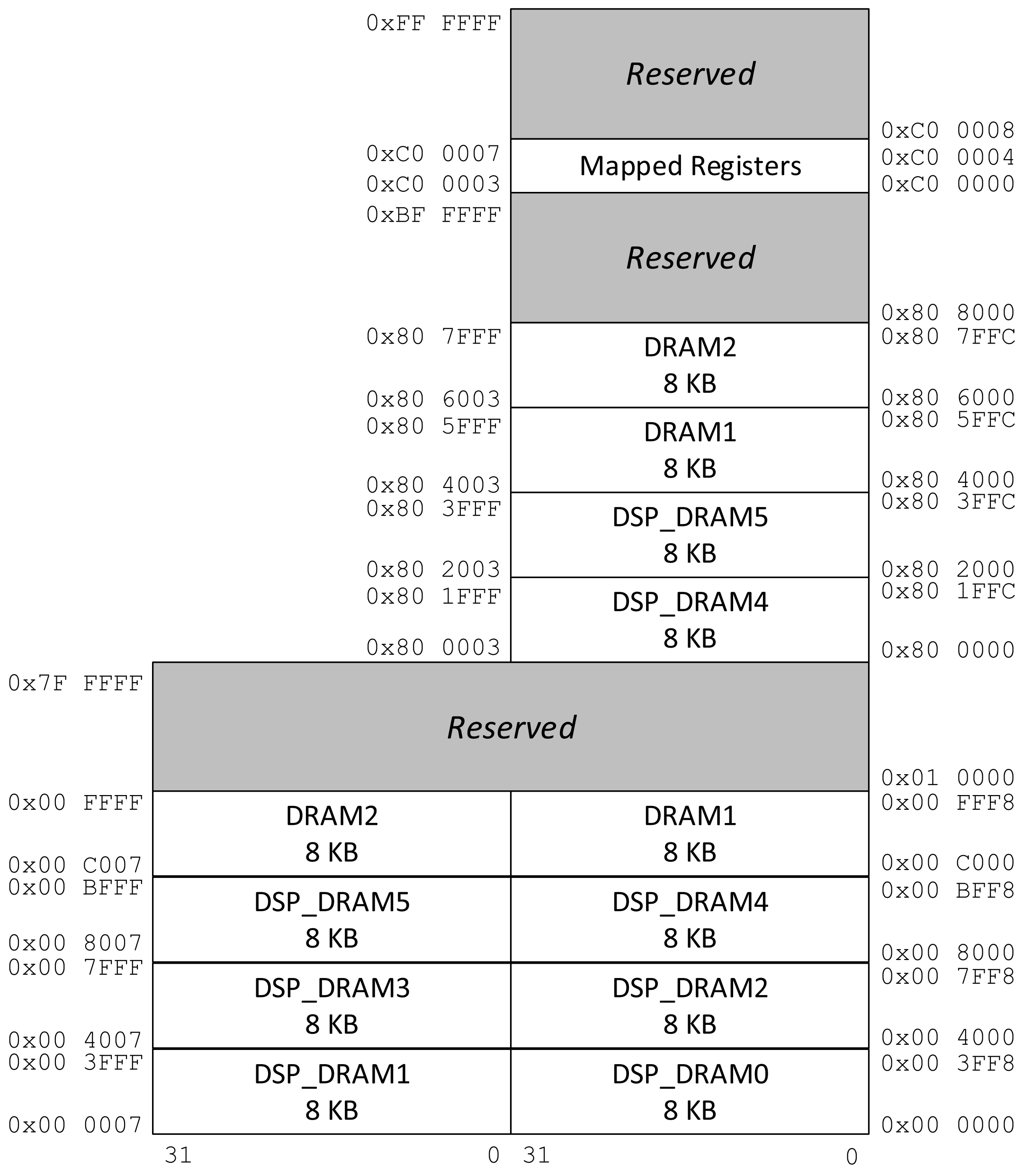


Figure 10. LPDSP32 Data Memory Mapping

NOTE: The different data memory mappings do not require any hardware configuration. The mapping that is used must simply be respected by the software that is running on the LPDSP32.

IMPORTANT: The last 16 KB of data memory (DRAM1 and DRAM2) is not used by the LPDSP32 in a normal application. This memory has been mapped to the LPDSP32's data memory space so that it can test these memories during production testing.

7.2.6 Bluetooth Low Energy Baseband (BB) Memory Usage

7.2.6.1 Exchange Memory

16 KB of BB DRAM act as the exchange memory between the Arm Cortex-M3 processor and the baseband controller. The BB DRAM is subdivided into 2 instances of 2048 words (`BB_DRAM0` and `BB_DRAM1`) that are directly accessible by the Arm Cortex-M3 processor and the DMA, parallel to the baseband controller. A configurable arbiter manages the simultaneous accesses between the Arm Cortex-M3 processor, the DMA, and the baseband controller. The arbiter can be configured through the `SYSCTRL_MEM_ARBITER_CFG` register.

7.3 Flash Memory Operations

7.3.1 Reading and Writing Flash Memory

Writing to flash memory consists of an erase cycle, followed by a program cycle. Following the erase cycle, the erased cells have a value of all ones. Programming the flash cells clears some of these cells to zero.

IMPORTANT: The flash write library contains functions that the user application can use to perform writes. A copy of this flash write library is provided in the RSL10 **program ROM, and can be used to write to the flash at any time. Refer to the *RSL10 Firmware Reference* for more information.**

NOTE: While writing to, or erasing, any flash memory, no flash memory instance can be accessed from the I-Code or D-Code buses. This includes any case where a low-level flash command is issued or when a low-level flash command is triggered by writing to the `FLASH_IF_CTRL` register.

Low-level commands, which are written with the `FLASH_CMD_CTRL` register, are used to write to the flash; see Table 9. Each command returns to idle state except the `CMD_PROGRAM_SEQ`. If the command is busy, the P-Bus write access is ignored, except in these circumstances:

- Writing to the `FLASH_DATA[0:1]` registers during the `CMD_PROGRAM_SEQ`, when new data is requested~~.~~
- Writing to the four `FLASH_COPY` (except `FLASH_COPY_CTRL`) registers~~.~~
- Writing to fields `FLASH_ECC_CTRL_CMD_ECC_CTRL`, `FLASH_ECC_CTRL_COPIER_ECC_CTRL` and `FLASH_ECC_CTRL_IDBUS_ECC_CTRL` of the `FLASH_ECC_CTRL` register~~.~~
- Clearing the `ECC_STATUS` bits~~.~~

Table 9. Flash Low-Level Commands

Command	Description
`CMD_IDLE`	Set the flash memory control signals so that the flash memory is in Standby Mode.
`CMD_WAKE_UP`	Power-up sequence. This sequence starts automatically when the flash memory is powered~~-~~up through the `SYSCTRL_MEM_POWER_CFG` register. Depending on the `LOAD_AUTO` setting, the `CMD_LOAD_TRIM` command is executed after this command.
`CMD_LOAD_TRIM`	Transfer settings from the NVR4 sector to the `PATCH_ADDR[3:0]` registers and to the flash memory internal configuration registers. The status bit `TRIMMED_STATUS` is updated at the end of the command, indicating if there was an uncorrectable ECC error or a word pair contained all ones or zeroes when reading NVR4, in which case the default trim values are used to guarantee the proper flash memory read functionality.
`CMD_READ`	Execute a read access: • If ECC is enabled, a pair of 32-bit words is read from flash memory. • If ECC is disabled, a single 36-bit word is read from flash memory.

Table 9. Flash Low-Level Commands (Continued)

Command	Description
CMD_PROGRAM_NOSEQ	Execute a non-sequential programming access: • If ECC is enabled, a pair of 32-bit words is written to flash memory. • If ECC is disabled, a single 36-bit word is written to flash memory.
CMD_PROGRAM_SEQ	Initiate a sequential programming access to flash memory: • If ECC is enabled, up to 32 pairs of 32-bit words can be written to the same row within a flash memory sector. • If ECC is disabled, up to 64 words of 36 bits can be written to the same row within a flash memory sector.
CMD_PROGRAM_SEQ_END	Terminate a sequential programming access. This command is only accepted when the sector addressed by FLASH_ADDR is in an unlocked flash memory zone.
CMD_SECTOR_ERASE	Erase a sector of the flash memory. This command is only accepted when the sector addressed by FLASH_ADDR is in an unlocked flash memory zone.
CMD_MASS_ERASE	Perform a mass erase. This command is only accepted when all of the areas in the main flash are unlocked or NVR1-3 are unlocked. • All sectors of the main flash array are erased when the main flash is unlocked. • The NVR1-3 sectors are erased when NVR1-3 are unlocked. • The RED1-2 sectors are erased when RED1-2 are unlocked. NOTE: By default the contents of NVR1-3 contain device information that should not be erased, and we recommend ensuring these sectors are locked when running the mass erase command. NOTE: The redundancy sectors are erased separately from both the main flash and the NVR1-3 sectors, as the redundancy sectors could replace any of these sectors. The user redundancy sectors are erased with the main flash. If these sectors were used to patch an NVR sector, use the mass erase command only if erasing both the main flash and NVR sectors.
CMD_SET_LOW_POWER	Set the LPWR flash memory pin respective to the hold & setup time. This command is called automatically when LP_MODE bit in FLASH_IF_CTRL is changed from 0 to 1.
CMD_UNSET_LOW_POWER	Unset the LPWR flash memory pin respective to the hold & setup time. This command is called automatically when LP_MODE bit in FLASH_IF_CTRL is changed from 1 to 0.
CMD_SET_RECALL	Set the RECALL flash memory pin respective to the hold & setup time. This command is called automatically when RECALL bit in FLASH_IF_CTRL is changed from 0 to 1.
CMD_UNSET_RECALL	Unset the RECALL flash memory pin respective to the hold & setup time. This command is called automatically when RECALL bit in FLASH_IF_CTRL is changed from 1 to 0.
CMD_SET_VREAD0	Set the VREAD0 flash memory pin respective to the hold & setup time. This command is called automatically when VREAD0_MODE bit in FLASH_IF_CTRL is changed from 0 to 1.
CMD_UNSET_VREAD0	Set the VREAD0 flash memory pin respective to the hold & setup time. This command is called automatically when VREAD0_MODE bit in FLASH_IF_CTRL is changed from 1 to 0.
CMD_SET_VREAD1	Set the VREAD1 flash memory pin respective to the hold & setup time. This command is called automatically when VREAD1_MODE bit in FLASH_IF_CTRL is changed from 0 to 1.
CMD_UNSET_VREAD1	Set the VREAD1 flash memory pin respective to the hold & setup time. This command is called automatically when VREAD1_MODE bit in FLASH_IF_CTRL is changed from 1 to 0.

Table 9. Flash Low-Level Commands (Continued)

Command	Description
CMD_WRITE_USER_RED1	Write contents of the FLASH_DATA register to MANU_FLASH_RR2 in NVR4. This value is loaded to the FLASH_PATCH_ADDR[2] register on boot, to configure USER_RED1 to patch the specified memory sector.
CMD_WRITE_USER_RED2	Write contents of the FLASH_DATA register to MANU_FLASH_RR3 in NVR4. This value is loaded to the FLASH_PATCH_ADDR[3] register on boot, to configure USER_RED2 to patch the specified memory sector.

The low-level commands can be written to the FLASH_CMD_CTRL register. These commands are then interpreted by a state machine, which acts on the different flash memory control signals to perform the relevant action

7.3.2 Low-Power Read Mode

- By setting the LP_MODE bit of the FLASH_IF_CTRL configuration register, it is possible to read the flash memory when its power supply voltage (VDDA) is below 2.75 V with reduced power consumption.
- The LP_MODE bit can only be modified when the flash is enabled. The state of the flash LP_MODE bit is maintained while the flash is disabled. When this LP_MODE bit is modified, the command CMD_SET_LOW_POWER is executed, and during the next 15 μs any flash memory read access is wait stated.

7.3.3 Reading and Programming Flash Memory Procedures

Once the flash memory is configured to enable writing to its desired areas, with the required delays set up, the flash memory programming procedure can be executed for either non-sequential or sequential programming as described in the following sections.

Reading from flash memory using the flash command interface can use non-sequential and sequential data accesses, using the same portions of the FLASH_DATA registers that are used for the write commands

CAUTION: Each group of 64 flash memory words (called a row) is allowed to be in programming mode for at most 5 ms between two erase events. This allows writing all words in each row three times with ECC disabled, or 5 times with ECC mode enabled. Exceeding this programming time between erase cycles can damage the flash cells.

7.3.3.1 Non-Sequential Programming

1. Set the address into FLASH_ADDR register.
2. Write to the data registers:
   - If the ECC is enabled, both FLASH_DATA registers are used to write both words in a double word pair along with their associated ECC information.

   - If the ECC is disabled, {FLASH_DATA[1](3:0), FLASH_DATA[0]} are treated as a 36-bit write value, written to one half of a double word pair and one half of the associated ECC bits

3. Write the command CMD_PROGRAM_NOSEQ to the FLASH_CMD_CTRL register to start the non-sequential sequence.
4. Poll the flash memory interface busy bit to check when the write is done.
5. The address is automatically incremented by 4 or 8, depending on the CMD_ECC_CTRL value.
6. The non-sequential programming sequence can be executed when the Arm Cortex-M3 processor runs a program from the flash memory, since the Arm Cortex-M3 processor is held in a wait state while the flash command is executing.

7.3.3.2 Sequential Programming

1. Set the address into the FLASH_ADDR register.
2. Write to the data registers as explained in Section 7.3.3.1, "Non-Sequential Programming".
3. If starting a new sequential write, write the command CMD_PROGRAM_SEQ to the FLASH_CMD_CTRL register.
4. Poll the SEQ_DATA_REG bit to check when the write is done.
5. The address is automatically incremented by 4 or 8, depending on the CMD_ECC_CTRL value. FLASH_ADDR bits 21:8 are not updated by this increment, and the address thus wraps around on a 256 boundary. This allows up to 64 different flash memory words to be written in one sequential programming sequence.
6. New data to write: jump to point 2, above, and writing to the FLASH_DATA1 register automatically applies the program sequence.
7. Close the program sequence: write the command CMD_PROGRAM_SEQ_END to the FLASH_CMD_CTRL register.
8. Poll the flash memory interface busy bit to check when the command is done.
9. The CMD_PROGRAM_SEQ command cannot be executed while the Arm Cortex-M3 processor runs a program from the flash memory.

7.3.4 Locking / Unlocking Mechanism

7.3.4.1 Locking Mechanism

A locking mechanism is included with the flash memory write hardware to prevent inadvertent writes to the flash memories. To lock the main flash memory, clear the FLASH_MAIN_CTRL and FLASH_NVR_CTRL registers.

7.3.4.2 Unlocking Mechanism

Similarly, there are 3 logical partitions that have their own permissions to unlock the main flash memory:

1. To write to the lower part of the main block of the flash memory (only possible if the main block is unlocked against Program/Erase), it is necessary to:
   a. Define the access permissions to the lower part of the flash memory (the MAIN_LOW_W_EN field in the FLASH_MAIN_CTRL register).
   b. Write a 32-bit unlock key to the FLASH_MAIN_WRITE_UNLOCK register.

2. To write to the middle part of the main block of the flash memory (only possible if the main block is unlocked against Program/Erase), it is necessary to:
   a. Define the access permissions to the middle part of the flash memory (field MAIN_MIDDLE_W_EN field in the FLASH_MAIN_CTRL register).
   b. Write a 32-bit unlock key to the FLASH_MAIN_WRITE_UNLOCK register.

3. To write to the high part of the main block of the flash memory (only possible if the main block is unlocked against Program/Erase), it is necessary to:
   a. Define the access permissions to the high part of the flash memory (field MAIN_HIGH_W_EN field in the FLASH_MAIN_CTRL register).
   b. Write a 32-bit unlock key to the FLASH_MAIN_WRITE_UNLOCK register.

4. A similar procedure needs to be applied to access the NVR block of the flash memory:
   a. Define the access permissions to the NVR block of the flash memory (field NVR[1|2|3]_W_EN field in the FLASH_NVR_CTRL register).
   b. Write a 32-bit unlock key to the FLASH_NVR_WRITE_UNLOCK register.

7.4 FLASH COPIER

This module copies data from flash memory into any DMA-accessible memory, or the CRC block. In 40-bit mode, the copier packs the 32-bit flash memory words into 40-bit words. This is useful to copy LPDSP32 programs from flash memory to `DSP_PRAM`. This module can also run in comparison mode. In this mode, the 36-bit data read from flash memory is verified against a reference value, but not written to any memory. This is useful to verify that a sector has been properly erased. It is also used for production testing purposes.

7.4.1 Block Characteristics

1. `FLASH_COPY_SRC_ADDR_PTR`: defines the flash memory source address (byte oriented). In 32-bit copier mode or in comparator mode, the pointer must point to the beginning of a word (2 LSBs are ignored).
2. `FLASH_COPY_DST_ADDR_PTR`: Defines the destination address into 32-bit word memory space. This pointer is not used in comparator mode, or when the copier destination is the CRC block. Addressing corresponds to logical memory instances in normal order, as viewed by the Arm Cortex-M3 processor.
3. `FLASH_COPY_WORD_CNT`: indicates how many words are to be written to the destination (copier mode), or the number of words to be read and verified (comparator mode).
4. `FLASH_COPY_CFG_MODE`: configures **`MODE`** (copier or comparator), flash memory access `COPY_MODE` (40-bit or 32-bit), `COPY_DEST` (memory or CRC), `COMP_MODE` (constant or checkerboard), `COMP_ADDR_DIR` (up or down), and `COMP_ADDR_STEP` (increment `FLASH_COPY_SRC_ADDR_PTR` by 1 or 2 words).
5. We recommend that the 40-bit copier mode only be used for `DSP_PRAM` destination. When a 32-bit memory is used as destination, then only 32-bit bytes are written and the 8 LSBs are discarded.
6. In copier mode, the flash memory is read with or without ECC, depending on the `COPIER_ECC_CTRL` bit of the `FLASH_ECC_CTRL` register. In comparator mode, the flash memory is always read without ECC.
7. `FLASH_COPY_CTRL`: provides **`START`** and **`STOP`** commands and read-only **`BUSY`** and **`ERROR`** status bits.
8. When both the **`START`** and **`STOP`** commands are issued at the same time, the **`STOP`** command takes priority.
9. When the **`START`** command is issued, the **`BUSY`** status bit is set and the **`ERROR`** bit is cleared.
10. While the copier is busy:
    a. 32-bit copier mode:
        i. A 32-bit word is read from flash memory.
        ii. The 32-bit word is written to the destination memory.
        iii. When the write has completed without error, the source address is incremented by 4, the destination address is incremented by 4, and the word counter is decremented.

    b. 40-bit copier mode:
        i. 32-bit words are read from flash memory, and when there is enough data to output a 40-bit word, the following occurs:
        ii. The 40-bit word is written to the destination memory.
        iii. When the write has completed without error, the source address is incremented by 5, the destination address is incremented by 4, and the word counter is decremented.

    c. CRC copier mode:
        i. A 32-bit word is read from the flash memory.
        ii. The 32-bit word is added to the CRC.
        iii. The source address is incremented by 4 and the word counter is decremented.

    d. Comparator mode:
        i. A 36-bit word is read from the flash memory.
        ii. In constant mode, or in checkerboard mode with an even source address, the 36-bit word is compared with `FLASH_DATA[1][3:0]` and `FLASH_DATA[0][31:0].`
        iii. In checkerboard mode with an odd source address, the 36-bit word is inverted and compared with `FLASH_DATA[1][3:0]` and `FLASH_DATA[0][31:0].`
        iv. When the comparison matches, the source address is updated according to `COMP_ADDR_DIR` and `COMP_ADDR_SIZE,` and the word counter is decremented.

11. When the word counter reaches zero, or when a write error has occurred (copier mode only), or when a verification error occurs (comparator mode only), then the **BUSY** status bit is cleared and an interrupt is generated for the Arm Cortex-M3 processor. In the case of an error, the **ERROR** bit is set.
12. The copy or comparator operation can be stopped before completion, through the **STOP** command. The operation can be continued by giving the **START** command, as the address pointers and word counter have the values required to continue the copy operation.
13. When the flash memory copier is running, the FLASH_COPY_CFG, FLASH_COPY_SRC_ADDR_PTR, FLASH_COPY_DST_ADDR_PTR and FLASH_WORD_CNT registers are not writable.
14. The priority handling between the flash memory copier and any DMA memory access, when writing to the flash memory copier's destination memory, is explained in Chapter 7, "Memory Arbitration" on page 94. The flash memory copier acts as a DMA access regarding the priority handling between the flash memory copier, Arm Cortex-M3 processor, LPDSP32 and baseband controller.

NOTE: While the flash memory copier is running, it constantly tries to read from the flash memory and write to its destination memory. If either memory instance is not available due to an access conflict with the Arm Cortex-M3 processor, LPDSP32, or DMA, the flash memory is re-read to ensure an atomic read and write occurs. We recommend avoiding such memory access conflicts, as they result in additional reads from flash memory and thus increase power consumption.

IMPORTANT: The flash copier has a known issue when configured for comparison mode. When verifying that an area of memory is the expected value, the flash copier indicates that an error has occurred, if the next address to compare is pointing to an area outside of memory. To work around this issue, instead of relying on the FLASH_COPY_CTRL_ERROR bit, a user application can verify that FLASH_COPY_SRC_ADDR_PTR points to the address one further than the last address that would have been checked. If FLASH_COPY_SRC_ADDR_PTR is set to this value, the comparison has succeeded; if not, the comparison points to the first word that has failed the comparison.

7.5 Memory Registers

Register Name	Register Description	Address
SYSCTRL_FLASH_OVERLAY_CFG	Flash memory / PRAM / DSPPRAM overlay control register	0x40000008
SYSCTRL_FLASH_READ_CNT	Flash memory read configuration register	0x40000040

7.5.1 SYSCTRL_FLASH_OVERLAY_CFG

Bit Field	Field Name	Description
7	DSP_PRAM0_OVERLAY_CFG	DSP_PRAM0 flash overlay configuration
6	DSP_PRAM1_OVERLAY_CFG	DSP_PRAM1 flash overlay configuration
5	DSP_PRAM2_OVERLAY_CFG	DSP_PRAM2 flash overlay configuration
4	DSP_PRAM3_OVERLAY_CFG	DSP_PRAM3 flash overlay configuration
3	PRAM3_OVERLAY_CFG	PRAM3 flash overlay configuration
2	PRAM2_OVERLAY_CFG	PRAM2 flash overlay configuration
1	PRAM1_OVERLAY_CFG	PRAM1 flash overlay configuration
0	PRAM0_OVERLAY_CFG	PRAM0 flash overlay configuration

Field Name	Value Symbol	Value Description	Hex Value
DSP_PRAM0_OVERLAY_CFG	DSP_PRAM0_OVERLAY_DISABLE	DSP_PRAM0 is not mapped on the flash addressing range	0x0*
	DSP_PRAM0_OVERLAY_ENABLE	DSP_PRAM0 is also mapped on the flash addressing range	0x1
DSP_PRAM1_OVERLAY_CFG	DSP_PRAM1_OVERLAY_DISABLE	DSP_PRAM1 is not mapped on the flash addressing range	0x0*
	DSP_PRAM1_OVERLAY_ENABLE	DSP_PRAM1 is also mapped on the flash addressing range	0x1
DSP_PRAM2_OVERLAY_CFG	DSP_PRAM2_OVERLAY_DISABLE	DSP_PRAM2 is not mapped on the flash addressing range	0x0*
	DSP_PRAM2_OVERLAY_ENABLE	DSP_PRAM2 is also mapped on the flash addressing range	0x1
DSP_PRAM3_OVERLAY_CFG	DSP_PRAM3_OVERLAY_DISABLE	DSP_PRAM3 is not mapped on the flash addressing range	0x0*
	DSP_PRAM3_OVERLAY_ENABLE	DSP_PRAM3 is also mapped on the flash addressing range	0x1
PRAM3_OVERLAY_CFG	PRAM3_OVERLAY_DISABLE	PRAM3 is not mapped on the flash addressing range	0x0*
	PRAM3_OVERLAY_ENABLE	PRAM3 is also mapped on the flash addressing range	0x1
PRAM2_OVERLAY_CFG	PRAM2_OVERLAY_DISABLE	PRAM2 is not mapped on the flash addressing range	0x0*
	PRAM2_OVERLAY_ENABLE	PRAM2 is also mapped on the flash addressing range	0x1
PRAM1_OVERLAY_CFG	PRAM1_OVERLAY_DISABLE	PRAM1 is not mapped on the flash addressing range	0x0*
	PRAM1_OVERLAY_ENABLE	PRAM1 is also mapped on the flash addressing range	0x1
PRAM0_OVERLAY_CFG	PRAM0_OVERLAY_DISABLE	PRAM0 is not mapped on the flash addressing range	0x0*
	PRAM0_OVERLAY_ENABLE	PRAM0 is also mapped on the flash addressing range	0x1

7.5.2 SYSCTRL_FLASH_READ_CNT

Bit Field	Field Name	Description
31:0	FLASH_READ_CNT	Flash read access counter value

7.5.3 SYSCTRL_MEM_ERROR

Bit Field	Field Name	Description
5	MEM_ERROR_CLEAR	Write a 1 to clear the memory error flags
4	BB_MEM_ERROR	Baseband memory error flag
3	FLASH_COPIER_MEM_ERROR	Flash copier memory error flag
2	DMA_MEM_ERROR	DMA memory error flag

Bit Field	Field Name	Description
1	LPDSP32_DMEM_ERROR	LPDSP32 data memory error flag
0	LPDSP32_PMEM_ERROR	LPDSP32 program memory error flag

Field Name	Value Symbol	Value Description	Hex Value
MEM_ERROR_CLEAR	MEM_ERROR_CLEAR	Clear the memory error flags	0x1
BB_MEM_ERROR	BB_MEM_NO_ERROR_DETECTED	No baseband memory error detected	0x0*
	BB_MEM_ERROR_DETECTED	Baseband has accessed an isolated memory	0x1
FLASH_COPIER_MEM_ERROR	FLASH_COPIER_MEM_NO_ERROR_DETECTED	No flash copier memory error detected	0x0*
	FLASH_COPIER_MEM_ERROR_DETECTED	Flash copier has accessed an isolated memory	0x1
DMA_MEM_ERROR	DMA_MEM_NO_ERROR_DETECTED	No DMA memory error detected	0x0*
	DMA_MEM_ERROR_DETECTED	DMA has accessed an isolated memory	0x1
LPDSP32_DMEM_ERROR	LPDSP32_DMEM_NO_ERROR_DETECTED	No LPDSP32 data memory error detected	0x0*
	LPDSP32_DMEM_ERROR_DETECTED	LPDSP32 has accessed an isolated data memory	0x1
LPDSP32_PMEM_ERROR	LPDSP32_PMEM_NO_ERROR_DETECTED	No LPDSP32 program memory error detected	0x0*
	LPDSP32_PMEM_ERROR_DETECTED	LPDSP32 has accessed an isolated program memory	0x1

7.5.4 SYSCTRL_MEM_POWER_CFG

Bit Field	Field Name	Description
21	DSP_DRAM5_POWER	DSP DRAM5 power configuration
20	DSP_DRAM4_POWER	DSP DRAM4 power configuration
19	DSP_DRAM3_POWER	DSP DRAM3-0 power configuration
18	DSP_DRAM2_POWER	DSP DRAM2 power configuration
17	DSP_DRAM1_POWER	DSP DRAM1 power configuration
16	DSP_DRAM0_POWER	DSP DRAM0 power configuration
15	DSP_PRAM3_POWER	DSP PRAM3 power configuration
14	DSP_PRAM2_POWER	DSP PRAM2 power configuration
13	DSP_PRAM1_POWER	DSP PRAM1 power configuration
12	DSP_PRAM0_POWER	DSP PRAM0 power configuration
11	BB_DRAM1_POWER	Baseband DRAM1 power configuration
10	BB_DRAM0_POWER	Baseband DRAM0 power configuration
8	DRAM2_POWER	DRAM2 power configuration
7	DRAM1_POWER	DRAM1 power configuration

Bit Field	Field Name	Description
6	DRAM0_POWER	DRAM0 power configuration
5	PRAM3_POWER	PRAM3 power configuration
4	PRAM2_POWER	PRAM2 power configuration
3	PRAM1_POWER	PRAM1 power configuration
2	PRAM0_POWER	PRAM0 power configuration
1	FLASH_POWER	Flash power configuration
0	PROM_POWER	PROM power configuration

Field Name	Value Symbol	Value Description	Hex Value
DSP_DRAM5_POWER	DSP_DRAM5_POWER_DISABLE	DSP DRAM5 power disabled	0x0
	DSP_DRAM5_POWER_ENABLE	DSP DRAM5 power enabled	0x1*
DSP_DRAM4_POWER	DSP_DRAM4_POWER_DISABLE	DSP DRAM4 power disabled	0x0
	DSP_DRAM4_POWER_ENABLE	DSP DRAM4 power enabled	0x1*
DSP_DRAM3_POWER	DSP_DRAM3_POWER_DISABLE	DSP DRAM3 power disabled	0x0
	DSP_DRAM3_POWER_ENABLE	DSP DRAM3 power enabled	0x1*
DSP_DRAM2_POWER	DSP_DRAM2_POWER_DISABLE	DSP DRAM2 power disabled	0x0
	DSP_DRAM2_POWER_ENABLE	DSP DRAM2 power enabled	0x1*
DSP_DRAM1_POWER	DSP_DRAM1_POWER_DISABLE	DSP DRAM1 power disabled	0x0
	DSP_DRAM1_POWER_ENABLE	DSP DRAM1 power enabled	0x1*
DSP_DRAM0_POWER	DSP_DRAM0_POWER_DISABLE	DSP DRAM0 power disabled	0x0
	DSP_DRAM0_POWER_ENABLE	DSP DRAM0 power enabled	0x1*
DSP_PRAM3_POWER	DSP_PRAM3_POWER_DISABLE	DSP PRAM3 power disabled	0x0*
	DSP_PRAM3_POWER_ENABLE	DSP PRAM3 power enabled	0x1
DSP_PRAM2_POWER	DSP_PRAM2_POWER_DISABLE	DSP PRAM2 power disabled	0x0*
	DSP_PRAM2_POWER_ENABLE	DSP PRAM2 power enabled	0x1
DSP_PRAM1_POWER	DSP_PRAM1_POWER_DISABLE	DSP PRAM1 power disabled	0x0*
	DSP_PRAM1_POWER_ENABLE	DSP PRAM1 power enabled	0x1
DSP_PRAM0_POWER	DSP_PRAM0_POWER_DISABLE	DSP PRAM0 power disabled	0x0*
	DSP_PRAM0_POWER_ENABLE	DSP PRAM0 power enabled	0x1
BB_DRAM1_POWER	BB_DRAM1_POWER_DISABLE	Baseband DRAM1 power disabled	0x0
	BB_DRAM1_POWER_ENABLE	Baseband DRAM1 power enabled	0x1*
BB_DRAM0_POWER	BB_DRAM0_POWER_DISABLE	Baseband DRAM0 power disabled	0x0
	BB_DRAM0_POWER_ENABLE	Baseband DRAM0 power enabled	0x1*
DRAM2_POWER	DRAM2_POWER_DISABLE	DRAM2 power disabled	0x0
	DRAM2_POWER_ENABLE	DRAM2 power enabled	0x1*
DRAM1_POWER	DRAM1_POWER_DISABLE	DRAM1 power disabled	0x0
	DRAM1_POWER_ENABLE	DRAM1 power enabled	0x1*
DRAM0_POWER	DRAM0_POWER_DISABLE	DRAM0 power disabled	0x0
	DRAM0_POWER_ENABLE	DRAM0 power enabled	0x1*

Field Name	Value Symbol	Value Description	Hex Value
PRAM3_POWER	PRAM3_POWER_DISABLE	PRAM3 power disabled	0x0
	PRAM3_POWER_ENABLE	PRAM3 power enabled	0x1*
PRAM2_POWER	PRAM2_POWER_DISABLE	PRAM2 power disabled	0x0
	PRAM2_POWER_ENABLE	PRAM2 power enabled	0x1*
PRAM1_POWER	PRAM1_POWER_DISABLE	PRAM1 power disabled	0x0
	PRAM1_POWER_ENABLE	PRAM1 power enabled	0x1*
PRAM0_POWER	PRAM0_POWER_DISABLE	PRAM0 power disabled	0x0
	PRAM0_POWER_ENABLE	PRAM0 power enabled	0x1*
FLASH_POWER	FLASH_POWER_DISABLE	Flash power disabled	0x0*
	FLASH_POWER_ENABLE	Flash power enabled	0x1
PROM_POWER	PROM_POWER_DISABLE	PROM power disabled	0x0
	PROM_POWER_ENABLE	PROM power enabled	0x1*

7.5.5 SYSCTRL_MEM_ACCESS_CFG

Bit Field	Field Name	Description
30:24	WAKEUP_ADDR_PACKED	Wakeup restore address in packed 7-bit format. When written, SYSCTRL_WAKEUP_ADDR is updated. This field reads back as zero when SYSCTRL_WAKEUP_ADDR does not point to an enabled RAM instance.
21	DSP_DRAM5_ACCESS	DSP DRAM5 access configuration
20	DSP_DRAM4_ACCESS	DSP DRAM4 access configuration
19	DSP_DRAM3_ACCESS	DSP DRAM3 access configuration
18	DSP_DRAM2_ACCESS	DSP DRAM2 access configuration
17	DSP_DRAM1_ACCESS	DSP DRAM1 access configuration
16	DSP_DRAM0_ACCESS	DSP DRAM0 access configuration
15	DSP_PRAM3_ACCESS	DSP PRAM3 access configuration
14	DSP_PRAM2_ACCESS	DSP PRAM2 access configuration
13	DSP_PRAM1_ACCESS	DSP PRAM1 access configuration
12	DSP_PRAM0_ACCESS	DSP PRAM0 access configuration
11	BB_DRAM1_ACCESS	Baseband DRAM1 access configuration
10	BB_DRAM0_ACCESS	Baseband DRAM0 access configuration
8	DRAM2_ACCESS	DRAM2 access configuration
7	DRAM1_ACCESS	DRAM1 access configuration
6	DRAM0_ACCESS	DRAM0 access configuration
5	PRAM3_ACCESS	PRAM3 access configuration
4	PRAM2_ACCESS	PRAM2 access configuration
3	PRAM1_ACCESS	PRAM1 access configuration
2	PRAM0_ACCESS	PRAM0 access configuration
1	FLASH_ACCESS	Flash access configuration
0	PROM_ACCESS	PROM access configuration

Field Name	Value Symbol	Value Description	Hex Value
DSP_DRAM5_ACCESS	DSP_DRAM5_ACCESS_DISABLE	DSP DRAM5 access disabled	0x0*
	DSP_DRAM5_ACCESS_ENABLE	DSP DRAM5 access enabled	0x1
DSP_DRAM4_ACCESS	DSP_DRAM4_ACCESS_DISABLE	DSP DRAM4 access disabled	0x0*
	DSP_DRAM4_ACCESS_ENABLE	DSP DRAM4 access enabled	0x1
DSP_DRAM3_ACCESS	DSP_DRAM3_ACCESS_DISABLE	DSP DRAM3 access disabled	0x0*
	DSP_DRAM3_ACCESS_ENABLE	DSP DRAM3 access enabled	0x1
DSP_DRAM2_ACCESS	DSP_DRAM2_ACCESS_DISABLE	DSP DRAM2 access disabled	0x0*
	DSP_DRAM2_ACCESS_ENABLE	DSP DRAM2 access enabled	0x1
DSP_DRAM1_ACCESS	DSP_DRAM1_ACCESS_DISABLE	DSP DRAM1 access disabled	0x0*
	DSP_DRAM1_ACCESS_ENABLE	DSP DRAM1 access enabled	0x1
DSP_DRAM0_ACCESS	DSP_DRAM0_ACCESS_DISABLE	DSP DRAM0 access disabled	0x0*
	DSP_DRAM0_ACCESS_ENABLE	DSP DRAM0 access enabled	0x1
DSP_PRAM3_ACCESS	DSP_PRAM3_ACCESS_DISABLE	DSP PRAM3 access disabled	0x0*
	DSP_PRAM3_ACCESS_ENABLE	DSP PRAM3 access enabled	0x1
DSP_PRAM2_ACCESS	DSP_PRAM2_ACCESS_DISABLE	DSP PRAM2 access disabled	0x0*
	DSP_PRAM2_ACCESS_ENABLE	DSP PRAM2 access enabled	0x1
DSP_PRAM1_ACCESS	DSP_PRAM1_ACCESS_DISABLE	DSP PRAM1 access disabled	0x0*
	DSP_PRAM1_ACCESS_ENABLE	DSP PRAM1 access enabled	0x1
DSP_PRAM0_ACCESS	DSP_PRAM0_ACCESS_DISABLE	DSP PRAM0 access disabled	0x0*
	DSP_PRAM0_ACCESS_ENABLE	DSP PRAM0 access enabled	0x1
BB_DRAM1_ACCESS	BB_DRAM1_ACCESS_DISABLE	Baseband DRAM1 access disabled	0x0*
	BB_DRAM1_ACCESS_ENABLE	Baseband DRAM1 access enabled	0x1
BB_DRAM0_ACCESS	BB_DRAM0_ACCESS_DISABLE	Baseband DRAM0 access disabled	0x0*
	BB_DRAM0_ACCESS_ENABLE	Baseband DRAM0 access enabled	0x1
DRAM2_ACCESS	DRAM2_ACCESS_DISABLE	DRAM2 access disabled	0x0*
	DRAM2_ACCESS_ENABLE	DRAM2 access enabled	0x1
DRAM1_ACCESS	DRAM1_ACCESS_DISABLE	DRAM1 access disabled	0x0*
	DRAM1_ACCESS_ENABLE	DRAM1 access enabled	0x1
DRAM0_ACCESS	DRAM0_ACCESS_DISABLE	DRAM0 access disabled	0x0
	DRAM0_ACCESS_ENABLE	DRAM0 access enabled	0x1*
PRAM3_ACCESS	PRAM3_ACCESS_DISABLE	PRAM3 access disabled	0x0*
	PRAM3_ACCESS_ENABLE	PRAM3 access enabled	0x1
PRAM2_ACCESS	PRAM2_ACCESS_DISABLE	PRAM2 access disabled	0x0*
	PRAM2_ACCESS_ENABLE	PRAM2 access enabled	0x1
PRAM1_ACCESS	PRAM1_ACCESS_DISABLE	PRAM1 access disabled	0x0*
	PRAM1_ACCESS_ENABLE	PRAM1 access enabled	0x1
PRAM0_ACCESS	PRAM0_ACCESS_DISABLE	PRAM0 access disabled	0x0*
	PRAM0_ACCESS_ENABLE	PRAM0 access enabled	0x1

Field Name	Value Symbol	Value Description	Hex Value
FLASH_ACCESS	FLASH_ACCESS_DISABLE	Flash access disabled	0x0*
	FLASH_ACCESS_ENABLE	Flash access enabled	0x1
PROM_ACCESS	PROM_ACCESS_DISABLE	PROM access disabled	0x0
	PROM_ACCESS_ENABLE	PROM access enabled	0x1*

7.5.6 SYSCTRL_MEM_RETENTION_CFG

Bit Field	Field Name	Description
21	DSP_DRAM5_RETENTION	DSP PRAM5 retention configuration
20	DSP_DRAM4_RETENTION	DSP PRAM4 retention configuration
19	DSP_DRAM3_RETENTION	DSP PRAM3 retention configuration
18	DSP_DRAM2_RETENTION	DSP PRAM2 retention configuration
17	DSP_DRAM1_RETENTION	DSP PRAM1 retention configuration
16	DSP_DRAM0_RETENTION	DSP PRAM0 retention configuration
15	DSP_PRAM3_RETENTION	DSP PRAM3 retention configuration
14	DSP_PRAM2_RETENTION	DSP PRAM2 retention configuration
13	DSP_PRAM1_RETENTION	DSP PRAM1 retention configuration
12	DSP_PRAM0_RETENTION	DSP PRAM0 retention configuration
11	BB_DRAM1_RETENTION	Baseband DRAM1 retention configuration
10	BB_DRAM0_RETENTION	Baseband DRAM0 retention configuration
8	DRAM2_RETENTION	DRAM2 retention configuration
7	DRAM1_RETENTION	DRAM1 retention configuration
6	DRAM0_RETENTION	DRAM0 retention configuration
5	PRAM3_RETENTION	PRAM3 retention configuration
4	PRAM2_RETENTION	PRAM2 retention configuration
3	PRAM1_RETENTION	PRAM1 retention configuration
2	PRAM0_RETENTION	PRAM0 retention configuration

Field Name	Value Symbol	Value Description	Hex Value
DSP_DRAM5_RETENTION	DSP_DRAM5_NORMAL_MODE	DSP DRAM5 Normal Mode	0x0
	DSP_DRAM5_RETENTION_MODE	DSP DRAM5 Retention Mode	0x1*
DSP_DRAM4_RETENTION	DSP_DRAM4_NORMAL_MODE	DSP DRAM4 Normal Mode	0x0
	DSP_DRAM4_RETENTION_MODE	DSP DRAM4 Retention Mode	0x1*
DSP_DRAM3_RETENTION	DSP_DRAM3_NORMAL_MODE	DSP DRAM3 Normal Mode	0x0
	DSP_DRAM3_RETENTION_MODE	DSP DRAM3 Retention Mode	0x1*
DSP_DRAM2_RETENTION	DSP_DRAM2_NORMAL_MODE	DSP DRAM2 Normal Mode	0x0
	DSP_DRAM2_RETENTION_MODE	DSP DRAM2 Retention Mode	0x1*
DSP_DRAM1_RETENTION	DSP_DRAM1_NORMAL_MODE	DSP DRAM1 Normal Mode	0x0
	DSP_DRAM1_RETENTION_MODE	DSP DRAM1 Retention Mode	0x1*

Field Name	Value Symbol	Value Description	Hex Value
DSP_DRAM0_RETENTION	DSP_DRAM0_NORMAL_MODE	DSP DRAM0 Normal Mode	0x0
	DSP_DRAM0_RETENTION_MODE	DSP DRAM0 Retention Mode	0x1*
DSP_PRAM3_RETENTION	DSP_PRAM3_NORMAL_MODE	DSP PRAM3 Normal Mode	0x0*
	DSP_PRAM3_RETENTION_MODE	DSP PRAM3 Retention Mode	0x1
DSP_PRAM2_RETENTION	DSP_PRAM2_NORMAL_MODE	DSP PRAM2 Normal Mode	0x0*
	DSP_PRAM2_RETENTION_MODE	DSP PRAM2 Retention Mode	0x1
DSP_PRAM1_RETENTION	DSP_PRAM1_NORMAL_MODE	DSP PRAM1 Normal Mode	0x0*
	DSP_PRAM1_RETENTION_MODE	DSP PRAM1 Retention Mode	0x1
DSP_PRAM0_RETENTION	DSP_PRAM0_NORMAL_MODE	DSP PRAM0 Normal Mode	0x0*
	DSP_PRAM0_RETENTION_MODE	DSP PRAM0 Retention Mode	0x1
BB_DRAM1_RETENTION	BB_DRAM1_NORMAL_MODE	Baseband DRAM1 Normal Mode	0x0
	BB_DRAM1_RETENTION_MODE	Baseband DRAM1 Retention Mode	0x1*
BB_DRAM0_RETENTION	BB_DRAM0_NORMAL_MODE	Baseband DRAM0 Normal Mode	0x0
	BB_DRAM0_RETENTION_MODE	Baseband DRAM0 Retention Mode	0x1*
DRAM2_RETENTION	DRAM2_NORMAL_MODE	DRAM2 Normal Mode	0x0
	DRAM2_RETENTION_MODE	DRAM2 Retention Mode	0x1*
DRAM1_RETENTION	DRAM1_NORMAL_MODE	DRAM1 Normal Mode	0x0
	DRAM1_RETENTION_MODE	DRAM1 Retention Mode	0x1*
DRAM0_RETENTION	DRAM0_NORMAL_MODE	DRAM0 Normal Mode	0x0*
	DRAM0_RETENTION_MODE	DRAM0 Retention Mode	0x1
PRAM3_RETENTION	PRAM3_NORMAL_MODE	PRAM3 Normal Mode	0x0
	PRAM3_RETENTION_MODE	PRAM3 Retention Mode	0x1*
PRAM2_RETENTION	PRAM2_NORMAL_MODE	PRAM2 Normal Mode	0x0
	PRAM2_RETENTION_MODE	PRAM2 Retention Mode	0x1*
PRAM1_RETENTION	PRAM1_NORMAL_MODE	PRAM1 Normal Mode	0x0
	PRAM1_RETENTION_MODE	PRAM1 Retention Mode	0x1*
PRAM0_RETENTION	PRAM0_NORMAL_MODE	PRAM0 Normal Mode	0x0
	PRAM0_RETENTION_MODE	PRAM0 Retention Mode	0x1*

7.5.7 SYSCTRL_MEM_ARBITER_CFG

Bit Field	Field Name	Description
29:28	DSP_DRAM45_ARBITER	DSP DRAM4 and DRAM5 arbiter configuration
27:26	DSP_DRAM23_ARBITER	DSP DRAM2 and DRAM3 arbiter configuration
25:24	DSP_DRAM01_ARBITER	DSP DRAM0 and DRAM1 arbiter configuration
23:22	DSP_PRAM3_ARBITER	DSP PRAM3 arbiter configuration
21:20	DSP_PRAM2_ARBITER	DSP PRAM2 arbiter configuration
19:18	DSP_PRAM1_ARBITER	DSP PRAM1 arbiter configuration
17:16	DSP_PRAM0_ARBITER	DSP PRAM0 arbiter configuration

Bit Field	Field Name	Description
11:10	BB_DRAM1_ARBITER	Baseband DRAM1 arbiter configuration
9:8	BB_DRAM0_ARBITER	Baseband DRAM0 arbiter configuration
5:4	DRAM12_ARBITER	DRAM1 and DRAM2 arbiter configuration
2	DRAM0_ARBITER	DRAM0 arbiter configuration
1	PRAM_ARBITER	PRAM0 to PRAM3 arbiter configuration
0	ROUND_ROBIN_TOKEN	Round-robin token generation configuration

Field Name	Value Symbol	Value Description	Hex Value
DSP_DRAM45_ARBITER	DSP_DRAM45_DSP_PRIORITY	DSP has priority access to the DSP DRAM4 and DRAM5 (above Arm Cortex-M3 core and DMA)	0x0*
	DSP_DRAM45_CM3_PRIORITY	Arm Cortex-M3 core has priority access to the DSP DRAM4 and DRAM5 (above DSP and DMA)	0x1
	DSP_DRAM45_ROUND_ROBIN_PRIORITY	Round robin priority access to the DSP DRAM4 and DRAM5	0x2
DSP_DRAM23_ARBITER	DSP_DRAM23_DSP_PRIORITY	DSP has priority access to the DSP DRAM2 and DRAM3 (above Arm Cortex-M3 core and DMA)	0x0*
	DSP_DRAM23_CM3_PRIORITY	Arm Cortex-M3 core has priority access to the DSP DRAM2 and DRAM3 (above DSP and DMA)	0x1
	DSP_DRAM23_ROUND_ROBIN_PRIORITY	Round robin priority access to the DSP DRAM2 and DRAM3	0x2
DSP_DRAM01_ARBITER	DSP_DRAM01_DSP_PRIORITY	DSP has priority access to the DSP DRAM0 and DRAM1 (above Arm Cortex-M3 core and DMA)	0x0*
	DSP_DRAM01_CM3_PRIORITY	Arm Cortex-M3 core has priority access to the DSP DRAM0 and DRAM1 (above DSP and DMA)	0x1
	DSP_DRAM01_ROUND_ROBIN_PRIORITY	Round robin priority access to the DSP DRAM0 and DRAM1	0x2
DSP_PRAM3_ARBITER	DSP_PRAM3_DSP_PRIORITY	DSP has priority access to the DSP PRAM3 (above Arm Cortex-M3 core and DMA)	0x0*
	DSP_PRAM3_CM3_PRIORITY	Arm Cortex-M3 core has priority access to the DSP PRAM3 (above DSP and DMA)	0x1
	DSP_PRAM3_ROUND_ROBIN_PRIORITY	Round robin priority access to the DSP PRAM3	0x2
DSP_PRAM2_ARBITER	DSP_PRAM2_DSP_PRIORITY	DSP has priority access to the DSP PRAM2 (above Arm Cortex-M3 core and DMA)	0x0*
	DSP_PRAM2_CM3_PRIORITY	Arm Cortex-M3 core has priority access to the DSP PRAM2 (above DSP and DMA)	0x1
	DSP_PRAM2_ROUND_ROBIN_PRIORITY	Round robin priority access to the DSP PRAM2	0x2

Field Name	Value Symbol	Value Description	Hex Value
DSP_PRAM1_ARBITER	DSP_PRAM1_DSP_PRIORITY	DSP has priority access to the DSP PRAM1 (above Arm Cortex-M3 core and DMA)	0x0*
	DSP_PRAM1_CM3_PRIORITY	Arm Cortex-M3 core has priority access to the DSP PRAM1 (above DSP and DMA)	0x1
	DSP_PRAM1_ROUND_ROBIN_PRIORITY	Round robin priority access to the DSP PRAM1	0x2
DSP_PRAM0_ARBITER	DSP_PRAM0_DSP_PRIORITY	DSP has priority access to the DSP PRAM0 (above Arm Cortex-M3 core and DMA)	0x0*
	DSP_PRAM0_CM3_PRIORITY	Arm Cortex-M3 core has priority access to the DSP PRAM0 (above DSP and DMA)	0x1
	DSP_PRAM0_ROUND_ROBIN_PRIORITY	Round robin priority access to the DSP PRAM0	0x2
BB_DRAM1_ARBITER	BB_DRAM1_BB_PRIORITY	Baseband controller has priority access to the BB DRAM1 (above Arm Cortex-M3 core and DMA)	0x0
	BB_DRAM1_CM3_PRIORITY	Arm Cortex-M3 core has priority access to the BB DRAM1 (above baseband and DMA)	0x1
	BB_DRAM1_ROUND_ROBIN_PRIORITY	Round robin priority access to the BB DRAM1	0x2
	BB_DRAM1_SMART_PRIORITY	Smart priority access to the BB DRAM1	0x3*
BB_DRAM0_ARBITER	BB_DRAM0_BB_PRIORITY	Baseband controller has priority access to the BB DRAM0 (above Arm Cortex-M3 core and DMA)	0x0
	BB_DRAM0_CM3_PRIORITY	Arm Cortex-M3 core has priority access to the BB DRAM0 (above baseband and DMA)	0x1
	BB_DRAM0_ROUND_ROBIN_PRIORITY	Round robin priority access to the BB DRAM0	0x2
	BB_DRAM0_SMART_PRIORITY	Smart priority access to the BB DRAM0	0x3*
DRAM12_ARBITER	DRAM12_CM3_PRIORITY	Arm Cortex-M3 core has priority access to the DRAM1 and DRAM2 (above DSP and DMA)	0x0*
	DRAM12_DSP_PRIORITY	DSP has priority access to the DRAM1 and DRAM2 (above Arm Cortex-M3 core and DMA)	0x1
	DRAM12_ROUND_ROBIN_PRIORITY	Round robin priority access to the DRAM1 and DRAM2	0x2
DRAM0_ARBITER	DRAM0_CM3_PRIORITY	Arm Cortex-M3 core has priority access to the DRAM0 (above DMA)	0x0*
	DRAM0_ROUND_ROBIN_PRIORITY	Round robin priority access to the DRAM0	0x1

Field Name	Value Symbol	Value Description	Hex Value
PRAM_ARBITER	PRAM_CM3_PRIORITY	Arm Cortex-M3 core has priority access to the PRAM0 to PRAM3 (above DMA)	0x0*
	PRAM_ROUND_ROBIN_PRIORITY	Round robin priority access to the PRAM0 to PRAM3	0x1
ROUND_ROBIN_TOKEN	REALTIME_DMA_MODE	Real-time DMA priority mode: After 7 cycles a pending high priority DMA access automatically gets the token	0x0*
	ROUND_ROBIN_MODE	Continuous round-robin mode: Token is rotating every SYSCLK cycle	0x1

7.5.8 SYSCTRL_MEM_TIMING_CFG

Bit Field	Field Name	Description
9:8	DSP_PRAM_EMAW	DSP_PRAM extra write margin configuration
6:4	DSP_PRAM_EMA	DSP_PRAM extra margin configuration
3	PROM_KEN	PROM bitlines keeper configuration
2:0	PROM_EMA	PROM extra margin configuration

Field Name	Value Symbol	Value Description	Hex Value
DSP_PRAM_EMAW	DSP_PRAM_EMAW_DEFAULT	DSP_PRAM default/minimum extra write margin	0x0*
	DSP_PRAM_EMAW_MAX	DSP_PRAM maximum extra write margin	0x3
DSP_PRAM_EMA	DSP_PRAM_EMA_MIN	DSP_PRAM minimum extra margin	0x0
	DSP_PRAM_EMA_DEFAULT	DSP_PRAM default extra margin	0x2*
	DSP_PRAM_EMA_MAX	DSP_PRAM maximum extra margin	0x7
PROM_KEN	PROM_KEN_ENABLED	PROM bitlines keeper enabled	0x0
	PROM_KEN_DISABLED	PROM bitlines keeper disabled	0x1*
PROM_EMA	PROM_EMA_MIN	PROM minimum extra margin	0x0
	PROM_EMA_DEFAULT	PROM default extra margin	0x5*
	PROM_EMA_MAX	PROM maximum extra margin	0x7

7.6 Flash Memory Registers

Register Name	Register Description	Address
FLASH_IF_CTRL	Flash Interface Control Register	0x40000500
FLASH_MAIN_WRITE_UNLOCK	Flash Main Write Unlock Register	0x40000504
FLASH_MAIN_CTRL	Flash Main Write Control Register	0x40000508
FLASH_DELAY_CTRL	Flash, Memory and RF Power-Up Delay Configuration	0x40000510
FLASH_CMD_CTRL	Flash Command Control Register	0x40000534
FLASH_IF_STATUS	Flash Interface Status Register	0x40000538
FLASH_ADDR	Flash Address Register	0x4000053C

Register Name	Register Description	Address
FLASH_DATA	Flash Read/Write Data Register	0x40000540
FLASH_NVR_WRITE_UNLOCK	Flash NVR Write Unlock Register	0x40000548
FLASH_NVR_CTRL	Flash NVR Control Register	0x4000054C
FLASH_PATCH_ADDR	Flash Patch Address Register	0x40000568
FLASH_COPY_CFG	Flash Copier Config Register	0x40000580
FLASH_COPY_CTRL	Flash-to-Memory Copier Control and Status	0x400005C8
FLASH_COPY_SRC_ADDR_PTR	Flash-to-Memory Copier Source Address Pointer	0x400005D0
FLASH_COPY_DST_ADDR_PTR	Flash-to-Memory Copier Destination Address Pointer	0x400005D4
FLASH_COPY_WORD_CNT	Flash-to-Memory Copier Word Count	0x400005D8
FLASH_ECC_CTRL	Flash ECC Control Register	0x400005DC
FLASH_ECC_STATUS	Flash ECC Status Register	0x400005E0
FLASH_ECC_ERROR_ADDR	Flash Address of the Latest Detected Error	0x400005E4
FLASH_ECC_UNCOR_ERROR_CNT	Flash ECC Uncorrected Error Counter	0x400005E8
FLASH_ECC_COR_ERROR_CNT	Flash ECC Corrected Error Counter	0x400005EC

7.6.1 FLASH_IF_CTRL

Bit Field	Field Name	Description
18	PREFETCH_D_BUS	Pre-fetch on D-Bus control
17	PREFETCH_I_BUS	Pre-fetch on I-Bus control
16	NOT_LOAD_AUTO	Do not automatically load the configuration registers and the patch information from NVR4 sector after the command WAKEUP is completed.
12	VREAD1_MODE	Control VREAD1: Read data after erase with more stringent condition than normal read. Changing this bit will execute the CMD_SET_VREAD1 or CMD_UNSET_VREAD1 command.
11	VREAD0_MODE	Control VREAD0: Read data after program with more stringent condition than normal read. Changing this bit will execute the CMD_SET_VREAD0 or CMD_UNSET_VREAD0 command.
10	RECALL	Set the recall pins mode during CMD_READ. Changing this bit will execute the CMD_SET_RECALL or CMD_UNSET_RECALL command.
9:8	RETRY	Configures the erase retry iteration. This impacts the eFlash endurance time. Also used by flash programming.
0	LP_MODE	Set the low power mode. Changing this bit will execute the CMD_SET_LOW_POWER or CMD_UNSET_LOW_POWER command.

Field Name	Value Symbol	Value Description	Hex Value
PREFETCH_D_BUS	FLASH_PREFETCH_D_BUS_DISABLE	Do not pre-fetch the n+1 address on D-Bus	0x0*
	FLASH_PREFETCH_D_BUS_ENABLE	Pre-fetch the n+1 address on D-Bus	0x1
PREFETCH_I_BUS	FLASH_PREFETCH_I_BUS_DISABLE	Do not pre-fetch the n+1 address on I-Bus	0x0*
	FLASH_PREFETCH_I_BUS_ENABLE	Pre-fetch the n+1 address on I-Bus	0x1

Field Name	Value Symbol	Value Description	Hex Value
NOT_LOAD_AUTO	FLASH_LOAD_AUTO_ENABLE	No automatic load after the WAKEUP command	0x0*
	FLASH_LOAD_AUTO_DISABLE	The CMD_WAKEUP includes the loading of internal registers and patch information.	0x1
VREAD1_MODE	FLASH_VREAD1_DISABLE	After erase, read data with a normal condition	0x0*
	FLASH_VREAD1_ENABLE	After erase, read data with a more stringent condition	0x1
VREAD0_MODE	FLASH_VREAD0_DISABLE	After programming, read data with a normal condition	0x0*
	FLASH_VREAD0_ENABLE	After programming, read data with a more stringent condition	0x1
RECALL	FLASH_RECALL_DISABLE	RECALL pin low during read command	0x0*
	FLASH_RECALL_ENABLE	RECALL pin high during read command	0x1
RETRY	FLASH_RETRY_1	For 1st erase pulse	0x0*
	FLASH_RETRY_2	For 2nd erase pulse	0x1
	FLASH_RETRY_3	For 3rd erase pulse	0x2
	FLASH_RETRY_4	For 4th erase pulse or required during programming	0x3
LP_MODE	FLASH_LOW_POWER_DISABLE	Disable the flash low power mode	0x0*
	FLASH_LOW_POWER_ENABLE	Enable the flash low power mode	0x1

7.6.2 FLASH_MAIN_WRITE_UNLOCK

Bit Field	Field Name	Description
31:0	UNLOCK_KEY	32-bit key to allow for write accesses into the flash main block

Field Name	Value Symbol	Value Description	Hex Value
UNLOCK_KEY	FLASH_MAIN_KEY	32-bit key to allow for Read and Write accesses into the flash main block	0xDBC8264E

7.6.3 FLASH_MAIN_CTRL

Bit Field	Field Name	Description
2	MAIN_HIGH_W_EN	Authorize write access to the high part of the flash main block through the FLASH_IF registers.
1	MAIN_MIDDLE_W_EN	Authorize write access to the middle part of the flash main block through the FLASH_IF registers.
0	MAIN_LOW_W_EN	Authorize write access to the lower part of the flash main block through the FLASH_IF registers.

Field Name	Value Symbol	Value Description	Hex Value
MAIN_HIGH_W_EN	MAIN_HIGH_W_DISABLE	The high part of the flash main block is protected against write access	0x0*
	MAIN_HIGH_W_ENABLE	The high part of the flash main block can be written	0x1
MAIN_MIDDLE_W_EN	MAIN_MIDDLE_W_DISABLE	The middle part of the flash main block is protected against write access	0x0*
	MAIN_MIDDLE_W_ENABLE	The middle part of the flash main block can be written	0x1
MAIN_LOW_W_EN	MAIN_LOW_W_DISABLE	The lower part of the flash main block is protected against write access	0x0*
	MAIN_LOW_W_ENABLE	The lower part of the flash main block can be written	0x1

7.6.4 FLASH_DELAY_CTRL

Bit Field	Field Name	Description
7	READ_MARGIN	Flash Read access time margin
3:0	SYSCLK_FREQ	Configure flash, memory and RF power-up delays

Field Name	Value Symbol	Value Description	Hex Value
READ_MARGIN	DEFAULT_READ_MARGIN	Used default read margins	0x0*
	FAST_READ_MARGIN	Used fast read margins	0x1
SYSCLK_FREQ	FLASH_DELAY_FOR_SYSCLK_3MHZ	FLASH_DELAY_CTRLx set for a SYSCLK = 3 MHz	0x0
	FLASH_DELAY_FOR_SYSCLK_4MHZ	FLASH_DELAY_CTRLx set for a SYSCLK = 4 MHz	0x1
	FLASH_DELAY_FOR_SYSCLK_5MHZ	FLASH_DELAY_CTRLx set for a SYSCLK = 5 MHz	0x2*
	FLASH_DELAY_FOR_SYSCLK_8MHZ	FLASH_DELAY_CTRLx set for a SYSCLK = 8 MHz	0x3
	FLASH_DELAY_FOR_SYSCLK_10MHZ	FLASH_DELAY_CTRLx set for a SYSCLK = 10 MHz	0x4
	FLASH_DELAY_FOR_SYSCLK_12MHZ	FLASH_DELAY_CTRLx set for a SYSCLK = 12 MHz	0x5
	FLASH_DELAY_FOR_SYSCLK_16MHZ	FLASH_DELAY_CTRLx set for a SYSCLK = 16 MHz	0x6
	FLASH_DELAY_FOR_SYSCLK_20MHZ	FLASH_DELAY_CTRLx set for a SYSCLK = 20 MHz	0x7
	FLASH_DELAY_FOR_SYSCLK_24MHZ	FLASH_DELAY_CTRLx set for a SYSCLK = 24 MHz	0x8
	FLASH_DELAY_FOR_SYSCLK_48MHZ	FLASH_DELAY_CTRLx set for a SYSCLK = 48 MHz	0x9

7.6.5 FLASH_CMD_CTRL

Bit Field	Field Name	Description
6:5	CMD_END	Terminates an active flash command if possible (e.g. sequential programming sequence)
4:0	COMMAND	Flash access command only writable when equal to CMD_IDLE

Field Name	Value Symbol	Value Description	Hex Value
CMD_END	CMD_END	Terminates an active flash command if possible	0x3
COMMAND	CMD_IDLE	Idle command	0x0*
	CMD_WAKE_UP	Wake up the flash	0x1
	CMD_LOAD_TRIM	Load patch and trimming values from NVR4	0x2
	CMD_READ	Execute a read cycle	0x5
	CMD_PROGRAM_NOSEQ	Execute a non-sequential programming cycle	0x6
	CMD_PROGRAM_SEQ	Starts a sequential programming sequence	0x7
	CMD_SECTOR_ERASE	Execute a sector erase cycle	0x8
	CMD_MASS_ERASE	Execute a mass erase cycle	0x9
	CMD_SET_LOW_POWER	Wait time to set the LPWR pin	0xA
	CMD_UNSET_LOW_POWER	Wait time to unset the LPWR pin	0xB
	CMD_SET_RECALL	Wait time to set the RECALL pin	0xC
	CMD_UNSET_RECALL	Wait time to unset the RECALL pin	0xD
	CMD_SET_VREAD1	Wait time to set the VREAD1 pin	0xE
	CMD_UNSET_VREAD1	Wait time to unset the VREAD1 pin	0xF
	CMD_SET_VREAD0	Wait time to set the VREAD0 pin	0x10
	CMD_UNSET_VREAD0	Wait time to unset the VREAD0 pin	0x11
	CMD_WRITE_USER_RED1	Write FLASH_DATA to PATCH2 of NVR4	0x12
	CMD_WRITE_USER_RED2	Write FLASH_DATA to PATCH3 of NVR4	0x13

7.6.6 FLASH_IF_STATUS

Bit Field	Field Name	Description
13	TRIMMED_STATUS	Flash trimming status
12	ISOLATE_STATUS	Flash isolate status
11	PROG_SEQ_DATA_REQ	Request new data while in sequential program mode
10	BUSY	Flash interface busy status bit
9	RED2_W_UNLOCK	Flash RED2 write unlock status bit
8	RED1_W_UNLOCK	Flash RED1 write unlock status bit

Bit Field	Field Name	Description
6	NVR3_W_UNLOCK	Flash NVR3 write unlock status bit
5	NVR2_W_UNLOCK	Flash NVR2 write unlock status bit
4	NVR1_W_UNLOCK	Flash NVR1 write unlock status bit
2	MAIN_HIGH_W_UNLOCK	Write unlock status bit of the high part of the flash main block
1	MAIN_MIDDLE_W_UNLOCK	Write unlock status bit of the middle part of the flash main block
0	MAIN_LOW_W_UNLOCK	Write unlock status bit of the lower part of the flash main block

Field Name	Value Symbol	Value Description	Hex Value
TRIMMED_STATUS	FLASH_UNTRIMMED	All NVR4 CBD0-CDB7 contents are equal to 0xFFFF. eFlash untrimmed.	0x0*
	FLASH_TRIMMED	Some registers CBD0-CBD7 contents are not equal to 0xFFFF. eFlash trimmed.	0x1
ISOLATE_STATUS	FLASH_ACCESSIBLE	Flash can be accessed (isolation inactive)	0x0
	FLASH_ISOLATE	Flash cannot be accessed (isolation active)	0x1*
PROG_SEQ_DATA_REQ	FLASH_PROG_SEQ_IDLE	No new data is requested by a Sequential Program sequence	0x0*
	FLASH_PROG_SEQ_REQ_NEW_DATA	New data is requested by a Sequential Program sequence	0x1
BUSY	FLASH_IF_IDLE	Indicates that the flash interface is ready	0x0*
	FLASH_IF_BUSY	Indicates that the flash interface is busy	0x1
RED2_W_UNLOCK	FLASH_RED2_W_LOCKED	Indicates that the flash RED2 sector is protected against write accesses by the flash interface	0x0*
	FLASH_RED2_W_UNLOCKED	Indicates that the flash RED2 sector can be write accessed by the flash interface	0x1
RED1_W_UNLOCK	FLASH_RED1_W_LOCKED	Indicates that the flash RED1 sector is protected against write accesses by the flash interface	0x0*
	FLASH_RED1_W_UNLOCKED	Indicates that the flash RED1 sector can be write accessed by the flash interface	0x1
NVR3_W_UNLOCK	FLASH_NVR3_W_LOCKED	Indicates that the flash NVR3 sector is protected against write accesses by the flash interface	0x0*
	FLASH_NVR3_W_UNLOCKED	Indicates that the flash NVR3 sector can be write accessed by the flash interface	0x1

Field Name	Value Symbol	Value Description	Hex Value
NVR2_W_UNLOCK	FLASH_NVR2_W_LOCKED	Indicates that the flash NVR2 sector is protected against write accesses by the flash interface	0x0*
	FLASH_NVR2_W_UNLOCKED	Indicates that the flash NVR2 sector can be write accessed by the flash interface	0x1
NVR1_W_UNLOCK	FLASH_NVR1_W_LOCKED	Indicates that the flash NVR1 sector is protected against write accesses by the flash interface	0x0*
	FLASH_NVR1_W_UNLOCKED	Indicates that the flash NVR1 sector can be write accessed by the flash interface	0x1
MAIN_HIGH_W_UNLOCK	FLASH_MAIN_HIGH_W_LOCKED	Indicates that the high part of the flash main section is protected against write accesses by the flash interface	0x0*
	FLASH_MAIN_HIGH_W_UNLOCKED	Indicates that the high part of the flash main section can be write accessed by the flash interface	0x1
MAIN_MIDDLE_W_UNLOCK	FLASH_MAIN_MIDDLE_W_LOCKED	Indicates that the middle part of the flash main section is protected against write accesses by the flash interface	0x0*
	FLASH_MAIN_MIDDLE_W_UNLOCKED	Indicates that the middle part of the flash main section can be write accessed by the flash interface	0x1
MAIN_LOW_W_UNLOCK	FLASH_MAIN_LOW_W_LOCKED	Indicates that the lower part of the flash main section is protected against write accesses by the flash interface	0x0*
	FLASH_MAIN_LOW_W_UNLOCKED	Indicates that the lower part of the flash main section can be write accessed by the flash interface	0x1

7.6.7 FLASH_ADDR

Bit Field	Field Name	Description
20:2	FLASH_ADDR	Flash Byte Address

7.6.8 FLASH_DATA

Bit Field	Field Name	Description
31:0	DATA	32-bit flash Data

7.6.9 FLASH_NVR_WRITE_UNLOCK

Bit Field	Field Name	Description
31:0	UNLOCK_KEY	32-bit key to allow for write access to NVR sectors of the flash

Field Name	Value Symbol	Value Description	Hex Value
UNLOCK_KEY	FLASH_NVR_KEY	32-bit key to allow for write access to the flash NVR sector	0x71B371F5

7.6.10 FLASH_NVR_CTRL

Bit Field	Field Name	Description
3	NVR3_W_EN	Authorize Write access to the flash NVR3 sector through the FLASH_IF registers.
2	NVR2_W_EN	Authorize Write access to the flash NVR2 sector through the FLASH_IF registers.
1	NVR1_W_EN	Authorize Write access to the flash NVR1 sector through the FLASH_IF registers.

Field Name	Value Symbol	Value Description	Hex Value
NVR3_W_EN	NVR3_WRITE_DISABLE	The flash NVR3 block is protected against write access.	0x0*
	NVR3_WRITE_ENABLE	The flash NVR3 block can be written.	0x1
NVR2_W_EN	NVR2_WRITE_DISABLE	The flash NVR2 block is protected against write access.	0x0*
	NVR2_WRITE_ENABLE	The flash NVR2 block can be written.	0x1
NVR1_W_EN	NVR1_WRITE_DISABLE	The flash NVR1 block is protected against write access.	0x0*
	NVR1_WRITE_ENABLE	The flash NVR1 block can be written.	0x1

7.6.11 FLASH_PATCH_ADDR

Bit Field	Field Name	Description
20:11	PATCH_ADDR	

7.6.12 FLASH_COPY_CFG

Bit Field	Field Name	Description
18	COMP_ADDR_STEP	Comparator address increment/decrement by 1 or 2
17	COMP_ADDR_DIR	Comparator address-up or address-down
16	COMP_MODE	Comparator Mode
9	COPY_DEST	Destination copier is the CRC or memories
8	COPY_MODE	Select copier mode (32-bit or 40-bit)
0	MODE	Copier or Comparator Mode Configuration

Field Name	Value Symbol	Value Description	Hex Value
COMP_ADDR_STEP	COMP_ADDR_STEP_1	Address increment/decrement by 1 between two reads	0x0*
	COMP_ADDR_STEP_2	Address increment/decrement by 2 between two reads	0x1
COMP_ADDR_DIR	COMP_ADDR_DOWN	FLASH_COPIER address count-down	0x0
	COMP_ADDR_UP	FLASH_COPIER address count-up	0x1*
COMP_MODE	COMP_MODE_CONSTANT	FLASH_DATA[1:0] compare with eFlash DOUT	0x0*
	COMP_MODE_CHBK	Odd address compare with FLASH_DATA[1:0], even address compare with inverse FLASH_DATA[1:0]	0x1
COPY_DEST	COPY_TO_MEM	Copy flash to memory	0x0*
	COPY_TO_CRC	Copy flash to CRC	0x1
COPY_MODE	COPY_TO_32BIT	Copy flash to 32-bit memory	0x0*
	COPY_TO_40BIT	Copy flash to 40-bit memory	0x1
MODE	COPY_MODE	Flash copier mode	0x0*
	COMPARATOR_MODE	Flash comparator mode	0x1

7.6.13 FLASH_COPY_CTRL

Bit Field	Field Name	Description
3	ERROR	Error status
2	STOP	Stop the transfer
1	START	Start the transfer
0	BUSY	Busy status

Field Name	Value Symbol	Value Description	Hex Value
ERROR	COPY_NO_ERROR	No write / comparison error	0x0*
	COPY_ERROR	Write or comparison error	0x1
STOP	COPY_STOP	Stop the current transfer	0x1
START	COPY_START	Start the current transfer	0x1
BUSY	COPY_IDLE	Flash copier is idle	0x0*
	COPY_BUSY	Flash copier is busy	0x1

7.6.14 FLASH_COPY_SRC_ADDR_PTR

Bit Field	Field Name	Description
20:0	COPY_SRC_ADDR_PTR	Source address pointer

7.6.15 FLASH_COPY_DST_ADDR_PTR

Bit Field	Field Name	Description
31:2	COPY_DST_ADDR_PTR	Destination address pointer

7.6.16 FLASH_COPY_WORD_CNT

Bit Field	Field Name	Description
16:0	COPY_WORD_CNT	Number of words to copy / compare

7.6.17 FLASH_ECC_CTRL

Bit Field	Field Name	Description
15:8	ECC_COR_CNT_INT_THRESHOLD	Select the number of corrected errors before sending an Arm Cortex-M3 core interrupt
3	COPIER_ECC_CTRL	
2	CMD_ECC_CTRL	
0	IDBUS_ECC_CTRL	Select the operating mode of the flash ECC

Field Name	Value Symbol	Value Description	Hex Value
ECC_COR_CNT_INT_THRESHOLD	FLASH_ECC_COR_INT_THRESHOLD_DISABLED	Interrupt is disabled	0x0
	FLASH_ECC_COR_INT_THRESHOLD_1	Send a Arm Cortex-M3 core interrupt when one or more correctable errors are detected.	0x1*
	FLASH_ECC_COR_INT_THRESHOLD_255	Send a Arm Cortex-M3 core interrupt when 255 or more correctable errors are detected.	0xFF
COPIER_ECC_CTRL	FLASH_COPIER_ECC_DISABLE	Disables ECC when reading flash through flash copier	0x0
	FLASH_COPIER_ECC_ENABLE	Enables ECC when reading flash through flash copier	0x1*
CMD_ECC_CTRL	FLASH_CMD_ECC_DISABLE	Disables ECC when reading flash through flash mapped register	0x0
	FLASH_CMD_ECC_ENABLE	Enables ECC when reading flash through flash mapped register	0x1*
IDBUS_ECC_CTRL	FLASH_IDBUS_ECC_DISABLE	Disables ECC when reading flash through I-Bus and D-Bus	0x0
	FLASH_IDBUS_ECC_ENABLE	Enables ECC when reading flash through I-Bus and D-Bus	0x1*

7.6.18 FLASH_ECC_STATUS

Bit Field	Field Name	Description
6	`ECC_COR_ERROR_CNT_CLEAR`	Reset the flash corrected errors counter
5	`ECC_UNCOR_ERROR_CNT_CLEAR`	Reset the flash uncorrected errors counter
4	`ECC_ERROR_ADDR_CLEAR`	Reset the flash address of the last detected error
1	`ECC_COR_ERROR_CNT_STATUS`	FLASH_ECC_ERROR_COR_CNT status
0	`ECC_UNCOR_ERROR_CNT_STATUS`	FLASH_ECC_ERROR_UNCOR_CNT status

Field Name	Value Symbol	Value Description	Hex Value
`ECC_COR_ERROR_CNT_CLEAR`	`FLASH_ECC_COR_ERROR_CNT_CLEAR`	Reset the flash corrected errors counter	0x1
`ECC_UNCOR_ERROR_CNT_CLEAR`	`FLASH_ECC_UNCOR_ERROR_CNT_CLEAR`	Reset the flash uncorrected errors counter	0x1
`ECC_ERROR_ADDR_CLEAR`	`FLASH_ECC_ERROR_ADDR_CLEAR`	Reset the flash address of the latest detected error	0x1
`ECC_COR_ERROR_CNT_STATUS`	`FLASH_ECC_NO_CORRECTED_ERROR`	Indicates FLASH_ECC_COR_ERROR_CNT is zero	0x0*
	`FLASH_ECC_CORRECTED_ERROR`	Indicates FLASH_ECC_COR_ERROR_CNT is not zero	0x1
`ECC_UNCOR_ERROR_CNT_STATUS`	`FLASH_ECC_NO_UNCORRECTED_ERROR`	Indicates FLASH_ECC_UNCOR_ERROR_CNT is zero	0x0*
	`FLASH_ECC_UNCORRECTED_ERROR`	Indicates FLASH_ECC_UNCOR_ERROR_CNT is not zero	0x1

7.6.19 FLASH_ECC_ERROR_ADDR

Bit Field	Field Name	Description
20:2	`ECC_ERROR_ADDR`	Store the flash address of the latest flash ECC error

7.6.20 FLASH_ECC_UNCOR_ERROR_CNT

Bit Field	Field Name	Description
7:0	`ECC_UNCOR_ERROR_CNT`	Flash ECC uncorrected error counter

7.6.21 FLASH_ECC_COR_ERROR_CNT

Bit Field	Field Name	Description
7:0	`ECC_COR_ERROR_CNT`	Flash ECC corrected error counter

CHAPTER 8

RF Front-End

8.1 Overview

The RF front-end (RFFE) transceiver is an ultra low-power 2.4 GHz radio, handling data rates up to 2 Mbps. It supports several wireless protocols such as Bluetooth low energy technology, custom, or proprietary protocols. The RF front-end communicates with:

- The Arm Cortex-M3 processor and the DMA, through a dedicated APB bridge that accesses the RF front-end internal configuration registers (refer to Appendix A, “Control and Configuration Registers” on page 415). The Arm Cortex-M3 processor always has priority over the DMA. A read operation inserts two wait states, and additional wait states are inserted when an SPI operation is active. A write operation inserts two or more wait states, and additional wait states are inserted when an SPI operation is active.
- The Arm Cortex-M3 processor uses a simple baseband to provide packet handling, data transfers (supported by interrupts and GPIOs as proprietary debug resources).
- The baseband controller communicates through the internal SPI interface and dedicated CLK and DATA signals. All these signals are multiplexed through the DIO block, as shown in Figure 34 on page 348.

An RF front-end arbiter deals with simultaneous accesses between the Arm Cortex-M3 processor, the DMA, and the internal SPI (baseband controller), with priority given to the SPI interface. The RF is powered by the VDDRF and VDDM regulators. Once these are enabled, the registers described in this chapter control the power, access, and interrupts.

Within the RFFE system block, as seen in Figure 11 on page 132, the analog components consists of the:

- Integrated inductors
- 48 MHz crystal oscillator and PLL
- Full Tx and Rx chains including:
    - On chip matching
    - Frequency synthesis
    - Power amplifier (PA) for Tx
    - Low-noise amplifier (LNA) for receiving with phase
    - Signal strength (received signal strength indication or RSSO) components for Rx

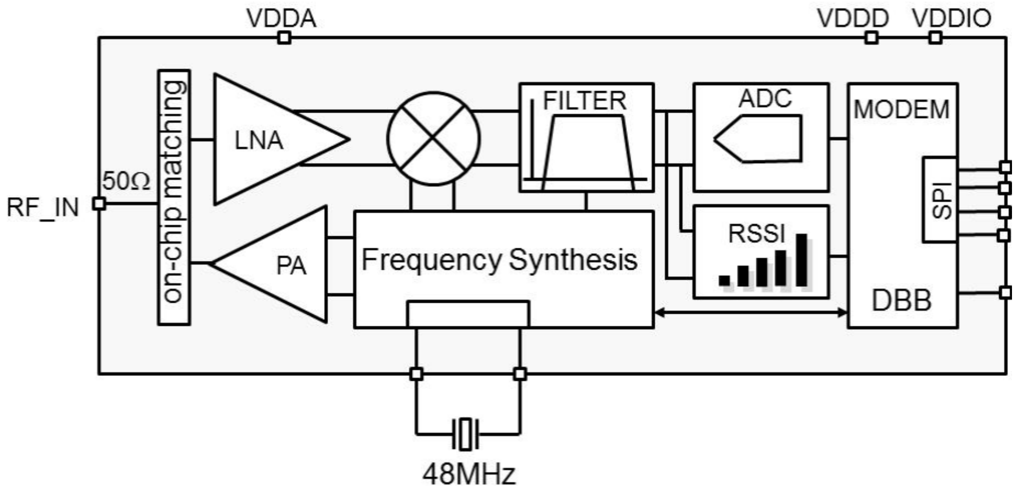


Figure 11. RFFE Block Diagram

The RF front-end implements a full transceiver, with the following digital features:

- FSK modem with programmable pulse shape and modulation index
- Data-rate programmable from 3 Mbps to 62.5 kbps (4 Mbps with 4-FSK)
- Packet handling (refer to Section 8.3, "Packet Handling" on page 135)
    - Automatic preamble and sync word insertion
    - Automatic packet length handler
    - Basic address check
    - Automatic CRC (Cyclic Redundancy Check) calculation and verification with a programmable CRC polynomial
    - Multi-frame support
    - 128 byte Tx FIFO, 128 byte Rx FIFO

- Encoding (refer to Section 8.4, "Modulator and Radio Configuration" on page 140)
    - IEEE 802.15.4 chip encoding and decoding
    - Manchester encoding
    - Data whitening

Combined with the 2 Mbps analog front end, the RFFE chip is capable of addressing Bluetooth low energy technology and custom or proprietary protocols.

In order to modify the RF power and access configurations, the `SYSCTRL_RF_POWER_CFG` (refer to Section 8.1.2, "SYSCTRL_RF_POWER_CFG") and `SYSCTRL_RF_ACCESS_CFG` (refer to Section 8.1.3, "SYSCTRL_RF_ACCESS_CFG") registers respectively are used. By enabling the `RF_POWER` bit in `SYSCTRL_RF_POWER_CFG`, the VDDM is connected to the RF block. To remove the RF block isolation, enable the `RF_ACCESS` bit in `SYSCTRL_RF_ACCESS_CFG`.

8.1.1 RF Front-End Registers

The registers configuring the RF front-end are accessed internally, directly by the analog Bluetooth baseband or through an SPI bus tied to the peripheral bus, with the internal RFFE SPI interface having priority over the SPI bus tied to the APB interface. For this reason, registers are grouped into their byte accesses only. In many instances where the grouped registers do not share a common use case, the registers are considered unnamed, and are only given a number. For these registers, this chapter uses the ungrouped register names.

IMPORTANT: The RF front-end registers support read-only bit-band access. Any register bit that normally has read-write access only has read access defined when used in a bit-band configuration, as bit-band write access requires a read-modify-write that is not supported by the SPI bus tied to the APB interface.

For example, register `RF_REG00` is an unnamed register that groups together the `MODE`, `MODE2`, `FOURFSK_CODING`, and `DATAWHITE_BTLE` registers, and an explicit description of `RF_REG00` is not included (whereas descriptions of the four grouped registers are).

Many of the RF front-end registers are supported by two distinct register banks to allow efficient switching between register configurations needed to change the modulation parameters during run time. The banked registers are marked in Section 8.5, “RF Front-End Registers” on page 159. Bank selection is controlled using the `BANK` bit-field from the `BANK` register (`RF_REG05`).

NOTE: Register bank 0 is typically used by the Bluetooth baseband for 1 Mbps configurations, and register bank 1 is typically used for 2 Mbps configurations.

8.1.2 SYSCTRL_RF_POWER_CFG

Register Name	Register Description	Address
`SYSCTRL_RF_POWER_CFG`	RF Power Configuration	0x40000050

Bit Field	Field Name	Description
0	`RF_POWER`	RF power configuration

Field Name	Value Symbol	Value Description	Hex Value
`RF_POWER`	`RF_POWER_DISABLE`	RF power disabled	0x0*
	`RF_POWER_ENABLE`	RF power enabled	0x1

8.1.3 SYSCTRL_RF_ACCESS_CFG

Register Name	Register Description	Address
`SYSCTRL_RF_ACCESS_CFG`	RF Access Configuration	0x40000054

Bit Field	Field Name	Description
1	`RF_IRQ_ACCESS`	RF IRQ access configuration
0	`RF_ACCESS`	RF access configuration

Field Name	Value Symbol	Value Description	Hex Value
RF_IRQ_ACCESS	RF_IRQ_ACCESS_DISABLE	RF IRQ access disabled	0x0*
	RF_IRQ_ACCESS_ENABLE	RF IRQ access enabled	0x1
RF_ACCESS	RF_ACCESS_DISABLE	RF access disabled	0x0*
	RF_ACCESS_ENABLE	RF access enabled	0x1

8.2 SYSTEM INTEGRATION

The RFFE implements the physical layer requirements of Bluetooth low energy technology, as accessed through the Bluetooth baseband hardware (see Chapter 9, "Bluetooth Low Energy Baseband" on page 201). It can also be used for a variety of standard (e.g., 802.15.4-based) protocols, proprietary protocols, and user- or ON Semiconductor-defined custom protocols.

IMPORTANT: When controlled by the Bluetooth baseband hardware, the RF front end is configured during BLE initialization, setting the contents of all registers in both register banks (with the exception of the `PA_PWR` register). If a non-standard configuration is required for a use case that also uses the Bluetooth baseband, any register updates must be applied after initialization of the Bluetooth baseband.

The `RF_REG00` `MODE` and `MODE2` sub-registers and configuration options effectively define how the RFFE functions. (refer to Section 8.5.1, "RF_REG00" for full register details.)

NOTE: There are restrictions on writing to registers (i.e., no bit-band access, access to internal bus, etc.).

The following are the various configurations for `MODE` and `MODE2` operations:

- `MODE_MODE`: select the working mode of the digital baseband.
    - 00: the digital baseband is off (no clock).
    - 01: the clock is generated but the blocks are reset (Tx, Rx, FIFOs and finite state machine (FSM)).
    - 10: the digital baseband is frozen.
    - 11: working

- MODE
    - `MODE_TX_NRX`: if set to 1, uses Tx, otherwise Rx.
    - `MODE_EN_SERIALIZER`: if set to 1, enables the serializer.
    - `MODE_EN_DESERIALIZER`: if set to 1, enables the deserializer.
    - `MODE_EN_FSM`: if set to 1, enables the radio FSM.
    - `MODE_NOT_TO_IDLE`: in FSM Mode, if set to 1, indicates to the FSM to go into Suspend Mode after a Tx or Rx packet.

- MODE 2
    - `MODE2_TESTMODE`: set the output Test Mode.
    - `MODE2_PSK_NFSK`: if set to 1, the PSK Mode is selected, FSK otherwise.
    - `MODE2_DIFF_CODING`: if set to 1, enables the differential coding/decoding.

8.2.1 48 MHz Crystal Oscillator

The `RF_XTAL_CTRL` register allows modifications to the crystal timing trim settings (from 43 to 341 μs), to bypass control algorithms, and provides configuration used to vary power consumption and control the oscillator.

IMPORTANT: When using the Bluetooth baseband or Bluetooth stack library, the default configuration of the 48 MHz crystal is required. For other configurations, the default configuration options for this register are recommended.

Additionally, the `ANALOG_INFO_CLK` register provides status information for the RFFE analog components, indicating when the various components of the oscillator are ready and providing information regarding the RF sub-band comparator outputs

For more information on integration and use of this clock by the rest of the system, see Section 6.2.2, “48 MHz Crystal Oscillator” on page 74.

IMPORTANT: 48 MHz crystal oscillator cannot be enabled unless the RFFE is powered and accessible.

NOTE: A delay of 1.3 μs is enforced at startup prior to any RF related block being accessible after enable.

The 48 MHz crystal oscillator can be trimmed by changing the `PLL_CTRL_XTAL_TRIM` bit-field from the `PLL_CTRL` register. This trim bit field is divided into 5 MSBs that provide coarse trimming, and 3 LSBs for fine trimming.

If the `XTAL_CTRL_BYPASS` bit in the `XTAL_CTRL` register is cleared, the crystal is automatically trimmed to 48 MHz using an iterative algorithm, with the trimming rate configured using the `XTAL_CTRL_XTAL_CKDIV` bit-field (larger divisors provide longer clock trimming periods for more averaging). If the `XTAL_CTRL_BYPASS` bit in the `XTAL_CTRL` register is set, the value set to the `PLL_CTRL_XTAL_TRIM` bit-field is used directly.

8.3 Packet Handling

The digital Tx and Rx contain a full packet handler. It has various features, including:

- Automatic preamble and pattern insertion
- Pattern detection
- Fixed and variable packet length with various tunings
- Automatic address insertion and checking
- Automatic full custom CRC insertion and checking
- Support for Multi-Frame Mode (preamble – pattern – data – CRC – data – CRC)

The RFFE has an internal SPI and 10 GPIOs; for more information see Section 11.9, “Support Interfaces” on page 346.

The RFFE has a number of GPIOs that can be configured to assist in monitoring IRQs and other signals while debugging protocol implementations that use the RFFE. The configuration of the GPIOs is specified by the `PAD_CONF_*_PAD_*_CONF` bit-fields from the `PAD_CONF_1` to `PAD_CONF_5` registers. The values of these fields are associated with the following functions, as shown in Table 10:

Table 10. Functions Associated with Register Values Configuring the GPIO

Value	Direction	Description
0000	In	Off. The GPIO is configured as an input, but the input value is not used.
0001	Out	IRQ_TX
0010	Out	IRQ_RXSTOP
0011	Out	IRQ_RECEIVED
0100	Out	IRQ_SYNC
0101	Out	IRQ_TXFIFO
0110	Out	IRQ_RXFIFO
0111	In	Tx start. The GPIO is configured as an input. A rising edge on this input notifies the FSM of a Tx start.
1000	Out	Rx data
1001	Out	Rx recovered data
1010	Out	Tx data
1011	Out	Tx clock
1100	Out	Digital clock

The RFFE chip has six IRQs that can be used to increase the usability of the chip. These are defined below:

RF_TX_IRQ

Interrupt is raised at the end of a packet transmission.
The IRQ is cleared by reading the `IRQ_STATUS` register.

RF_RXSTOP_IRQ

Interrupt is raised when the FSM stops the Rx Mode, independently of whether a packet has been received or not.
The IRQ is cleared by reading the `IRQ_STATUS` or the `DESER_STATUS` register.

RF_IRQ_RECEIVED_IRQ

Interrupt is raised when a packet is received and stored in the FIFO.
The IRQ is cleared by reading the `IRQ_STATUS` or the `DESER_STATUS` register.

RF_SYNC_IRQ

Interrupt is raised when the sync word is detected in Rx Mode.
The IRQ is cleared by reading the `IRQ_STATUS` or the `DESER_STATUS` register.

RF_TXFIFO_IRQ

Interrupt is raised when the `TXFIFO_NEAR_UNDERFLOW` is high.
Since the IRQ is tied to the “near underflow” flag of the FIFO, it can be cleared by filling the FIFO with enough data.

RF_RXFIFO_IRQ

Interrupt is raised when the `RXFIFO_NEAR_OVERFLOW` is high.
Since the IRQ is tied to the “near overflow” flag of the FIFO, it can be cleared by emptying the FIFO.

The IRQs can be activated by the IRQS_MASK field of the IRQ_CONF register. For example, the RF_RXSTOP_IRQn can be activated by setting bit field IRQS_MASK to 1. By default, the IRQs are active high, but this behavior can be switched by writing 1 to the IRQ_ACTIVE_LOW field of the IRQ_CONF register. The pad can also be configured to be in HIZ state when the IRQ is not active, by setting the IRQ_CONF_IRQ_HIGH_Z field of the same register to 1.

8.3.1 Packet Format

The packet handler supports several packet formats. Some of the possible packet formats are shown in Figure 12.

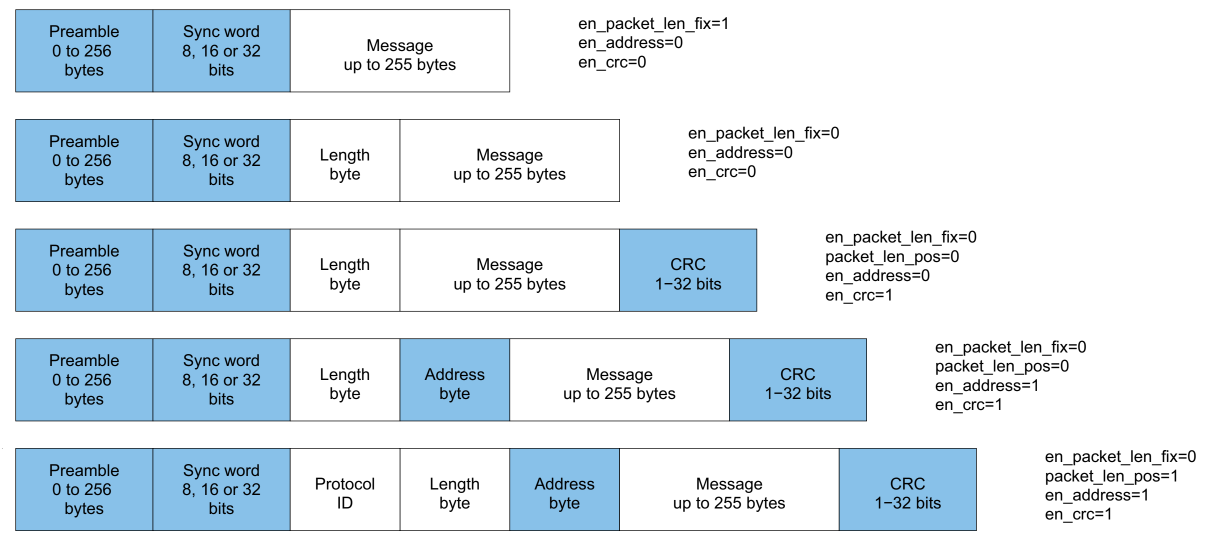


Figure 12. Various packet formats supported by the serializer. The bytes in blue can be inserted automatically.

8.3.1.1 Preamble

The preamble can be added automatically to the data, even if the packet structure is handled completely by the micro-controller. This feature can be turned on by simply setting the EN_PREAMBLE field of the PACKET_HANDLING register to 1. The length of the preamble in bytes is found in the PREAMBLE_LEN field of the PREAMBLE_LENGTH register increased by 1, and the preamble itself is located in the PREAMBLE register.

8.3.1.2 Pattern

The pattern (or synchronization word) is introduced automatically if the preamble is present. The length of the pattern is contained in the PATTERN_WORD_LEN field of the PACKET_EXTRA register. The pattern itself is found in the PATTERN registers. In the case of 8 or 16 bits, it is always the LSBs that are used. Pattern detection is enabled by setting the EN_PATTERN bit of the PACKET_HANDLING register. Pattern detection can accept some errors: this is useful with very short preambles, since the clock recovery is not yet complete. The maximum number of errors accepted in pattern recognition is located in the PATTERN_MAX_ERR field of the PACKET_EXTRA register.

8.3.1.3 Packet Length

If the EN_PACKET field of the PACKET_HANDLING register is set to 1, the packet structure is used, and so the serializer needs to know the length of the packet. This can be either fixed or variable. If the fixed format is chosen, the EN_PACKET_LEN_FIX field of the PACKET_LENGTH_OPTS register must be set to 1. In such a case, the length of the packet is found in the PACKET_LEN bit of the PACKET_LENGTH register. In the case of a variable packet length, this is normally specified as one of the first bytes of the packet. The PACKET_LEN_POS field of the PACKET_LENGTH_OPTS register specifies the position of this byte. If it is set to 0, this means that the first byte of the serializer contains the packet length; if it is set to 1, the second bite contains the packet length; and so on.

The packet length can be specified in several ways: for instance, it can take into account the CRC, or the packet length itself. The PACKET_LEN_CORR of the previous register contains the correction to apply to the packet length byte. The packet handler always considers the length of the packet, from the first byte after the packet length byte, until the last byte before the CRC. This field corrects the packet length. For example, if a standard protocol, such as Bluetooth low energy technology, considers that the CRC is taken into account in the packet length, a packet length of 5 means that there are 3 data bytes and 2 CRC bytes. So the PACKET_LEN_CORR has to be set to - 2.

Figure 13 is an example of how the packet length is handled.

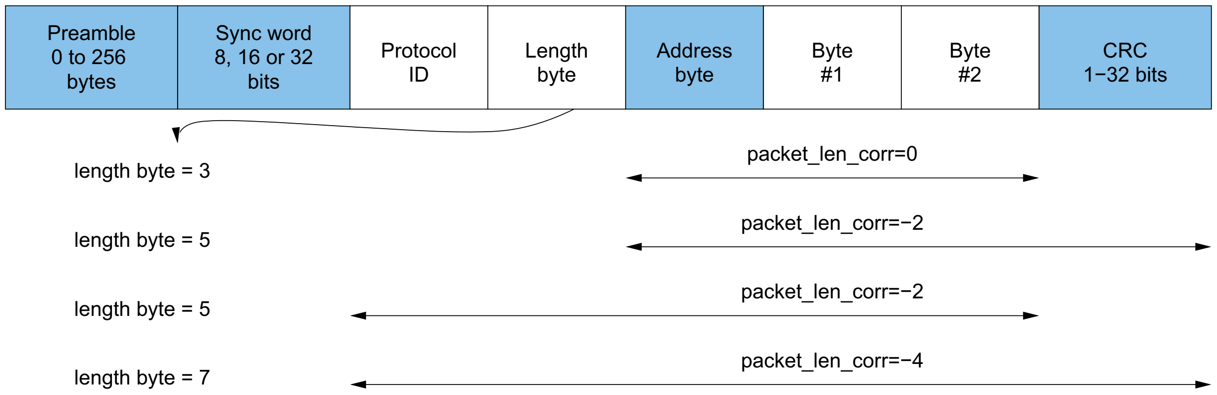


Figure 13. Examples of Packet Length Definition and the Associated Correction

In the case of a variable packet length, the PACKET_LENGTH register has another meaning: on the Rx side, this register can be used to specify the maximum packet length. If a protocol supports only packets with a maximum of 64 bytes, this register can be set to 64. If a received packet has a length greater than 64, a packet length error is generated by the deserializer.

8.3.1.4 Address

An address can be inserted automatically after the packet length, if the ADDRESS_TX field of the ADDRESS_CONF_EN register is set to 1. The address is given by the ADDRESS bit of the ADDRESS register. On the Rx side, if the ADDRESS_CONF_EN_ADDRESS_RX field is set to 1, an address comparison is made. If the addresses do not match, an address error is generated by the deserializer. Moreover, a broadcast address can be specified in the ADDRESS_BR bit of the ADDRESS_BROADCAST register. The Rx broadcast address reception can be enabled by setting the ADDRESS_RX_BR of the ADDRESS_CONF_EN register to 1; in such a case, the RS accepts the normal address and the broadcast address during reception. Confirmation of broadcast address reception is found in the IS_ADDRESS_BR field of the RF_DESER_STATUS register.

The address length can be 8 or 16 bits depending on the value of the ADDRESS_LEN of the ADDRESS_CONF register.

8.3.1.5 Multi-Frame

If the MULTI_FRAME bit of the PACKET_HANDLING_EN register is set to 1, Multi-Frame Mode is enabled. (A frame is composed of the data and the corresponding CRC.) In this mode, a preamble and a single synchronization word are followed by multiple frames. As long as the MULTI_FRAME bit is set to 1, Multi-Frame Mode is enabled.

8.3.2 CRC

The CRC is a hash function of the data, used to detect errors during transmission. The value of the CRC is added at the end of the packet. Errors during transmission are detected by a difference between the calculated CRC and the received one. The CRCs are generally specified by a polynomial.

The digital baseband can calculate the CRC on the fly, and insert it at the end of the packet. The CRC polynomial is programmable, and it can have a length from 1 to 32 bits. The length of the polynomial is specified by the polynomial itself. The CRC polynomial is contained in the `RF_CRC_POLYNOMIAL` register. The polynomial is represented in Koopman notation: the nth bit specifies the (n+1) order; the order 0 (the 1 at the end of the polynomial) is ignored. Some examples:

Table 11. CRC-CCITT Algorithm Parameters

CRC Parameter	Parameter Value
Order	16
Polynomial	$x^{16} + x^{12} + x^5 + 1$
Polynomial (hex)	0x1021
Initial Value (hex)	0xFFFF
Final XOR Value (hex)	0x0000

Table 12. CRC16 Algorithm Parameters

CRC Parameter	Parameter Value
Order	16
Polynomial	$x^{16} + x^{15} + x^2 + 1$
Polynomial (hex)	0x8005
Initial Value (hex)	0xFFFF
Final XOR Value (hex)	0x0000

Table 13. Bluetooth CRC24 Algorithm Parameters

CRC Parameter	Parameter Value
Order	24
Polynomial	$x^{24} + x^{10} + x^9 + x^6 + x^4 + x^3 + x + 1$
Polynomial (hex)	0x00065B
Initial Value (hex)	0x555555 or protocol defined
Final XOR Value (hex)	0x000000

This hardware CRC implementation works on the serialized stream, so the bit order depends on the `PACKET_HANDLING_LSB_FIRST` value. At the insertion of the CRC, the value of the CRC is simply shifted out.

The start value of the CRC register is contained in the `RF_CRC_RST` register.

CRC calculation, insertion, and validation are performed automatically if the `EN_CRC` field of the `PACKET_HANDLING_EN` register is set to 1.

The CRC calculation of the packet length value can be controlled through the CRC_ON_PKTLEN field of the same register. This is useful for the standards in which the CRC is on the MAC layer – for example, the IEEE 802.15.4 standard.

8.3.3 Accessing the FIFOs

The two FIFOs' data is accessible via the RF_TXFIFO and RF_RXFIFO registers. This access can be achieved in burst mode without having to manually increment addresses. A write to the RF_TXFIFO register corresponds to a push, while a read of the RF_RXFIFO register corresponds to a pop. Reading the RF_TXFIFO register is possible, but this results in no action on the FIFO (no pop implied). Writing to the RF_RXFIFO register is not possible.

8.3.4 FIFO Status

The status of each FIFO can be read in the TXFIFO_STATUS_BIST and RXFIFO_STATUS_BIST registers. There are indications regarding overflows, underflows, or if the FIFO is empty or full. The NEAR_UNDERFLOW and NEAR_OVERFLOW fields are controlled by the FIFO_FIFO_THR value of the FIFO register for the Rx, and the FIFO_THR_TX value of the FIFO2 register of the Tx. Table 14 on page 140 gives the thresholds for the value of FIFO_FIFO_THR.

Table 14. Available FIFO Thresholds

fifo_thr	Near Underflow Threshold	Near Overflow Threshold
000	8	120
001	24	104
010	40	88
011	64	72
100	72	64
101	88	40
110	104	24
111	120	8

The FIFOs can be flushed at any time by setting bit 0 of their respective status registers to 1. Alternatively, the FIFOs can be flushed at the beginning of a reception (Rx case), or at the end of a transmission (Tx case).

During the reception of a message, many events can occur: for instance, a CRC error, or a packet length error. In all cases, the data must be stored in the FIFO, at least temporarily. If an event occurs, there are two choices: keep the data in the FIFO and let the external controller examine the content, or simply flush the received data. To avoid a situation in which the user starts to look at the FIFO's content before the end of the packet, it is important that the FIFO status is only updated at the end of the packet.

The Rx FIFO supports the above two choices. The automatic flush is controlled by the FIFO_FLUSH_ON_ADDR_ERR, FIFO_FLUSH_ON_PL_ERR, FIFO_FLUSH_ON_CRC_ERR, and FIFO_FLUSH_ON_OVFLW fields of the FIFO register. The RX_FIFO_ACK field of the same register is responsible for choosing the behavior of the FIFO status. If it is set to 1, the packet has to be received correctly before updating the FIFO status.

8.4 Modulator and Radio Configuration

Table 15 contains the supported encoding and decoding options, in order:

Table 15. Supported Encoding and Decoding Options

Option	Description
Data whitening	The data can be whitened by setting the `EN_DATAWHITE` bit of the `CODING` register. The whitening sequence is a PN9. Whitening is used to avoid long sequences of 0 or 1. In the case of the Bluetooth low energy technology standard, the LFSR used is a Galois LFSR7 with a specific reset status. This particular LFSR can be activated by setting the `DW_BTLE` bit of the `DATAWHITE_BTLE` register; note that the `EN_DATAWHITE` bit field of the `CODING` register also needs to be set. On the same register, the `DW_BTLE_RST` field specifies the reset status of the LFSR.
Manchester	Manchester encoding is available through the `EN_MANCHESTER` bit of the `CODING` register. The code is the same as that used in the IEEE 802.3 standard: a 1 is coded by a rising edge (01) and a 0 is coded by a falling edge (10). NOTE: This can be inverted by using the `BIT_INVERT` configuration.
IEEE 802.15.4 Bit to Chip	The IEEE 802.15.4 standard specifies a conversion from a sequence of 4 bits to a transmitted sequence of 32 chips. This conversion can be activated by setting the `EN_802154_B2C` bit of the `CODING` register to 1. NOTE: On the Rx side when this bit is set, the chip sequence synchronization is made on sequences of 0000: this is mainly due to the fact that on the IEEE 802.15.4 standard, the preamble is composed by this sequence. However, pattern recognition is working transparently. For further information, see the IEEE Computer Society publication Part 15.4 of Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs).
Linear to Frequency	The IEEE 802.15.4 standard specifies an O-QPSK modulation. This can be considered linear, since the In phase and Quadrature phases are encoded directly. However, on the RFFE chip, there is a direct modulation, meaning that the frequency is encoded directly. To maintain the linear code and to be able to modulate in the frequency domain, a linear to frequency coding is available, and can be activated by setting the `EN_802154_L2F` bit of the `CODING` register to 1. NOTE: On the Rx side, a phase ambiguity can arise: to get rid of it, the Rx correlators work on the frequency spreading sequences, and not on the phase sequences.
Bit Inversion	When the bit `BIT_INVERT` of the `CODING` register is set to 1, the encoding stream and the decoding stream are inverted.
Bit Order in Quadrature Modulations	The bit order in quadrature modulation (especially O-QPSK) can be determined by the `EVEN_BEFORE_ODD`, `OFFSET`, and `I_NQ_DELAYED` bits of the `CODING` modulation.
Differential Encoding	This can be activated by setting the `DIFF_CODING` of the `MODE2` register. This encoding is not available for every coding option. However, on the Rx side, every coding option (especially the 2 bits/symbol modulation) is available.

8.4.1 Radio Configuration

8.4.1.1 Data Rate

The symbol rate is specified indirectly through the `_DR_M_` field of the `MOD_INFO` register. The `_DR_M_` field specifies the oversampling ratio frequency, but since the oversampling is fixed to 8, it also specifies the symbol rate. It can be calculated using the following equation, where f_{sys} is the system frequency – that is, 16 MHz or 24 MHz.:

$$dr\_m = \frac{f_{sys}}{8DR} - 1$$

8.4.1.1.1 Fractional Data Rate

To increase the number of possible data rates, it is possible to have a fractional data rate, which is a data-rate that is a fraction of the actual data rate. The effective data rate is given by the following equation, where N and D are the numerator and denominator specified in the `TX_FRAC_CONF` and `RX_FRAC_CONF` registers.:

$$\text{dr\_eff} = \frac{N}{D}\text{dr\_m}$$

The fractional data rate is achieved through interpolation.

In Tx, this interpolation adds a gain on the signal, meaning that the modulation index changes. The amplitude is multiplied by the following, where the `TX_FRAC_GAIN` bit is specified in the register `FRAC_CONF`:

$$G = \frac{D}{2^{3 + \text{tx\_frac\_gain}}}$$

So if D = 10 and `TX_FRAC_GAIN` is 0, the gain is equal to 10/8 = 1.25.

For the Rx, there is also an amplitude gain due to the fractional data rate. The gain is given by the same calculation as is used in Tx. Since the fractional data-rate block is located at the input of the demodulator, this has to be taken into account when calculating the amplitude of the signal at the output of the matched filter (see Section 8.4.3.4, “Matched Filtering” on page 153).

8.4.1.2 Central Frequency

The central frequency can be calculated using the following equation, where f_{RF} is the central RF frequency, and f_{refTx} is the reference frequency – that is, f_{refTx} = 144 MHz:

$$\text{center\_frequency} = \frac{f_{RF} \times 2^{19(21)}}{f_{refTx}}$$

The RFFE chip has the option of having two different clock references, depending on the operational mode: Tx or Rx. The Tx reference clock is five times larger than the Rx clock. So to have the same frequency in Rx and Tx, the digital central frequency needs to be changed. However, the digital block has the capability of switching automatically between the Tx value and the Rx value. To switch this feature on, the `ADAPT_CFREQ` bit of the `CENTER_FREQ` register must be set to 1.

NOTE: The meaning of the `CENTER_FREQUENCY` field changes if automatic adaptation is turned on. When it is set to 0, its meaning is that specified in the previous equation.

If it is set to 1, it is specified in the following formula, where f_{RF} is the RF frequency, and f_{refTx} is the reference frequency in Tx Mode:

$$\text{center\_frequency} = \frac{f_{RF} \times 2^{21}}{f_{refTx}}$$

The corresponding frequencies are obtained by dividing the value by 4 for the Tx Mode, and by adding this value to the same value divided by 4 in Rx Mode.

8.4.1.3 Channels

In the RFFE it is possible to work with channels.This means that only a base frequency and the channel spacing need to be specified; then the user needs only to specify the channel wanted. The advantage of this is that channel specification requires only one register access, while the central frequency specification requires four register accesses.

This function is activated by setting the `EN_CHANNEL_SEL` bit of the `CHANNELS_2` register to 1. The channel spacing is specified in the `CHANNEL_SPACING` field of the `CHANNELS_1` register. The value in this field is given by the following formula, where f_{sp} is the channel spacing:

$$\text{CHANNEL\_SPACING} = \frac{f_{SP} \times 2^{25}}{f_{refTx}}$$

If channel 4 is wanted (that means $4 \times f_{sp}$ from the central frequency), the value 0x4 must be written to the `CHANNEL` register.

8.4.1.4 On/Off Timing

Each analog block can take a certain amount of time to switch on. Therefore, the transitional states need to be maintained for the right amount of time, to avoid—for example—sending a message when the PLL is not yet stabilized. The time needed by each category of blocks can be specified in the `TIMINGS_1` to `TIMINGS_5` registers. The timings of the categories are specified by an integer value. The corresponding time is given by this value increased by 1, multiplied by the time granularity. This is specified by the `T_GRANULARITY_TX(3:0)` and `T_GRANULARITY_RX(3:0)` fields of the `TIMINGS_1` register. The time granularity is given by:

$$\text{T\_GRAN\_TX} = 2^{\text{T\_GRANULARITY\_TX} - 2}$$

$$\text{T\_GRAN\_RX} = 2^{\text{T\_GRANULARITY\_RX} - 2}$$

Therefore, the timings can range from 0.25 μs to 262.1 ms, a range of sufficient size for the purpose.

8.4.1.4.1 Tx On/Off Timing

The timing from the idle state to Tx On is calculated in this way:

$$(\max(\text{T\_PLL\_TX} + \text{TX\_SUBBAND}, \text{T\_DLL} + \text{TX\_SUBBAND}, \text{T\_TX\_RF}) + 1) * \text{T\_GRAN\_TX}$$

Here is an example: a system is configured with the following values:

- `T_GRANULARITY_TX` = 2
- `T_SUBBAND_TX` = 4
- `T_TX_RF` = 0
- `T_DLL` = 4
- `T_PLL_TX` = 2

If the FSM is activated with the Activate Tx Only command, which means without performing subband selection, then during the startup sequence of the Tx, the FSM timer is set to 4; this is the maximum value between `T_TX_RF`, `T_PLL_TX`, and `T_DLL`, meaning Tx will be activated after 5 μs.

If subband selection is performed during the activation, the `T_PLL_TX` and `T_DLL` values are increased by the value of `T_SUBBAND_TX`. As result, the internal timer starts with a value of 8, which is the maximum value between `T_PLL_TX` + `T_SUBBAND_TX`, `T_DLL` + `T_SUBBAND_TX`, and `T_TX_RF`.

Tx Off time is equivalent to Tx ramp-down from the instant that the FSM state is set to idle.

8.4.1.4.2 Rx On/Off Timing

The timing from the idle state to Rx On is calculated in this way:

(max (T_PLL_RX + RX_SUBBAND, T_RX_RF, T_RX_BB) + 1) * T_GRAN_RX

The granularity is 4 μs and in this case the maximum value is 4, so the internal timer is set to 4.

NOTE: Since the granularity of the counter is 4 μs, this whole sequence takes 20 μs.

NOTE: This is only the power-up sequence for the Rx; if the required data is at the antenna at the exact moment of the rising edge of the digital enable signal, the data is only made available after the Rx processing delay. If subband selection is activated, the behavior is similar to the Tx case; the subband time is added to the PLL power-up time only.

Rx Off time happens immediately when the FSM state changes to idle.

8.4.1.4.3 Power Amplifier (PA) Power-Up

To reduce the spectral regrowth of the PA during power-up, a ramp-up and a ramp-down can be activated. The ramp-up works on the power back-off of the PA. The PA ramp-up is activated by setting the `EN_PA_RAMPUP` bit of the `PA_RAMPUP` register to 1. The ramp-down is activated by setting the `EN_PA_RAMPDOWN` bit of the same register to 0. Note that the ramp-down only works if the ramp-up is activated. The ramp-up does not start with the activation of the PA. If the ramp-up is enabled, the PA back-off is set to the minimum as soon as the activation command reaches the PA. Then a variable counter controlled by `DEL_PA_RAMPUP(2:0)` waits for the PA to be on. The values of the delay are:

- 0b000: 0.25 μs
- 0b001: 0.31 μs
- 0b010: 0.5 μs
- 0b011: 0.81 μs
- 0b100: 1.5 μs
- 0b101: 2.1 μs
- 0b110: 2.8 μs
- 0b111: 4.1 μs

The steepness of the ramp-up is controlled by the `TAU_PA_RAMPUP(1:0)` parameter. The ramp-up duration depends on the final value of the PA back-off (in the `PA_CONF` register). In the case of the maximum final value of the PA back off and starting from step 0, the conversion table is as follows:

- 0b00: 6.0 μs
- 0b01: 3.0 μs
- 0b10: 2.0 μs
- 0b11: 1.5 μs

The PA ramp-down takes exactly the same amount of time.

For Bluetooth Low Energy configuration, subband selection is used in Tx but not in Rx. Tx On time takes 30 μs and Rx On time takes 4 μs. Tx ramp-up and ramp-down times are equal to 2 μs for maximum PA final value.

8.4.2 Tx Specific Configuration

8.4.2.1 Pulse Shape

The pulse shape is specified through the `TX_PULSE_SHAPE` registers. The pulse shape is composed as follows:

```
coef_1, coef_2, ... coef_14, coef_15, coef_15, coef_14, ..., coef_2, coef_1
```

The over-sampling is set to 8; hence, the pulse shape is four symbols long.

If the `PULSE_NSYM` bit of the `MOD_TX` register is set to 1, the second half of the pulse shape is inverted.

NOTE: The modulation is obtained by converting both the convolution of the pulse shape and the data-stream into a series of pulses (not rectangles). In the case of a GFSK modulation, the specified pulse shape is not the impulse response of an exponential filter, but the convolution of this response with a rectangle that is 1 symbol long.

8.4.2.2 Modulation Index

The modulation index, h, is specified by the following equation, where Δf is the frequency deviation from the central frequency, and DR is the data rate:

$$h = \frac{2 \times \Delta f}{DR}$$

After the pulse shaper, the data is multiplied by a specified factor M, and then added to the central frequency. If a series of 1 is assumed, the output of the pulse shape has an output of Q. The modulation index can be rewritten as:

$$h = \frac{2 \times M \times Q \times f_{refTx}}{DR \times 2^{19}}$$

The factor M is specified through mantissa (man) and exponent (exp), by the formula:

$$M = \left(1 + \frac{man}{16}\right)2^{exp}$$

Here there are two ways of choosing the modulation index:

- Fix the pulse shape and adapt the multiplication factor to achieve the desired modulation index.
- Fix the multiplication factor and adapt the pulse shape.

The second method is not optimal. In fact, it can occur that the exponent part of the multiplicative factor has to be used. Moreover, there is a loss of granularity if this method is used.

The exponent and the mantissa value can be specified in the `TX_MULT` register (`TX_MULT_EXP` and `TX_MULT_MAN` fields).

NOTE: If the interpolator is used, the interpolation gain has to be taken into account.

Example: an MSK modulation at 800 kbps.

A rectangular pulse shape is chosen with a value of 120 (coef_0,.., coef_11 = 0, coef_12...coef_15=120). The value Q is equal to 120: The modulation index for an MSK modulation is 0.5. The multiplicative factor M is:

$$M = \frac{h \times DR \times 2^{19}}{2Q \times f_{refTx}} = 7.282$$

This value must be split into mantissa and exponent. The exponent is the floor of the $\log_2$ of M, which is 2. The resulting mantissa is 13.

$$M_{\alpha} = \left(1 + \frac{13}{16}\right)2^{2} = 7.25$$

In the case of a 4-FSK modulation, the definition of index modulation must be considered to be the same, but this results in the additional deviations being at +3 and -3 times the nominal deviation. So, for example, if a 2-FSK configuration is defined for ±250 kHz, it results in the 4FSK version also having ±750 kHz as frequency deviations.

8.4.2.3 Interpolator

At the end of the Tx chain there is an interpolator. Its main purpose is to avoid quantization steps when the Tx is working with low data rates. In fact, a quantization step can result in a wider spectrum. The interpolator is a simple first order cascaded integrator-comb (CIC) interpolator. The input clock is the 8x symbol frequency. The output frequency f_{out} is specified by the ck_tx_m(4:0) of the `MOD_TX` register. Its definition is:

$$f_{out} = \frac{f_{in}}{ck\_tx\_m + 1}$$

The interpolator is enabled using the `EN_INTERP` bit of the same register. If the interpolator is disabled, the output signal is re-sampled at the f_{out} frequency.

NOTE: It is preferable that the f_{out} frequency is an integer multiple of the f_{in} frequency (8x symbol rate).

8.4.2.4 Power Amplifier (PA) Source

The output power provided by the PA is tunable using the `PA_PWR` field of the `PA_PWR` register. The field is composed of the back-off control (`PA_PWR`) and a control on the PA common mode. The maximum value of `PA_PWR` is 12 (or 0xc).

NOTE: The back-off value is also used by the PA ramp-up algorithm. The value at the end of the ramp-up is the actual value in the register.

The bias current PA backoff bias (`IQ_RXTX_1` in register `BIAS_0`) allows the slope of the back-off curve to be changed. However, this generates a discontinuity on the curve between maximum output power (`PA_PWR`=12) and the first back-off step (`PA_PWR`=11)

When the bit `PA_PWR`(4) = 1, additional (but discontinuous) power reduction steps can be configured. Power levels are -30dBm for -1 (`PA_PWR`(4:0) = 11111), and -40dBm for -3 (`PA_PWR`(4:0) = 11101), the step -2 (`PA_PWR`(4:0) = 11110) being very much dependent on the PA backoff bias (`IQ_RXTX_1`(3:0)), in the range from -26dto -40dBm.

NOTE: The `PA_PWR`(5) does not affect output power, but lengthens the PA power ramp-up sequence to provide less a lower transient draw from the supply.

8.4.3 Rx Specific Configuration

8.4.3.1 Channel Filter Configuration

The channel filter is not a digital block, but is digitally configurable, and its configurations might affect the digital baseband fine tuning. It is a polyphase filter, so its transfer function is not symmetrical; therefore, it rejects the image. Its central frequency and its bandwidth can be configured with three parameters. Both central frequency and bandwidth are tunable via the bias of the transconductance (Gm) of the filter. The bias is also tunable: it is generated by a switched capacitor PTAT, and the frequency of the switched caps can be changed. The frequency is defined by the following equation, where f_{sys} is the system frequency, and so it is either 16 MHz or 24 MHz, and K_f is the value of the field `DIV_FILT` of the register `CLK_CH_FILTER`:

$$f_{sc} = \frac{f_{sys}}{1 + K_f}$$

The bias of the filter can be tuned via the `IQ_FI_BW` and `IQ_FI_FC` fields of the register `FILTER_BIAS`.

An approximated configuration of the filter can be made by using the following equations:

$$f_c = \frac{f_{sc}}{2\text{MHz}}(238 + 93.6 \times \text{iq\_fi\_fc})[\text{kHz}]$$

$$bw = \frac{f_{sc}}{2\text{MHz}}(178 + 63.5 \times \text{iq\_fi\_bw})[\text{kHz}]$$

For example, in Bluetooth low energy technology, `DIV_FILT`=7, `IQ_FI_FC`=8, and `IQ_FI_BW`=14, so:

$$f_c = \frac{\frac{16\text{MHz}}{8}}{2\text{MHz}}(238 + 93.6 \times 8) = 986.8\text{kHz}$$

$$bw = \frac{\frac{16\text{MHz}}{8}}{2\text{MHz}}(178 + 63.5 \times 14) = 1067\text{kHz}$$

8.4.3.2 Phase and RSSI Fractional Decimation

The purpose of the decimation blocks is to change the sampling frequency of the signal from the analog and digital front end to the demodulation blocks; these work with a constant oversampling ratio, while the front end is fully configurable.

In the case of an analog baseband, there are some issues regarding the analog baseband blocks. In fact, the intermodulation frequency needs to be kept high enough to avoid pulling. Moreover, some modulations require a larger bandwidth of the channel filter: this can be achieved only by increasing the clock frequency of the channel filter and the phase ADC.

Resampling is realized with a fractional decimator. (It is supposed that the front end sampling frequency is always higher than the demodulator.) Fractional decimation is realized through an interpolator followed by a decimator.

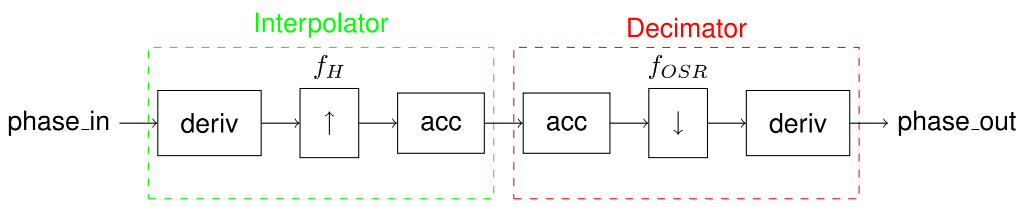


Figure 14. Simplified Block Diagram of the Resampler Block for the Phase

Note that in Figure 14, acc stands for “accumulator”, and deriv stands for “derivator”. While the RSSI can be resampled without any major problems, there might be an issue with the phase with this configuration if the signals are not handled correctly. Since it can have a gain that is not a power of 2, the periodicity of the phase cannot be respected. Moreover, because of the implicit filtering, there can be errors when the phase rolls over. This is not the case for the first interpolator, since the first derivation gives the frequency, which has no rollover. The accumulator generates the phase correctly, since in the accumulator the saturation implicitly recreates a good phase. The chosen solution to this issue is to consider the signal to be a frequency, then perform a second order CIC decimator, without the second differentiator. The resulting signal is simply the frequency without a differentiator, and so it is the phase.

There are several parameters that control the phase and RSSI decimation: `EN_RESAMPLE_PH`, `EN_RESAMPLE_RSSI`, and `DIV_PHADC` from the `FRONTEND` register; `RESAMPLE_RSSI_G1`, `RESAMPLE_RSSI_G2`, and `RESAMPLE_PH_GAIN` from the `FRONTEND2` register; and `RESAMPLE_PH_IF` from the `RX_IF` register.

The incoming signals are clocked at the frequency given by the `DIV_PHADC` field. At the first stage they are upsampled at the f_{sys} frequency, giving a gain of `DIV_PHADC`+1. At the second stage, the gain is given by the ratio between f_{sys} and the oversampled frequency, and is equal to `DR_M`+1. These gains have to be compensated for, at least to avoid overflows.

Since the phase is converted in frequency and the signal is at an IF, a DC value is present that is amplified during the gain stages. It is interesting to cancel this DC value directly, which can be done using the `RESAMPLE_PH_IF` field. The value of this field is given by the following equation, where f_{IF} is the IF frequency, and f_R is the frequency at the input of the decimation block:

$$\text{resample\_ph\_if} = \frac{16f_{IF}}{f_R}$$

The phase then has two gain stages, due to the presence of the interpolator and the decimator. The first gain is given by the ratio between the maximum frequency of the baseband – that is to say, 16MHz or 24MHz – and the phADC frequency. This gain is equivalent to:

$$G_1 = \frac{f_{sys}}{f_R}$$

The second gain stage is due to the presence of the decimator, and is equivalent to the ratio between the maximum baseband frequency and the oversampled frequency – that is to say, 8x the data rate. This gain is given by:

$$G_2 = \frac{f_{sys}}{f_{OSR}}$$

These gain are compensated for, through the RESAMPLE_PH_GAIN: this variable is an unsigned word. The gain is given by:

$$G_c = 2^{\text{resample\_ph\_gain} - 7}$$

On the RSSI side, an additional gain is added after the interpolator. This gain is given by $2^{-\text{resample\_rssi\_g1}}$. A second gain is placed after the decimator, and its value is given by $2^{-\text{resample\_rssi\_g2}}$.

8.4.3.3 Carrier Recovery

8.4.3.3.1 Rough Carrier Recovery

The rough carrier recovery is a simple algorithm that tries to fix the signal frequency average to 0. In the case of an FSK, this corresponds to a threshold determination. The time constant of this algorithm is specified in the TAU_ROUGH_RECOV register. Rough carrier recovery is enabled by setting the EN_ROUGH_RECOV bit of the CARRIER_RECOVERY register to 1.

In this block, the IF is also canceled. The IF is specified in the IF2_CLK_OS field of the RX_IF register. This field is given by the following equation,

where f_{IF} is the intermediate frequency and f_{sym} is the symbol rate:

$$\text{if2\_clk\_os} = \frac{128 f_{IF}}{f_{sym}}$$

8.4.3.3.2 Fine Carrier Recovery

The fine carrier recovery algorithm uses the decision made on the stream to estimate the carrier error and to fix it. In practice, the decision is converted in amplitude and compared to the actual amplitude. The conversion is made through the FSK_FCR_AMP registers.

In the case of an FSK modulation with 1 bit per symbol, the three values are used to recreate the ISI, so three bits of the decision output are used: one corresponding to the present state, one for the previous, and one for the next. So the recreated signal has a value of FSK_FCR_AMP3 in the case of a sequence [1;1;1], FSK_FCR_AMP2 for [1;1;0] or [0;1;1], and FSK_FCR_AMP1 for [0;1;0]. Respectively, it is -FSK_FCR_AMP3 for [0;0;0], -FSK_FCR_AMP2 for [0;0;1] or [1;0;0], and -FSK_FCR_AMP1 for [1;0;1].

In the case of a 4-FSK, the mapping is done with the decision values of the I and Q signals. When I and Q are equal to [0;0] the recreated signal is -FSK_FCR_AMP1. It is FSK_FCR_AMP1 when I and Q are [0;1], - FSK_FCR_AMP3 if I and Q are [1;0], and finally FSK_FCR_AMP3 with [1;1]. This configuration can be changed by changing the FOURFSK_CODING register.

If rough carrier recovery is far from being completed, the +Δf and the – Δf might be on the same side. Hence, the decision block only sees a sequence of 0s or 1s. In such a case, the fine carrier recovery works against the correct center frequency by trying to put the – Δf to match the +Δf, which the fine carrier recovery sees as 1s. For this reason, fine carrier recovery is only applied once the pattern has been detected.

The time constant of the fine carrier recovery block is found in the TAU_PHASE_RECOV register. The block is enabled by setting the EN_FINE_RECOV bit of the CARRIER_RECOVERY register.

8.4.3.3.3 Carrier Recovery Boundaries

The carrier recovery block can recover the carrier in a specified range. Theoretically the range is expected to be as wide as possible, but there are some limitations. First of all, when there is no signal at the input, the block tries to recover a carrier from the noise. Since the noise is usually white noise, the average is generally null; but this is not always true, especially in the presence of interferers, or noise injected from the digital blocks into the analog path. In these cases, the carrier recovery diverges, and in the presence of a signal it is not able to recover the carrier in time, if the preamble is short. For this reason, a boundary for carrier recovery can be specified through the mantissa `FREQ_LIMIT_MAN` and exponent `FREQ_LIMIT_EXP` fields of the `CARRIER_RECOVERY_EXTRA` register. The boundary of the carrier recovery is given by the following formula, where ***m*** is the mantissa, ***e*** the exponent, and ***f_{sym}*** the symbol frequency:

$$f_l = \frac{3\left(1 + \frac{m}{8}\right)2^e f_{sym}}{2^4}$$

The mantissa needs to be specified as an unsigned value, while the exponent is signed. The carrier is searched for in the range:

$$[f_{IF} - f_l : f_{IF} + f_l]$$

In practice, to calculate these values, this set of equations is used:

$$f_l = \frac{3Kf_{sym}}{2^4}$$

$$K = \left(1 + \frac{m}{8}\right)2^e$$

Example

In the Bluetooth low energy technology standard, the carrier precision is given by ±150 kHz. This means that f_l = 150 kHz. The symbol rate in Bluetooth low energy technology is the same as the bit rate, 1 Mbps. So the first equation is:

$$150 \times 10^3 = \frac{3 \times K \times 1 \times 10^6}{16}$$

$$K = 0{,}8$$

Now 0.8 must be expressed with mantissa and exponent.

$$\left(1 + \frac{m}{8}\right)2^e = 0{,}8$$

It is easy to calculate that the closest values are *e* = -1 and *m* = 5: these values give K = 0.8125, and a carrier recovery range of ±152 kHz. So freq_lim_man(2:0) = 0b101 and freq_lim_exp(2:0) = 0b111 (the prefix 0b means a binary representation).

8.4.3.3.4 RSSI Detection

The previous algorithms work well on a continuous stream. However, in Packet Mode, when the radio is activated and there is no signal, the noise at the output of the phADC is not white, and carrier recovery is perturbed. To avoid this situation, detection is made on the RSSI to estimate the packet's arrival. RSSI detection can be made on the absolute value of the RSSI, or on the differential value, or both. Differential detection is activated by setting the `RSSI_DET_EN_DIFF` bit of the register `RSSI_DETECT`, while absolute detection is activated using the `RSSI_DET_EN_ABS` bit of the same register. The thresholds are specified in the `RSSI_DETECT_DIFF_THR` and `RSSI_DETECT_ABS_THR` registers, respectively. If both absolute and differential are activated, the RSSI is detected if the differential value is higher than the threshold, and the absolute value is also higher than its respective threshold. The detection is made on the filtered and corrected RSSI value, so the speed is controlled by the `TAU_RSSI_FILTERING` value. If the RSSI filtering is too low, and too much noise is still present on the RSSI value, an additional filtering can be applied by setting the `RSSI_DET_FILT` bit of the register `DEMOD_CTRL` to 1. This additional filtering is equivalent to a 4-tap FIR with all taps set to 1.

The differential RSSI is not just the simple derivative of the RSSI; because of its structure, it might miss some ramp-up. The differential RSSI is the output of an FIR with the following transfer function, where τ is equal to 1, 2, 4, or 6, depending on the value of `RSSI_DET_DIFF_LL` of the `RSSI_DETECT` register:

$$Hz(z) = 1 - z^{-\tau}$$

The RSSI detection is fed to a state machine that controls the status of the carrier recovery and other blocks. The detection can be sent directly or can be delayed: the delay is controlled by the `RSSI_DET_WAIT` field of the `RSSI_DETECT` register. The delays are:

0b00	no delay
0b01	2 symbol delay
0b10	4 symbol delay
0b11	8 symbol delay

As soon as the state machine receives the RSSI detection, a series of tasks can be launched.

Reset and Slow-down

Once the RSSI ramp is detected, the carrier recovery algorithm is reset, and the Starter Mode is set to 0 – that is to say, carrier recovery slows down. The slow-down lasts for the time necessary to get the sync word read on the delayed path, and is calculated automatically (with some margin).

Carrier Offset Estimation

This is always performed; the only way to block it is to disable RSSI detection. Carrier offset estimation is carried out by accumulating the actual frequency for a variable number of samples. The number of samples is chosen using the `RSSI_DET_CR_LEN` field of the `RSSI_DETECT` register. The available values are:

- 0b00: 32 samples -> 4 symbols
- 0b01: 64 samples -> 8 symbols
- 0b10: 128 samples -> 16 symbols
- 0b11: 256 samples -> 32 symbols

This system is supposed to work with an 8-bit preamble, so the first two cases correspond to half of, and the entirety of, the preamble, respectively. The other two cases will average on the sync word too; in order to get rid of a biased sync word, sync word bias compensation should be switched on.

Sync Word Bias Compensation

After the RSSI ramp is detected, the state machine will consider 8 bits of preamble, followed by the sync word. The sync word typically has an average of 0. However, since the sync word can be chosen arbitrarily, there may be a bias on the average. In order to compensate for this bias, the state machine will correct the carrier estimation by reading the content of the `RF_SYNC_PATTERN` register; this clearly only applies to estimations of 16 symbols or 32 symbols. Sync word bias compensation can be activated by setting the `EN_SYNC_WORD_CORR` bit of the `SYNC_WORD_CORR` register to 1. The amplitude of the compensation is controlled by the `SYNC_WORD_BIAS` field and essentially depends on the modulation index. For a modulation index of 0.5, the value 0x4 should be applied.

Early Fine Recovery

Normally, fine recovery is only activated after sync word detection. If the RSSI ramp is detected and carrier recovery is estimated correctly, fine recovery can be turned on earlier. In order to do so, the `EARLY_FINE_RECOV` bit of the `DEMOD_CTRL` register is set to 1.

Enable Pre-Sync Word Detection

The sync word is normally detected only on the delayed path. However, there may be an opportunity to detect the sync word, or at least the end part of it, on the non-delayed path as well. If this is the case, the sync word detection on the delayed path should arrive with a deterministic delay, so the state machine will know precisely how long it has to slow down the system. This functionality is activated by setting the `EN_PRE_SYNC` bit of the `DEMOD_CTRL` register to 1.

Enable Min-Max Detection

An offset is always possible, especially if the transmitter is not sending the central frequency but a 0 or a 1 before the preamble, and if the preamble comes a long time after the PA ramp-up. In such a case, the carrier estimation may be biased. However, an alternative algorithm can be used: it looks at the output of the matched filter for the minimum and maximum values, and sets the threshold at the middle. If this functionality is enabled, the search for the min and max is performed only between the 10th and 42nd symbol after the RSSI detection (in Bluetooth low energy technology, it is from the 3rd and the last bit of the synchronization word). This block should not be activated if the carrier estimation is longer than eight symbols; otherwise it will give a false value, since the early estimation is made on data not yet corrected. The functionality is activated by the `EN_MIN_MAX_MF` bit of the `DEMOD_CTRL` register being set to 1.

Fast Clock Recovery

On the 4-FSK modulation, clock recovery is critical because the horizontal eye is quite close. In order to have a clock recovery that performs well, the time constant needs to be increased to filter the excessive noise on the zero crossings. However, during the preamble, the eye is not close at all because there is no inter-symbol interference. The idea is to have a short period during the preamble, in which clock recovery is sped up to get the correct phase quickly. This functionality is activated by setting the `EN_FAST_CLK_RECOV` of the register `DEMOD_CTRL` to 1.

8.4.3.3.5 Delay Line Synchronization

For some particular protocols, including Bluetooth low energy technology, there is an additional synchronization mechanism that works well for carrier recovery. This mechanism uses the delay line to look at the synchronization word. When a flaw is found, the mechanism is able to evaluate the frequency offset of the carrier recovery.

NOTE: This mode only works with 32-bit sync words and LSB first. It is enabled by setting the `EN_DELLINE_SYNC_DET` bit of the register `DEMOD_CTRL` to 1.

When this mode is activated, it is recommended that the sync word correction bias be activated by setting the `EN_SYNC_WORD_CORR` to 1. The `SYNC_WORD_BIAS` field of the same register also must be set. As a rule of thumb, it needs to be fixed at ~12 × h, where h is the modulation index. The internal correlator will look for a peak. The precision of this peak search can be controlled by `MAX_ERR_IN_DL_SYNC`. In practice, it defines the maximum number of errors in the sync word, from 0 to 3.

The correction for carrier recovery will be available only after the entire sync word has entered the delay line. This correction needs to be applied to the sync word in order to provide the decision block with a good input. Because of this, the delay line needs to be set to a delay greater than 32 symbols.

There is an additional mode that can be used. The delay line is capable of detecting the sync word, so in theory the sync detection in the deserializer is no longer needed. Moreover, the correlation peak also gives information regarding the optimal sampling position of the sync word: it is in the middle of the peak. This information can be used to trigger clock recovery. So the sync word detection in the delay line can be used to trigger correct packet reception. To enable this functionality, the `EN_SYNC_OK_DELAY_LINE` bit of the register `CARRIER_RECOVERY_EXTRA` must be set to 1. In this case, non-causal processing needs to be disabled. This mode gives the minimum delay on packet reception. Note that this mode has been tested only for Bluetooth low energy technology-type modulation.

8.4.3.4 Matched Filtering

The matched filter is used in order to filter the signal by maximizing the SNR value. This block is also responsible for choosing the right data representation (phase or frequency). There are actually two FIRs present in the digital baseband: in the case of a PSK modulation they are both used for I and Q signals, while in the case of an FSK modulation the FIR is used for anti-causal processing. This block cannot be disabled.

The filter is an FIR, and the coefficients are specified in the `COEF*` fields of the `RX_PULSE_SHAPE` registers. The FIR is symmetrical and its impulse response is given by:

```
[coef1, coef2, ... coef7, coef8, coef8, coef7, ..., coef2, coef1]
```

In the case of an FSK modulation, the phase signal at the input of the filter is converted to frequency, and goes through the FIR. In the case of a PSK modulation, the phase is converted to the linear domain by a simple look-up table (LUT), and the I and Q signals go through the filter.

At the output of the filter there is a gain stage. This stage is used to normalize the amplitude of the signal. It is mostly useful in the case of FSK modulation to normalize the modulation index, or in the case of a pre-processing in the RX path that has a non-controllable gain. The gain is specified by a mantissa and exponent combination. The values of these coefficients are specified by the `FILTER_GAIN_M` and `FILTER_GAIN_E` fields of the `FILTER_GAIN` register. The mantissa has to be specified as an unsigned value and the exponent as a signed value. The gain after the FIR is specified as:

$$G = \left(1 + \frac{m}{8}\right)2^{e}$$

8.4.3.5 Clock and Data-Rate Recovery

This block recovers the clock of the signal, and its data rate (inside a specific range): in practice it will generate an enabled signal working at the clock frequency.

A distinction has to be made between clock recovery and data-rate recovery:

- Clock recovery refers to the recovery of the sampling instant on the eye diagram. In practice, it is the capacity to determine the best instant in which to sample the signal, in order to avoid ISI and to sample in the middle of the eye.
- Data-rate recovery refers to the capability of determining the transmitter data rate. Generally, data rate recovery is not needed, because the matching of the crystals between the TX and the RX should be good enough, and the few tenths of ppm can be recovered by the simpler clock recovery algorithm. However, in some special cases, the mismatch can be too high for simple clock recovery; for example, in the case of a transmitter with only an RC oscillator, accuracy cannot be guaranteed. Note that once the data rate has been recovered, the clock still needs to be recovered. Nothing ensures that once the data rate has been recovered, the sampling instant is in the middle of the eye.

Both clock and data-rate recovery work together on the zero crossings of the signal. In particular, a correlation is made between the input signal and an expected crossing signal.

This block takes several parameters: the time constants `TAU_CLK_RECOV` and `TAU_DATARATE_RECOV` which are grouped together under the same register, the data-rate recover limit `DR_LIMIT` from the `FILTER_GAIN` register, and the data-rate offset `DATARATE_OFFSET` from the `DATARATE_OFFSET` register. The time constants determine the time that the block needs in order to achieve clock or data-rate recovery, respectively.

`DATARATE_OFFSET` specifies the initial expected data-rate offset. The offset is specified with a signed 8-bit word. The full scale corresponds to 12.5% of mismatch.

`DR_LIMIT` specifies the range of data-rate recovery. The values are given in Table 16 on page 154.

Table 16. Data-Rate Recovery Search Range

dr_limit	Search Range
00	0
01	±3.125%
10	±6.25%
11	±12.5%

NOTE: For small data-rate mismatches – for example, if only ppm of crystal oscillators are responsible for a DR mismatch – a simple clock recovery is enough. Also, there is a potential issue in data-rate recovery if carrier recovery is not performed correctly. This issue is seen in Figure 15.

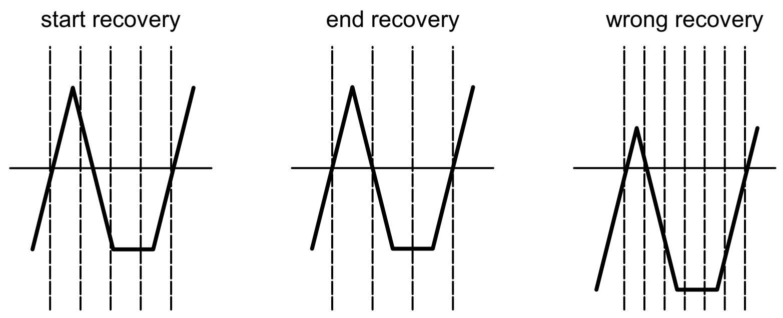


Figure 15. Data-Rate Recovery Issue

As Figure 15 shows, data recovery aims to align an internal counter to the zero crossings of the signal. In the case shown by the right-hand image in Figure 15, the data rate is lowered in order to align the zero crossings. If the carrier is not recovered correctly, the zero crossings are misaligned and appear as though they came from a faster signal. For this reason, the conditions for the zero crossing detection of data-rate recovery are stricter.

Clock recovery does not change in the case of a 4-FSK modulation: it is always based on the zero crossing detection. However, because of the 4-FSK modulation, the eye diagram horizontal opening is narrower than that of a 2-FSK modulation, as can be seen in Figure 16 on page 155.

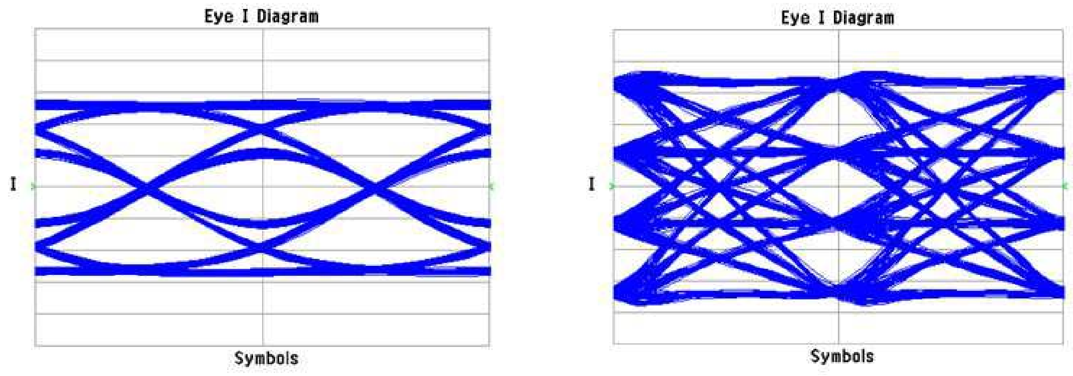


Figure 16. Eye Diagram for 2-FSK Modulation and 4-FSK Modulation

This means that the time constant for the 4-FSK modulation needs to be increased in order to achieve better filtering and be more precise regarding the sampling time. The latter is especially important because, as can be seen in Figure 16, if the sampling time is not exact, there may be a wrong decision regarding the level. The same is not true for the 2-FSK, for which, ideally, the sampling time can be between the two zero crossings.

8.4.3.6 Decision

Due to the Gaussian filter, the GFSK modulation scheme introduces inter-symbol interference (ISI). The ISI decreases the sensitivity of the receiver, because during the decision the signal level can be smaller than in the case of a rectangular pulse shape. ISI cancellation is carried out using the Viterbi algorithm.

8.4.3.6.1 Viterbi Algorithm

The most elegant solution for getting rid of ISI is the Viterbi algorithm, because it is a maximum likelihood sequence estimator.

The Viterbi algorithm is simply enabled by setting the `EN_VITERBI_GFSK` bit of the `DECISION` register. The amplitudes of the expected signal are specified in the `FSK_FCR_AMP` registers. The path length of the estimator is specified by the `VITERBI_LEN` field of the `DECISION` register.

8.4.3.7 RSSI Filtering and AGC

The RSSI filter is a block that filters the instantaneous RSSI; the filter is a multi-rate filter, so a large choice of filter rates is available. The time constant of the filter is given by the `TAU_RSSI_FILTERING` field in the `RSSI_BANK` register. A fast mode can also be made available, by setting the `FAST_RSSI` bit of the same register to 1: this results in the time window being eight times shorter. During the averaging period, two blocks will also evaluate a minimum and a maximum value of the RSSI. These RSSI values are available in the `RSSI_AVG`, `RSSI_MAX` and `RSSI_MIN` registers. Note that the controlled (AGC) attenuation in the RX signal path is compensated for automatically by the block, so these values have to be considered absolutes.

The RSSI-filtered value is also used by an AGC algorithm. The AGC consists of a simple counter. If the RSSI-filtered value is greater than the value specified in the `AGC_THR_HIGH` register, the counter will increase; if it is lower than the value specified in the `AGC_THR_LOW` register, the counter will decrease. The counter has three bits and starts at 0. Its maximum value is fixed by the value of the `ATT_CTRL_MAX` field of the `ATT_CTRL` register. The value of the counter is then used as the input of the AGC look-up table specified in the `AGC_LUT` registers. This LUT is composed of 11-bit words that correspond to the attenuation of the analog RX path. The bits of the fields of the `AGC_LUT_*` register are distributed in the following order:

- agc_level(1:0) — LNA2 configuration
    - 00: max gain
    - 01: 6 dB attenuation
    - 10: not valid setting
    - 11: 12 dB attenuation

- agc_level(2): if set, adds 6 dB of attenuation by LNA current reduction (changed mirror ratio).
- agc_level(3): if set, adds 5 dB of attenuation by LNA1 load resistive degeneration.
- agc_level(4): if set, adds 5 dB of attenuation by LNA1 load resistive degeneration.
- agc_level(6:5) — intermediate frequency amplifier Gm control
    - 00: max gain
    - 01: 6 dB attenuation
    - 10: not valid setting
    - 11: 12 dB attenuation

- agc_level(8:7) — load of the intermediate frequency amplifier
    - 00: 16 kΩ, max gain
    - 01: 8 kΩ, 6dB of attenuation
    - 10: 4 kΩ, 12dB of attenuation
    - 11: 2 kΩ, 18dB of attenuation

- agc_level(10:9) — select the LNA bias current.
    - 00: lna_agc_bias_0
    - 01: lna_agc_bias_1
    - 10: lna_agc_bias_2

- 11: lna_agc_bias_3

To increase the speed of the AGC, an improved version of the AGC algorithm has been implemented. When an RSSI value is received, the AGC predicts what should be the AGC step. To do so, it needs to know the attenuation between every AGC level. These attenuations can be specified by the fields `RF_AGC_ATTXX` of the register `AGC_ATT`. The field `AGC_ATT_01`, for example, specifies the attenuation level between level 0 and level 1 of the AGC. These steps must be specified with a resolution of 2 dB. For example, the value 0x3 means that the AGC step attenuates by 6 dB. The attenuations can be optionally specified between 4 dB and 11 dB, with a resolution of 1 dB. In such a case, the value 0x3 means that the AGC step attenuates by 4+3 = 7 dB. This option is selected by setting bit 33 of register `AGC_ATT` to 1. To activate the mode of RSSI correction, the `AGC_MODE` bit of the register `RSSI_CTRL` needs to be set to 1.

The stability of the AGC can be improved for both algorithms by setting a wait state after the AGC changes its state. The AGC algorithm can wait 0, 1, 2, or 3 RSSI measurements before updating the AGC state. This wait time can be selected using the `AGC_WAIT` field of the `RSSI_CTRL` register.

The AGC algorithm can be switched off by setting the `BYPASS_AGC` bit of the `RSSI_CTRL` register to 1. The RX chain attenuation is then determined by the `SET_RX_ATT_CTRL` field of the `ATT_CTRL` register.

8.4.3.7.1 Peak Detector

The peak detector is used to increase the adjacent channel rejection in case of a close interferer. If the interferer is strong enough to trigger the peak detector, an AGC step increase is requested. This procedure is repeated until the peak detector trigger returns to zero.

To use the peak detector, the FSM needs to activate it: the bit `USE_PEAK_DETECTOR` of the `CTRL_RX` register needs to be set to 1. The AGC algorithm will use the peak detector information if the `EN_AGC_PEAK` bit of the `AGC_PEAK_DET` register is set to 1. The peak detector has three thresholds; the AGC algorithm uses one of these thresholds to determine if the interferer is too strong, and another one to determine that no more interferer is present. These two thresholds are selected via the `PEAK_DET_THR_LOW` field and the `PEAK_DET_THR_HIGH` bit of the `AGC_PEAK_DET` register. In the same register there is also `PEAK_DET_TAU,` which defines a time constant for the filtering of the peak detector signals.

- `PEAK_DET_THR_LOW` — peak detector low threshold (AGC decrement indicator)
  - 00: below level 1
  - 01: below level 2
  - 10: below level 3
  - 11: N.A.

- `PEAK_DET_THR_HIGH` — peak detector high threshold (AGC increment indicator)
  - 0: above level 2
  - 1: above level 3

8.4.3.7.2 RSSI and Peak-Detector Combined AGC Strategy

If the peak detector is activated, there are 4 distinct signals:

rssi_over	Set to 1 if the RSSI value is larger than `AGC_THR_HIGH.`
rssi_under	Set to 1 if the RSSI value is smaller than `AGC_THR_LOW.`
peak_over	Set to 1 if the peak detector output is larger than `PEAK_DET_THR_HIGH.`

peak_under	Set to 1 if the peak detector output is smaller than `PEAK_DET_THR_LOW`.

These signals define the following actions:

inc_att (increase attenuation)

Set to 1 if rssi_over *or* peak_over is 1.

In this case, the required number of steps is estimated using the RSSI value above `AGC_THR_HIGH`; the peak detector alone increases by one AGC step per cycle.

dec_att (decrease attenuation)

Set to 1 if rssi_under *and* peak_under are 1.

Attenuation is always decreased by one AGC step per cycle.

8.4.4 Continuous Wave (CW) Configuration

In the course of your output power and frequency testing, you might need to configure the RF front-end to output a CW signal. For instance, if your testing equipment uses a frequency divider/counter to measure output frequency, it requires RSL10 to output an unmodulated signal.

The following steps describe how to use register settings to configure a CW signal output, for Tx or Rx, at a rate of either 1 Mbps or 2Mbps:

1. Load the *hci_app* hex file into flash memory, and then reset the RSL10 Evaluation and Development Board. This ensures that the RF registers are set correctly.

NOTE: The register-setting steps that follow can be performed using JTAG commander, or implemented in the Arm Cortex-M3 processor code itself.

2. Set `RF_REG00->MODE2_BYTE` to 0.
3. Set `RF_CENTER_FREQ` to the frequency you desire. Find the required frequency using the equation $\text{frequency} = 0x8215c71c + (n \times 0x71c7)$, where *n* is the RF Bluetooth low energy channel number from 0 to 39.
4. `RF_REG05->BANK_BYTE` is set to 0, for 1 Mbps, by default. For 2 Mbps, set `RF_REG05->BANK_BYTE` to 1.
5. To configure for Rx mode, set `RF_REG30->FSM_MODE_BYTE` to 3.
6. To configure for Tx mode, set `RF_REG30->FSM_MODE_BYTE` to 7.
7. To configure for idle mode (disable RF), set `RF_REG30->FSM_MODE_BYTE` to 8.
8. To disable pulse shaping during Tx to keep the frequency from being offset from the center, set registers `RF_TX_PULSE0`, `RF_TX_PULSE1`, `RF_TX_PULSE2`, and `RF_TX_PULSE3` to zero.

When working in CW configuration, use your own preferred settings for VDDRF, VDDPA enabling, VCC, VDDPA, DCCLK, the charge pump clock, and buck enabling.

8.4.5 Direct Test Mode (DTM)

DTM is a standard mechanism defined in the Bluetooth Specification for testing the radio performance and interoperability of Bluetooth Low Energy devices.

Through DTM, an external Bluetooth test instrument can use a 2-wire UART interface to issue standardized HCI (Host Control Interface) commands.

DTM is required for Bluetooth and some regulatory approval processes. Therefore, if your product design needs DTM, you must expose a UART interface.

Refer to the *hci_app* sample application for details on how to use DTM.

8.5 RF Front-End Registers

Register Name	Banked	Register Description	Address
`SYSCTRL_RF_POWER_CFG`	N/A	RF Power Configuration	0x40000050
`SYSCTRL_RF_ACCESS_CFG`	N/A	RF Access Configuration	0x40000054
`RF_REG00`	No	REG00	0x40010000
`RF_REG01`	-	REG01 - The TAU_ROUGH_RECOV, and TAU_PHASE_RECOV registers are banked; all other registers are not banked.	0x40010004
`RF_REG02`	-	REG02 - The TAU_CLK_RECOV register is banked; all other registers are not banked.	0x40010008
`RF_REG03`	No	REG03	0x4001000C
`RF_REG04`	No	REG04	0x40010010
`RF_REG05`	No	REG05	0x40010014
`RF_CENTER_FREQ`	No	CENTER_FREQ	0x40010018
`RF_REG07`	Yes	REG07	0x4001001C
`RF_REG08`	Yes	REG08	0x40010020
`RF_REG09`	Yes	REG09	0x40010024
`RF_REG0A`	Yes	REG0A	0x40010028
`RF_SYNC_PATTERN`	Yes	SYNC_PATTERN	0x4001002C
`RF_REG0C`	Yes	REG0C	0x40010030
`RF_CRC_POLYNOMIAL`	Yes	CRC_POLYNOMIAL	0x40010034
`RF_CRC_RST`	Yes	CRC_RST	0x40010038
`RF_REG0F`	Yes	REG0F	0x4001003C
`RF_REG10`	Yes	REG10	0x40010040
`RF_TX_PULSE0`	Yes	TX_PULSE0	0x40010044
`RF_TX_PULSE1`	Yes	TX_PULSE1	0x40010048
`RF_TX_PULSE2`	Yes	TX_PULSE2	0x4001004C
`RF_TX_PULSE3`	Yes	TX_PULSE3	0x40010050
`RF_RX_PULSE`	Yes	RX_PULSE	0x40010054
`RF_REG16`	Yes	REG16	0x40010058
`RF_REG17`	Yes	REG17	0x4001005C
`RF_REG18`	Yes	REG18	0x40010060
`RF_REG19`	Yes	REG19	0x40010064
`RF_REG1A`	-	REG1A - The `FILTER_BIAS_IQ_FI` register is banked; all other registers are not banked.	0x40010068
`RF_REG1B`	No	REG1B	0x4001006C
`RF_AGC_LUT1`	No	AGC_LUT1	0x40010070
`RF_AGC_LUT2`	No	AGC_LUT2	0x40010074

Register Name	Banked	Register Description	Address
RF_AGC_LUT3	No	AGC_LUT3	0x40010078
RF_AGC_LUT4	No	AGC_LUT4	0x4001007C
RF_REG20	No	REG20	0x40010080
RF_AGC_ATT1	No	AGC_ATT1	0x40010084
RF_REG22	No	REG22	0x40010088
RF_REG23	No	REG23	0x4001008C
RF_REG24	No	REG24	0x40010090
RF_REG25	No	REG25	0x40010094
RF_REG26	No	REG26	0x40010098
RF_REG27	-	REG27- The CTRL_ADC register is banked; all other registers are not banked.	0x4001009C
RF_REG28	No	REG28	0x400100A0
RF_PLL_CTRL	No	PLL_CTRL	0x400100A4
RF_REG2A	No	REG2A	0x400100A8
RF_XTAL_CTRL	No	XTAL_CTRL	0x400100AC
RF_REG2C	No	REG2C	0x400100B0
RF_REG2D	No	REG2D	0x400100B4
RF_REG2E	No	REG2E	0x400100B8
RF_REG2F	No	REG2F	0x400100BC
RF_REG30	No	REG30	0x400100C0
RF_REG31	No	REG31	0x400100C4
RF_REG32	No	REG32	0x400100C8
RF_TXFIFO	No	TXFIFO	0x400100CC
RF_RXFIFO	No	RXFIFO	0x400100D0
RF_DESER_STATUS	No	DESER_STATUS	0x400100D4
RF_IRQ_STATUS	No	IRQ_STATUS	0x400100D8
RF_REG37	No	REG37	0x400100DC
RF_REG38	No	REG38	0x400100E0
RF_REG39	No	REG39	0x400100E4
RF_REVISION	No	REVISION	0x400100FC

8.5.1 RF_REG00

Bit Field	Field Name	Description
31	DATAWHITE_BTLE_DW_BTLE	If set to 1, the data whitening specified in the Bluetooth LE standard is used. Note that the en_datawhite field of the CODING register has also to be set to 1
30:24	DATAWHITE_BTLE_DW_BTLE_RST	Reset value to put on the Bluetooth LE data whitening shift register
23	FOURFSK_CODING_EN_FOURFSK_CODING	If set to 1 the 4-FSK coding is activated

Bit Field	Field Name	Description
22:20	FOURFSK_CODING_TX_FOURFSK_CODING	Set the 4-FSK coding (Tx): bit 0 determine if the sign is given by the Q signal (0) or I signal (1), bit 1 select if the signal is inverted for the sign, it 2 select if the signal is inverted for the abs amplitude
18:16	FOURFSK_CODING_RX_FOURFSK_CODING	Set the 4-FSK decoding (Rx): bit 0 determine if the sign is given by the Q signal (0) or I signal (1), bit 1 select if the signal is inverted for the sign, it 2 select if the signal is inverted for the abs amplitude
14	MODE2_DIFF_CODING	If set to 1 enables the differential coding/decoding
13	MODE2_PSK_NFSK	If set to 1, the PSK mode is selected, FSK otherwise.
12:8	MODE2_TESTMODE	set the output testmode
7	MODE_NOT_TO_IDLE	In FSM mode, if set to 1 indicates to the FSM to go in suspend mode after a Tx or Rx packet
5	MODE_EN_FSM	If set to 1 enables the radio FSM
4	MODE_EN_DESERIALIZER	If set to 1 enables the deserializer
3	MODE_EN_SERIALIZER	If set to 1 enables the serializer
2	MODE_TX_NRX	if set to 1 use the Tx, otherwise the Rx
1:0	MODE_MODE	Select the working mode of the digital baseband: 00) the digital baseband is off (no clock) 01) the clock is generated but the blocks are reset (Tx,Rx,FIFOs and FSM) 10) the digital baseband is frozen 11) working

Field Name	Value Symbol	Value Description	Hex Value
DATAWHITE_BTLE_DW_BTLE	DATAWHITE_BTLE_DW_BTLE_DEFAULT		0x0*
DATAWHITE_BTLE_DW_BTLE_RST	DATAWHITE_BTLE_DW_BTLE_RST_DEFAULT		0x0*
FOURFSK_CODING_EN_FOURFSK_CODING	FOURFSK_CODING_EN_FOURFSK_CODING_DEFAULT		0x0*
FOURFSK_CODING_TX_FOURFSK_CODING	FOURFSK_CODING_TX_FOURFSK_CODING_DEFAULT		0x0*
FOURFSK_CODING_RX_FOURFSK_CODING	FOURFSK_CODING_RX_FOURFSK_CODING_DEFAULT		0x0*
MODE2_DIFF_CODING	MODE2_DIFF_CODING_DEFAULT		0x0*
MODE2_PSK_NFSK	MODE2_PSK_NFSK_DEFAULT		0x0*
MODE2_TESTMODE	MODE2_TESTMODE_DEFAULT		0x0*
MODE_NOT_TO_IDLE	MODE_NOT_TO_IDLE_DEFAULT		0x0*
MODE_EN_FSM	MODE_EN_FSM_DEFAULT		0x0*
MODE_EN_DESERIALIZER	MODE_EN_DESERIALIZER_DEFAULT		0x0*
MODE_EN_SERIALIZER	MODE_EN_SERIALIZER_DEFAULT		0x0*
MODE_TX_NRX	MODE_TX_NRX_DEFAULT		0x0*
MODE_MODE	MODE_MODE_DEFAULT		0x0*

8.5.2 RF_REG01

Bit Field	Field Name	Description
31:24	TAU_PHASE_RECOV_TAU_PHASE_RECOV	Time constant of the fine carrier recovery block
23:16	TAU_ROUGH_RECOV_TAU_ROUGH_RECOV	Time constant of the rough carrier recovery block
15	CARRIER_RECOVERY_EN_CORRECT_CFREQ_AFC	If set to 1, enables the automatic AFC correction.
14	CARRIER_RECOVERY_CORRECT_CFREQ_IF_NEG	If set to 1, the IF correction is negative
13	CARRIER_RECOVERY_EN_CORRECT_CFREQ_IF	If set to 1, enables the automatic IF correction
12	CARRIER_RECOVERY_AFC_NEG	If set to 1 correct the AFC negatively
11	CARRIER_RECOVERY_STARTER_MODE	If set to 1 enables the starter mode, i.e. a 32x faster carrier recovery.
10	CARRIER_RECOVERY_EN_AFC	if set to 1 enables the Automatic Frequency Control
9	CARRIER_RECOVERY_EN_FINE_RECOV	If set to 1 enables the fine carrier recovery
8	CARRIER_RECOVERY_EN_ROUGH_RECOV	If set to 1 enables the rough carrier recovery
6	MOD_TX_PULSE_NSYM	If set to 1, the Tx pulse shape is an odd function.
5	MOD_TX_EN_INTERP	If set to 1, enables the Tx CIC interpolator.
4:0	MOD_TX_CK_TX_M	Unsigned value that determines the Tx interpolator frequency. The formula is similar to the evaluation of the oversampling frequency.

Field Name	Value Symbol	Value Description	Hex Value
TAU_PHASE_RECOV_TAU_PHASE_RECOV	TAU_PHASE_RECOV_TAU_PHASE_RECOV_DEFAULT		0x0*
TAU_ROUGH_RECOV_TAU_ROUGH_RECOV	TAU_ROUGH_RECOV_TAU_ROUGH_RECOV_DEFAULT		0x0*
CARRIER_RECOVERY_EN_CORRECT_CFREQ_AFC	CARRIER_RECOVERY_EN_CORRECT_CFREQ_AFC_DEFAULT		0x0*
CARRIER_RECOVERY_CORRECT_CFREQ_IF_NEG	CARRIER_RECOVERY_CORRECT_CFREQ_IF_NEG_DEFAULT		0x0*
CARRIER_RECOVERY_EN_CORRECT_CFREQ_IF	CARRIER_RECOVERY_EN_CORRECT_CFREQ_IF_DEFAULT		0x0*
CARRIER_RECOVERY_AFC_NEG	CARRIER_RECOVERY_AFC_NEG_DEFAULT		0x0*
CARRIER_RECOVERY_STARTER_MODE	CARRIER_RECOVERY_STARTER_MODE_DEFAULT		0x0*
CARRIER_RECOVERY_EN_AFC	CARRIER_RECOVERY_EN_AFC_DEFAULT		0x0*
CARRIER_RECOVERY_EN_FINE_RECOV	CARRIER_RECOVERY_EN_FINE_RECOV_DEFAULT		0x0*
CARRIER_RECOVERY_EN_ROUGH_RECOV	CARRIER_RECOVERY_EN_ROUGH_RECOV_DEFAULT		0x0*
MOD_TX_PULSE_NSYM	MOD_TX_PULSE_NSYM_DEFAULT		0x0*
MOD_TX_EN_INTERP	MOD_TX_EN_INTERP_DEFAULT		0x0*
MOD_TX_CK_TX_M	MOD_TX_CK_TX_M_DEFAULT		0x0*

8.5.3 RF_REG02

Bit Field	Field Name	Description
31	FIFO_FIFO_FLUSH_ON_OVFLW	If set to 1, stops the Rx and flushes the FIFO in case of overflow
30	FIFO_FIFO_FLUSH_ON_ADDR_ERR	If set to 1, stops the Rx and flushes the FIFO in case of address error
29	FIFO_FIFO_FLUSH_ON_PL_ERR	If set to 1, stops the Rx and flushes the FIFO in case of packet length error
28	FIFO_FIFO_FLUSH_ON_CRC_ERR	If set to 1, stops the Rx and flushes the FIFO in case of CRC error
27	FIFO_RX_FIFO_ACK	If set to 1, the Rx FIFO needs an acknowledgement (packet received correctly) to change its state.
26:24	FIFO_FIFO_THR	Threshold indicating the 'almost full' state
23:16	DATARATE_OFFSET_DATARATE_OFFSET	Data-rate offset. Is a signed value and the full scale (0x7f) corresponds to a data-rate offset of 12.5%.
15:8	TAU_DATARATE_RECOV_TAU_DATARATE_RECOV	Time constant of the data-rate recovery
7:0	TAU_CLK_RECOV_TAU_CLK_RECOV	Time constant of the clock recovery

Field Name	Value Symbol	Value Description	Hex Value
FIFO_FIFO_FLUSH_ON_OVFLW	FIFO_FIFO_FLUSH_ON_OVFLW_DEFAULT		0x0*
FIFO_FIFO_FLUSH_ON_ADDR_ERR	FIFO_FIFO_FLUSH_ON_ADDR_ERR_DEFAULT		0x0*
FIFO_FIFO_FLUSH_ON_PL_ERR	FIFO_FIFO_FLUSH_ON_PL_ERR_DEFAULT		0x0*
FIFO_FIFO_FLUSH_ON_CRC_ERR	FIFO_FIFO_FLUSH_ON_CRC_ERR_DEFAULT		0x0*
FIFO_RX_FIFO_ACK	FIFO_RX_FIFO_ACK_DEFAULT		0x0*
FIFO_FIFO_THR	FIFO_FIFO_THR_DEFAULT		0x0*
DATARATE_OFFSET_DATARATE_OFFSET	DATARATE_OFFSET_DATARATE_OFFSET_DEFAULT		0x0*
TAU_DATARATE_RECOV_TAU_DATARATE_RECOV	TAU_DATARATE_RECOV_TAU_DATARATE_RECOV_DEFAULT		0x0*
TAU_CLK_RECOV_TAU_CLK_RECOV	TAU_CLK_RECOV_TAU_CLK_RECOV_DEFAULT		0x0*

8.5.4 RF_REG03

Bit Field	Field Name	Description
31:28	PAD_CONF_2_PAD_3_CONF	Configuration of GPIO pad 3
27:24	PAD_CONF_2_PAD_2_CONF	Configuration of GPIO pad 2
23:20	PAD_CONF_1_PAD_1_CONF	Configuration of GPIO pad 1
19:16	PAD_CONF_1_PAD_0_CONF	Configuration of GPIO pad 0
15	IRQ_CONF_IRQ_HIGH_Z	If set to 1, the pads are set to High-Z when the IRQ is not active.
14	IRQ_CONF_IRQ_ACTIVE_LOW	If set to 1, the IRQ are active low
13:8	IRQ_CONF_IRQS_MASK	Mask to determine which IRQs are enabled (active high)
7:5	FIFO_2_FIFO_THR_TX	Threshold indicating the 'almost empty' state

Bit Field	Field Name	Description
4	FIFO_2_WAIT_TXFIFO_WR	If set to 1, the FSM will wait a Tx FIFO write before starting the Tx in case of an empty Tx FIFO.
3	FIFO_2_STOP_ON_RXFF_OVFLW	If set to 1, stops the reception in case of a FIFO overflow.
2	FIFO_2_STOP_ON_TXFF_UNFLW	If set to 1, stops the transmission in case of a FIFO underflow.
1	FIFO_2_RXFF_FLUSH_ON_START	If set to 1, flushes the Rx FIFO when the Rx is enabled, in order to receive a packet with an empty FIFO.
0	FIFO_2_TXFF_FLUSH_ON_STOP	If set to 1, flushes the Tx FIFO after the end of a packet transmission in order to have an empty FIFO.

Field Name	Value Symbol	Value Description	Hex Value
PAD_CONF_2_PAD_3_CONF	PAD_CONF_2_PAD_3_CONF_DEFAULT		0x0*
PAD_CONF_2_PAD_2_CONF	PAD_CONF_2_PAD_2_CONF_DEFAULT		0x0*
PAD_CONF_1_PAD_1_CONF	PAD_CONF_1_PAD_1_CONF_DEFAULT		0x0*
PAD_CONF_1_PAD_0_CONF	PAD_CONF_1_PAD_0_CONF_DEFAULT		0x0*
IRQ_CONF_IRQ_HIGH_Z	IRQ_CONF_IRQ_HIGH_Z_DEFAULT		0x0*
IRQ_CONF_IRQ_ACTIVE_LOW	IRQ_CONF_IRQ_ACTIVE_LOW_DEFAULT		0x0*
IRQ_CONF_IRQS_MASK	IRQ_CONF_IRQS_MASK_DEFAULT		0x0*
FIFO_2_FIFO_THR_TX	FIFO_2_FIFO_THR_TX_DEFAULT		0x0*
FIFO_2_WAIT_TXFIFO_WR	FIFO_2_WAIT_TXFIFO_WR_DEFAULT		0x0*
FIFO_2_STOP_ON_RXFF_OVFLW	FIFO_2_STOP_ON_RXFF_OVFLW_DEFAULT		0x0*
FIFO_2_STOP_ON_TXFF_UNFLW	FIFO_2_STOP_ON_TXFF_UNFLW_DEFAULT		0x0*
FIFO_2_RXFF_FLUSH_ON_START	FIFO_2_RXFF_FLUSH_ON_START_DEFAULT		0x0*
FIFO_2_TXFF_FLUSH_ON_STOP	FIFO_2_TXFF_FLUSH_ON_STOP_DEFAULT		0x0*

8.5.5 RF_REG04

Bit Field	Field Name	Description
31:30	MAC_CONF_MAC_TIMER_GR	MAC timer granularity. The granularity is given by (2^(2mac_timer_gr))x1us
29	MAC_CONF_RX_MAC_ACT	If set to 1, the FSM will switch to Rx or Tx after an Rx mode.
28	MAC_CONF_RX_MAC_TX_NRX	If set to 1, the FSM will switch to Tx after an Rx mode, Rx otherwise.
27	MAC_CONF_RX_MAC_START_NSTOP	If set to 1, the MAC timer is activated at the reception of the sync word, at the end of the packet otherwise.
26	MAC_CONF_TX_MAC_ACT	If set to 1, the FSM will switch to Rx or Tx after a Tx mode.
25	MAC_CONF_TX_MAC_TX_NRX	If set to 1, the FSM will switch to Tx after a Tx mode, Rx otherwise.
24	MAC_CONF_TX_MAC_START_NSTOP	If set to 1, the MAC timer is activated at beginning of the packet, otherwise at the end of the packet transmission.
23:20	PAD_CONF_5_PAD_9_CONF	Configuration of GPIO pad 9

Bit Field	Field Name	Description
19:16	PAD_CONF_5_PAD_8_CONF	Configuration of GPIO pad 8
15:12	PAD_CONF_4_PAD_7_CONF	Configuration of GPIO pad 7
11:8	PAD_CONF_4_PAD_6_CONF	Configuration of GPIO pad 6
7:4	PAD_CONF_3_PAD_5_CONF	Configuration of GPIO pad 5
3:0	PAD_CONF_3_PAD_4_CONF	Configuration of GPIO pad 4

Field Name	Value Symbol	Value Description	Hex Value
MAC_CONF_MAC_TIMER_GR	MAC_CONF_MAC_TIMER_GR_DEFAULT		0x0*
MAC_CONF_RX_MAC_ACT	MAC_CONF_RX_MAC_ACT_DEFAULT		0x0*
MAC_CONF_RX_MAC_TX_NRX	MAC_CONF_RX_MAC_TX_NRX_DEFAULT		0x0*
MAC_CONF_RX_MAC_START_ NSTOP	MAC_CONF_RX_MAC_START_NSTOP_ DEFAULT		0x0*
MAC_CONF_TX_MAC_ACT	MAC_CONF_TX_MAC_ACT_DEFAULT		0x0*
MAC_CONF_TX_MAC_TX_NRX	MAC_CONF_TX_MAC_TX_NRX_DEFAULT		0x0*
MAC_CONF_TX_MAC_START_ NSTOP	MAC_CONF_TX_MAC_START_NSTOP_ DEFAULT		0x0*
PAD_CONF_5_PAD_9_CONF	PAD_CONF_5_PAD_9_CONF_DEFAULT		0x0*
PAD_CONF_5_PAD_8_CONF	PAD_CONF_5_PAD_8_CONF_DEFAULT		0x0*
PAD_CONF_4_PAD_7_CONF	PAD_CONF_4_PAD_7_CONF_DEFAULT		0x0*
PAD_CONF_4_PAD_6_CONF	PAD_CONF_4_PAD_6_CONF_DEFAULT		0x0*
PAD_CONF_3_PAD_5_CONF	PAD_CONF_3_PAD_5_CONF_DEFAULT		0x0*
PAD_CONF_3_PAD_4_CONF	PAD_CONF_3_PAD_4_CONF_DEFAULT		0x0*

8.5.6 RF_REG05

Bit Field	Field Name	Description
30	CHANNEL_SWITCH_IQ	Switch I and Q channels
29:24	CHANNEL_CHANNEL	Channel number
18	BANK_DATARATE_TX_NRX	Select the data-rate register: 0-> Rx data-rate, 1-> Tx data-rate
17:16	BANK_BANK	Select the used bank
15:8	TX_MAC_TIMER_TX_MAC_TIMER	Time to wait after the Tx mode.
7:0	RX_MAC_TIMER_RX_MAC_TIMER	Time to wait after the Rx mode.

Field Name	Value Symbol	Value Description	Hex Value
CHANNEL_SWITCH_IQ	CHANNEL_SWITCH_IQ_DEFAULT		0x0*
CHANNEL_CHANNEL	CHANNEL_CHANNEL_DEFAULT		0x0*
BANK_DATARATE_TX_NRX	BANK_DATARATE_TX_NRX_ DEFAULT		0x0*
BANK_BANK	BANK_BANK_DEFAULT		0x0*

Field Name	Value Symbol	Value Description	Hex Value
TX_MAC_TIMER_TX_MAC_TIMER	TX_MAC_TIMER_TX_MAC_TIMER_DEFAULT		0x0*
RX_MAC_TIMER_RX_MAC_TIMER	RX_MAC_TIMER_RX_MAC_TIMER_DEFAULT		0x0*

8.5.7 RF_CENTER_FREQ

Bit Field	Field Name	Description
31	CENTER_FREQ_ADAPT_CFREQ	If set to 1, automatically adapt frequency between Tx and Rx.
30	CENTER_FREQ_RX_DIV_5_N6	If set to 1, the ratio of the pll reference between Tx and Rx is 5 instead of 6.
29:0	CENTER_FREQ_CENTER_FREQUENCY	Set the center frequency

Field Name	Value Symbol	Value Description	Hex Value
CENTER_FREQ_ADAPT_CFREQ	CENTER_FREQ_ADAPT_CFREQ_DEFAULT		0x0*
CENTER_FREQ_RX_DIV_5_N6	CENTER_FREQ_RX_DIV_5_N6_DEFAULT		0x0*
CENTER_FREQ_CENTER_FREQUENCY	CENTER_FREQ_CENTER_FREQUENCY_DEFAULT		0x0*

8.5.8 RF_REG07

Bit Field	Field Name	Description
31:16	CHANNELS_1_CHANNEL_SPACING_LO	channel spacing: the formula that determines this value is the same as for the central frequency. v=ch_sp/144e6*2^25
14	MOD_INFO_RX_EN_DIV_2_N3_RX	If set to 1 the clock divider will provide a clock divided by 2 instead of 3.
13	MOD_INFO_RX_SYMBOL_2BIT_RX	If set to 1, each symbol is composed by 2 bits (OQPSK or 4-FSK)
12:8	MOD_INFO_RX_DR_M_RX	Unsigned value that determines the oversampling frequency and consequently the data-rate. This frequency is the system frequency (16 or 24 MHz) divided by this value+1.
6	MOD_INFO_TX_EN_DIV_2_N3_TX	If set to 1 the clock divider will provide a clock divided by 2 instead of 3.
5	MOD_INFO_TX_SYMBOL_2BIT_TX	If set to 1, each symbol is composed by 2 bits (OQPSK or 4-FSK)
4:0	MOD_INFO_TX_DR_M_TX	Unsigned value that determines the oversampling frequency and consequently the data-rate. This frequency is the system frequency (16 or 24 MHz) divided by this value+1.

Field Name	Value Symbol	Value Description	Hex Value
CHANNELS_1_CHANNEL_SPACING_LO	CHANNELS_1_CHANNEL_SPACING_LO_DEFAULT		0x0*
MOD_INFO_RX_EN_DIV_2_N3_RX	MOD_INFO_RX_EN_DIV_2_N3_RX_DEFAULT		0x0*
MOD_INFO_RX_SYMBOL_2BIT_RX	MOD_INFO_RX_SYMBOL_2BIT_RX_DEFAULT		0x0*

Field Name	Value Symbol	Value Description	Hex Value
MOD_INFO_RX_DR_M_RX	MOD_INFO_RX_DR_M_RX_DEFAULT		0x0*
MOD_INFO_TX_EN_DIV_2_N3_TX	MOD_INFO_TX_EN_DIV_2_N3_TX_DEFAULT		0x0*
MOD_INFO_TX_SYMBOL_2BIT_TX	MOD_INFO_TX_SYMBOL_2BIT_TX_DEFAULT		0x0*
MOD_INFO_TX_DR_M_TX	MOD_INFO_TX_DR_M_TX_DEFAULT		0x0*

8.5.9 RF_REG08

Bit Field	Field Name	Description
31:24	PACKET_LENGTH_PACKET_LEN	The packet length in the fixed packet length mode. In the variable packet length mode, it specifies the maximal packet length defined by the standard. In case of error, a packet_len_err is raised.
23	PACKET_HANDLING_LSB_FIRST	If set to 1, the LSB is the first bit to be sent, the MSB otherwise
22	PACKET_HANDLING_EN_CRC	If set to 1, enables the automatic CRC evaluation and insertion
21	PACKET_HANDLING_EN_CRC_ON_PKTLEN	If set to 1, enables the CRC calculation on the packet length part of the packet.
20	PACKET_HANDLING_EN_PREAMBLE	If set to 1, enables the automatic preamble insertion
19	PACKET_HANDLING_EN_MULTI_FRAME	If set to 1, enables the multi-frame packet (preamble-pattern-data-CRC-data-CRC-...)
18	PACKET_HANDLING_ENB_DW_ON_CRC	Enables the data-whitening on the CRC (active low)
17	PACKET_HANDLING_EN_PATTERN	If set to 1, enables the automatic pattern insertion and recognition
16	PACKET_HANDLING_EN_PACKET	If set to 1 enables the packet handler
15	CODING_EN_DATAWHITE	If set to 1 enables the data-whitening
14	CODING_I_NQ_DELAYED	If set to 1, the channel I is considered 'delayed' in case of a 2bit per symbol modulation
13	CODING_OFFSET	If set to 1, an offset (delay) is introduced in one of the two channels (2 bits per symbol modulation).
12	CODING_BIT_INVERT	If set to 1, it inverts the bit value (Tx and Rx)
11	CODING_EVEN_BEFORE_ODD	Determines the bit order in case of a 2 bits per symbol modulation: if set to 1 the first bit (bit 0, even) goes to the I path
10	CODING_EN_802154_L2F	If set to 1 enables the linear to frequency encoding needed in order to modulate an OQPSK as an MSK.
9	CODING_EN_802154_B2C	If set to 1 enables the bit to chips encoding used in the IEEE 802.15.4 standard
8	CODING_EN_MANCHESTER	If set to 1 enables the Manchester encoding
7	CHANNELS_2_EN_CHANNEL_SEL	If set to 1 enables the definition of channels
3:0	CHANNELS_2_CHANNEL_SPACING_HI	channel spacing: the formula that determines this value is the same as for the central frequency. v=ch_sp/144e6*2^25

Field Name	Value Symbol	Value Description	Hex Value
PACKET_LENGTH_PACKET_LEN	PACKET_LENGTH_PACKET_LEN_DEFAULT		0xFF*
PACKET_HANDLING_LSB_FIRST	PACKET_HANDLING_LSB_FIRST_DEFAULT		0x0*
PACKET_HANDLING_EN_CRC	PACKET_HANDLING_EN_CRC_DEFAULT		0x0*
PACKET_HANDLING_EN_CRC_ON_PKTLEN	PACKET_HANDLING_EN_CRC_ON_PKTLEN_DEFAULT		0x0*
PACKET_HANDLING_EN_PREAMBLE	PACKET_HANDLING_EN_PREAMBLE_DEFAULT		0x0*
PACKET_HANDLING_EN_MULTI_FRAME	PACKET_HANDLING_EN_MULTI_FRAME_DEFAULT		0x0*
PACKET_HANDLING_ENB_DW_ON_CRC	PACKET_HANDLING_ENB_DW_ON_CRC_DEFAULT		0x0*
PACKET_HANDLING_EN_PATTERN	PACKET_HANDLING_EN_PATTERN_DEFAULT		0x0*
PACKET_HANDLING_EN_PACKET	PACKET_HANDLING_EN_PACKET_DEFAULT		0x0*
CODING_EN_DATAWHITE	CODING_EN_DATAWHITE_DEFAULT		0x0*
CODING_I_NQ_DELAYED	CODING_I_NQ_DELAYED_DEFAULT		0x0*
CODING_OFFSET	CODING_OFFSET_DEFAULT		0x0*
CODING_BIT_INVERT	CODING_BIT_INVERT_DEFAULT		0x0*
CODING_EVEN_BEFORE_ODD	CODING_EVEN_BEFORE_ODD_DEFAULT		0x0*
CODING_EN_802154_L2F	CODING_EN_802154_L2F_DEFAULT		0x0*
CODING_EN_802154_B2C	CODING_EN_802154_B2C_DEFAULT		0x0*
CODING_EN_MANCHESTER	CODING_EN_MANCHESTER_DEFAULT		0x0*
CHANNELS_2_EN_CHANNEL_SEL	CHANNELS_2_EN_CHANNEL_SEL_DEFAULT		0x0*
CHANNELS_2_CHANNEL_SPACING_HI	CHANNELS_2_CHANNEL_SPACING_HI_DEFAULT		0x0*

8.5.10 RF_REG09

Bit Field	Field Name	Description
27	ADDRESS_CONF_ADDRESS_LEN	If set to 1 the address length is 16 bits, 8 otherwise.
26	ADDRESS_CONF_EN_ADDRESS_RX_BR	If set to 1 enables the broadcast address detection on Rx.
25	ADDRESS_CONF_EN_ADDRESS_RX	If set to 1 enables the address detection on Rx
24	ADDRESS_CONF_EN_ADDRESS_TX	If set to 1 enables the address insertion on Tx
23:16	PREAMBLE_LENGTH_PREAMBLE_LEN	Length of the preamble -1
15:8	PREAMBLE_PREAMBLE	Preamble to be inserted

Bit Field	Field Name	Description
6	PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX	If set to 1, the packet length is fixed and specified in the PACKET_LEN register
5:2	PACKET_LENGTH_OPTS_PACKET_LEN_CORR	Signed value that specifies the correction to apply to the specified packet length (due to differences between standards). The packet length here is specified by the byte number after the packet length byte, with the exclusion of the CRC.
1:0	PACKET_LENGTH_OPTS_PACKET_LEN_POS	Unsigned value that specifies the position of the packet length after the pattern

Field Name	Value Symbol	Value Description	Hex Value
ADDRESS_CONF_ADDRESS_LEN	ADDRESS_CONF_ADDRESS_LEN_DEFAULT		0x0*
ADDRESS_CONF_EN_ADDRESS_RX_BR	ADDRESS_CONF_EN_ADDRESS_RX_BR_DEFAULT		0x0*
ADDRESS_CONF_EN_ADDRESS_RX	ADDRESS_CONF_EN_ADDRESS_RX_DEFAULT		0x0*
ADDRESS_CONF_EN_ADDRESS_TX	ADDRESS_CONF_EN_ADDRESS_TX_DEFAULT		0x0*
PREAMBLE_LENGTH_PREAMBLE_LEN	PREAMBLE_LENGTH_PREAMBLE_LEN_DEFAULT		0x0*
PREAMBLE_PREAMBLE	PREAMBLE_PREAMBLE_DEFAULT		0x0*
PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX	PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX_DEFAULT		0x0*
PACKET_LENGTH_OPTS_PACKET_LEN_CORR	PACKET_LENGTH_OPTS_PACKET_LEN_CORR_DEFAULT		0x0*
PACKET_LENGTH_OPTS_PACKET_LEN_POS	PACKET_LENGTH_OPTS_PACKET_LEN_POS_DEFAULT		0x0*

8.5.11 RF_REG0A

Bit Field	Field Name	Description
31:16	ADDRESS_BROADCAST_ADDRESS_BR	Broadcast address
15:0	ADDRESS_ADDRESS	Address of the node

Field Name	Value Symbol	Value Description	Hex Value
ADDRESS_BROADCAST_ADDRESS_BR	ADDRESS_BROADCAST_ADDRESS_BR_DEFAULT		0x0*
ADDRESS_ADDRESS	ADDRESS_ADDRESS_DEFAULT		0x0*

8.5.12 RF_SYNC_PATTERN

Bit Field	Field Name	Description
31:0	PATTERN	Pattern (sync word) to be inserted or recognized.

Field Name	Value Symbol	Value Description	Hex Value
PATTERN	PATTERN_DEFAULT		0x0*

8.5.13 RF_REG0C

Bit Field	Field Name	Description
30:26	CONV_CODES_POLY_CC_POLY_2	polynom of the third convolutional code
25:21	CONV_CODES_POLY_CC_POLY_1	polynom of the second convolutional code
20:16	CONV_CODES_POLY_CC_POLY_0	polynom of the first convolutional code
11:10	CONV_CODES_CONF_CC_VITERBI_LEN	Set the memory length of the Viterbi decoder: 00 => 5, 01 => 10, 10 => 20, 11 => 30
9	CONV_CODES_CONF_CC_EN_TX_STOP	if set to 1 enables the stop word at the end of the transmission. Necessary in order to keep a stream coherent with the convolutional coding
8	CONV_CODES_CONF_EN_CONV_CODE	If set to 1 enables the convolutional codes
7:6	PACKET_EXTRA_STOP_WORD_LEN	length of the stop word, same as the pattern word length
5	PACKET_EXTRA_EN_STOP_WORD	If set to 1 adds the stop word (0x00) after the CRC
4	PACKET_EXTRA_PKT_INFO_PRE_NPOST	If set to 1 the packet information are sampled at the end of the packet instead of the sync word detection.
3:2	PACKET_EXTRA_PATTERN_MAX_ERR	unsigned value that specifies the maximum number of errors in the pattern recognition
1:0	PACKET_EXTRA_PATTERN_WORD_LEN	Pattern word length: 00 => 8bits, 01 => 16 bits, 10 => 24 bits, 11 => 32 bits

Field Name	Value Symbol	Value Description	Hex Value
CONV_CODES_POLY_CC_POLY_2	CONV_CODES_POLY_CC_POLY_2_DEFAULT		0x0*
CONV_CODES_POLY_CC_POLY_1	CONV_CODES_POLY_CC_POLY_1_DEFAULT		0x0*
CONV_CODES_POLY_CC_POLY_0	CONV_CODES_POLY_CC_POLY_0_DEFAULT		0x0*
CONV_CODES_CONF_CC_VITERBI_LEN	CONV_CODES_CONF_CC_VITERBI_LEN_DEFAULT		0x0*
CONV_CODES_CONF_CC_EN_TX_STOP	CONV_CODES_CONF_CC_EN_TX_STOP_DEFAULT		0x0*
CONV_CODES_CONF_EN_CONV_CODE	CONV_CODES_CONF_EN_CONV_CODE_DEFAULT		0x0*
PACKET_EXTRA_STOP_WORD_LEN	PACKET_EXTRA_STOP_WORD_LEN_DEFAULT		0x0*
PACKET_EXTRA_EN_STOP_WORD	PACKET_EXTRA_EN_STOP_WORD_DEFAULT		0x0*
PACKET_EXTRA_PKT_INFO_PRE_NPOST	PACKET_EXTRA_PKT_INFO_PRE_NPOST_DEFAULT		0x0*

Field Name	Value Symbol	Value Description	Hex Value
PACKET_EXTRA_PATTERN_MAX_ERR	PACKET_EXTRA_PATTERN_MAX_ERR_DEFAULT		0x0*
PACKET_EXTRA_PATTERN_WORD_LEN	PACKET_EXTRA_PATTERN_WORD_LEN_DEFAULT		0x0*

8.5.14 RF_CRC_POLYNOMIAL

Bit Field	Field Name	Description
31:0	CRC_POLYNOMIAL_CRC_POLY	CRC polynomial. It is coded using the Koopman notation, i.e. the nth bit codes the (n+1) coefficient. Example: x^16+x^12+x^5+1 => 0x8810

Field Name	Value Symbol	Value Description	Hex Value
CRC_POLYNOMIAL_CRC_POLY	CRC_POLYNOMIAL_CRC_POLY_DEFAULT		0x0*

8.5.15 RF_CRC_RST

Bit Field	Field Name	Description
31:0	CRC_RST_CRC_RST	CRC reset value

Field Name	Value Symbol	Value Description	Hex Value
CRC_RST_CRC_RST	CRC_RST_CRC_RST_DEFAULT		0x0*

8.5.16 RF_REG0F

Bit Field	Field Name	Description
31:28	RX_FRAC_CONF_RX_FRAC_DEN	
27:24	RX_FRAC_CONF_RX_FRAC_NUM	
19	FRAC_CONF_TX_FRAC_GAIN	
18	FRAC_CONF_RX_FRAC_GAIN	
17	FRAC_CONF_TX_EN_FRAC	
16	FRAC_CONF_RX_EN_FRAC	
14:10	CONV_CODES_PUNCT_CC_PUNCT_2	puncture of the third convolutional code
9:5	CONV_CODES_PUNCT_CC_PUNCT_1	puncture of the second convolutional code
4:0	CONV_CODES_PUNCT_CC_PUNCT_0	puncture of the first convolutional code

Field Name	Value Symbol	Value Description	Hex Value
RX_FRAC_CONF_RX_FRAC_DEN	RX_FRAC_CONF_RX_FRAC_DEN_DEFAULT		0x0*
RX_FRAC_CONF_RX_FRAC_NUM	RX_FRAC_CONF_RX_FRAC_NUM_DEFAULT		0x0*
FRAC_CONF_TX_FRAC_GAIN	FRAC_CONF_TX_FRAC_GAIN_DEFAULT		0x0*
FRAC_CONF_RX_FRAC_GAIN	FRAC_CONF_RX_FRAC_GAIN_DEFAULT		0x0*
FRAC_CONF_TX_EN_FRAC	FRAC_CONF_TX_EN_FRAC_DEFAULT		0x0*
FRAC_CONF_RX_EN_FRAC	FRAC_CONF_RX_EN_FRAC_DEFAULT		0x0*
CONV_CODES_PUNCT_CC_PUNCT_2	CONV_CODES_PUNCT_CC_PUNCT_2_DEFAULT		0x0*
CONV_CODES_PUNCT_CC_PUNCT_1	CONV_CODES_PUNCT_CC_PUNCT_1_DEFAULT		0x0*
CONV_CODES_PUNCT_CC_PUNCT_0	CONV_CODES_PUNCT_CC_PUNCT_0_DEFAULT		0x0*

8.5.17 RF_REG10

Bit Field	Field Name	Description
31:29	FRONTEND2_RESAMPLE_PH_GAIN	Gain of the phase resampling block
28:26	FRONTEND2_RESAMPLE_RSSI_G2	Gain of the decimator in the RSSI resampling block
25:24	FRONTEND2_RESAMPLE_RSSI_G1	Gain of the interpolator in the RSSI resampling block
22	FRONTEND_EN_PHADC_DEGLITCH	If set to 1 enables the phADC deglitcher
21	FRONTEND_EN_RESAMPLE_RSSI	If set to 1 enables the RSSI resampling
20	FRONTEND_EN_RESAMPLE_PHADC	If set to 1 enables the phase resampling
19:16	FRONTEND_DIV_PHADC	Unsigned value that specifies the divider to obtain the phADC clock (and RSSI).
15:12	TX_MULT_TX_MULT_EXP	Exponent of the Tx multiplier
11:8	TX_MULT_TX_MULT_MAN	Mantissa of the Tx multiplier
7:4	TX_FRAC_CONF_TX_FRAC_DEN	
3:0	TX_FRAC_CONF_TX_FRAC_NUM	

Field Name	Value Symbol	Value Description	Hex Value
FRONTEND2_RESAMPLE_PH_GAIN	FRONTEND2_RESAMPLE_PH_GAIN_DEFAULT		0x0*
FRONTEND2_RESAMPLE_RSSI_G2	FRONTEND2_RESAMPLE_RSSI_G2_DEFAULT		0x0*
FRONTEND2_RESAMPLE_RSSI_G1	FRONTEND2_RESAMPLE_RSSI_G1_DEFAULT		0x0*
FRONTEND_EN_PHADC_DEGLITCH	FRONTEND_EN_PHADC_DEGLITCH_DEFAULT		0x0*

Field Name	Value Symbol	Value Description	Hex Value
FRONTEND_EN_RESAMPLE_RSSI	FRONTEND_EN_RESAMPLE_RSSI_ DEFAULT		0x0*
FRONTEND_EN_RESAMPLE_PHADC	FRONTEND_EN_RESAMPLE_PHADC_ DEFAULT		0x0*
FRONTEND_DIV_PHADC	FRONTEND_DIV_PHADC_DEFAULT		0x0*
TX_MULT_TX_MULT_EXP	TX_MULT_TX_MULT_EXP_DEFAULT		0x0*
TX_MULT_TX_MULT_MAN	TX_MULT_TX_MULT_MAN_DEFAULT		0x0*
TX_FRAC_CONF_TX_FRAC_DEN	TX_FRAC_CONF_TX_FRAC_DEN_ DEFAULT		0x0*
TX_FRAC_CONF_TX_FRAC_NUM	TX_FRAC_CONF_TX_FRAC_NUM_ DEFAULT		0x0*

8.5.18 RF_TX_PULSE0

Bit Field	Field Name	Description
31:24	TX_PULSE_SHAPE_1_TX_COEF4	
23:16	TX_PULSE_SHAPE_1_TX_COEF3	
15:8	TX_PULSE_SHAPE_1_TX_COEF2	
7:0	TX_PULSE_SHAPE_1_TX_COEF1	These registers specify the Tx pulse shape. The pulse shape is formed by: coef1-coef16-coef16-coef1. Since the oversampling ratio is 8, the pulse shape is 4 symbols long. Every coefficient is an 8 bits signed.

Field Name	Value Symbol	Value Description	Hex Value
TX_PULSE_SHAPE_1_TX_COEF4	TX_PULSE_SHAPE_1_TX_COEF4_ DEFAULT		0x0*
TX_PULSE_SHAPE_1_TX_COEF3	TX_PULSE_SHAPE_1_TX_COEF3_ DEFAULT		0x0*
TX_PULSE_SHAPE_1_TX_COEF2	TX_PULSE_SHAPE_1_TX_COEF2_ DEFAULT		0x0*
TX_PULSE_SHAPE_1_TX_COEF1	TX_PULSE_SHAPE_1_TX_COEF1_ DEFAULT		0x0*

8.5.19 RF_TX_PULSE1

Bit Field	Field Name	Description
31:24	TX_PULSE_SHAPE_2_TX_COEF8	
23:16	TX_PULSE_SHAPE_2_TX_COEF7	
15:8	TX_PULSE_SHAPE_2_TX_COEF6	
7:0	TX_PULSE_SHAPE_2_TX_COEF5	

Field Name	Value Symbol	Value Description	Hex Value
TX_PULSE_SHAPE_2_TX_COEF8	TX_PULSE_SHAPE_2_TX_COEF8_DEFAULT		0x0*
TX_PULSE_SHAPE_2_TX_COEF7	TX_PULSE_SHAPE_2_TX_COEF7_DEFAULT		0x0*
TX_PULSE_SHAPE_2_TX_COEF6	TX_PULSE_SHAPE_2_TX_COEF6_DEFAULT		0x0*
TX_PULSE_SHAPE_2_TX_COEF5	TX_PULSE_SHAPE_2_TX_COEF5_DEFAULT		0x0*

8.5.20 RF_TX_PULSE2

Bit Field	Field Name	Description
31:24	TX_PULSE_SHAPE_3_TX_COEF12	
23:16	TX_PULSE_SHAPE_3_TX_COEF11	
15:8	TX_PULSE_SHAPE_3_TX_COEF10	
7:0	TX_PULSE_SHAPE_3_TX_COEF9	

Field Name	Value Symbol	Value Description	Hex Value
TX_PULSE_SHAPE_3_TX_COEF12	TX_PULSE_SHAPE_3_TX_COEF12_DEFAULT		0x0*
TX_PULSE_SHAPE_3_TX_COEF11	TX_PULSE_SHAPE_3_TX_COEF11_DEFAULT		0x0*
TX_PULSE_SHAPE_3_TX_COEF10	TX_PULSE_SHAPE_3_TX_COEF10_DEFAULT		0x0*
TX_PULSE_SHAPE_3_TX_COEF9	TX_PULSE_SHAPE_3_TX_COEF9_DEFAULT		0x0*

8.5.21 RF_TX_PULSE3

Bit Field	Field Name	Description
31:24	TX_PULSE_SHAPE_4_TX_COEF16	
23:16	TX_PULSE_SHAPE_4_TX_COEF15	
15:8	TX_PULSE_SHAPE_4_TX_COEF14	
7:0	TX_PULSE_SHAPE_4_TX_COEF13	

Field Name	Value Symbol	Value Description	Hex Value
TX_PULSE_SHAPE_4_TX_COEF16	TX_PULSE_SHAPE_4_TX_COEF16_DEFAULT		0x0*
TX_PULSE_SHAPE_4_TX_COEF15	TX_PULSE_SHAPE_4_TX_COEF15_DEFAULT		0x0*

Field Name	Value Symbol	Value Description	Hex Value
TX_PULSE_SHAPE_4_TX_COEF14	TX_PULSE_SHAPE_4_TX_COEF14_DEFAULT		0x0*
TX_PULSE_SHAPE_4_TX_COEF13	TX_PULSE_SHAPE_4_TX_COEF13_DEFAULT		0x0*

8.5.22 RF_RX_PULSE

Bit Field	Field Name	Description
31:28	RX_PULSE_SHAPE_RX_COEF8	
27:24	RX_PULSE_SHAPE_RX_COEF7	
23:20	RX_PULSE_SHAPE_RX_COEF6	
19:16	RX_PULSE_SHAPE_RX_COEF5	
15:12	RX_PULSE_SHAPE_RX_COEF4	
11:8	RX_PULSE_SHAPE_RX_COEF3	
7:4	RX_PULSE_SHAPE_RX_COEF2	
3:0	RX_PULSE_SHAPE_RX_COEF1	These registers specify the Rx pulse shape. The pulse shape is formed by: coef1-coef8-coef8-coef1. Since the oversampling ratio is 8, the pulse shape is 2 symbols long. Coefficients from coef4 to coef8 are unsigned, while coef1 to coef3 are signed.

Field Name	Value Symbol	Value Description	Hex Value
RX_PULSE_SHAPE_RX_COEF8	RX_PULSE_SHAPE_RX_COEF8_DEFAULT		0x0*
RX_PULSE_SHAPE_RX_COEF7	RX_PULSE_SHAPE_RX_COEF7_DEFAULT		0x0*
RX_PULSE_SHAPE_RX_COEF6	RX_PULSE_SHAPE_RX_COEF6_DEFAULT		0x0*
RX_PULSE_SHAPE_RX_COEF5	RX_PULSE_SHAPE_RX_COEF5_DEFAULT		0x0*
RX_PULSE_SHAPE_RX_COEF4	RX_PULSE_SHAPE_RX_COEF4_DEFAULT		0x0*
RX_PULSE_SHAPE_RX_COEF3	RX_PULSE_SHAPE_RX_COEF3_DEFAULT		0x0*
RX_PULSE_SHAPE_RX_COEF2	RX_PULSE_SHAPE_RX_COEF2_DEFAULT		0x0*
RX_PULSE_SHAPE_RX_COEF1	RX_PULSE_SHAPE_RX_COEF1_DEFAULT		0x0*

8.5.23 RF_REG16

Bit Field	Field Name	Description
28:25	RX_IF_RESAMPLE_PH_IF	IF value for the phase resampler.
24:16	RX_IF_IF2_CLK_OS	IF value for the carrier recovery
15:8	FSK_FCR_AMP_1_FSK_FCR_AMP1	FSK amplitude 1 (lowest): in FSK w/o ISI is used to specify the expected amplitude. In 4-FSK is the lowest amplitude (+/-1). in FSK w/ ISI it specifies the lowest amplitude (generally, it corresponds to a sequence 0-1-0.
7:6	FILTER_GAIN_DR_LIMIT	Set the data-rate recovery limits: 00 => 0%, 01 => 3.125%, 10 => 6.25%, 11 => 12.5%
5:3	FILTER_GAIN_FILTER_GAIN_M	Mantissa of the final stage gain of the matched filter
2:0	FILTER_GAIN_FILTER_GAIN_E	Exponent of the final stage gain of the matched filter

Field Name	Value Symbol	Value Description	Hex Value
RX_IF_RESAMPLE_PH_IF	RX_IF_RESAMPLE_PH_IF_DEFAULT		0x0*
RX_IF_IF2_CLK_OS	RX_IF_IF2_CLK_OS_DEFAULT		0x0*
FSK_FCR_AMP_1_FSK_FCR_AMP1	FSK_FCR_AMP_1_FSK_FCR_AMP1_DEFAULT		0x0*
FILTER_GAIN_DR_LIMIT	FILTER_GAIN_DR_LIMIT_DEFAULT		0x0*
FILTER_GAIN_FILTER_GAIN_M	FILTER_GAIN_FILTER_GAIN_M_DEFAULT		0x0*
FILTER_GAIN_FILTER_GAIN_E	FILTER_GAIN_FILTER_GAIN_E_DEFAULT		0x0*

8.5.24 RF_REG17

Bit Field	Field Name	Description
31:24	FSK_FCR_AMP_3_FSK_FCR_AMP3	FSK amplitude 3 (highest): in 4-FSK is the high amplitude (+/-3). in FSK w/ ISI it specify the highest amplitude (generally it corresponds to a sequence 1-1-1.
23:16	FSK_FCR_AMP_2_FSK_FCR_AMP2	FSK amplitude 2 (mid): in 4-FSK is the threshold. in FSK w/ ISI it specify the mid amplitude (generally it corresponds to a sequence 0-1-1 or 1-1-0.
14:13	CARRIER_RECOVERY_EXTRA_MAX_ERR_IN_DL_SYNC	Set the maximum errors in the delay line sync detection
12	CARRIER_RECOVERY_EXTRA_EN_SYNC_OK_DELAY_LINE	If set to 1 uses the pattern_ok signal in delay line to synchronize the deserializer
11:9	CARRIER_RECOVERY_EXTRA_NC_SEL_OUT	Select the output position for the 'not-causal processing': 000 => 4 symbol, 001 => 6 symbols, 010 => 8 symbols, 011 => 12 symbols, 100 => 16 symbols, 101 => 24 symbols, 110 => 32 symbols, 111 => 40 symbols
8	CARRIER_RECOVERY_EXTRA_EN_NOT_CAUSAL	if set to 1 enables the not causal processing
6:4	CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_MAN	Mantissa of the carrier recovery frequency limit (unsigned).
2:0	CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_EXP	Exponent of the carrier recovery frequency limit (signed). Formula: carrier_offset_max=(1+m/8)*2^e/4*f_sym

Field Name	Value Symbol	Value Description	Hex Value
FSK_FCR_AMP_3_FSK_FCR_AMP3	FSK_FCR_AMP_3_FSK_FCR_AMP3_DEFAULT		0x0*
FSK_FCR_AMP_2_FSK_FCR_AMP2	FSK_FCR_AMP_2_FSK_FCR_AMP2_DEFAULT		0x0*
CARRIER_RECOVERY_EXTRA_MAX_ERR_IN_DL_SYNC	CARRIER_RECOVERY_EXTRA_MAX_ERR_IN_DL_SYNC_DEFAULT		0x0*
CARRIER_RECOVERY_EXTRA_EN_SYNC_OK_DELAY_LINE	CARRIER_RECOVERY_EXTRA_EN_SYNC_OK_DELAY_LINE_DEFAULT		0x0*
CARRIER_RECOVERY_EXTRA_NC_SEL_OUT	CARRIER_RECOVERY_EXTRA_NC_SEL_OUT_DEFAULT		0x0*

Field Name	Value Symbol	Value Description	Hex Value
CARRIER_RECOVERY_EXTRA_EN_NOT_CAUSAL	CARRIER_RECOVERY_EXTRA_EN_NOT_CAUSAL_DEFAULT		0x0*
CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_MAN	CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_MAN_DEFAULT		0x0*
CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_EXP	CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_EXP_DEFAULT		0x0*

8.5.25 RF_REG18

Bit Field	Field Name	Description
31:16	CORRECT_CFREQ_IF_CORRECT_CFREQ_IF	Unsigned value that specifies the IF for the Rx mode.
15:14	RSSI_BANK_RSSI_TRI_CK_DIV	Speed on the RSSI triangular dithering signal (cf reg RSSI_TUN)
13	RSSI_BANK_FAST_RSSI	If set to 1, the RSSI filtering is 8x faster
12	RSSI_BANK_EN_FAST_PRE_SYNC	If the packet mode is set, indicates to switch the fast modes during the preamble reception
11:8	RSSI_BANK_TAU_RSSI_FILTERING	Time constant of the RSSI filtering block: 0: 4symbols, 1: 8symbols, 2: 16 symbols, 3: 32symbols, 4: 64symbols, 5: 128symbols, 6: 256symbols, 7: 512symbols, 8: 1024symbols
4	DECISION_USE_VIT_SOFT	If set to 1 uses the Viterbi soft decoding
3:2	DECISION_VITERBI_LEN	Sets the Viterbi path length: 00: 1 bit, 01: 2 bits, 10: 4 bits, 11: 8 bits
1	DECISION_VITERBI_POW_NLIN	if set to 1, the Viterbi algorithm uses power instead of amplitude to evaluate the error on the path
0	DECISION_EN_VITERBI_GFSK	If set to 1 enables the Viterbi algorithm for the GFSK decoding; this will override the old ISI correction algorithm.

Field Name	Value Symbol	Value Description	Hex Value
CORRECT_CFREQ_IF_CORRECT_CFREQ_IF	CORRECT_CFREQ_IF_CORRECT_CFREQ_IF_DEFAULT		0x0*
RSSI_BANK_RSSI_TRI_CK_DIV	RSSI_BANK_RSSI_TRI_CK_DIV_DEFAULT		0x0*
RSSI_BANK_FAST_RSSI	RSSI_BANK_FAST_RSSI_DEFAULT		0x0*
RSSI_BANK_EN_FAST_PRE_SYNC	RSSI_BANK_EN_FAST_PRE_SYNC_DEFAULT		0x0*
RSSI_BANK_TAU_RSSI_FILTERING	RSSI_BANK_TAU_RSSI_FILTERING_DEFAULT		0x0*
DECISION_USE_VIT_SOFT	DECISION_USE_VIT_SOFT_DEFAULT		0x0*
DECISION_VITERBI_LEN	DECISION_VITERBI_LEN_DEFAULT		0x0*

Field Name	Value Symbol	Value Description	Hex Value
DECISION_VITERBI_POW_NLIN	DECISION_VITERBI_POW_NLIN_DEFAULT		0x0*
DECISION_EN_VITERBI_GFSK	DECISION_EN_VITERBI_GFSK_DEFAULT		0x0*

8.5.26 RF_REG19

Bit Field	Field Name	Description
29:28	PLL_BANK_PLL_FILTER_RES_TRIM_TX	Same as pll_filter_res_trim but for Tx case. Real value in Tx is pll_filter_res_trim xor pll_filter_res_trim_tx. If set to 0, Tx and Rx have the same value.
27:24	PLL_BANK_IQ_PLL_0_TX	Charge pump bias for Tx case. Real value in Tx is iq_pll_0 xor iq_pll_0_tx. If set to 0, Tx and Rx have the same value.
22	PLL_BANK_LOW_DR_TX	If set to 1 the Tx will work in low data-rate mode
21:20	PLL_BANK_PLL_FILTER_RES_TRIM	Allow to modify the value of the loop filter resistor R2 when bit 5 is high (TX mode): 00 => normal resistor (R_2_typ), 01 => 123%, 10 => 130% 11 => 170%
19:16	PLL_BANK_IQ_PLL_0	Charge pump bias
13	PA_PWR_MIN_PA_PWR	Sets the minimum power during the PA ramp-up: if 0 the ramp-up starts at -3, if 1 the ramp-up starts at -1
12:8	PA_PWR_PA_PWR	Signed value that sets the PA power: minimum value is -3 (-40dBm), max value is 12 (3.3dBm).
7:4	CLK_CH_FILTER_DIV_RSSI	Unsigned value that specifies the division factor for the clock controlling the RSSI.
3:0	CLK_CH_FILTER_DIV_FILT	Unsigned value that specifies the division factor for the clock controlling the channel filter.

Field Name	Value Symbol	Value Description	Hex Value
PLL_BANK_PLL_FILTER_RES_TRIM_TX	PLL_BANK_PLL_FILTER_RES_TRIM_TX_DEFAULT		0x0*
PLL_BANK_IQ_PLL_0_TX	PLL_BANK_IQ_PLL_0_TX_DEFAULT		0x0*
PLL_BANK_LOW_DR_TX	PLL_BANK_LOW_DR_TX_DEFAULT		0x0*
PLL_BANK_PLL_FILTER_RES_TRIM	PLL_BANK_PLL_FILTER_RES_TRIM_DEFAULT		0x0*
PLL_BANK_IQ_PLL_0	PLL_BANK_IQ_PLL_0_DEFAULT		0x0*
PA_PWR_MIN_PA_PWR	PA_PWR_MIN_PA_PWR_DEFAULT		0x0*
PA_PWR_PA_PWR	PA_PWR_PA_PWR_DEFAULT		0x0*
CLK_CH_FILTER_DIV_RSSI	CLK_CH_FILTER_DIV_RSSI_DEFAULT		0x0*
CLK_CH_FILTER_DIV_FILT	CLK_CH_FILTER_DIV_FILT_DEFAULT		0x0*

8.5.27 RF_REG1A

Bit Field	Field Name	Description
31:28	ATT_CTRL_ATT_CTRL_MAX	Maximum attenuation level in AGC algorithm
27:24	ATT_CTRL_SET_RX_ATT_CTRL	Attenuation level if the AGC is bypassed
23:22	RSSI_CTRL_AGC_DECAY_TAU	Time constant of the decay speed; high values corresponds to a slow decay
21	RSSI_CTRL_AGC_USE_LNA	If set to 1 the AGC algorithm uses the LNA bias.
20	RSSI_CTRL_AGC_MODE	Select the AGC algorithm: 0 -> old algorithm, 1 -> new algorithm
19:18	RSSI_CTRL_AGC_WAIT	Sets the wait time of the AGC after switching between states: 00 => don't wait, 01 => wait 1x RSSI filtering period, 10 => wait 2x RSSI filtering period, 11 => wait 3x RSSI filtering period
17	RSSI_CTRL_PAYLOAD_BLOCKS_AGC	If set to 1, the AGC is blocked during the payload
16	RSSI_CTRL_BYPASS_AGC	If set to 1, the AGC algorithm is bypassed
12:8	FILTER_BIAS_IQ_FI_BW	Bias for the bandwidth of the channel filter
4:0	FILTER_BIAS_IQ_FI_FC	Bias for the central frequency of the channel filter

Field Name	Value Symbol	Value Description	Hex Value
ATT_CTRL_ATT_CTRL_MAX	ATT_CTRL_ATT_CTRL_MAX_DEFAULT		0x0*
ATT_CTRL_SET_RX_ATT_CTRL	ATT_CTRL_SET_RX_ATT_CTRL_DEFAULT		0x0*
RSSI_CTRL_AGC_DECAY_TAU	RSSI_CTRL_AGC_DECAY_TAU_DEFAULT		0x0*
RSSI_CTRL_AGC_USE_LNA	RSSI_CTRL_AGC_USE_LNA_DEFAULT		0x0*
RSSI_CTRL_AGC_MODE	RSSI_CTRL_AGC_MODE_DEFAULT		0x0*
RSSI_CTRL_AGC_WAIT	RSSI_CTRL_AGC_WAIT_DEFAULT		0x0*
RSSI_CTRL_PAYLOAD_BLOCKS_AGC	RSSI_CTRL_PAYLOAD_BLOCKS_AGC_DEFAULT		0x0*
RSSI_CTRL_BYPASS_AGC	RSSI_CTRL_BYPASS_AGC_DEFAULT		0x0*
FILTER_BIAS_IQ_FI_BW	FILTER_BIAS_IQ_FI_BW_DEFAULT		0x0*
FILTER_BIAS_IQ_FI_FC	FILTER_BIAS_IQ_FI_FC_DEFAULT		0x0*

8.5.28 RF_REG1B

Bit Field	Field Name	Description
31	IEEE802154_OPTS_EN_DW_TEST	If set to 1 enables the Tx data-whitening before the convolutional code block
30:29	IEEE802154_OPTS_BER_CLK_MODE	sets the clock output mode for BER mode or RW mode: 00 => data change on falling edge, 01 => data change on rising edge, 10 => clock signal is a toggled signal, 11 => enable signal from clock recovery
28	IEEE802154_OPTS_RX_DATA_NOT_SAMPLED	If set to 1, the signal rx_data in testmodes is not sampled. Used for debug purposes
27	IEEE802154_OPTS_EN_L2F_RX	if set to 1 enables the frequency to linear conversion in the Rx side (always controlled by the en_802154_l2f configuration bit).

Bit Field	Field Name	Description
26:24	IEEE802154_OPTS_C2B_THR	Threshold of the chip2bit correlator of the IEEE 802.15.4 protocol.
23:20	AGC_PEAK_DET_PEAK_DET_TAU	Time constant of the peak detector monostable circuit; if set to 0 the monostable is bypassed
19:18	AGC_PEAK_DET_PEAK_DET_THR_LOW	Threshold for the low level of the peak detector: 0 => 0, 1 => 1, 2 => 2, 3 => N.A.
17	AGC_PEAK_DET_PEAK_DET_THR_HIGH	Threshold for the high level of the peak detector: 0 => 2, 1 => 3
16	AGC_PEAK_DET_EN_AGC_PEAK	If set to 1 enables the AGC peak detector
15:8	AGC_THR_HIGH_AGC_THR_HIGH	AGC threshold high level
7:0	AGC_THR_LOW_AGC_THR_LOW	AGC threshold low level

Field Name	Value Symbol	Value Description	Hex Value
IEEE802154_OPTS_EN_DW_TEST	IEEE802154_OPTS_EN_DW_TEST_DEFAULT		0x0*
IEEE802154_OPTS_BER_CLK_MODE	IEEE802154_OPTS_BER_CLK_MODE_DEFAULT		0x0*
IEEE802154_OPTS_RX_DATA_NOT_SAMPLED	IEEE802154_OPTS_RX_DATA_NOT_SAMPLED_DEFAULT		0x0*
IEEE802154_OPTS_EN_L2F_RX	IEEE802154_OPTS_EN_L2F_RX_DEFAULT		0x0*
IEEE802154_OPTS_C2B_THR	IEEE802154_OPTS_C2B_THR_DEFAULT		0x0*
AGC_PEAK_DET_PEAK_DET_TAU	AGC_PEAK_DET_PEAK_DET_TAU_DEFAULT		0x0*
AGC_PEAK_DET_PEAK_DET_THR_LOW	AGC_PEAK_DET_PEAK_DET_THR_LOW_DEFAULT		0x0*
AGC_PEAK_DET_PEAK_DET_THR_HIGH	AGC_PEAK_DET_PEAK_DET_THR_HIGH_DEFAULT		0x0*
AGC_PEAK_DET_EN_AGC_PEAK	AGC_PEAK_DET_EN_AGC_PEAK_DEFAULT		0x0*
AGC_THR_HIGH_AGC_THR_HIGH	AGC_THR_HIGH_AGC_THR_HIGH_DEFAULT		0x0*
AGC_THR_LOW_AGC_THR_LOW	AGC_THR_LOW_AGC_THR_LOW_DEFAULT		0x0*

8.5.29 RF_AGC_LUT1

Bit Field	Field Name	Description
31:22	AGC_LUT_1_AGC_LEVEL_2_LO	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
21:11	AGC_LUT_1_AGC_LEVEL_1	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
10:0	AGC_LUT_1_AGC_LEVEL_0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.

Field Name	Value Symbol	Value Description	Hex Value
AGC_LUT_1_AGC_LEVEL_2_LO	AGC_LUT_1_AGC_LEVEL_2_LO_DEFAULT		0x0*
AGC_LUT_1_AGC_LEVEL_1	AGC_LUT_1_AGC_LEVEL_1_DEFAULT		0x0*
AGC_LUT_1_AGC_LEVEL_0	AGC_LUT_1_AGC_LEVEL_0_DEFAULT		0x0*

8.5.30 RF_AGC_LUT2

Bit Field	Field Name	Description
31:23	AGC_LUT_2_AGC_LEVEL_5_LO	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
22:12	AGC_LUT_2_AGC_LEVEL_4	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
11:1	AGC_LUT_2_AGC_LEVEL_3	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
0	AGC_LUT_2_AGC_LEVEL_2_HI	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.

Field Name	Value Symbol	Value Description	Hex Value
AGC_LUT_2_AGC_LEVEL_5_LO	AGC_LUT_2_AGC_LEVEL_5_LO_DEFAULT		0x0*
AGC_LUT_2_AGC_LEVEL_4	AGC_LUT_2_AGC_LEVEL_4_DEFAULT		0x0*
AGC_LUT_2_AGC_LEVEL_3	AGC_LUT_2_AGC_LEVEL_3_DEFAULT		0x0*
AGC_LUT_2_AGC_LEVEL_2_HI	AGC_LUT_2_AGC_LEVEL_2_HI_DEFAULT		0x0*

8.5.31 RF_AGC_LUT3

Bit Field	Field Name	Description
31:24	AGC_LUT_3_AGC_LEVEL_8_LO	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
23:13	AGC_LUT_3_AGC_LEVEL_7	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
12:2	AGC_LUT_3_AGC_LEVEL_6	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
1:0	AGC_LUT_3_AGC_LEVEL_5_HI	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.

Field Name	Value Symbol	Value Description	Hex Value
AGC_LUT_3_AGC_LEVEL_8_LO	AGC_LUT_3_AGC_LEVEL_8_LO_DEFAULT		0x0*
AGC_LUT_3_AGC_LEVEL_7	AGC_LUT_3_AGC_LEVEL_7_DEFAULT		0x0*

Field Name	Value Symbol	Value Description	Hex Value
AGC_LUT_3_AGC_LEVEL_6	AGC_LUT_3_AGC_LEVEL_6_DEFAULT		0x0*
AGC_LUT_3_AGC_LEVEL_5_HI	AGC_LUT_3_AGC_LEVEL_5_HI_ DEFAULT		0x0*

8.5.32 RF_AGC_LUT4

Bit Field	Field Name	Description
31:25	AGC_LUT_4_AGC_LEVEL_11_LO	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
24:14	AGC_LUT_4_AGC_LEVEL_10	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
13:3	AGC_LUT_4_AGC_LEVEL_9	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
2:0	AGC_LUT_4_AGC_LEVEL_8_HI	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.

Field Name	Value Symbol	Value Description	Hex Value
AGC_LUT_4_AGC_LEVEL_11_LO	AGC_LUT_4_AGC_LEVEL_11_LO_ DEFAULT		0x0*
AGC_LUT_4_AGC_LEVEL_10	AGC_LUT_4_AGC_LEVEL_10_DEFAULT		0x0*
AGC_LUT_4_AGC_LEVEL_9	AGC_LUT_4_AGC_LEVEL_9_DEFAULT		0x0*
AGC_LUT_4_AGC_LEVEL_8_HI	AGC_LUT_4_AGC_LEVEL_8_HI_ DEFAULT		0x0*

8.5.33 RF_REG20

Bit Field	Field Name	Description
31:28	TIMINGS_3_T_DLL	Time needed by the DLL blocks to switch on.
27:24	TIMINGS_3_T_PLL_TX	Time needed by the PLL blocks in Tx mode to switch on.
23:20	TIMINGS_2_T_SUBBAND_TX	Time needed by the subband algorithm to calibrate in Tx.
19:16	TIMINGS_2_T_TX_RF	Time needed by the Tx RF blocks to switch on.
14:12	TIMINGS_1_T_GRANULARITY_TX	Fixes the granularity of the timer in Tx mode. The granularity is given by (2^(t_granularity-2))x1us
10:8	TIMINGS_1_T_GRANULARITY_RX	Fixes the granularity of the timer in Rx mode. The granularity is given by (2^(t_granularity))x1us
3:0	AGC_LUT_5_AGC_LEVEL_11_HI	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.

Field Name	Value Symbol	Value Description	Hex Value
TIMINGS_3_T_DLL	TIMINGS_3_T_DLL_DEFAULT		0x0*
TIMINGS_3_T_PLL_TX	TIMINGS_3_T_PLL_TX_DEFAULT		0x1*
TIMINGS_2_T_SUBBAND_TX	TIMINGS_2_T_SUBBAND_TX_DEFAULT		0xF*

Field Name	Value Symbol	Value Description	Hex Value
TIMINGS_2_T_TX_RF	TIMINGS_2_T_TX_RF_DEFAULT		0xF*
TIMINGS_1_T_GRANULARITY_TX	TIMINGS_1_T_GRANULARITY_TX_ DEFAULT		0x0*
TIMINGS_1_T_GRANULARITY_RX	TIMINGS_1_T_GRANULARITY_RX_ DEFAULT		0x1*
AGC_LUT_5_AGC_LEVEL_11_HI	AGC_LUT_5_AGC_LEVEL_11_HI_ DEFAULT		0xF*

8.5.34 RF_AGC_ATT1

Bit Field	Field Name	Description
31:30	AGC_ATT_1_AGC_ATT_AB_LO	
29:27	AGC_ATT_1_AGC_ATT_9A	
26:24	AGC_ATT_1_AGC_ATT_89	
23:21	AGC_ATT_1_AGC_ATT_78	
20:18	AGC_ATT_1_AGC_ATT_67	
17:15	AGC_ATT_1_AGC_ATT_56	
14:12	AGC_ATT_1_AGC_ATT_45	
11:9	AGC_ATT_1_AGC_ATT_34	
8:6	AGC_ATT_1_AGC_ATT_23	
5:3	AGC_ATT_1_AGC_ATT_12	
2:0	AGC_ATT_1_AGC_ATT_01	These fields specify the attenuation levels

Field Name	Value Symbol	Value Description	Hex Value
AGC_ATT_1_AGC_ATT_AB_LO	AGC_ATT_1_AGC_ATT_AB_LO_DEFAULT		0x3*
AGC_ATT_1_AGC_ATT_9A	AGC_ATT_1_AGC_ATT_9A_DEFAULT		0x3*
AGC_ATT_1_AGC_ATT_89	AGC_ATT_1_AGC_ATT_89_DEFAULT		0x3*
AGC_ATT_1_AGC_ATT_78	AGC_ATT_1_AGC_ATT_78_DEFAULT		0x3*
AGC_ATT_1_AGC_ATT_67	AGC_ATT_1_AGC_ATT_67_DEFAULT		0x3*
AGC_ATT_1_AGC_ATT_56	AGC_ATT_1_AGC_ATT_56_DEFAULT		0x3*
AGC_ATT_1_AGC_ATT_45	AGC_ATT_1_AGC_ATT_45_DEFAULT		0x3*
AGC_ATT_1_AGC_ATT_34	AGC_ATT_1_AGC_ATT_34_DEFAULT		0x3*
AGC_ATT_1_AGC_ATT_23	AGC_ATT_1_AGC_ATT_23_DEFAULT		0x3*
AGC_ATT_1_AGC_ATT_12	AGC_ATT_1_AGC_ATT_12_DEFAULT		0x3*
AGC_ATT_1_AGC_ATT_01	AGC_ATT_1_AGC_ATT_01_DEFAULT		0x3*

8.5.35 RF_REG22

Bit Field	Field Name	Description
29	TIMING_FAST_RX_EN_FAST_RX_TXFILT	If set to 1 enables filter Tx configuration for the fast Rx PLL
28	TIMING_FAST_RX_EN_FAST_RX	If set to 1 enables the fast Rx PLL
27:24	TIMING_FAST_RX_T_RX_FAST_CHP	Time to switch off the fast CHP in Rx mode
23:20	TIMINGS_5_T_RX_RF	Time needed by the Rx RF blocks to switch on.
19:16	TIMINGS_5_T_RX_BB	Time needed by the Rx BB blocks to switch on.
15:12	TIMINGS_4_T_SUBBAND_RX	Time needed by the subband algorithm to calibrate in Rx
11:8	TIMINGS_4_T_PLL_RX	Time needed by the PLL blocks in Rx mode to switch on.
1	AGC_ATT_2_AGC_ATT_1DB	If set to 1 the attenuation is specified by 1dB steps from 4dB to 11dB
0	AGC_ATT_2_AGC_ATT_AB_HI	

Field Name	Value Symbol	Value Description	Hex Value
TIMING_FAST_RX_EN_FAST_RX_TXFILT	TIMING_FAST_RX_EN_FAST_RX_TXFILT_DEFAULT		0x0*
TIMING_FAST_RX_EN_FAST_RX	TIMING_FAST_RX_EN_FAST_RX_DEFAULT		0x0*
TIMING_FAST_RX_T_RX_FAST_CHP	TIMING_FAST_RX_T_RX_FAST_CHP_DEFAULT		0x0*
TIMINGS_5_T_RX_RF	TIMINGS_5_T_RX_RF_DEFAULT		0x0*
TIMINGS_5_T_RX_BB	TIMINGS_5_T_RX_BB_DEFAULT		0x0*
TIMINGS_4_T_SUBBAND_RX	TIMINGS_4_T_SUBBAND_RX_DEFAULT		0x0*
TIMINGS_4_T_PLL_RX	TIMINGS_4_T_PLL_RX_DEFAULT		0x0*
AGC_ATT_2_AGC_ATT_1DB	AGC_ATT_2_AGC_ATT_1DB_DEFAULT		0x0*
AGC_ATT_2_AGC_ATT_AB_HI	AGC_ATT_2_AGC_ATT_AB_HI_DEFAULT		0x0*

8.5.36 RF_REG23

Bit Field	Field Name	Description
31:28	BIAS_1_IQ_RXTX_3	PrePA Casc bias
27:24	BIAS_1_IQ_RXTX_2	PrePA In bias
23:20	BIAS_0_IQ_RXTX_1	PA backoff bias
19:16	BIAS_0_IQ_RXTX_0	PA bias
14:12	INTERFACE_CONF_APB_WAIT_STATE	Select the number of wait states during the APB transaction
9:8	INTERFACE_CONF_SPI_SELECT	Select the spi mode: 00 legacy spi, 01 advanced spi, 10 BLIM4SME spi
7	TIMEOUT_EN_RX_TIMEOUT	If set to 1 enables the timeout of the Rx when the system is on FSM mode

Bit Field	Field Name	Description
6:4	TIMEOUT_T_TIMEOUT_GR	Granularity of the timer in timeout Rx mode
3:0	TIMEOUT_T_RX_TIMEOUT	Time that has to occur before the timeout.

Field Name	Value Symbol	Value Description	Hex Value
BIAS_1_IQ_RXTX_3	BIAS_1_IQ_RXTX_3_DEFAULT		0x0*
BIAS_1_IQ_RXTX_2	BIAS_1_IQ_RXTX_2_DEFAULT		0x0*
BIAS_0_IQ_RXTX_1	BIAS_0_IQ_RXTX_1_DEFAULT		0x0*
BIAS_0_IQ_RXTX_0	BIAS_0_IQ_RXTX_0_DEFAULT		0x0*
INTERFACE_CONF_APB_WAIT_STATE	INTERFACE_CONF_APB_WAIT_STATE_DEFAULT		0x0*
INTERFACE_CONF_SPI_SELECT	INTERFACE_CONF_SPI_SELECT_DEFAULT		0x0*
TIMEOUT_EN_RX_TIMEOUT	TIMEOUT_EN_RX_TIMEOUT_DEFAULT		0x0*
TIMEOUT_T_TIMEOUT_GR	TIMEOUT_T_TIMEOUT_GR_DEFAULT		0x0*
TIMEOUT_T_RX_TIMEOUT	TIMEOUT_T_RX_TIMEOUT_DEFAULT		0x0*

8.5.37 RF_REG24

Bit Field	Field Name	Description
31:28	BIAS_5_IQ_PLL_4_RX	VCO bias for Rx
27:24	BIAS_5_IQ_PLL_4_TX	VCO bias for Tx
23:20	BIAS_4_IQ_PLL_2	Sub-band comparator bias
19:16	BIAS_4_IQ_PLL_1	Dynamic divider bias
15:12	BIAS_3_IQ_RXTX_8	IFA ctrl_c bias
11:8	BIAS_3_IQ_RXTX_7	IFA ctrl_r bias
7:4	BIAS_2_IQ_RXTX_6	VCOM_MX bias
3:0	BIAS_2_IQ_RXTX_5	VCOM_LO bias

Field Name	Value Symbol	Value Description	Hex Value
BIAS_5_IQ_PLL_4_RX	BIAS_5_IQ_PLL_4_RX_DEFAULT		0x0*
BIAS_5_IQ_PLL_4_TX	BIAS_5_IQ_PLL_4_TX_DEFAULT		0x0*
BIAS_4_IQ_PLL_2	BIAS_4_IQ_PLL_2_DEFAULT		0x0*
BIAS_4_IQ_PLL_1	BIAS_4_IQ_PLL_1_DEFAULT		0x0*
BIAS_3_IQ_RXTX_8	BIAS_3_IQ_RXTX_8_DEFAULT		0x0*
BIAS_3_IQ_RXTX_7	BIAS_3_IQ_RXTX_7_DEFAULT		0x0*
BIAS_2_IQ_RXTX_6	BIAS_2_IQ_RXTX_6_DEFAULT		0x0*
BIAS_2_IQ_RXTX_5	BIAS_2_IQ_RXTX_5_DEFAULT		0x0*

8.5.38 RF_REG25

Bit Field	Field Name	Description
31:28	BIAS_9_IQ_BB_6	Peak detector threshold bias 0
27:24	BIAS_9_IQ_BB_5	Peak detector bias
23:20	BIAS_8_IQ_BB_4	RSSI_D bias
19:16	BIAS_8_IQ_BB_3	RSSI_G bias
15:12	BIAS_7_IQ_BB_2	ACD_L bias
11:8	BIAS_7_IQ_BB_1	ACD_C bias
7:4	BIAS_6_IQ_BB_0	ACD_O bias
3:0	BIAS_6_IQ_PLL_3	DLL bias

Field Name	Value Symbol	Value Description	Hex Value
BIAS_9_IQ_BB_6	BIAS_9_IQ_BB_6_DEFAULT		0x0*
BIAS_9_IQ_BB_5	BIAS_9_IQ_BB_5_DEFAULT		0x0*
BIAS_8_IQ_BB_4	BIAS_8_IQ_BB_4_DEFAULT		0x0*
BIAS_8_IQ_BB_3	BIAS_8_IQ_BB_3_DEFAULT		0x0*
BIAS_7_IQ_BB_2	BIAS_7_IQ_BB_2_DEFAULT		0x0*
BIAS_7_IQ_BB_1	BIAS_7_IQ_BB_1_DEFAULT		0x0*
BIAS_6_IQ_BB_0	BIAS_6_IQ_BB_0_DEFAULT		0x0*
BIAS_6_IQ_PLL_3	BIAS_6_IQ_PLL_3_DEFAULT		0x0*

8.5.39 RF_REG26

Bit Field	Field Name	Description
28	SD_MASH_MASH_ENABLE	Enable the sigma delta mash
27	SD_MASH_MASH_DITHER	Enable dithering on the sigma delta mash
26:25	SD_MASH_MASH_ORDER	Order of the sigma delta mash
24	SD_MASH_MASH_RSTB	Reset of the sigma delta mash (active low)
23:20	BIAS_12_LNA_AGC_BIAS_3	LNA bias for AGC lvl 3
19:16	BIAS_12_LNA_AGC_BIAS_2	LNA bias for AGC lvl 2
15:12	BIAS_11_LNA_AGC_BIAS_1	LNA bias for AGC lvl 1
11:8	BIAS_11_LNA_AGC_BIAS_0	LNA bias for AGC lvl 0
7:4	BIAS_10_IQ_BB_8	Peak detector threshold bias 1
3:0	BIAS_10_IQ_BB_7	Peak detector threshold bias 2

Field Name	Value Symbol	Value Description	Hex Value
SD_MASH_MASH_ENABLE	SD_MASH_MASH_ENABLE_DEFAULT		0x0*
SD_MASH_MASH_DITHER	SD_MASH_MASH_DITHER_DEFAULT		0x0*
SD_MASH_MASH_ORDER	SD_MASH_MASH_ORDER_DEFAULT		0x0*

Field Name	Value Symbol	Value Description	Hex Value
SD_MASH_MASH_RSTB	SD_MASH_MASH_RSTB_DEFAULT		0x0*
BIAS_12_LNA_AGC_BIAS_3	BIAS_12_LNA_AGC_BIAS_3_DEFAULT		0x0*
BIAS_12_LNA_AGC_BIAS_2	BIAS_12_LNA_AGC_BIAS_2_DEFAULT		0x0*
BIAS_11_LNA_AGC_BIAS_1	BIAS_11_LNA_AGC_BIAS_1_DEFAULT		0x0*
BIAS_11_LNA_AGC_BIAS_0	BIAS_11_LNA_AGC_BIAS_0_DEFAULT		0x0*
BIAS_10_IQ_BB_8	BIAS_10_IQ_BB_8_DEFAULT		0x0*
BIAS_10_IQ_BB_7	BIAS_10_IQ_BB_7_DEFAULT		0x0*

8.5.40 RF_REG27

Bit Field	Field Name	Description
31	CTRL_ADC_ONE_CK_RSSI_PHADC	If set to 1, the RSSI and the phADC share the same clock
30:29	CTRL_ADC_PHADC_DELLATCH	phADC delay latch trimming
28:24	CTRL_ADC_CTRL_ADC	bits(1:0) => phADC reset delay, bits(3:2) phADC clock delay, bit(4) phADC latch idle
19	BIAS_EN_2_EN_PTAT	Enable PTAT
18:16	BIAS_EN_2_EN_BIAS_BB_HI	Bias enable for BB (same order as biases)
15:12	BIAS_EN_1_EN_BIAS_BB_LO	Bias enable for BB (same order as biases)
11:7	BIAS_EN_1_EN_BIAS_PLL	Bias enable for PLL (same order as biases)
6:0	BIAS_EN_1_EN_BIAS_RXTX	Bias enable for RxTx (same order as biases)

Field Name	Value Symbol	Value Description	Hex Value
CTRL_ADC_ONE_CK_RSSI_PHADC	CTRL_ADC_ONE_CK_RSSI_PHADC_DEFAULT		0x0*
CTRL_ADC_PHADC_DELLATCH	CTRL_ADC_PHADC_DELLATCH_DEFAULT		0x0*
CTRL_ADC_CTRL_ADC	CTRL_ADC_CTRL_ADC_DEFAULT		0x0*
BIAS_EN_2_EN_PTAT	BIAS_EN_2_EN_PTAT_DEFAULT		0x0*
BIAS_EN_2_EN_BIAS_BB_HI	BIAS_EN_2_EN_BIAS_BB_HI_DEFAULT		0x0*
BIAS_EN_1_EN_BIAS_BB_LO	BIAS_EN_1_EN_BIAS_BB_LO_DEFAULT		0x0*
BIAS_EN_1_EN_BIAS_PLL	BIAS_EN_1_EN_BIAS_PLL_DEFAULT		0x0*
BIAS_EN_1_EN_BIAS_RXTX	BIAS_EN_1_EN_BIAS_RXTX_DEFAULT		0x0*

8.5.41 RF_REG28

Bit Field	Field Name	Description
31	CTRL_RX_SWITCH_LP	If set to 1 switch the low-pass filter in the Rx chain
30	CTRL_RX_USE_PEAK_DETECTOR	If set to 1, the peak detector is powered on during the Rx by the FSM
29	CTRL_RX_START_MIX_ON_CAL	If set to 1, the mixer is enabled during the sub-band selection phase
28:24	CTRL_RX_CTRL_RX	bits(1:0) => resonance 1 LNA, bits(3:2) => resonance 2 LNA, bit(4) => IFA PTAT-R only
23:20	SWCAP_FSM_SB_CAP_RX	VCO subband selection (Rx in FSM mode)

Bit Field	Field Name	Description
19:16	SWCAP_FSM_SB_CAP_TX	VCO subband selection (Tx in FSM mode)
10	DLL_CTRL_CK_LAST_SEL_DELAY	
9	DLL_CTRL_CK_FIRST_SEL_DELAY	
8	DLL_CTRL_CK_EXT_SEL	Low: input clock comes from ck_xtal pin (default). High: input clock comes from ck_ext pin
7	DLL_CTRL_CK_DIG_EN	Debug: enable to use the alternate ck_dig pin to output the PLL reference clock signal
6	DLL_CTRL_CK_TEST_EN	Debug: enable to output on GPIO the PLL reference clock signal via ck_test pin
5	DLL_CTRL_TOO_FAST_ENB	When low, enable auxiliary wide lock range phase detector when fast mode locking is enabled (fast_enb = 0). When high, only the narrow lock range phase detector is enabled and bit 2 (fast_enb) must be high to avoid false frequency lock (slow mode locking)
4	DLL_CTRL_LOCKED_DET_EN	Enable reference frequency multiplier locked detector. When this signal is high, the dll_locked output goes high when the output multiplied clock is nearly about three times the frequency of the input clock.
3	DLL_CTRL_LOCKED_AUTO_CHECK_EN	If for some reason the reference frequency multiplier is out of lock (usually because some input clocks from ck_xtal or ck_ext are missing) and this signal is high, the frequency multiplier will try to lock again automatically. Otherwise, a manual reset should be performed via dll_rstb input (see Table 3) to relock the frequency multiplier. This mode only works if bit 4 is also high (locked detector enabled, see below)
2	DLL_CTRL_FAST_ENB	Enable, when low, fast mode locking of the reference frequency multiplier (default). Bit 5 must also be set low in this mode of operation (see below)
1:0	DLL_CTRL_CK_SEL	Selection of the clock used as frequency reference of the PLL (also to ck_test and ck_dig outputs): 00 => ref = ck_xtal or ck_ext (if bit 8 is high), 01 => ref = same as ck_sel = 00 if dll_en = 0, otherwise frequency(ref) = 3x frequency(ck_xtal) or 3x frequency(ck_ext) (if bit 8 is high), 10 => ref = same as ck_sel = 01 but output frequency divided by 2 (used in normal RX mode when dll_en = 0), 11 => ref = same as ck_sel = 01 but output frequency divided by 5 (used for RX mode with external signal at 132 MHz when dll_en = 0)

Field Name	Value Symbol	Value Description	Hex Value
CTRL_RX_SWITCH_LP	CTRL_RX_SWITCH_LP_DEFAULT		0x0*
CTRL_RX_USE_PEAK_DETECTOR	CTRL_RX_USE_PEAK_DETECTOR_DEFAULT		0x0*
CTRL_RX_START_MIX_ON_CAL	CTRL_RX_START_MIX_ON_CAL_DEFAULT		0x0*
CTRL_RX_CTRL_RX	CTRL_RX_CTRL_RX_DEFAULT		0x0*
SWCAP_FSM_SB_CAP_RX	SWCAP_FSM_SB_CAP_RX_DEFAULT		0x0*
SWCAP_FSM_SB_CAP_TX	SWCAP_FSM_SB_CAP_TX_DEFAULT		0x0*
DLL_CTRL_CK_LAST_SEL_DELAY	DLL_CTRL_CK_LAST_SEL_DELAY_DEFAULT		0x0*
DLL_CTRL_CK_FIRST_SEL_DELAY	DLL_CTRL_CK_FIRST_SEL_DELAY_DEFAULT		0x0*
DLL_CTRL_CK_EXT_SEL	DLL_CTRL_CK_EXT_SEL_DEFAULT		0x0*

Field Name	Value Symbol	Value Description	Hex Value
DLL_CTRL_CK_DIG_EN	DLL_CTRL_CK_DIG_EN_DEFAULT		0x0*
DLL_CTRL_CK_TEST_EN	DLL_CTRL_CK_TEST_EN_DEFAULT		0x0*
DLL_CTRL_TOO_FAST_ENB	DLL_CTRL_TOO_FAST_ENB_DEFAULT		0x0*
DLL_CTRL_LOCKED_DET_EN	DLL_CTRL_LOCKED_DET_EN_DEFAULT		0x0*
DLL_CTRL_LOCKED_AUTO_CHECK_EN	DLL_CTRL_LOCKED_AUTO_CHECK_EN_DEFAULT		0x0*
DLL_CTRL_FAST_ENB	DLL_CTRL_FAST_ENB_DEFAULT		0x0*
DLL_CTRL_CK_SEL	DLL_CTRL_CK_SEL_DEFAULT		0x2*

8.5.42 RF_PLL_CTRL

Bit Field	Field Name	Description
31:24	XTAL_TRIM_XTAL_TRIM	trimming of the xtal: 5MSB thermometric, 3LSB direct
20	PLL_CTRL_2_PLL_RX_48MEG	If set to 1 the PLL is set to 48MHz in Rx instead of 24MHz (need also to change ck_sel)
19	PLL_CTRL_2_SWCAP_TX_SAME_RX	If set to 1, in case of swcap_fsm=1, the register for Rx and Tx swcap is the same
18	PLL_CTRL_2_SWCAP_FSM	If set to 1 use the swcap_fsm register as reference for the sub-band selection
17	PLL_CTRL_2_DLL_RSTB	Reset signal of the DLL (active low)
16	PLL_CTRL_2_VCO_SUBBAND_TRIM_HI	VCO sub-band selection bits
15:13	PLL_CTRL_1_VCO_SUBBAND_TRIM_LO	VCO sub-band selection bits
12	PLL_CTRL_1_SUB_SEL_OFFS_EN	Add offset to sub-band selection comparator
11	PLL_CTRL_1_DIV2_CLKVCO_TEST_EN	Debug: VCO signal divided by the programmable divider is divided by a: 0 => division ratio set to 1, 1 => division ratio set to 2; before to be outputted to ck_div_test
10	PLL_CTRL_1_VCODIV_CLK_TEST_EN	Debug: enable to output on GPIO the VCO signal divided by the programmable divider
9	PLL_CTRL_1_EN_LOW_CHP_BIAS	When high (recommended), allows use of a lower bias current for the required output pumping current.
8	PLL_CTRL_1_CHP_DEAD_ZONE_EN	Debug: enable charge-pump dead zone (degraded PLL characteristics for test)
7:6	PLL_CTRL_1_CHP_CURR_OFFSET_TRIM	Debug: charge-pump offset current values selection bits (see bit 6 to enable this mode): 00 => d_phi = 15, 01 => d_phi=22.5, 10 => d_phi = 30, 11 => d_phi = 60. Also sets the bias current of the common mode control block of the charge-pump. Must be sets to 01 to ensure a proper operation of the VCO tuning voltage comparator for sub-band selection, if used
5	PLL_CTRL_1_HIGH_BW_FILTER_EN	Enable the PLL filter high bandwidth needed in TX (must be high together with bit 4 in TX, low in RX)
4	PLL_CTRL_1_FAST_CHP_EN	Enable the high current output of the charge-pump for PLL TX high bandwidth mode (must be high together with bit 5 in TX, low in RX)

Bit Field	Field Name	Description
3:2	PLL_CTRL_1_CHP_MODE_TRIM	Charge-pump active if 00 else this allow to open the PLL and force the VCO tune voltage to reach: 01 => minimum frequency inside sub-band selection, 10 => medium frequency inside sub-band selection, 11 => maximum frequency inside sub-band selection.
1	PLL_CTRL_1_CHP_CMC_EN	Enable the common mode control block of the charge-pump. Must be high to ensure proper operation of the VCO tuning voltage comparator for sub-band selection, if used
0	PLL_CTRL_1_CHP_CURR_ OFFSET_EN	Debug: enable the charge-pump offset current (see bits 7:6 for offset current value)

Field Name	Value Symbol	Value Description	Hex Value
XTAL_TRIM_XTAL_TRIM	XTAL_TRIM_XTAL_TRIM_DEFAULT		0x80*
PLL_CTRL_2_PLL_RX_48MEG	PLL_CTRL_2_PLL_RX_48MEG_ DEFAULT		0x0*
PLL_CTRL_2_SWCAP_TX_SAME_RX	PLL_CTRL_2_SWCAP_TX_SAME_RX_ DEFAULT		0x0*
PLL_CTRL_2_SWCAP_FSM	PLL_CTRL_2_SWCAP_FSM_DEFAULT		0x0*
PLL_CTRL_2_DLL_RSTB	PLL_CTRL_2_DLL_RSTB_DEFAULT		0x0*
PLL_CTRL_2_VCO_SUBBAND_TRIM_ HI	PLL_CTRL_2_VCO_SUBBAND_TRIM_ HI_DEFAULT		0x0*
PLL_CTRL_1_VCO_SUBBAND_TRIM_ LO	PLL_CTRL_1_VCO_SUBBAND_TRIM_ LO_DEFAULT		0x0*
PLL_CTRL_1_SUB_SEL_OFFS_EN	PLL_CTRL_1_SUB_SEL_OFFS_EN_ DEFAULT		0x0*
PLL_CTRL_1_DIV2_CLKVCO_TEST_ EN	PLL_CTRL_1_DIV2_CLKVCO_TEST_ EN_DEFAULT		0x0*
PLL_CTRL_1_VCODIV_CLK_TEST_EN	PLL_CTRL_1_VCODIV_CLK_TEST_ EN_DEFAULT		0x0*
PLL_CTRL_1_EN_LOW_CHP_BIAS	PLL_CTRL_1_EN_LOW_CHP_BIAS_ DEFAULT		0x0*
PLL_CTRL_1_CHP_DEAD_ZONE_EN	PLL_CTRL_1_CHP_DEAD_ZONE_EN_ DEFAULT		0x0*
PLL_CTRL_1_CHP_CURR_OFFSET_ TRIM	PLL_CTRL_1_CHP_CURR_OFFSET_ TRIM_DEFAULT		0x0*
PLL_CTRL_1_HIGH_BW_FILTER_EN	PLL_CTRL_1_HIGH_BW_FILTER_EN_ DEFAULT		0x0*
PLL_CTRL_1_FAST_CHP_EN	PLL_CTRL_1_FAST_CHP_EN_DEFAULT		0x0*
PLL_CTRL_1_CHP_MODE_TRIM	PLL_CTRL_1_CHP_MODE_TRIM_ DEFAULT		0x0*
PLL_CTRL_1_CHP_CMC_EN	PLL_CTRL_1_CHP_CMC_EN_DEFAULT		0x0*
PLL_CTRL_1_CHP_CURR_OFFSET_EN	PLL_CTRL_1_CHP_CURR_OFFSET_EN_ DEFAULT		0x0*

8.5.43 RF_REG2A

Bit Field	Field Name	Description
28	ENABLES_SEPARATE_PPA_CASC	If set to 1, the en PPA cascode bit is independent from the en PA
27:22	ENABLES_EN_RXTX	Enable signals: 0 => LNA, 1 => LNA, 2 => IFA, 3 => Tx, 4 => PA, 5 => PPA casc
21:16	ENABLES_EN_BB	Enable signals for the BB: 0 => Filter, 1 => Filter central frequency bias, 2 => Filter bandwidth bias, 3 => ADC, 4 => RSSI, 5 => peak detector
15:13	RSSI_TUN_RSSI_TUN_GAIN	RSSI tuning for gain
12:8	RSSI_TUN_RSSI_ODD_OFFSET	RSSI tuning for odd stages: offset to the even triangular wave
7:4	RSSI_TUN_RSSI_EVEN_MAX	RSSI tuning for even stages: maximum value of the triangular wave. If max = min, static signal.
3:0	RSSI_TUN_RSSI_EVEN_MIN	RSSI tuning for even stages: minimum value of the triangular wave

Field Name	Value Symbol	Value Description	Hex Value
ENABLES_SEPARATE_PPA_CASC	ENABLES_SEPARATE_PPA_CASC_DEFAULT		0x0*
ENABLES_EN_RXTX	ENABLES_EN_RXTX_DEFAULT		0x0*
ENABLES_EN_BB	ENABLES_EN_BB_DEFAULT		0x0*
RSSI_TUN_RSSI_TUN_GAIN	RSSI_TUN_RSSI_TUN_GAIN_DEFAULT		0x3*
RSSI_TUN_RSSI_ODD_OFFSET	RSSI_TUN_RSSI_ODD_OFFSET_DEFAULT		0x0*
RSSI_TUN_RSSI_EVEN_MAX	RSSI_TUN_RSSI_EVEN_MAX_DEFAULT		0x7*
RSSI_TUN_RSSI_EVEN_MIN	RSSI_TUN_RSSI_EVEN_MIN_DEFAULT		0x7*

8.5.44 RF_XTAL_CTRL

Bit Field	Field Name	Description
31:28	XTAL_CTRL_XO_THR_HIGH	High threshold for xtal trimming
27:24	XTAL_CTRL_XO_THR_LOW	Low threshold for xtal trimming
23:22	XTAL_CTRL_XO_A_S_CURR_SEL_HIGH	Value of after_startup_curr_sel when level is higher than xo_thr_high
21:20	XTAL_CTRL_XO_A_S_CURR_SEL_LOW	Value of after_startup_curr_sel when level is lower than xo_thr_low
18	XTAL_CTRL_XTAL_CTRL_BYPASS	Bypass the Xtal control algorithm
17	XTAL_CTRL_DIG_CLK_IN_SEL	If set to 1 selects the clk_in_dig signal for the digital block, otherwise the internal xtal
16	XTAL_CTRL_XO_EN_B_REG	Xtal oscillator enable (active low)
15:14	XTAL_CTRL_XTAL_CKDIV	Xtal trimming speed: 00 => 43us, 01 => 85us, 10 => 171us, 11 => 341us
13	XTAL_CTRL_CLK_OUT_EN_B	When high, disable the output clock to go to main IP (clk_out output stay low).
12	XTAL_CTRL_REG_VALUE_SEL	When low, all main ctrl signals are used instead of corresponding ctrl signal or some control bits of xtal_reg. They are: xo_en_b, ext_clk_mode and lp_mode. When high, corresponding ctrl signal and some control bits of xtal_reg are used instead of main ctrl signals. They are: xo_en_b_reg, ext_clk_mode (bit 0) and lp_mode (bit 1).

Bit Field	Field Name	Description
11:10	XTAL_CTRL_AFTERSTARTUP_CURR_SEL	Selection of the current before amplitude stabilization but after starting-up in active transistors of the core oscillator: '00': typ. 0.15 mA, '01': typ. 0.24 mA, '10': typ. 0.40 mA, '11': typ. 0.61 mA
9:8	XTAL_CTRL_STARTUP_CURR_SEL	Selection of the starting-up current in active transistors of the core oscillator: '00': typ. 0.41 mA, '01': typ. 0.59 mA, '10': typ. 0.88 mA, '11': typ. 1.24 mA
7	XTAL_CTRL_INV_CLK_DIG	Invert clock on clk_dig output
6	XTAL_CTRL_INV_CLK_PLL	Invert clock on clk_pll output
5	XTAL_CTRL_FORCE_CLK_READY	Debug: allow to force output clocks on clk_pll, clk_dig and clk_out (if these outputs are enabled) and bypass the xtal internal clock detector that gates these clock outputs.
4	XTAL_CTRL_CLK_DIG_EN_B	When high, disable the output clock to go to digital (clk_dig output stay low).
3	XTAL_CTRL_BUFF_EN_B	When low (and if xtal_en_b(_reg) is low), the xtal buffer is enabled otherwise it is disabled. Could be used to decrease the power consumption of the xtal while maintaining oscillation in the xtal oscillator
2	XTAL_CTRL_HP_MODE	When high, bias current in the clock buffer is increased compared to normal operation (high bandwidth mode in 132 MHz clock input buffer).
1	XTAL_CTRL_LP_MODE	When high, bias current in the clock buffer is reduced compared to normal operation (low power mode). Usable only if bit 12 is high (see below) otherwise it is bypassed by lp_mode pin input on main interface
0	XTAL_CTRL_EXT_CLK_MODE	When high, allow to uses xtal_p (and eventually xtal_n) has external clock input(s). The XTAL oscillator core is disabled. Usable only if bit 12 is high (see below) otherwise it is bypassed by ext_clk_mode pin input on main interface

Field Name	Value Symbol	Value Description	Hex Value
XTAL_CTRL_XO_THR_HIGH	XTAL_CTRL_XO_THR_HIGH_DEFAULT		0xC*
XTAL_CTRL_XO_THR_LOW	XTAL_CTRL_XO_THR_LOW_DEFAULT		0x3*
XTAL_CTRL_XO_A_S_CURR_SEL_HIGH	XTAL_CTRL_XO_A_S_CURR_SEL_HIGH_DEFAULT		0x2*
XTAL_CTRL_XO_A_S_CURR_SEL_LOW	XTAL_CTRL_XO_A_S_CURR_SEL_LOW_DEFAULT		0x0*
XTAL_CTRL_XTAL_CTRL_BYPASS	XTAL_CTRL_XTAL_CTRL_BYPASS_DEFAULT		0x0*
XTAL_CTRL_DIG_CLK_IN_SEL	XTAL_CTRL_DIG_CLK_IN_SEL_DEFAULT		0x0*
XTAL_CTRL_XO_EN_B_REG	XTAL_CTRL_ENABLE_OSCILLATOR		0x0
	XTAL_CTRL_DISABLE_OSCILLATOR		0x1*
XTAL_CTRL_XTAL_CKDIV	XTAL_CTRL_XTAL_CKDIV_DEFAULT		0x0*
XTAL_CTRL_CLK_OUT_EN_B	XTAL_CTRL_CLK_OUT_EN_B_DEFAULT		0x0*
XTAL_CTRL_REG_VALUE_SEL	XTAL_CTRL_REG_VALUE_SEL_EXTERNAL		0x0*
	XTAL_CTRL_REG_VALUE_SEL_INTERNAL		0x1
XTAL_CTRL_AFTERSTARTUP_CURR_SEL	XTAL_CTRL_AFTERSTARTUP_CURR_SEL_DEFAULT		0x1*

Field Name	Value Symbol	Value Description	Hex Value
XTAL_CTRL_STARTUP_CURR_SEL	XTAL_CTRL_STARTUP_CURR_SEL_DEFAULT		0x1*
XTAL_CTRL_INV_CLK_DIG	XTAL_CTRL_INV_CLK_DIG_DEFAULT		0x0*
XTAL_CTRL_INV_CLK_PLL	XTAL_CTRL_INV_CLK_PLL_DEFAULT		0x0*
XTAL_CTRL_FORCE_CLK_READY	XTAL_CTRL_FORCE_CLK_READY_DEFAULT		0x0*
XTAL_CTRL_CLK_DIG_EN_B	XTAL_CTRL_CLK_DIG_EN_B_DEFAULT		0x0*
XTAL_CTRL_BUFF_EN_B	XTAL_CTRL_BUFF_EN_B_DEFAULT		0x0*
XTAL_CTRL_HP_MODE	XTAL_CTRL_HP_MODE_DEFAULT		0x0*
XTAL_CTRL_LP_MODE	XTAL_CTRL_LP_MODE_DEFAULT		0x0*
XTAL_CTRL_EXT_CLK_MODE	XTAL_CTRL_EXT_CLK_MODE_DEFAULT		0x0*

8.5.45 RF_REG2C

Bit Field	Field Name	Description
31:24	SUBBAND_OFFSET_SB_OFFSET	Offset to add in frequency count in order to compensate the offset of the varicap.
23:20	SWCAP_LIM_SB_MAX_VAL	maximum subband value in linear search subband (freq and comp)
19:16	SWCAP_LIM_SB_MIN_VAL	minimum subband value in linear search subband (freq and comp)
15	SUBBAND_CONF_SB_FLL_MODE	Enables the FLL mode for the subband selection (overrides other settings)
14	SUBBAND_CONF_SB_INV_BAND	invert the meaning of sb_high and sb_low
13:12	SUBBAND_CONF_SB_FREQ_CNT	The length to count in frequency mode: 00 => 256 (Rx: 10.7us, Tx: 2.13us),01 => 512 (Rx: 21.3us, Tx: 4.26us),11 => 1024 (Rx: 42.7us, Tx: 8.53us),01 => 4096 (Rx: 171us, Tx: 34.1us)
11:10	SUBBAND_CONF_SB_WAIT_T	time to wait to the PLL to settle: 00 => Rx 8us, Tx 2us, 01 => Rx 12us, Tx 3us, 10 => Rx 16us, Tx 4us, 11 => Rx 24us, Tx 6u
9:8	SUBBAND_CONF_SB_MODE	sub-band algorithm mode: 00 => SAR w/ comparators, 01 => linear w/ comparators, 00 => SAR w/ frequency ratios, 01 => linear w/ frequency ratios
5:4	PA_CONF_SW_CN	Harmonic 2 notch tuning
3	PA_CONF_TX_SWITCHPA	If set to 1, enables the PA only with the digital block, otherwise it's the RF Tx timing
2	PA_CONF_TX_0DBM	If set to 1 enables the PA, otherwise only the PPA is used (-20dBm)
1:0	PA_CONF_CTRL_PA	N.U.

Field Name	Value Symbol	Value Description	Hex Value
SUBBAND_OFFSET_SB_OFFSET	SUBBAND_OFFSET_SB_OFFSET_DEFAULT		0x0*
SWCAP_LIM_SB_MAX_VAL	SWCAP_LIM_SB_MAX_VAL_DEFAULT		0x0*
SWCAP_LIM_SB_MIN_VAL	SWCAP_LIM_SB_MIN_VAL_DEFAULT		0x0*
SUBBAND_CONF_SB_FLL_MODE	SUBBAND_CONF_SB_FLL_MODE_DEFAULT		0x0*
SUBBAND_CONF_SB_INV_BAND	SUBBAND_CONF_SB_INV_BAND_DEFAULT		0x0*
SUBBAND_CONF_SB_FREQ_CNT	SUBBAND_CONF_SB_FREQ_CNT_DEFAULT		0x0*
SUBBAND_CONF_SB_WAIT_T	SUBBAND_CONF_SB_WAIT_T_DEFAULT		0x0*
SUBBAND_CONF_SB_MODE	SUBBAND_CONF_SB_MODE_DEFAULT		0x0*
PA_CONF_SW_CN	PA_CONF_SW_CN_DEFAULT		0x0*

Field Name	Value Symbol	Value Description	Hex Value
PA_CONF_TX_SWITCHPA	PA_CONF_TX_SWITCHPA_DEFAULT		0x0*
PA_CONF_TX_0DBM	PA_CONF_TX_0DBM_DEFAULT		0x0*
PA_CONF_CTRL_PA	PA_CONF_CTRL_PA_DEFAULT		0x0*

8.5.46 RF_REG2D

Bit Field	Field Name	Description
31	SUBBAND_CORR_SUBBAND_CORR_EN	Enable the subband correction
30:28	SUBBAND_CORR_SUBBAND_CORR_RX	Subband correction in Rx
26:24	SUBBAND_CORR_SUBBAND_CORR_TX	Subband correction in Tx
23	PLL_CONF_TX_NRX_INV_CLK_PLL_TX	
22	PLL_CONF_TX_NRX_INV_CLK_DIG_TX	
21:20	PLL_CONF_TX_NRX_CK_SEL_TX	Xor value between Tx and Rx for the ck_sel field of register DLL_CTRL
18:17	PLL_CONF_TX_NRX_CHP_CURR_OFF_TRIM_TX	
16	PLL_CONF_TX_NRX_CHP_CURR_OFF_EN_TX	
15	PA_RAMPUP_FULL_PA_RAMPUP	If set to 1, the PA rampup uses the PA backoff enable bit (from -40 dBm)
14:12	PA_RAMPUP_DEL_PA_RAMPUP	time to wait to start the ramp-up after the PA enable is detected
11:10	PA_RAMPUP_TAU_PA_RAMPUP	time constant of the Ramp-up/Ramp-down
9	PA_RAMPUP_EN_PA_RAMPDOWN	if set to 1 enables the PA ramp-down. Only valid in case of ramp-up
8	PA_RAMPUP_EN_PA_RAMPUP	if set to 1 enables the PA ramp-up
7:3	MISC_SPARES	Unused bits
2:1	MISC_RSSI_PRE_ATT	RSSI pre-attenuator: 00 => 0dB, 01 => 4dB, 10 => 8dB, 11 => 12dB
0	MISC_XTAL_LOW_CLK_READY_TH_EN	XTAL: if set to 1, the clk_ready threshold is set to a lower value

Field Name	Value Symbol	Value Description	Hex Value
SUBBAND_CORR_SUBBAND_CORR_EN	SUBBAND_CORR_SUBBAND_CORR_EN_DEFAULT		0x0*
SUBBAND_CORR_SUBBAND_CORR_RX	SUBBAND_CORR_SUBBAND_CORR_RX_DEFAULT		0x0*
SUBBAND_CORR_SUBBAND_CORR_TX	SUBBAND_CORR_SUBBAND_CORR_TX_DEFAULT		0x0*
PLL_CONF_TX_NRX_INV_CLK_PLL_TX	PLL_CONF_TX_NRX_INV_CLK_PLL_TX_DEFAULT		0x0*
PLL_CONF_TX_NRX_INV_CLK_DIG_TX	PLL_CONF_TX_NRX_INV_CLK_DIG_TX_DEFAULT		0x0*
PLL_CONF_TX_NRX_CK_SEL_TX	PLL_CONF_TX_NRX_CK_SEL_TX_DEFAULT		0x3*

Field Name	Value Symbol	Value Description	Hex Value
PLL_CONF_TX_NRX_CHP_CURR_OFF_TRIM_TX	PLL_CONF_TX_NRX_CHP_CURR_OFF_TRIM_TX_DEFAULT		0x0*
PLL_CONF_TX_NRX_CHP_CURR_OFF_EN_TX	PLL_CONF_TX_NRX_CHP_CURR_OFF_EN_TX_DEFAULT		0x0*
PA_RAMPUP_FULL_PA_RAMPUP	PA_RAMPUP_FULL_PA_RAMPUP_DEFAULT		0x0*
PA_RAMPUP_DEL_PA_RAMPUP	PA_RAMPUP_DEL_PA_RAMPUP_DEFAULT		0x0*
PA_RAMPUP_TAU_PA_RAMPUP	PA_RAMPUP_TAU_PA_RAMPUP_DEFAULT		0x0*
PA_RAMPUP_EN_PA_RAMPDOWN	PA_RAMPUP_EN_PA_RAMPDOWN_DEFAULT		0x0*
PA_RAMPUP_EN_PA_RAMPUP	PA_RAMPUP_EN_PA_RAMPUP_DEFAULT		0x0*
MISC_SPARES	MISC_SPARES_DEFAULT		0x0*
MISC_RSSI_PRE_ATT	MISC_RSSI_PRE_ATT_DEFAULT		0x0*
MISC_XTAL_LOW_CLK_READY_TH_EN	MISC_XTAL_LOW_CLK_READY_TH_EN_DEFAULT		0x0*

8.5.47 RF_REG2E

Bit Field	Field Name	Description
31:24	RSSI_DETECT_ABS_THR_RSSI_DET_ABS_THR	Threshold used for absolute RSSI detection
23:16	RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR	Threshold used for differential RSSI detection
14	DEMOD_CTRL_EN_DELLINE_SYNC_DET	If set to 1 enable the sync word detection in the delay line. This implies that nc_sel_out = 0x7
13	DEMOD_CTRL_RSSI_DET_FILT	Add an additional filtering on the RSSI value
12	DEMOD_CTRL_EN_FAST_CLK_RECOV	If set to 1 speed up the clock recovery during the rest of the preamble
11	DEMOD_CTRL_EN_MIN_MAX_MF	If set to 1 enables the min max algo after the matched filter
10	DEMOD_CTRL_EN_PRE_SYNC	If set to 1 enables the sync detection on the non-delayed path; not working in 4-FSK
9	DEMOD_CTRL_BLOCK_RSSI_DET	If set to 1 blocks the rssi detection during the slow-down period
8	DEMOD_CTRL_EARLY_FINE_RECOV	If set to 1 enables the early fine recovery after the packet detection or pre-sync
7:6	RSSI_DETECT_RSSI_DET_CR_LEN	Number of samples to estimate the carrier offset: 0 -> 32, 1 -> 64, 2 -> 128, 3->256
5:4	RSSI_DETECT_RSSI_DET_WAIT	Symbols to wait after the RSSI detection: 00 -> 0, 01 -> 1, 10 -> 2, 11 -> 4
3:2	RSSI_DETECT_RSSI_DET_DIFF_LL	Set the distance between the actual value and the subtracted one (0->1 sample,1->2 samples, etc)
1	RSSI_DETECT_EN_ABS_RSSI_DETECT	If set to 1 enables the absolute RSSI detection
0	RSSI_DETECT_EN_DIFF_RSSI_DETECT	If set to 1 enables the differential RSSI detection

Field Name	Value Symbol	Value Description	Hex Value
RSSI_DETECT_ABS_THR_RSSI_DET_ABS_THR	RSSI_DETECT_ABS_THR_RSSI_DET_ABS_THR_DEFAULT		0x0*
RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR	RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR_DEFAULT		0x0*
DEMOD_CTRL_EN_DELLINE_SYNC_DET	DEMOD_CTRL_EN_DELLINE_SYNC_DET_DEFAULT		0x0*
DEMOD_CTRL_RSSI_DET_FILT	DEMOD_CTRL_RSSI_DET_FILT_DEFAULT		0x0*
DEMOD_CTRL_EN_FAST_CLK_RECOV	DEMOD_CTRL_EN_FAST_CLK_RECOV_DEFAULT		0x0*
DEMOD_CTRL_EN_MIN_MAX_MF	DEMOD_CTRL_EN_MIN_MAX_MF_DEFAULT		0x0*
DEMOD_CTRL_EN_PRE_SYNC	DEMOD_CTRL_EN_PRE_SYNC_DEFAULT		0x0*
DEMOD_CTRL_BLOCK_RSSI_DET	DEMOD_CTRL_BLOCK_RSSI_DET_DEFAULT		0x0*
DEMOD_CTRL_EARLY_FINE_RECOV	DEMOD_CTRL_EARLY_FINE_RECOV_DEFAULT		0x0*
RSSI_DETECT_RSSI_DET_CR_LEN	RSSI_DETECT_RSSI_DET_CR_LEN_DEFAULT		0x0*
RSSI_DETECT_RSSI_DET_WAIT	RSSI_DETECT_RSSI_DET_WAIT_DEFAULT		0x0*
RSSI_DETECT_RSSI_DET_DIFF_LL	RSSI_DETECT_RSSI_DET_DIFF_LL_DEFAULT		0x0*
RSSI_DETECT_RSSI_DET_EN_ABS	RSSI_DETECT_RSSI_DET_EN_ABS_DEFAULT		0x0*
RSSI_DETECT_RSSI_DET_EN_DIFF	RSSI_DETECT_RSSI_DET_EN_DIFF _DEFAULT		0x0*

8.5.48 RF_REG2F

Bit Field	Field Name	Description
26:24	CK_DIV_1_6_CK_DIV_1_6	Clock division factor for ck_div_1_6
22:0	RESERVED	

Field Name	Value Symbol	Value Description	Hex Value
CK_DIV_1_6_CK_DIV_1_6	CK_DIV_1_6_NO_CLOCK	This means that no clock is generated	0x0*
	CK_DIV_1_6_PRESCALE_1		0x1
	CK_DIV_1_6_PRESCALE_2		0x2
	CK_DIV_1_6_PRESCALE_3		0x3
	CK_DIV_1_6_PRESCALE_4		0x4
	CK_DIV_1_6_PRESCALE_5		0x5
	CK_DIV_1_6_PRESCALE_6		0x6
	CK_DIV_1_6_PRESCALE_7		0x7
PADS_PE_DS_GPIO_DS	PADS_PE_DS_GPIO_DS_DEFAULT		0x0*

Field Name	Value Symbol	Value Description	Hex Value
PADS_PE_DS_GPIO_PE	PADS_PE_DS_GPIO_PE_DEFAULT		0x0*
PADS_PE_DS_NRESET_PE	PADS_PE_DS_NRESET_PE_DEFAULT		0x0*
PADS_PE_DS_SPI_MISO_PE	PADS_PE_DS_SPI_MISO_PE_DEFAULT		0x0*
PADS_PE_DS_SPI_MOSI_PE	PADS_PE_DS_SPI_MOSI_PE_DEFAULT		0x0*
PADS_PE_DS_SPI_SCLK_PE	PADS_PE_DS_SPI_SCLK_PE_DEFAULT		0x0*
PADS_PE_DS_SPI_CS_N_PE	PADS_PE_DS_SPI_CS_N_PE_DEFAULT		0x0*
SUBBAND_FLL_SB_FLL_DITHER	SUBBAND_FLL_SB_FLL_DITHER_ DEFAULT		0x0*
SUBBAND_FLL_SB_FLL_CIC_TAU	SUBBAND_FLL_SB_FLL_CIC_TAU_ DEFAULT		0x0*
SUBBAND_FLL_SB_FLL_PH_4_N8	SUBBAND_FLL_SB_FLL_PH_4_N8_ DEFAULT		0x0*
SUBBAND_FLL_SB_FLL_WAIT	SUBBAND_FLL_SB_FLL_WAIT_ DEFAULT		0x0*
SYNC_WORD_CORR_EN_SYNC_WORD_CORR	SYNC_WORD_CORR_EN_SYNC_WORD_ CORR_DEFAULT		0x0*
SYNC_WORD_CORR_SYNC_WORD_BIAS	SYNC_WORD_CORR_SYNC_WORD_BIAS_ DEFAULT		0x0*

8.5.49 RF_REG30

Bit Field	Field Name	Description
31:25	RXFIFO_STATUS_BIST	Start the bist test on the Rx FIFO (code 0x5d)
31:30	RXFIFO_STATUS_BIST_ERRORS	Indicate the BIST error: 00 => no error, 01 => error in checkboard test, 10 => error in inversed checkboard test, 11 => error in decoder test
29	RXFIFO_STATUS_NEAR_UNDERFLOW	Is set to 1 if the Rx FIFO is close to the underflow
28	RXFIFO_STATUS_NEAR_OVERFLOW	Is set to 1 if the Rx FIFO is close to the overflow
27	RXFIFO_STATUS_UNDERFLOW	Is set to 1 if there has been an underflow
26	RXFIFO_STATUS_OVERFLOW	Is set to 1 if there has been an overflow
25	RXFIFO_STATUS_FULL	Is set to 1 if the Rx FIFO is full
24	RXFIFO_STATUS_FLUSH	If set to 1 the Rx FIFO is flushed
24	RXFIFO_STATUS_EMPTY	Is set to 1 if the Rx FIFO is empty
23:17	TXFIFO_STATUS_BIST	Start the bist test on the Tx FIFO (code 0x5d)
23:22	TXFIFO_STATUS_BIST_ERRORS	Indicate the BIST error: 00 => no error, 01 => error in checkboard test, 10 => error in inversed checkboard test, 11 => error in decoder test
21	TXFIFO_STATUS_NEAR_UNDERFLOW	Is set to 1 if the Tx FIFO is close to the underflow
20	TXFIFO_STATUS_NEAR_OVERFLOW	Is set to 1 if the Tx FIFO is close to the overflow
19	TXFIFO_STATUS_UNDERFLOW	Is set to 1 if there has been an underflow
18	TXFIFO_STATUS_OVERFLOW	Is set to 1 if there has been an overflow
17	TXFIFO_STATUS_FULL	Is set to 1 if the Tx FIFO is full
16	TXFIFO_STATUS_FLUSH	If set to 1 the Tx FIFO is flushed
16	TXFIFO_STATUS_EMPTY	Is set to 1 if the Tx FIFO is empty

Bit Field	Field Name	Description
10	`FSM_STATUS_TX_NRX`	Is set to 0 if the radio is in Rx mode, to 1 if in Tx mode
9:8	`FSM_STATUS_STATUS`	Status of the FSM: 00 => Idle, 01 => Tx mode, 10 => Rx mode, 11 => Suspend
3	`FSM_MODE_RESET`	If set to 1, the FSM is reset. If mode is set to 0 the FSM is reset asynchronously. If is set to 1 the Tx or Rx (depending on tx_nrx) is stopped gently via the serializer or the deserializer
2	`FSM_MODE_TX_NRX`	Sets the Radio in Tx (1) or Rx (0) mode
2	`FSM_MODE_RX_MODE`	The field stay with value 1 as long as the reception isn't over
1:0	`FSM_MODE_MODE`	Sets the FSM mode: 00: nothing is done, 01: activate, 10: calibrate the PLL, 11: calibrate the PLL then Tx/Rx
1	`FSM_MODE_TX_MODE`	The field keep the value 1 as long as the transmission isn't over
0	`FSM_MODE_N_IDLE`	The field is set to 1 if the FSM is not in the Idle mode.

8.5.50 RF_REG31

Bit Field	Field Name	Description
31:24	`RSSI_MAX_RSSI_MAX`	Maximum RSSI value over a filtering period
23:16	`RSSI_MIN_RSSI_MIN`	Minimum RSSI value over a filtering period
15:8	`RXFIFO_COUNT_RX_COUNT`	Number of bytes in the Rx FIFO
7:0	`TXFIFO_COUNT_TX_COUNT`	Number of bytes in the Tx FIFO

8.5.51 RF_REG32

Bit Field	Field Name	Description
30:28	`RX_ATT_LEVEL_RX_ATT_LEVEL_PKT_LVL`	Rx attenuation level (AGC level) during the packet reception
26:24	`RX_ATT_LEVEL_RX_ATT_LEVEL`	Rx attenuation level (AGC level)
23:16	`RSSI_AVG_RSSI_AVG`	Filtered RSSI value
15:8	`DR_ERR_IND_DR_ERR_IND`	Data-rate error indicator
7:0	`RSSI_PKT_RSSI_PKT`	Filtered RSSI value sampled during the packet reception

8.5.52 RF_TXFIFO

Bit Field	Field Name	Description
7:0	`TXFIFO_TX_DATA`	Data to be sent

8.5.53 RF_RXFIFO

Bit Field	Field Name	Description
7:0	`RXFIFO_RX_DATA`	Received data

8.5.54 RF_DESER_STATUS

Bit Field	Field Name	Description
7	DESER_STATUS_SIGNAL_RECEIVING	Is set to 1 if the deserializer is on
6	DESER_STATUS_SYNC_DETECTED	Is set to 1 is the sync word (pattern) has been detected
5	DESER_STATUS_WAIT_SYNC	Is set to 1 if the deserializer is waiting the sync word
4	DESER_STATUS_IS_ADDRESS_BR	Is set to 1 if the received address is the broadcast address.
3	DESER_STATUS_PKT_LEN_ERR	Is set to 1 in case of the packet length is longer than the maximum acceptable packet length
2	DESER_STATUS_ADDRESS_ERR	Is set to 1 in case of an address error
1	DESER_STATUS_CRC_ERR	Is set to 1 in case of a CRC error
0	DESER_STATUS_DESER_FINISH	Is set to 1 when the deserializer has finished

8.5.55 RF_IRQ_STATUS

Bit Field	Field Name	Description
5	IRQ_STATUS_FLAG_RXFIFO	Is set to 1 when the IRQ RXFIFO is active
4	IRQ_STATUS_FLAG_TXFIFO	Is set to 1 when the IRQ TXFIFO is active
3	IRQ_STATUS_FLAG_SYNC	Is set to 1 when the IRQ SYNC is active
2	IRQ_STATUS_FLAG_RECEIVED	Is set to 1 when the IRQ RECEIVED is active
1	IRQ_STATUS_FLAG_RXSTOP	Is set to 1 when the IRQ RXSTOP is active
0	IRQ_STATUS_FLAG_TX	Is set to 1 when the IRQ TX is active

8.5.56 RF_REG37

Bit Field	Field Name	Description
31:16	FEI_PKT_FEI_PKT	Frequency error indicator sampled during the packet reception.
15:0	FEI_FEI_OUT	Frequency error indicator

8.5.57 RF_REG38

Bit Field	Field Name	Description
31:24	LINK_QUAL_PKT_LINK_QUALITY_PKT	Link quality indicator sampled during the packet reception. Note that the Viterbi algorithm as to be enabled.
23:16	LINK_QUAL_LINK_QUALITY	Instantaneous link quality indicator. Note that the Viterbi algorithm as to be enabled.
15:0	FEI_AFC_FEI_AFC	Frequency error indicator sampled during the AFC.

8.5.58 RF_REG39

Bit Field	Field Name	Description
13	ANALOG_INFO_XTAL_FINISH	If set to 1, the Xtal algorithm has finished
12	ANALOG_INFO_DLL_LOCKED	DLL locked signal

Bit Field	Field Name	Description
11	ANALOG_INFO_CLK_DIG_READY	Ready signal of the digital clock
10	ANALOG_INFO_CLK_PLL_READY	Ready signal of the PLL clock
9	ANALOG_INFO_SUBBAND_HI	Status of the subband comparator Hi
8	ANALOG_INFO_SUBBAND_LO	Status of the subband comparator Lo
7:0	SUBBAND_ERR_SB_FLL_ERR	distance from the subband center (only available with the FLL method)

Field Name	Value Symbol	Value Description	Hex Value
ANALOG_INFO_XTAL_FINISH	ANALOG_INFO_XTAL_TRIM_RUNNING		0x0*
	ANALOG_INFO_XTAL_TRIM_FINISHED		0x1
ANALOG_INFO_DLL_LOCKED	ANALOG_INFO_DLL_UNLOCKED		0x0*
	ANALOG_INFO_DLL_LOCKED		0x1
ANALOG_INFO_CLK_DIG_READY	ANALOG_INFO_CLK_DIG_NOT_READY		0x0*
	ANALOG_INFO_CLK_DIG_READY		0x1

8.5.59 RF_REVISION

Bit Field	Field Name	Description
29:24	CHIP_ID	Version of the chip: 0x00: v1, 0x10: v2A, 0x11: v2B, 0x12: v2C, 0x13: v2D, 0x14: v2E, 0x20: v3

CHAPTER 9

Bluetooth Low Energy Baseband

9.1 OVERVIEW

The Bluetooth low energy baseband controller is responsible for real-time operations, and performs packet and frame processing. Its architecture is illustrated in Figure 17.

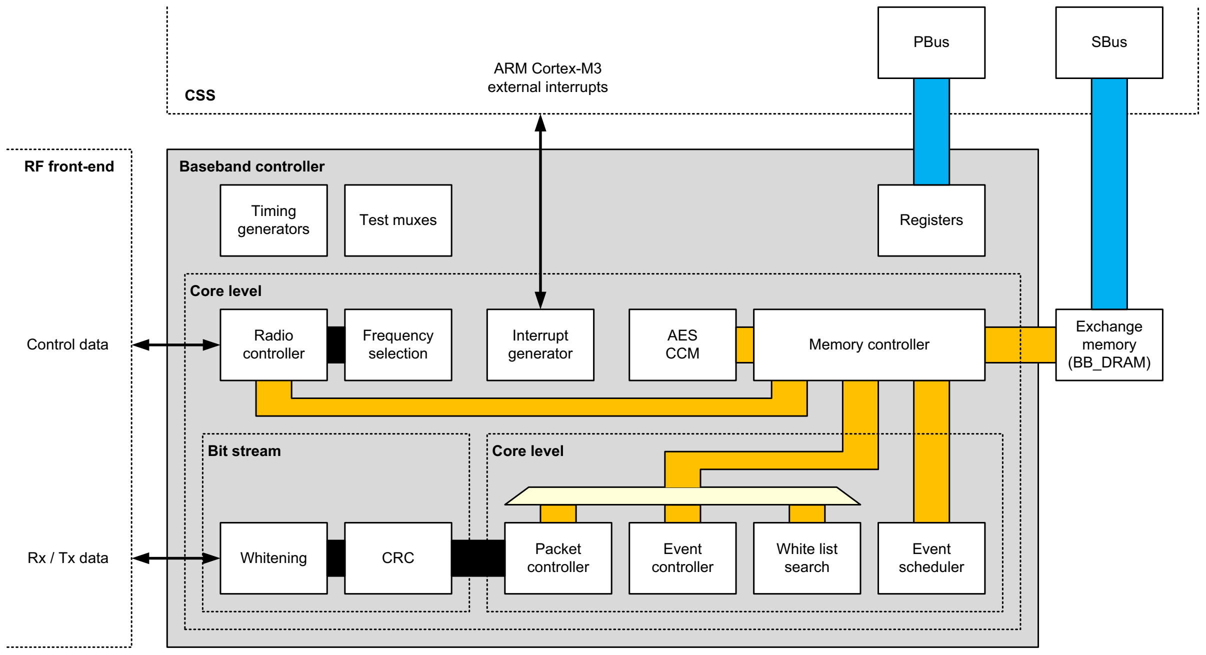


Figure 17. Bluetooth Low Energy Baseband Controller Architecture

The baseband controller communicates with:

- The Arm Cortex-M3 processor, through:
  - The Bluetooth baseband controller sends interrupts to the Arm Cortex-M3 processor. For information on the interrupt support, see Section 14.1, “Nested Vectored Interrupt Controller (NVIC)” on page 400.
  - The S-Bus and the DMA that access the exchange memory (`BB_DRAM0` and `BB_DRAM1`) parallel to the baseband controller. The memory access is managed by an external arbiter that can be configured. For information on these memories, see Section 7.2.6, “Bluetooth Low Energy Baseband (BB) Memory Usage” on page 105.
  - The control and configuration registers listed in Section 9.4, “Baseband Registers” on page 213.

- The RF front-end through the support interfaces, as described in Section 11.9, “Support Interfaces” on page 346.

The Bluetooth baseband hardware fills the logical layers of the Bluetooth low energy data transfer architecture shown in Figure 18. This supplements the physical layers implemented by the RF front-end (refer to Chapter 8, “RF Front-End” on page 131) and the L2CAP and host layers that are supported by the Bluetooth stack firmware (refer to the *“Bluetooth Stack and Profiles”* chapter from the *RSL10 Firmware Reference*).

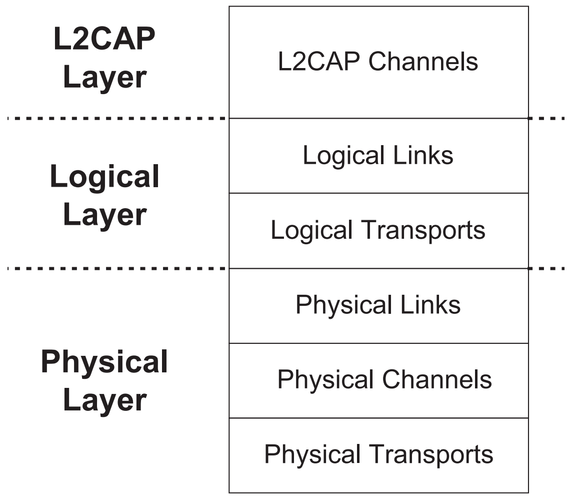


Figure 18. Bluetooth Generic Data Transport Architecture

The Bluetooth low energy baseband controller is implemented with the features listed in Table 17.

Table 17. Baseband Controller Hardware Implementation Features

Hardware Implementation Settings	Description
Maximum Frequency	Baseband can execute up to a maximum frequency of 24 MHz
Exchange P-Bus Access	The processor accesses the baseband controller registers via an extension of the P-Bus. The R/W registers' accesses are limited to 32 and 16 bits.
Interrupts	The interrupts are pulse triggered
Coexistence Support	A coexistence interface is instantiated identifying when the RF front-end is busy for Bluetooth or other traffic
External low-power timing access	The low power timing generator is implemented parallel to the Bluetooth low energy technology controller, so external access is possible
Audio Support	For devices containing AOBLE support, three audio channels are supported

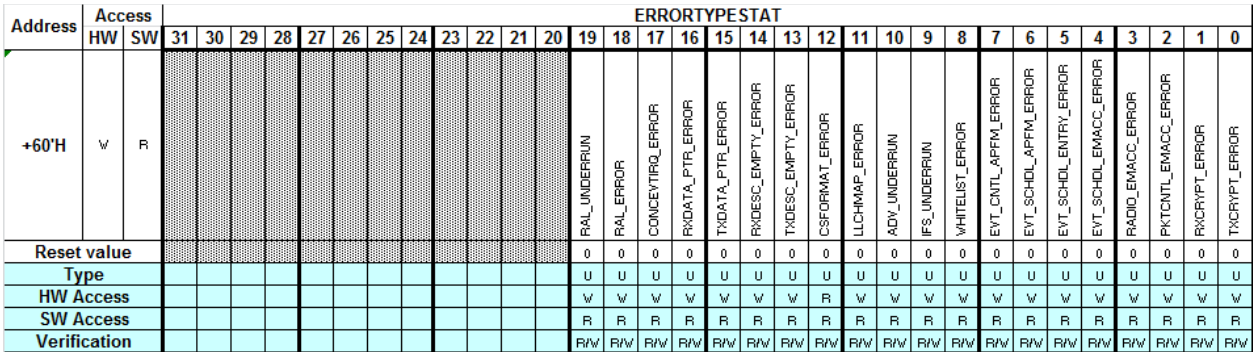
This chapter includes information about the following:

- The registers that can be used to access the baseband hardware, and how the baseband uses the exchange memory and other control structures to control data transferred to the RF front-end
- The baseband timing
- The hardware aspects of the security manager as accessible through the GAP layer

See the *RSL10 Firmware Reference* for more information about host and profile layer support.

9.1.1 Bluetooth Baseband Error Handling

This interrupt indicates that a hardware error has been detected. The error type can be recovered by reading the `ERRORTYPESTAT` register (refer to Figure 19 for register overview and Table 18 on page 203 for a detailed description of each bit).



Address	Access		ERRORTYPESTAT																															
	HW	SW	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+60'H	W	R													RAL_UNDERRUN	RAL_ERROR	CONCEVTIRQ_ERROR	RXDATA_PTR_ERROR	TXDATA_PTR_ERROR	RXDESC_EMPTY_ERROR	TXDESC_EMPTY_ERROR	CSFORMAT_ERROR	LLCHMAP_ERROR	ADV_UNDERRUN	IFS_UNDERRUN	WHITELIST_ERROR	EVT_CNTL_APFM_ERROR	EVT_SCHDL_APFM_ERROR	EVT_SCHDL_ENTRY_ERROR	EVT_SCHDL_EMACC_ERROR	RADIO_EMACC_ERROR	PKTCNTL_EMACC_ERROR	RXCRYPT_ERROR	TXCRYPT_ERROR
Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type															U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
HW Access															W	W	W	W	W	W	W	R	W	W	W	W	W	W	W	W	W	W	W	W
SW Access															R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Verification															R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Figure 19. ERRORTYPESTAT Error Interrupt Register

Table 18. ERRORTYPESTAT Description

Command	Value	Description
`RAL_UNDERRUN`	0: No error 1: Error occurred	Indicates Resolving Address List engine Underrun issue: happens when RAL List parsing is not finished on time.
`RAL_ERROR`	0: No error 1: Error occurred	Indicates Resolving Address List engine faced a bad setting.
`CONCEVTIRQ_ERROR`	0: No error 1: Error occurred	Indicates whether two consecutive and concurrent `ble_event_irq` have been generated, and not acknowledged in time by the RSL10 software.
`RXDATA_PTR_ERROR`	0: No error 1: Error occurred	Indicates whether Rx data buffer pointer value programmed is null: this is a major programming failure.
`TXDATA_PTR_ERROR`	0: No error 1: Error occurred	Indicates whether Tx data buffer pointer value programmed is null during Advertising / Scanning / Initiating events, or during Master / Slave connections with non-null packet length: this is a major programming failure.
`RXDESC_EMPTY_ERROR`	0: No error 1: Error occurred	Indicates whether Rx Descriptor pointer value programmed in register is null: this is a major programming failure.
`TXDESC_EMPTY_ERROR`	0: No error 1: Error occurred	Indicates whether Tx Descriptor pointer value programmed in Control Structure is null during Advertising / Scanning / Initiating events: this is a major programming failure.
`CSFORMAT_ERROR`	0: No error 1: Error occurred	Indicates whether CS-FORMAT has been programmed with an invalid value: this is a major software programming failure.
`LLCHMAP_ERROR`	0: No error 1: Error occurred	Indicates Link Layer Channel Map error, happens when actual number of `CS-LLCHMAP` bits set to one is different from `CS-NBCHGOOD` at the beginning of Frequency Hopping process.
`ADV_UNDERRUN`	0: No error 1: Error occurred	Indicates Advertising Interval Underrun: occurs if time between two consecutive intervals.
`IFS_UNDERRUN`	0: No error 1: Error occurred	Indicates Inter Frame Space Underrun: occurs if IFS time is not enough to update and read Control Structure/Descriptors, and/or White List parsing is not finished and/or Decryption time is too long to be finished on time.
`WHITELIST_ERROR`	0: No error 1: Error occurred	Indicates White List Timeout error: occurs if White List parsing is not finished on time.

Table 18. ERRORTYPESTAT Description (Continued)

Command	Value	Description
`EVT_CNTL_APFM_ERROR`	0: No error 1: Error occurred	Indicates Anticipated Pre-Fetch Mechanism error: happens when 2 consecutive events are programmed, and the first event is not completely finished while second pre-fetch instant is reached.
`EVT_SCHDL_APFM_ERROR`	0: No error 1: Error occurred	Indicates Anticipated Pre-Fetch Mechanism error: happens when 2 consecutive events are scheduled, and the first event is not completely finished while second pre-fetch instant is reached.
`EVT_SCHDL_ENTRY_ERROR`	0: No error 1: Error occurred	Indicates Event Scheduler has faced invalid timing programing on two consecutive exchange table entries (e.g first one with 624 microseconds offset and second one with no offset).
`EVT_SCHDL_EMACC_ERROR`	0: No error 1: Error occurred	Indicates Event Scheduler exchange memory access error: happens when exchange memory accesses are not served in time, and blocks the Exchange Table entry read.
`RADIO_EMACC_ERROR`	0: No error 1: Error occurred	Indicates Radio Controller exchange memory access error: happens when exchange memory accesses are not served in time and data is corrupted.
`PKTCNTL_EMACC_ERROR`	0: No error 1: Error occurred	Indicates Packet Controller exchange memory access error: happens when exchange memory accesses are not served in time and Tx/Rx is are corrupted.
`RXCRYPT_ERROR`	0: No error 1: Error occurred	Indicates real time decryption error: happens when AES-CCM decryption is too slow compared to Packet Controller requests. A 16-bytes block has to be decrypted prior the next block is received by the Packet Controller.
`TXCRYPT_ERROR`	0: No error 1: Error occurred	Indicates Real Time encryption error: happens when AES-CCM encryption is too slow compared to Packet Controller requests. A 16-bytes block has to be encrypted and prepared on Packet Controller request, and needs to be ready before the Packet Controller has to send it.

9.1.2 Support Interfaces

For debugging purposes, the baseband controller instantiates a diagnostic port that can be configured through the following baseband controller registers.

- `BB_DIAGCNTL`
- `BB_DIAGSTAT`
- `BB_DEBUGADDMAX`
- `BB_DEBUGADDMIN`
- `BB_ERRORTYPESTAT`

The 8-bit diagnostic bus can be mapped to any DIO output, as described in Figure 34 on page 348.

9.2 Baseband Registers and Memory Usage

The link layer software interfaces with the baseband by writing/reading its registers and the exchange memory. Baseband access methods need to be adapted to the physical interface between the CPU and baseband (e.g. AHB, SPI, etc.). An abstraction layer is therefore implemented so that LL software is, as much as possible, independent from the physical interface. Only the link layer baseband abstraction layer is modified for compatibility with the physical interface.

The abstraction layer contains a few primitives, allowing atomic read/writes and copies of data from system RAM to exchange memory.

9.2.1 Baseband Abstraction Layer Primitives

The following functions and macros are used by the link layer software to access the baseband registers, exchange memory, and RF registers:

REG_BLE_RD(addr)
: This macro reads a 32-bit value from a Bluetooth low energy technology core register

REG_BLE_WR(addr, value)
: This function writes a 32-bit value to a Bluetooth low energy technology core register

EM_BLE_RD(addr)
: This function reads a 16-bit value from the Bluetooth low energy technology exchange memory

EM_BLE_WR(addr, value)
: This function writes a 16-bit value to the exchange memory

RF_BLE_RD(addr)
: This macro reads a value from an RF register

RF_BLE_WR(addr, value)
: This function writes a value to an RF register

*void em_ble_burst_rd(void *sys_addr, uint16_t em_addr, uint16_t len)*
: This function reads **`len`** bytes from address **`em_addr`** in the exchange memory and writes them at address **`sys_addr`** in system memory

*void em_ble_burst_wr(void const*sys_addr, uint16_t em_addr, uint16_t len)*
: This function reads **`len`** bytes from address **`sys_addr`** in the system memory and writes them at address **`em_addr`** in the exchange memory

These macros are defined in *reg_access.h* and provide direct access to the baseband outside of the link layer software.

9.2.2 Control Structures

The control structure contains information that instructs the hardware to perform actions relevant to a specific Bluetooth link layer state (advertising, scanning, initiating, master connection, and slave connection).

The control structure is located in the exchange memory and is statically allocated at build time. The number of allocated control structures is one more than the maximum number of links supported by the built firmware.

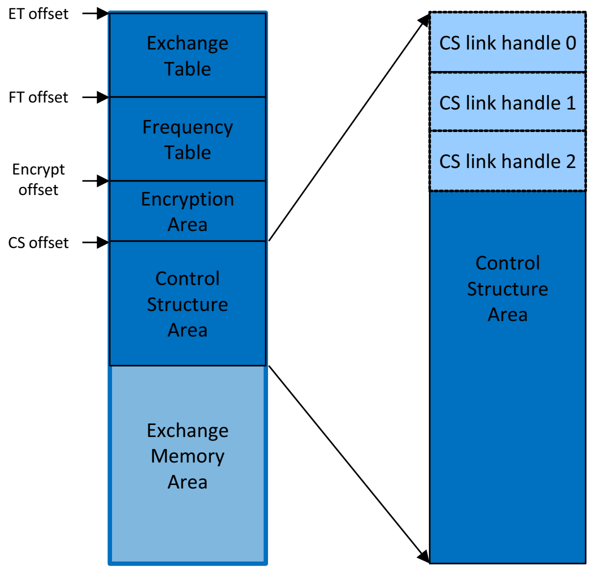


Figure 20. Control Structure Allocation

IMPORTANT: The exchange memory provided by the RSL10 SoC supports a maximum of 31 links; however, the default Bluetooth stack firmware may be built to support fewer links, for power and memory efficiency. If your user application requires more links than the Bluetooth library builds described in the RSL10 Firmware Reference can support, contact your ON Semiconductor Customer Service Representative for assistance.

9.3 Baseband Timing

9.3.1 Clock Structures

The Bluetooth low energy baseband is clocked using a baseband clock (BBCLK), a divided baseband clock (BBCLK_DIV), and a baseband timer clock. For more information about these clocks, see Section 6.3.4, “Baseband Clock (BBCLK) and Other Clocks for the Bluetooth Low Energy Baseband” on page 81.

BBCLK is a prescaled form of SYSCLK (see Section 6.3.1, “System Clock (SYSCLK)” on page 78) that is provided to the Bluetooth low energy baseband. BBCLK is derived from SYSCLK clock through a 3-bit integer division by the `CLK_DIV_CFG0_BBCLK_PRESCALE` bit field in the `CLK_DIV_CFG0` register. For proper baseband operation during RF transmissions, BBCLK must be configured in the range from 6 to 24 MHz with operation at 8 MHz or 16 MHz recommended. This prescaler provides a clock prescaled from SYSCLK by 1 to 8, and results in an BBCLK with a frequency as defined by the following equation:

$$f_{BBCLK} = \frac{f_{SYSCLK}}{(CLK\_DIV\_CFG0\_BBCLK\_PRESCALE + 1)}$$

For proper baseband operation BBCLK_DIV must be configured to supply a 1 MHz reference, divided from BBCLK. BBCLK_DIV is divided using the `BBIF_CTRL_CLK_SEL` bit-field of the `BBIF_CTRL` register. This bit-field should be set so that the following equation is correct:

$$f_{BBCLK\_DIV} = \frac{f_{BBCLK}}{(BBIF\_CTRL\_CLK\_SEL + 1)} = 1MHz$$

The reset and the low power timing generator clock (32 kHz), or a divided clock for lower power consumption, are divided from the standby clock (Section 6.3.2, "Standby Clock (STANDBYCLK)" on page 79) with the baseband timer configuring and potentially further dividing this using the `ACS_BB_TIMER_CTRL_BB_CLK_PRESCALE` bit from the `ACS_BB_TIMER_CTRL` register. The reset signal for this block is controlled by the `ACS_BB_TIMER_CTRL_BB_TIMER_NRESET` bit, and is re-synchronized on the baseband timer clock, and consequently, it can take up to one clock cycle of this divided clock before the baseband timer is effectively reset.

IMPORTANT: When `ACS_BB_TIMER_CTRL_BB_CLK_PRESCALE` is used to scale `STANDBYCLK` for use as the baseband time clock:

- **The RTC must be enabled.**
- **The `ACS_RTC_CFG_START_VALUE` bit-field of the `ACS_RTC_CFG` register must be set to provide division of `STANDBYCLK` of 2^N (`ACS_RTC_CFG_START_VALUE` set to 2^N-1), where *N* > *ACS_BB_TIMER_CTRL_BB_CLK_PRESCALE*.**

For more information on RTC configuration, see Section 6.3.5, "Real-Time Clock (RTC)" on page 81.

9.3.1.1 ACS_BB_TIMER_CTRL

Bit Field	Field Name	Description
9:8	BB_CLK_PRESCALE	Prescale value for the baseband timer clock
0	BB_TIMER_NRESET	nReset signal for the baseband timer

Field Name	Value Symbol	Value Description	Hex Value
BB_CLK_PRESCALE	BB_CLK_PRESCALE_1	Use 32 kHz clock from RTC	0x0*
	BB_CLK_PRESCALE_2	Use 16 kHz clock from RTC (counter bit0)	0x1
	BB_CLK_PRESCALE_4	Use 8 kHz clock from RTC (counter bit1)	0x2
	BB_CLK_PRESCALE_8	Use 4 kHz clock from RTC (counter bit2)	0x3
BB_TIMER_NRESET	BB_TIMER_RESET	Baseband timer is in reset state	0x0*
	BB_TIMER_NRESET	Baseband timer reset is released	0x1

9.3.2 Slot Timing

The system is synchronized as per a base time unit, called a "slot", which is equal to 625 μs. A hardware counter named "base time counter" manages the system slot counting. A hardware counter named "fine time counter" performs a 1 μs precision countdown within a slot.

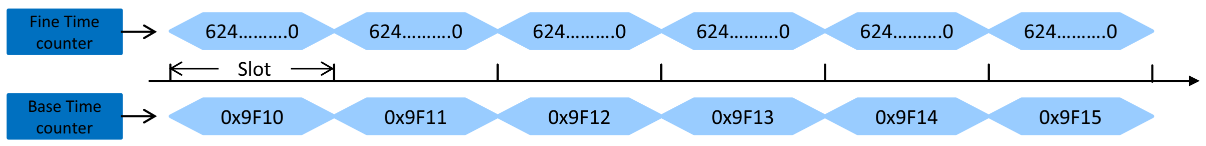


Figure 21. Slot Definition

The base time counter starts at 0 on power-up. It is 27 bits wide, and it is incremented every 625 μs.

9.3.3 Timing and Event-Related Interrupts

The real time scheduling and the system wakeup are synchronized over several interrupts. These interrupts are generated by the hardware (BaseBand / Core). Refer to Figure 22 on page 209.

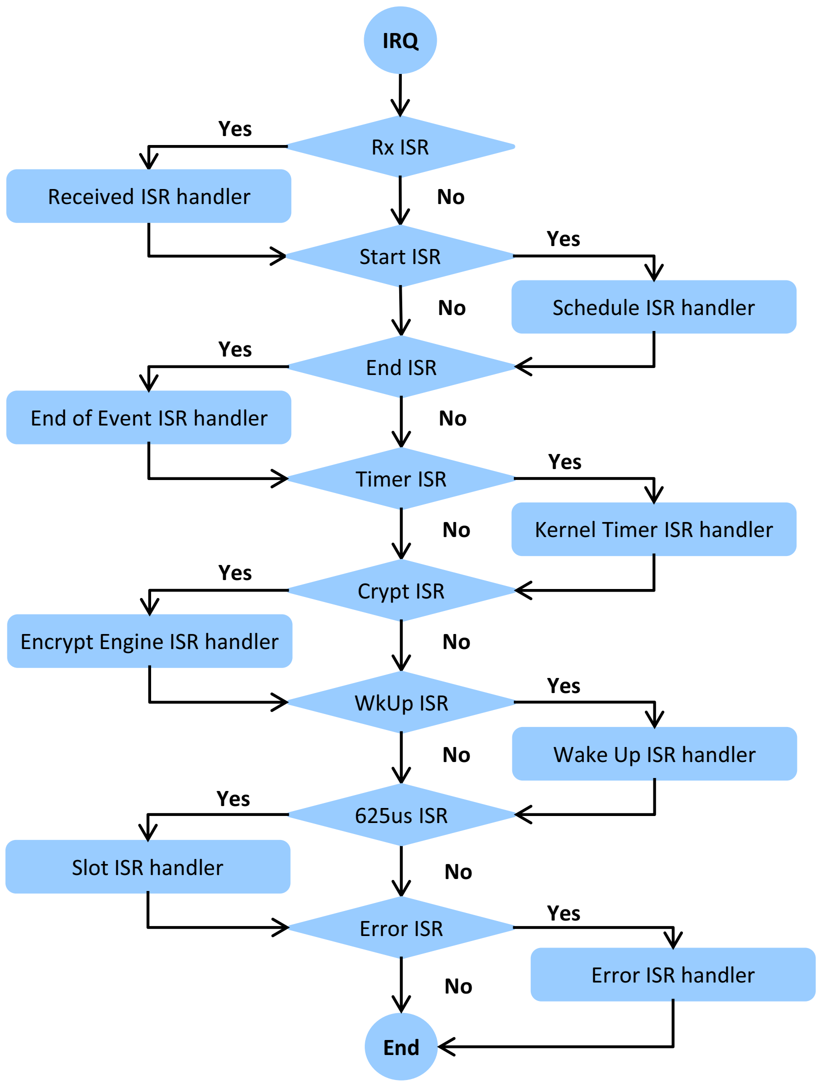


Figure 22. Interrupts Overview

9.3.3.1 Schedule Interrupt

The schedule interrupt allows the event arbiter to program the exchange table. This interrupt happens in advance of the execution time. This delay is called PROG LATENCY; it is counted in slot number and is set to 2 by definition in the firmware (i.e. 2 * 625 μs = 1.25 ms).

This latency allows the firmware to safely program the control structure fields and Tx buffers, before hardware fetching takes place.

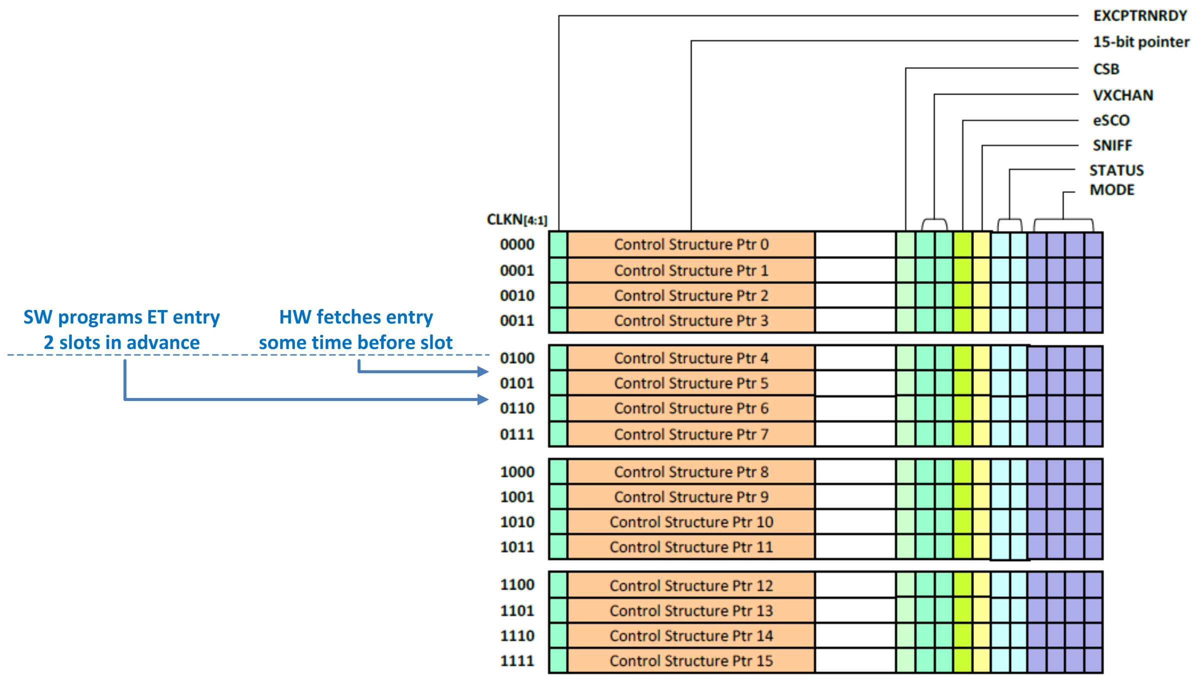


Figure 23. Programming Process Overview

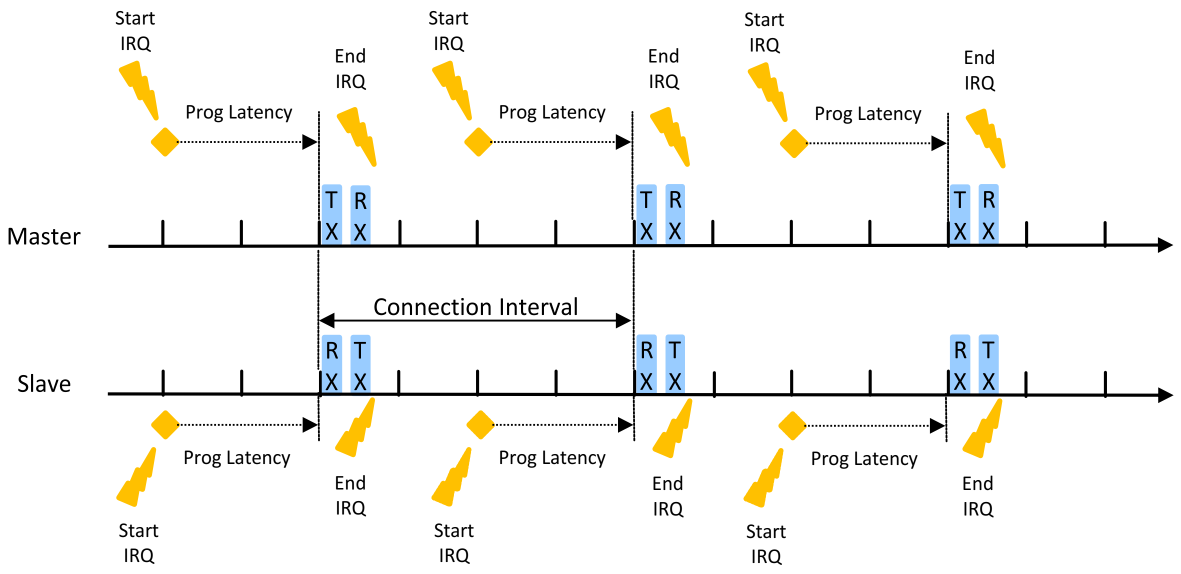


Figure 24. Schedule/End of Event Interrupts Overview

9.3.3.2 End of Event Interrupt

The end of event interrupt allows the firmware to check what has been received and sent. This interrupt happens at the end of the event, even if nothing has been received. See Figure 24 on page 210.

The firmware checks the control structure (i.e., CS-CONFLICT) bit and the number of descriptors consumed, to see if it is a normal or a dual mode arbitration conflict end of event.

9.3.3.3 Reception Interrupt

The reception interrupt allows the firmware to check what has been received and sent. This interrupt occurs when the threshold set in the control structure (CS-RXTHR - control structure value of the threshold for the Rx interrupt) has been reached, or when something is received during scanning activity.

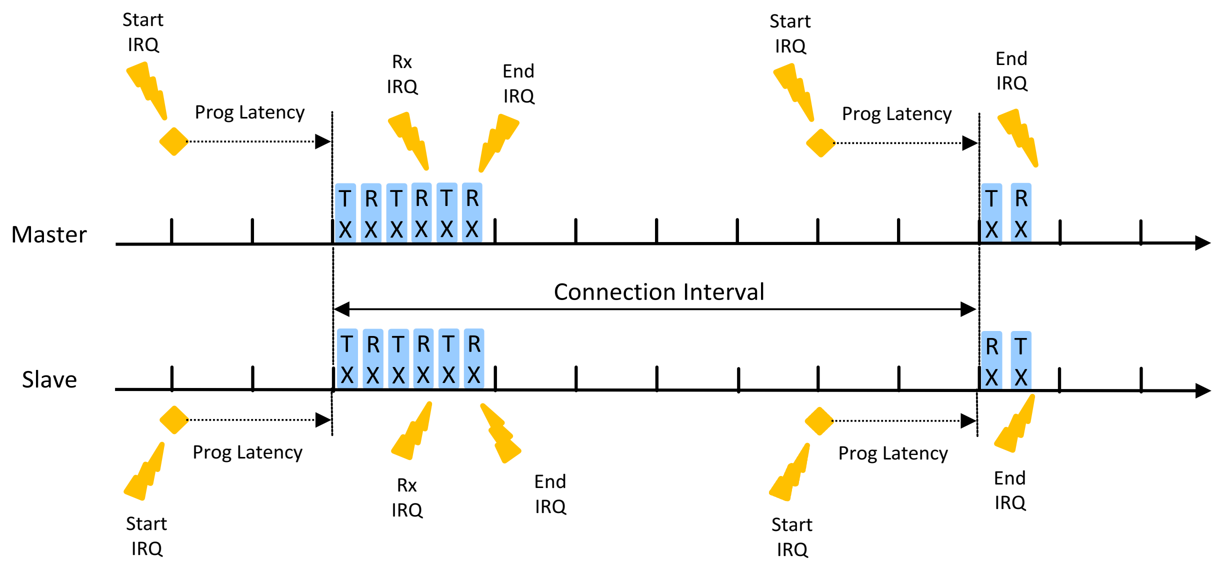


Figure 25. Rx interrupts, Rx Threshold Set to 2

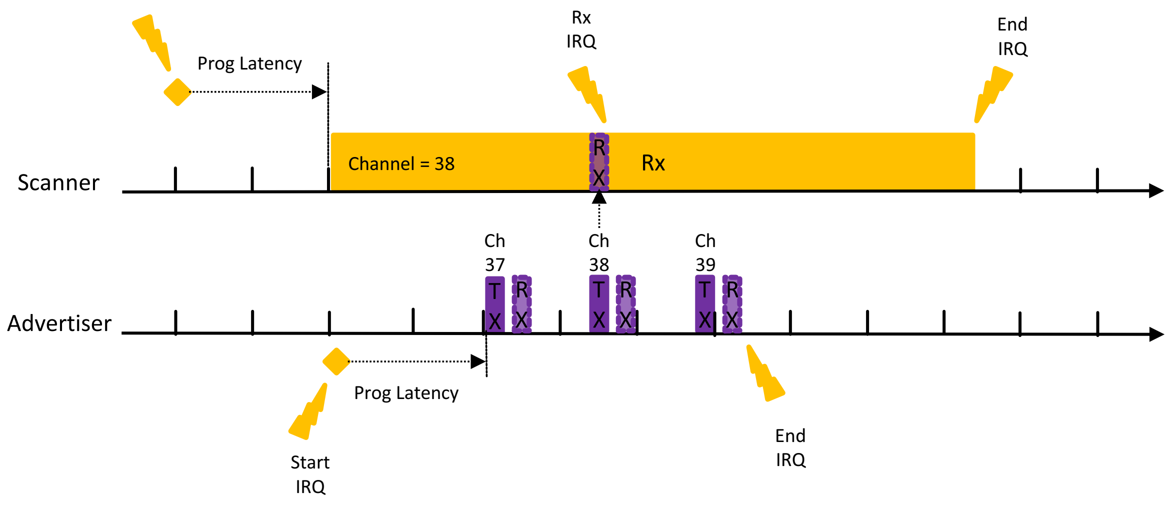


Figure 26. Rx interrupts, Advertising Packet Received

9.3.3.4 Wakeup Interrupt

The wakeup interrupt allows the firmware to turn on the system and start a Bluetooth low energy technology activity. The wakeup interrupt can occur asynchronously with a slot boundary. The Wakeup has 2 steps:

Wakeup	Start to compensate the low power clock drift (wakeup interrupt).
Slot synchronization	System is ready (slot interrupt), so the activity can start.

All those phases are shown in Figure 27.

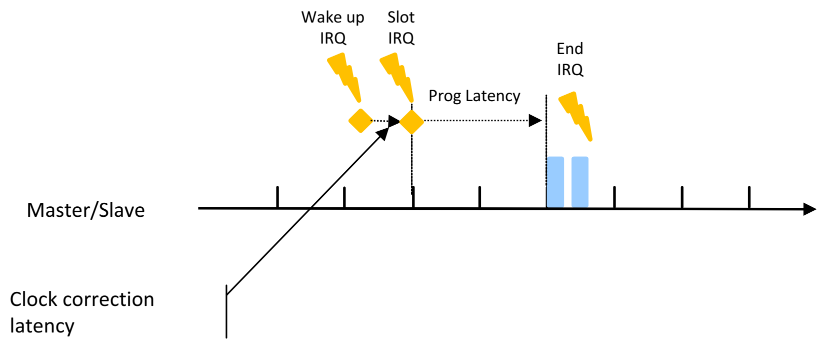


Figure 27. Wakeup Interrupts

Upon wakeup, the hardware base time counter has to be updated as to the passed sleep duration. For that, the firmware computes the correction values to apply on the base and fine counters by setting appropriate registers. The core applies the correction for the next slot interrupt. The maximum time to correct the clock value is 1 slot (default value) and is known in the firmware as “clock correction latency”.

The event arbiter checks whether the event should be programmed, or if it is in the past and should be pushed into the canceled queue.

9.3.3.5 Software Interrupt

This interrupt indicates an event arbiter cancellation.The firmware increments the priority and the programming time before trying to reinsert the event in the event arbiter wait queue.

9.3.3.6 Encryption Interrupt

This interrupt occurs when the encryption engine performs a ciphering process running AES-128 toolbox, and when the procedure ends.

The ciphering process is started by a command, defined by an API, with two parameters:

- Plain data: data to be encrypted
- Key: 128-bit key for the encryption.

When the interrupt is generated, a kernel message is sent to the requester with the ciphered data.

9.4 Baseband Registers

Register Name	Register Description	Address
Baseband Controller Interface		
BBIF_CTRL	Baseband controller control register	0x40001400
BBIF_STATUS	Baseband controller status register	0x40001404
BBIF_COEX_CTRL	RF coexistence control register	0x40001408
BBIF_COEX_STATUS	RF coexistence status register	0x4000140C
BBIF_COEX_INT_CFG	RF coexistence interrupt configuration register	0x40001410
BBIF_COEX_INT_STATUS	RF coexistence interrupt status register	0x40001414
BBIF_SYNC_CFG	Bluetooth low energy technology and RF link synchronization configuration register	0x40001418
Baseband Controller		
BB_RWBBCNTL	Baseband control register	0x40001500
BB_VERSION	Bluetooth low energy revision register	0x40001504
BB_RWBLEBCONF	Baseband configuration register (compilation options dependant)	0x40001508
BB_INTCNTL	Interrupts control register	0x4000150C
BB_INTSTAT	Interrupts status register	0x40001510
BB_INTRAWSTAT	Interrupts raw status register	0x40001514
BB_INTACK	Interrupts acknowledgement register	0x40001518
BB_BASETIMECNT	base timer configuration register	0x4000151C
BB_FINETIMECNT	Fine timer configuration register	0x40001520
BB_BDADDRL	Bluetooth low energy device address (LSB part) register[1]	0x40001524
BB_BDADDRU	Bluetooth low energy device address (MSB part) register[1]	0x40001528
BB_ET_CURRENTRXDESCPTR	Rx descriptor pointer register	0x4000152C
BB_DEEPSLCNTL	Deep sleep control register	0x40001530
BB_DEEPSLWKUP	Deep sleep wakeup register	0x40001534
BB_DEEPSLSTAT	Deep sleep status register	0x40001538
BB_ENBPRESET	Stabilization times	0x4000153C
BB_FINECNTCORR	Fine timer correction register	0x40001540
BB_BASETIMECNTCORR	Base timer correction register	0x40001544
BB_DIAGCNTL	Diagnostic ports control register	0x40001550
BB_DIAGSTAT	Diagnostic ports status register	0x40001554
BB_DEBUGADDMAX	Diagnostic ports upper limit	0x40001558
BB_DEBUGADDMIN	Diagnostic ports lower limit	0x4000155C
BB_ERRORTYPESTAT	Diagnostic ports errors register	0x40001560
BB_SWPROFILING	Software profiling register	0x40001564
BB_RADIOCNTL0	Principal control register for the radio interface	0x40001570
BB_RADIOCNTL1	Second control register for the radio interface	0x40001574
BB_RADIOCNTL2	Third control register for the radio interface	0x40001578
BB_RADIOPWRUPDN0	Principal control register for the radio interface power-up/down delays	0x40001580

Register Name	Register Description	Address
BB_RADIOPWRUPDN1	Second control register for the radio interface power-up/down delays	0x40001584
BB_RADIOTXRXTIM0	Principal control register for the radio interface timing compensation delays	0x40001590
BB_RADIOTXRXTIM1	Second control register for the radio interface timing compensation delays	0x40001594
BB_SPIPTRCNTL0	First control register for the radio interface SPI pointers	0x400015A0
BB_SPIPTRCNTL1	Second control register for the radio interface SPI pointers	0x400015A4
BB_SPIPTRCNTL2	Third control register for the radio interface SPI pointers	0x400015A8
BB_ADVCHMAP	Advertising channel map register	0x400015B0
BB_ADVTIM	Delay information register handling advertising event timers	0x400015C0
BB_ACTSCANSTAT	Active scan mode control register	0x400015C4
BB_WLPUBADDPTR	Address pointer of public devices	0x400015D0
BB_WLPRIVADDPTR	Address pointer of private devices	0x400015D4
BB_WLNBDEV	Devices in white list	0x400015D8
BB_AESCNTL	AES-128 ciphering control register	0x400015E0
BB_AESKEY31_0	AES encryption 128-bit key register (bits 31:0)	0x400015E4
BB_AESKEY63_32	AES encryption 128-bit key register (bits 63:32)	0x400015E8
BB_AESKEY95_64	AES encryption 128-bit key register (bits 95:64)	0x400015EC
BB_AESKEY127_96	AES encryption 128-bit key register (bits 127:96)	0x400015F0
BB_AESPTR	AES memory zone pointer	0x400015F4
BB_TXMICVAL	AES-CCM plain MIC value register in Tx	0x400015F8
BB_RXMICVAL	AES-CCM plain MIC value register in Rx	0x400015FC
BB_RFTESTCNTL	RF testing and regulatory body support register	0x40001600
BB_RFTESTTXSTAT	Number of transmitted packets during test modes	0x40001604
BB_RFTESTRXSTAT	Number of correctly received packet during test modes	0x40001608
BB_TIMGENCNTL	Timing generator control register	0x40001610
BB_GROSSTIMTGT	Gross timer control register	0x40001614
BB_FINETIMTGT	Fine timer control register	0x40001618
BB_COEXIFCNTL0	RF coexistence control register 0	0x40001620
BB_COEXIFCNTL1	RF coexistence control register 1	0x40001624
BB_COEXIFCNTL2	RF coexistence control register 2	0x40001628
BB_BBMPRIO0	Priority control register 0	0x4000162C
BB_BBMPRIO1	Priority control register 1	0x40001630
BB_RALPTR	Register used by the Resolving Address List engine	0x40001640
BB_RALNBDEV	Register used by the Resolving Address List engine	0x40001644
BB_RAL_LOCAL_RND	Register used by the Resolving Address List engine	0x40001648
BB_RAL_PEER_RND	Register used by the Resolving Address List engine	0x4000164C
BB_ISOCHANCNTL0	ISO Channel 0 control	0x40001650
BB_ISOMUTECNTL0	ISO Channel 0 mute control	0x40001654
BB_ISOCURRENTTXPTR0	ISO Channel 0 current Tx pointer	0x40001658

Register Name	Register Description	Address
BB_ISOCURRENTRXPTR0	ISO Channel 0 current Rx pointer	0x4000165C
BB_ISOTRCNL0	ISO Channel 0 payloads	0x40001660
BB_ISOEVTCNTLOFFSETL0	ISO Channel 0 mute control	0x40001664
BB_ISOEVTCNTLOFFSETU0	ISO Channel 0 mute control	0x40001668
BB_ISOCHANCNTL1	ISO Channel 1 control	0x40001670
BB_ISOMUTECNTL1	ISO Channel 1 mute control	0x40001674
BB_ISOCURRENTTXPTR1	ISO Channel 1 current Tx pointer	0x40001678
BB_ISOCURRENTRXPTR1	ISO Channel 1 current Rx pointer	0x4000167C
BB_ISOTRCNL1	ISO Channel 1 payloads	0x40001680
BB_ISOEVTCNTLOFFSETL1	ISO Channel 1 mute control	0x40001684
BB_ISOEVTCNTLOFFSETU1	ISO Channel 1 mute control	0x40001688
BB_ISOCHANCNTL2	ISO Channel 2 control	0x40001690
BB_ISOMUTECNTL2	ISO Channel 2 mute control	0x40001694
BB_ISOCURRENTTXPTR2	ISO Channel 2 current Tx pointer	0x40001698
BB_ISOCURRENTRXPTR2	ISO Channel 2 current Rx pointer	0x4000169C
BB_ISOTRCNL2	ISO Channel 2 payloads	0x400016A0
BB_ISOEVTCNTLOFFSETL2	ISO Channel 2 mute control	0x400016A4
BB_ISOEVTCNTLOFFSETU2	ISO Channel 2 mute control	0x400016A8
BB_BBPRIOSCHARB	Register controlling the decision instant for priority scheduling arbitration	0x400016B0

1. In typical use cases, the Bluetooth device address should be set to the value stored to the Device Configuration Record (NVR3) at the DEVICE_INFO_BLUETOOTH_ADDR location. For more information see the *RSL10 Firmware Reference*.

9.4.1 BBIF_CTRL

Bit Field	Field Name	Description
16	WAKEUP_REQ	External wakeup request used to sort out sleep modes
9:4	CLK_SEL	Configure the internal baseband controller clock divider in order to provide a 1MHz reference clock
0	CLK_ENABLE	Enable the baseband controller clocks generation

Field Name	Value Symbol	Value Description	Hex Value
WAKEUP_REQ	BB_DEEP_SLEEP	Keep the baseband controller in Deep Sleep Mode	0x0*
	BB_WAKEUP	Wake up the baseband controller and keep it active	0x1

Field Name	Value Symbol	Value Description	Hex Value
CLK_SEL	BBCLK_DIVIDER_6	Divide the BBCLK by 6 (minimum authorized value)	0x6
	BBCLK_DIVIDER_8	Divide the BBCLK by 8	0x8*
	BBCLK_DIVIDER_12	Divide the BBCLK by 12	0xC
	BBCLK_DIVIDER_16	Divide the BBCLK by 16	0x10
	BBCLK_DIVIDER_24	Divide the BBCLK by 24 (maximum authorized value)	0x18
CLK_ENABLE	BB_CLK_DISABLE	Baseband controller clocks are gated	0x0*
	BB_CLK_ENABLE	Baseband controller clocks are generated	0x1

9.4.2 BBIF_STATUS

Bit Field	Field Name	Description
15:11	LINK_FORMAT	Bluetooth low energy link format
8:4	LINK_LABEL	Bluetooth low energy link label
3	AOBLE_STATUS	Audio over Bluetooth low energy feature status
2	CLK_STATUS	Clock status defining the current active clock in use
1	OSC_EN	Oscillator front-end enabling
0	RADIO_EN	RF front-end enabling

Field Name	Value Symbol	Value Description	Hex Value
LINK_FORMAT	LINK_FORMAT_RESET	Reset value	0x0*
	MASTER_CONNECT	Master connect	0x2
	SLAVE_CONNECT	Slave connect	0x3
	LOW_DUTY_ADVERTISER	Low Duty Cycle Advertiser	0x4
	HIGH_DUTY_ADVERTISER	High Duty Cycle Advertiser	0x5
	PASSIVE_SCANNER	Passive Scanner	0x8
	ACTIVE_SCANNER	Active Scanner	0x9
	INITIATOR	Initiator	0xF
	TX_TEST_MODE	Tx Test Mode	0x1C
	RX_TEST_MODE	Rx Test Mode	0x1D
	TX_RX_TEST_MODE	Tx / Rx Test Mode	0x1E
AOBLE_STATUS	AOBLE_DISABLED_DEVICE	Device does not have the Audio over Bluetooth low energy technology feature	0x0
	AOBLE_ENABLED_DEVICE	Device has the Audio over Bluetooth low energy technology feature	0x1*
CLK_STATUS	MASTER_CLK	Baseband controller running on master1/2_clk	0x0*
	LOW_POWER_CLK	Baseband controller running on low_power_clk	0x1

Field Name	Value Symbol	Value Description	Hex Value
OSC_EN	OSC_DISABLED	Oscillator can be safely disabled	0x0*
	OSC_ENABLED	Oscillator must be enabled	0x1
RADIO_EN	RF_DISABLED	RF front-end can be safely disabled	0x0*
	RF_ENABLED	RF front-end must be enabled	0x1

9.4.3 BBIF_COEX_CTRL

Bit Field	Field Name	Description
4	TX	Indicates if the RF performs a Tx activity
0	RX	Indicates if the RF performs a Rx activity

Field Name	Value Symbol	Value Description	Hex Value
TX	COEX_TX_IDLE	RF performs no Tx activity	0x0*
	COEX_TX_BUSY	RF core performs Tx activity	0x1
RX	COEX_RX_IDLE	RF core performs no Rx activity	0x0*
	COEX_RX_BUSY	RF core performs Rx activity	0x1

9.4.4 BBIF_COEX_STATUS

Bit Field	Field Name	Description
15:12	BLE_PTI	Indicates the priority level of the current Bluetooth baseband core activity
8	BLE_IN_PROCESS	Indicates if the Bluetooth baseband core has an event in process, active high.
4	BLE_TX	Indicates if the Bluetooth baseband core is busy and performs Tx activity, active high.
0	BLE_RX	Indicates if the Bluetooth baseband core is busy and performs Rx activity, active high

Field Name	Value Symbol	Value Description	Hex Value
BLE_PTI	BLE_PTI_PRIORITY_0	BLE_PTI lowest priority	0x0*
	BLE_PTI_PRIORITY_15	BLE_PTI highest priority	0xF
BLE_IN_PROCESS	BLE_IDLE	Bluetooth baseband processes no event	0x0*
	BLE_IN_PROCESS	Bluetooth baseband processes an event	0x1
BLE_TX	BLE_TX_IDLE	Bluetooth baseband core performs no Tx activity	0x0*
	BLE_TX_BUSY	Bluetooth baseband core performs Tx activity	0x1
BLE_RX	BLE_RX_IDLE	Bluetooth baseband core performs no Rx activity	0x0*
	BLE_RX_BUSY	Bluetooth baseband core performs Rx activity	0x1

9.4.5 BBIF_COEX_INT_CFG

Bit Field	Field Name	Description
9:8	BLE_IN_PROCESS_EVENT	BLE_IN_PROCESS event interrupt configuration
5:4	BLE_TX_EVENT	BLE_TX event interrupt configuration
1:0	BLE_RX_EVENT	BLE_RX event interrupt configuration

Field Name	Value Symbol	Value Description	Hex Value
BLE_IN_PROCESS_EVENT	BLE_IN_PROCESS_EVENT_NONE	Interrupt not triggered	0x0*
	BLE_IN_PROCESS_EVENT_RISING_EDGE	Interrupt triggered on rising edge	0x1
	BLE_IN_PROCESS_EVENT_FALLING_EDGE	Interrupt triggered on falling edge	0x2
	BLE_IN_PROCESS_EVENT_TRANSITION	Interrupt triggered on any edge	0x3
BLE_TX_EVENT	BLE_TX_EVENT_NONE	Interrupt not triggered	0x0*
	BLE_TX_EVENT_RISING_EDGE	Interrupt triggered on rising edge	0x1
	BLE_TX_EVENT_FALLING_EDGE	Interrupt triggered on falling edge	0x2
	BLE_TX_EVENT_TRANSITION	Interrupt triggered on any edge	0x3
BLE_RX_EVENT	BLE_RX_EVENT_NONE	Interrupt not triggered	0x0*
	BLE_RX_EVENT_RISING_EDGE	Interrupt triggered on rising edge	0x1
	BLE_RX_EVENT_FALLING_EDGE	Interrupt triggered on falling edge	0x2
	BLE_RX_EVENT_TRANSITION	Interrupt triggered on any edge	0x3

9.4.6 BBIF_COEX_INT_STATUS

Bit Field	Field Name	Description
4	BLE_TX_EVENT_FLAG	Indicates if a BLE_TX_EVENT interrupt has been generated
0	BLE_RX_EVENT_FLAG	Indicates if a BLE_RX_EVENT interrupt has been generated

Field Name	Value Symbol	Value Description	Hex Value
BLE_TX_EVENT_FLAG	BLE_TX_EVENT_NO_INT	No BLE_TX_EVENT interrupt has been generated	0x0*
	BLE_TX_EVENT_INT	A BLE_TX_EVENT interrupt has been generated	0x1
BLE_RX_EVENT_FLAG	BLE_RX_EVENT_NO_INT	No BLE_RX_EVENT interrupt has been generated	0x0*
	BLE_RX_EVENT_INT	A BLE_RX_EVENT interrupt has been generated	0x1

9.4.7 BBIF_SYNC_CFG

Bit Field	Field Name	Description
17	`RF_RX`	Specify if the RF front-end is currently receiving the audio link
16	`RF_ACTIVE`	Specify if the RF front-end is currently processing the audio link
15:11	`LINK_FORMAT`	Configure the Bluetooth low energy link format for synchronization
8:4	`LINK_LABEL`	Configure the Bluetooth low energy link label for synchronization
3:1	`SOURCE`	Select the BLE/RF link synchronization source
0	`ENABLE`	Enable the frame synchronization pulse filter

Field Name	Value Symbol	Value Description	Hex Value
`RF_RX`	`RX_IDLE`	No audio link is currently received by the RF front-end	0x0*
	`RX_ACTIVE`	The audio link is currently received by the RF front-end	0x1
`RF_ACTIVE`	`IDLE`	No audio link is currently processed by the RF front-end	0x0*
	`ACTIVE`	The audio link is currently processed by the RF front-end	0x1
`SOURCE`	`SYNC_SOURCE_BLE_RX`	Use the Bluetooth low energy bb_sync_p signal as synchronization source	0x0*
	`SYNC_SOURCE_BLE_RX_AUDIO0`	Use the Bluetooth low energy audio0_sync_p signal as synchronization source	0x1
	`SYNC_SOURCE_BLE_RX_AUDIO1`	Use the Bluetooth low energy audio1_sync_p signal as synchronization source	0x2
	`SYNC_SOURCE_BLE_RX_AUDIO2`	Use the Bluetooth low energy E audio2_sync_p signal as synchronization source	0x3
	`SYNC_SOURCE_RF_RX`	Use the RF front-end rf_sync_p signal as synchronization source	0x4
	`SYNC_SOURCE_BLE_TX`	Use the Bluetooth low energy tx_en signal as synchronization source	0x5
`ENABLE`	`SYNC_DISABLE`	Disable the frame synchronization pulse filter	0x0*
	`SYNC_ENABLE`	Enable the frame synchronization pulse filter	0x1

9.4.8 BB_RWBBCNTL

Bit Field	Field Name	Description
31	`MASTER_SOFT_RST`	Reset the complete system except registers and timing generator
30	`MASTER_TGSOFT_RST`	Reset the timing generator
29	`REG_SOFT_RST`	Reset the complete register block
28	`SWINT_REQ`	Forces the generation of ble_sw_irq

Bit Field	Field Name	Description
26	RFTEST_ABORT	Abort the current RF testing defined as per CS-FORMAT
25	ADVERT_ABORT	Abort the current scan window
24	SCAN_ABORT	Abort the current advertising event
22	MD_DSB	Allow a single Tx/Rx exchange whatever the MD bits are
21	SN_DSB	Disable sequence number management
20	NESN_DSB	Disable acknowledge scheme
19	CRYPT_DSB	Disable encryption / decryption
18	WHIT_DSB	Disable whitening
17	CRC_DSB	Disable CRC stripping
16	HOP_REMAP_DSB	Disable frequency hopping remapping algorithm
9	ADVERTFILT_EN	Advertising channels error filtering enable control
8	RWBLE_EN	Enable Bluetooth baseband core exchange table pre-fetch mechanism
7:4	RXWINSZDEF	Default Rx Window size in us (used when device is master connected or performs its second receipt)
2:0	SYNCERR	Indicates the maximum number of errors allowed to recognize the synchronization word

Field Name	Value Symbol	Value Description	Hex Value
MASTER_SOFT_RST	MASTER_SOFT_RST_0	No action happens if it is written with 0	0x0*
	MASTER_SOFT_RST_1	Resets the complete system at 0	0x1
MASTER_TGSOFT_RST	MASTER_TGSOFT_RST_0	No action happens if it is written with 0	0x0*
	MASTER_TGSOFT_RST_1	Resets the timing generator at 0	0x1
REG_SOFT_RST	REG_SOFT_RST_0	No action happens if it is written with 0	0x0*
	REG_SOFT_RST_1	Resets the complete register block at 0	0x1
SWINT_REQ	SWINT_REQ_0	No action happens if it is written with 0	0x0*
	SWINT_REQ_1	When written with a 1 and proper masking is set, resets at 0	0x1
RFTEST_ABORT	RFTEST_ABORT_0	No action happens if it is written with 0	0x0*
	RFTEST_ABORT_1	Abort the current RF testing	0x1
ADVERT_ABORT	ADVERT_ABORT_0	No action happens if it is written with 0	0x0*
	ADVERT_ABORT_1	Abort the current scan window	0x1
SCAN_ABORT	SCAN_ABORT_0	No action happens if it is written with 0	0x0*
	SCAN_ABORT_1	Abort the current advertising event	0x1
MD_DSB	MD_DSB_0	Normal operation of MD bits management	0x0*
	MD_DSB_1	Allow a single Tx/Rx exchange whatever the MD bits are.	0x1
SN_DSB	SN_DSB_0	Normal operation of sequence number	0x0*
	SN_DSB_1	Sequence number management disabled	0x1

Field Name	Value Symbol	Value Description	Hex Value
NESN_DSB	NESN_DSB_0	Normal operation of acknowledge	0x0*
	NESN_DSB_1	Acknowledge scheme disabled	0x1
CRYPT_DSB	CRYPT_DSB_0	Normal operation (encryption / decryption enabled)	0x0*
	CRYPT_DSB_1	Encryption / decryption disabled	0x1
WHIT_DSB	WHIT_DSB_0	Normal operation (whitening enabled)	0x0*
	WHIT_DSB_1	Whitening disabled	0x1
CRC_DSB	CRC_DSB_0	Normal operation (CRC removed from data stream)	0x0*
	CRC_DSB_1	CRC stripping disabled on Rx packets, CRC replaced by 0x000 in Tx	0x1
HOP_REMAP_DSB	HOP_REMAP_DSB_0	Normal operation (frequency hopping remapping algorithm enabled)	0x0*
	HOP_REMAP_DSB_1	Frequency hopping remapping algorithm disabled	0x1
ADVERTFILT_EN	ADVERTFILT_EN_0	Bluetooth baseband core reports all errors to Bluetooth baseband software	0x0*
	ADVERTFILT_EN_1	Bluetooth baseband core reports only correctly received packet, without error to Bluetooth baseband software	0x1
RWBLE_EN	RWBLE_EN_0	Disable Bluetooth baseband core exchange table pre-fetch mechanism	0x0*
	RWBLE_EN_1	Enable Bluetooth baseband core exchange table pre-fetch mechanism	0x1
RXWINSZDEF	RXWINSZDEF_0		0x0*
SYNCERR	SYNCERR_0		0x0*

9.4.9 BB_VERSION

Bit Field	Field Name	Description
31:24	TYP	Bluetooth baseband core type (Bluetooth 5)
23:16	REL	Bluetooth baseband core version - major release number
15:8	UPG	Bluetooth baseband core upgrade - upgrade number
7:0	BUILD	Bluetooth baseband core build - build number

Field Name	Value Symbol	Value Description	Hex Value
TYP	TYP_8		0x8*
REL	REL_0		0x0*
UPG	UPG_09		0x9*
BUILD	BUILD_1		0x1*

9.4.10 BB_RWBLEBCONF

Bit Field	Field Name	Description
31	DMMODE	Bluetooth baseband core dual mode
25:24	ISOPORTNB	Number of supported isochronous channels
23	DECIPHER	AES deciphering present
21	COEX	RF coexistence mechanism
20:16	RFIF	Support of the RF front-end
15	USEDBG	Diagnostic port
14	USECRYPT	AES-CCM encryption
13:8	CLK_SEL	Operating frequency (in MHz)
7	INTMODE	Interruption Mode
6	BUSTYPE	Processor bus type
5	DATA_WIDTH	Processor bus width
4:0	ADD_WIDTH	Value of the RW_BLE_ADDRESS_WIDTH parameter converted into binary

Field Name	Value Symbol	Value Description	Hex Value
DMMODE	DMMODE_0	Bluetooth baseband core is used as a standalone BLE device	0x0*
	DMMODE_1	Bluetooth baseband core is used in a dual mode device	0x1
ISOPORTNB	NO_ISO_CH	No ISO/Audio Channel available	0x0
	ONE_ISO_CH	One ISO/Audio Channel available	0x1
	TWO_ISO_CH	Two ISO/Audio Channels available	0x2
	THREE_ISO_CH	Three ISO/Audio Channels available	0x3*
DECIPHER	DECIPHER_0	AES deciphering not present	0x0*
	DECIPHER_1	AES deciphering present	0x1
COEX	COEX_0	RF coexistence mechanism not present	0x0
	COEX_1	RF coexistence mechanism present	0x1*
RFIF	RFIF_RIPPLE	Ripple RF	0x0
	RFIF_EXT	External radio controller support	0x1
	RFIF_ATLAS	Atlas radio	0x2
	RFIF_ICYTRX	IcyTRx radio	0x3*
USEDBG	USEDBG_0	RF coexistence mechanism present	0x0
	USEDBG_1	Diagnostic port instantiated	0x1*
USECRYPT	USECRYPT_0	AES-CCM encryption block not present	0x0
	USECRYPT_1	AES-CCM encryption block present	0x1*
CLK_SEL	CLK_SEL_8	Default value is 8MHz	0x8*

Field Name	Value Symbol	Value Description	Hex Value
INTMODE	INTMODE_0	Interrupts are edge level generated, i.e. pulse	0x0*
	INTMODE_1	Interrupts are trigger level generated, i.e. stays active at 1 till acknowledgement	0x1
BUSTYPE	BUSTYPE_0	Processor bus type: AHB bus	0x0
	BUSTYPE_1	Processor bus type: XBAR bus	0x1*
DATA_WIDTH	DATA_WIDTH_0	Processor bus width: 16 bits	0x0
	DATA_WIDTH_1	Processor bus width: 32 bits	0x1*
ADD_WIDTH	ADD_WIDTH_14	EM size is 16 KB	0xE*

9.4.11 BB_INTCNTL

Bit Field	Field Name	Description
15	CSCNTDEVMSK	Mask CSCNT interrupts during events allowing CSCNT interrupt generation during events
12	AUDIOINT2MSK	Audio channel 2 interrupt mask
11	AUDIOINT1MSK	Audio channel 1 interrupt mask
10	AUDIOINT0MSK	Audio channel 0 interrupt mask
9	SWINTMSK	SW triggered interrupt mask
8	EVENTAPFAINTMSK	End of event / anticipated pre-fetch abort interrupt mask
7	FINETGTIMINTMSK	Fine target timer mask
6	GROSSTGTIMINTMSK	Gross target timer mask
5	ERRORINTMSK	Error interrupt mask
4	CRYPTINTMSK	Encryption engine interrupt mask
3	EVENTINTMSK	End of event interrupt mask
2	SLPINTMSK	Sleep Mode interrupt mask
1	RXINTMSK	Rx interrupt mask
0	CSCNTINTMSK	625us base time interrupt mask

Field Name	Value Symbol	Value Description	Hex Value
CSCNTDEVMSK	CSCNTDEVMSK_0	CSCNT interrupt not generated during events	0x0
	CSCNTDEVMSK_1	CSCNT interrupt generated during events	0x1*
AUDIOINT2MSK	AUDIOINT2MSK_0	Interrupt not generated	0x0*
	AUDIOINT2MSK_1	Interrupt generated	0x1
AUDIOINT1MSK	AUDIOINT1MSK_0	Interrupt not generated	0x0*
	AUDIOINT1MSK_1	Interrupt generated	0x1
AUDIOINT0MSK	AUDIOINT0MSK_0	Interrupt not generated	0x0*
	AUDIOINT0MSK_1	Interrupt generated	0x1

Field Name	Value Symbol	Value Description	Hex Value
SWINTMSK	SWINTMSK_0	Interrupt not generated	0x0*
	SWINTMSK_1	Interrupt generated	0x1
EVENTAPFAINTMSK	EVENTAPFAINTMSK_0	Interrupt not generated	0x0
	EVENTAPFAINTMSK_1	Interrupt generated	0x1*
FINETGTIMINTMSK	FINETGTIMINTMSK_0	Interrupt not generated	0x0*
	FINETGTIMINTMSK_1	Interrupt generated	0x1
GROSSTGTIMINTMSK	GROSSTGTIMINTMSK_0	Interrupt not generated	0x0*
	GROSSTGTIMINTMSK_1	Interrupt generated	0x1
ERRORINTMSK	ERRORINTMSK_0	Interrupt not generated	0x0*
	ERRORINTMSK_1	Interrupt generated	0x1
CRYPTINTMSK	CRYPTINTMSK_0	Interrupt not generated	0x0
	CRYPTINTMSK_1	Interrupt generated	0x1*
EVENTINTMSK	EVENTINTMSK_0	Interrupt not generated	0x0
	EVENTINTMSK_1	Interrupt generated	0x1*
SLPINTMSK	SLPINTMSK_0	Interrupt not generated	0x0
	SLPINTMSK_1	Interrupt generated	0x1*
RXINTMSK	RXINTMSK_0	Interrupt not generated	0x0
	RXINTMSK_1	Interrupt generated	0x1*
CSCNTINTMSK	CSCNTINTMSK_0	Interrupt not generated	0x0
	CSCNTINTMSK_1	Interrupt generated	0x1*

9.4.12 BB_INTSTAT

Bit Field	Field Name	Description
12	AUDIOINT2STAT	Audio channel 2 interrupt status
11	AUDIOINT1STAT	Audio channel 1 interrupt status
10	AUDIOINT0STAT	Audio channel 0 interrupt status
9	SWINTSTAT	SW triggered interrupt status
8	EVENTAPFAINTSTAT	End of event / anticipated pre-fetch abort interrupt status
7	FINETGTIMINTSTAT	Masked fine target timer error interrupt status
6	GROSSTGTIMINTSTAT	Masked gross target timer error interrupt status
5	ERRORINTSTAT	Masked error interrupt status
4	CRYPTINTSTAT	Masked encryption engine interrupt status
3	EVENTINTSTAT	Masked end of event interrupt status
2	SLPINTSTAT	Masked sleep interrupt status
1	RXINTSTAT	Masked packet reception interrupt status
0	CSCNTINTSTAT	Masked 625us base time reference interrupt status

Field Name	Value Symbol	Value Description	Hex Value
AUDIOINT2STAT	AUDIOINT2STAT_0	No Audio interrupt	0x0*
	AUDIOINT2STAT_1	An Audio interrupt is pending.	0x1
AUDIOINT1STAT	AUDIOINT1STAT_0	No Audio interrupt	0x0*
	AUDIOINT1STAT_1	An Audio interrupt is pending.	0x1
AUDIOINT0STAT	AUDIOINT0STAT_0	No Audio interrupt	0x0*
	AUDIOINT0STAT_1	An Audio interrupt is pending.	0x1
SWINTSTAT	SWINTSTAT_0	No SW triggered interrupt	0x0*
	SWINTSTAT_1	A SW triggered interrupt is pending	0x1
EVENTAPFAINTSTAT	EVENTAPFAINTSTAT_0	No end of event interrupt	0x0*
	EVENTAPFAINTSTAT_1	An end of event interrupt is pending	0x1
FINETGTIMINTSTAT	FINETGTIMINTSTAT_0	No fine target timer interrupt	0x0*
	FINETGTIMINTSTAT_1	A fine target timer interrupt is pending	0x1
GROSSTGTIMINTSTAT	GROSSTGTIMINTSTAT_0	No gross target timer interrupt	0x0*
	GROSSTGTIMINTSTAT_1	A gross target timer interrupt is pending	0x1
ERRORINTSTAT	ERRORINTSTAT_0	No error interrupt	0x0*
	ERRORINTSTAT_1	An error interrupt is pending	0x1
CRYPTINTSTAT	CRYPTINTSTAT_0	No encryption / decryption interrupt	0x0*
	CRYPTINTSTAT_1	An encryption / decryption interrupt is pending	0x1
EVENTINTSTAT	EVENTINTSTAT_0	No end of advertising / scanning / connection interrupt	0x0*
	EVENTINTSTAT_1	An end of advertising / scanning / connection interrupt is pending	0x1
SLPINTSTAT	SLPINTSTAT_0	No end of Sleep Mode interrupt	0x0*
	SLPINTSTAT_1	An end of Sleep Mode interrupt is pending	0x1
RXINTSTAT	RXINTSTAT_0	No Rx interrupt	0x0*
	RXINTSTAT_1	An Rx interrupt is pending	0x1
CSCNTINTSTAT	CSCNTINTSTAT_0	No 625us base time interrupt	0x0*
	CSCNTINTSTAT_1	A 625us base time interrupt is pending	0x1

9.4.13 BB_INTRAWSTAT

Bit Field	Field Name	Description
12	AUDIOINT2RAWSTAT	Audio channel 2 interrupt raw status
11	AUDIOINT1RAWSTAT	Audio channel 1 interrupt raw status
10	AUDIOINT0RAWSTAT	Audio channel 0 interrupt raw status
9	SWINTRAWSTAT	SW triggered interrupt raw status
8	EVENTAPFAINTRAWSTAT	End of event / anticipated pre-fetch abort interrupt raw status
7	FINETGTIMINTRAWSTAT	Masked fine target timer error interrupt raw status

Bit Field	Field Name	Description
6	GROSSTGTIMINTRAWSTAT	Masked gross target timer interrupt raw status
5	ERRORINTRAWSTAT	Masked error interrupt raw status
4	CRYPTINTRAWSTAT	Masked encryption engine interrupt raw status
3	EVENTINTRAWSTAT	Masked end of event interrupt raw status
2	SLPINTRAWSTAT	Masked sleep interrupt raw status
1	RXINTRAWSTAT	Masked packet reception interrupt raw status
0	CSCNTINTRAWSTAT	Masked 625us base time reference interrupt raw status

Field Name	Value Symbol	Value Description	Hex Value
AUDIOINT2RAWSTAT	AUDIOINT2RAWSTAT_0	No Audio interrupt	0x0*
	AUDIOINT2RAWSTAT_1	An Audio interrupt is pending.	0x1
AUDIOINT1RAWSTAT	AUDIOINT1RAWSTAT_0	No Audio interrupt	0x0*
	AUDIOINT1RAWSTAT_1	An Audio interrupt is pending.	0x1
AUDIOINT0RAWSTAT	AUDIOINT0RAWSTAT_0	No Audio interrupt	0x0*
	AUDIOINT0RAWSTAT_1	An Audio interrupt is pending.	0x1
SWINTRAWSTAT	SWINTRAWSTAT_0	No SW triggered interrupt	0x0*
	SWINTRAWSTAT_1	A SW triggered interrupt is pending	0x1
EVENTAPFAINTRAWSTAT	EVENTAPFAINTRAWSTAT_0	No end of event interrupt	0x0*
	EVENTAPFAINTRAWSTAT_1	An end of event interrupt is pending	0x1
FINETGTIMINTRAWSTAT	FINETGTIMINTRAWSTAT_0	No fine target timer interrupt	0x0*
	FINETGTIMINTRAWSTAT_1	A fine target timer interrupt is pending	0x1
GROSSTGTIMINTRAWSTAT	GROSSTGTIMINTRAWSTAT_0	No gross target timer interrupt	0x0*
	GROSSTGTIMINTRAWSTAT_1	A gross target timer interrupt is pending	0x1
ERRORINTRAWSTAT	ERRORINTRAWSTAT_0	No error interrupt	0x0*
	ERRORINTRAWSTAT_1	An error interrupt is pending	0x1
CRYPTINTRAWSTAT	CRYPTINTRAWSTAT_0	No encryption / decryption interrupt	0x0*
	CRYPTINTRAWSTAT_1	An encryption / decryption interrupt is pending	0x1
EVENTINTRAWSTAT	EVENTINTRAWSTAT_0	No end of advertising / scanning / connection interrupt	0x0*
	EVENTINTRAWSTAT_1	An end of advertising / scanning / connection interrupt is pending	0x1
SLPINTRAWSTAT	SLPINTRAWSTAT_0	No end of Sleep Mode interrupt	0x0*
	SLPINTRAWSTAT_1	An end of Sleep Mode interrupt is pending	0x1
RXINTRAWSTAT	RXINTRAWSTAT_0	No Rx interrupt	0x0*
	RXINTRAWSTAT_1	An Rx interrupt is pending	0x1
CSCNTINTRAWSTAT	CSCNTINTRAWSTAT_0	No 625us base time interrupt	0x0*
	CSCNTINTRAWSTAT_1	A 625us base time interrupt is pending	0x1

9.4.14 BB_INTACK

Bit Field	Field Name	Description
12	AUDIOINT2ACK	Audio channel 2 interrupt acknowledgement bit
11	AUDIOINT1ACK	Audio channel 1 interrupt acknowledgement bit
10	AUDIOINT0ACK	Audio channel 0 interrupt acknowledgement bit
9	SWINTACK	SW triggered interrupt acknowledgement bit
8	EVENTAPFAINTACK	End of event / anticipated pre-fetch abort interrupt acknowledgement bit
7	FINETGTIMINTACK	Fine target timer interrupt acknowledgement bit
6	GROSSTGTIMINTACK	Gross target timer interrupt acknowledgement bit
5	ERRORINTACK	Error interrupt acknowledgement bit
4	CRYPTINTACK	Encryption engine interrupt acknowledgement bit
3	EVENTINTACK	End of event interrupt acknowledgment bit
2	SLPINTACK	End of deep sleep interrupt acknowledgment bit
1	RXINTACK	Packet reception interrupt acknowledgment bit
0	CSCNTINTACK	625us base time reference interrupt acknowledgment bit

Field Name	Value Symbol	Value Description	Hex Value
AUDIOINT2ACK	AUDIOINT2ACK_0		0x0*
	AUDIOINT2ACK_1	Acknowledges the Audio channel 2 interrupt. This bit resets AUDIOINT2STAT and AUDIOINT2RAWSTAT flags.	0x1
AUDIOINT1ACK	AUDIOINT1ACK_0		0x0*
	AUDIOINT1ACK_1	Acknowledges the Audio channel 1 interrupt. This bit resets AUDIOINT2STAT and AUDIOINT2RAWSTAT flags.	0x1
AUDIOINT0ACK	AUDIOINT0ACK_0		0x0*
	AUDIOINT0ACK_1	Acknowledges the Audio channel 0 interrupt. This bit resets AUDIOINT2STAT and AUDIOINT2RAWSTAT flags.	0x1
SWINTACK	SWINTACK_0		0x0*
	SWINTACK_1	Acknowledges the SW triggered interrupt. This bit resets SWINTSTAT and SWINTRAWSTAT flags.	0x1
EVENTAPFAINTACK	EVENTAPFAINTACK_0		0x0*
	EVENTAPFAINTACK_1	Acknowledges the end of event / anticipated pre-fetch abort interrupt. This bit resets EVENTAPFAINTSTAT and EVENTAPFAINTRAWSTAT flags.	0x1
FINETGTIMINTACK	FINETGTIMINTACK_0		0x0*
	FINETGTIMINTACK_1	Acknowledges the fine timer interrupt. This bit resets FINETGTIMINTSTAT and FINETGTIMINTRAWSTAT flags	0x1

Field Name	Value Symbol	Value Description	Hex Value
GROSSTGTIMINTACK	GROSSTGTIMINTACK_0		0x0*
	GROSSTGTIMINTACK_1	Acknowledges the gross timer interrupt. This bit resets GROSSTGTIMINTSTAT and GROSSTGTIMINTRAWSTAT flags	0x1
ERRORINTACK	ERRORINTACK_0		0x0*
	ERRORINTACK_1	Acknowledges the error interrupt. This bit resets ERRORINTSTAT and ERRORINTRAWSTAT flags	0x1
CRYPTINTACK	CRYPTINTACK_0		0x0*
	CRYPTINTACK_1	Acknowledges the encryption engine interrupt. This bit resets CRYPTINTSTAT and CRYPTINTRAWSTAT flags	0x1
EVENTINTACK	EVENTINTACK_0		0x0*
	EVENTINTACK_1	Acknowledges the end of advertising / scanning / connection interrupt. This bit resets SLPINTSTAT and SLPINTRAWSTAT flags	0x1
SLPINTACK	SLPINTACK_0		0x0*
	SLPINTACK_1	Acknowledges the end of Sleep Mode interrupt. This bit resets SLPINTSTAT and SLPINTRAWSTAT flags	0x1
RXINTACK	RXINTACK_0		0x0*
	RXINTACK_1	Acknowledges the Rx interrupt. This bit resets RXINTSTAT and RXINTRAWSTAT flags	0x1
CSCNTINTACK	CSCNTINTACK_0		0x0*
	CSCNTINTACK_1	Acknowledges the CLKN interrupt. This bit resets CLKINTSTAT and CLKINTRAWSTAT flags	0x1

9.4.15 BB_BASETIMECNT

Bit Field	Field Name	Description
31	SAMP	Sample the base time counter
26:0	BASETIMECNT	Value of the 625us base time reference counter

Field Name	Value Symbol	Value Description	Hex Value
SAMP	SAMP_0		0x0*
	SAMP_1	Samples the base time counter value in BASETIMECNT register and resets at 0 when action is performed	0x1
BASETIMECNT	BASETIMECNT_0		0x0*

9.4.16 BB_FINETIMECNT

Bit Field	Field Name	Description
9:0	FINECNT	Value of the current us fine time reference counter

Field Name	Value Symbol	Value Description	Hex Value
FINECNT	FINECNT_0		0x0*

9.4.17 BB_BDADDRL

Bit Field	Field Name	Description
31:0	BDADDRL	Bluetooth low energy device address (LSB part)

Field Name	Value Symbol	Value Description	Hex Value
BDADDRL	BDADDRL_0		0x0*

9.4.18 BB_BDADDRU

Bit Field	Field Name	Description
16	PRIV_NPUB	Bluetooth low energy device address privacy indicator
15:0	BDADDRU	Bluetooth low energy device address (MSB part)

Field Name	Value Symbol	Value Description	Hex Value
PRIV_NPUB	PRIV_NPUB_0	Public Bluetooth device address	0x0*
	PRIV_NPUB_1	Private Bluetooth device address	0x1
BDADDRU	BDADDRU_0		0x0*

9.4.19 BB_ET_CURRENTRXDESCPTR

Bit Field	Field Name	Description
31:16	ETPTR	Exchange table pointer that determines the starting point of the exchange table
14:0	CURRENTRXDESCPTR	Rx descriptor pointer that determines the starting point of the receive buffer chained list

Field Name	Value Symbol	Value Description	Hex Value
ETPTR	ETPTR_0		0x0*
CURRENTRXDESCPTR	CURRENTRXDESCPTR_0		0x0*

9.4.20 BB_DEEPSLCNTL

Bit Field	Field Name	Description
31	EXTWKUPDSB	External wakeup disable
15	DEEP_SLEEP_STAT	Indicator of current deep sleep clock mux status
4	SOFT_WAKEUP_REQ	Wake up request from Bluetooth baseband software applying when system is in Deep Sleep Mode
3	DEEP_SLEEP_CORR_EN	625us base time reference integer and fractional part correction applying when system has been woken-up from Deep Sleep Mode
2	DEEP_SLEEP_ON	Bluetooth baseband core power mode control
1	RADIO_SLEEP_EN	Controls the radio module
0	OSC_SLEEP_EN	Controls the RF high frequency crystal oscillator

Field Name	Value Symbol	Value Description	Hex Value
EXTWKUPDSB	EXTWKUPDSB_0	Bluetooth baseband core can be woken by external wakeup	0x0*
	EXTWKUPDSB_1	Bluetooth baseband core cannot be woken up by external wakeup	0x1
DEEP_SLEEP_STAT	DEEP_SLEEP_STAT_0	Bluetooth baseband core is not yet in Deep Sleep Mode	0x0*
	DEEP_SLEEP_STAT_1	Bluetooth baseband core is in Deep Sleep Mode (only low_power_clk is running)	0x1
SOFT_WAKEUP_REQ	SOFT_WAKEUP_REQ_0	No action happens if it is written with 0	0x0*
	SOFT_WAKEUP_REQ_1	Wake up request from Bluetooth baseband software	0x1
DEEP_SLEEP_CORR_EN	DEEP_SLEEP_CORR_EN_0	No action happens if it is written with 0	0x0*
	DEEP_SLEEP_CORR_EN_1	Enables fine counter and base time counter when written	0x1
DEEP_SLEEP_ON	DEEP_SLEEP_ON_0	Bluetooth baseband core in normal active mode	0x0*
	DEEP_SLEEP_ON_1	Request Bluetooth baseband core to switch in Deep Sleep Mode	0x1
RADIO_SLEEP_EN	RADIO_SLEEP_EN_0	radio stands in normal active mode	0x0*
	RADIO_SLEEP_EN_1	Allow to disable radio	0x1
OSC_SLEEP_EN	OSC_SLEEP_EN_0	High frequency crystal oscillator stands in normal active mode	0x0*
	OSC_SLEEP_EN_1	Allow to disable high frequency crystal oscillator	0x1

9.4.21 BB_DEEPSLWKUP

Bit Field	Field Name	Description
31:0	DEEPSLTIME	Determines the time in low_power_clk clock cycles to spend in Deep Sleep Mode before waking up the device

Field Name	Value Symbol	Value Description	Hex Value
DEEPSLTIME	DEEPSLTIME_0		0x0*

9.4.22 BB_DEEPSLSTAT

Bit Field	Field Name	Description
31:0	DEEPSLDUR	Actual duration of the last deep sleep phase measured in low_power_clk clock cycle

Field Name	Value Symbol	Value Description	Hex Value
DEEPSLDUR	DEEPSLDUR_0		0x0*

9.4.23 BB_ENBPRESET

Bit Field	Field Name	Description
20:10	TWOSC	Time in low power oscillator cycles allowed for stabilization of the high frequency oscillator when the deep-Sleep Mode has been left due to sleep-timer expiry (DEEPSLWKUP-DEEPSLTIME])

Field Name	Value Symbol	Value Description	Hex Value
TWOSC	TWOSC_0		0x0*

9.4.24 BB_FINECNTCORR

Bit Field	Field Name	Description
9:0	FINECNTCORR	Phase correction value for the 625us reference counter (i.e. fine counter) in us

Field Name	Value Symbol	Value Description	Hex Value
FINECNTCORR	FINECNTCORR_0		0x0*

9.4.25 BB_BASETIMECNTCORR

Bit Field	Field Name	Description
26:0	BASETIMECNTCORR	Base time counter correction value

Field Name	Value Symbol	Value Description	Hex Value
BASETIMECNTCORR	BASETIMECNTCORR_0		0x0*

9.4.26 BB_DIAGCNTL

Bit Field	Field Name	Description
31	DIAG3_EN	Enable diagnostic port 3 output
29:24	DIAG3	
23	DIAG2_EN	Enable diagnostic port 2 output
21:16	DIAG2	
15	DIAG1_EN	Enable diagnostic port 1 output
13:8	DIAG1	
7	DIAG0_EN	Enable diagnostic port 0 output
5:0	DIAG0	

Field Name	Value Symbol	Value Description	Hex Value
DIAG3_EN	DIAG3_EN_0	Disable diagnostic port 3 output. All outputs are set to 0x0.	0x0*
	DIAG3_EN_1	Enable diagnostic port 3 output	0x1
DIAG3	DIAG3_0	Selection of the outputs that must be driven to the diagnostic port 3	0x0*
DIAG2_EN	DIAG2_EN_0	Disable diagnostic port 2 output. All outputs are set to 0x0.	0x0*
	DIAG2_EN_1	Enable diagnostic port 2 output	0x1
DIAG2	DIAG2_0	Selection of the outputs that must be driven to the diagnostic port 2	0x0*
DIAG1_EN	DIAG1_EN_0	Disable diagnostic port 1 output. All outputs are set to 0x0.	0x0*
	DIAG1_EN_1	Enable diagnostic port 1 output	0x1
DIAG1	DIAG1_0	Selection of the outputs that must be driven to the diagnostic port 1	0x0*
DIAG0_EN	DIAG0_EN_0	Disable diagnostic port 0 output. All outputs are set to 0x0.	0x0*
	DIAG0_EN_1	Enable diagnostic port 0 output	0x1
DIAG0	DIAG0_0	Selection of the outputs that must be driven to the diagnostic port 0	0x0*

9.4.27 BB_DIAGSTAT

Bit Field	Field Name	Description
31:24	DIAG3STAT	Directly connected to ble_dbg3[7:0] output (debug use only)
23:16	DIAG2STAT	Directly connected to ble_dbg2[7:0] output (debug use only)
15:8	DIAG1STAT	Directly connected to ble_dbg1[7:0] output (debug use only)
7:0	DIAG0STAT	Directly connected to ble_dbg0[7:0] output (debug use only)

Field Name	Value Symbol	Value Description	Hex Value
DIAG3STAT	DIAG3STAT_0		0x0*
DIAG2STAT	DIAG2STAT_0		0x0*
DIAG1STAT	DIAG1STAT_0		0x0*
DIAG0STAT	DIAG0STAT_0		0x0*

9.4.28 BB_DEBUGADDMAX

Bit Field	Field Name	Description
31:16	REG_ADDMAX	Upper limit for the register zone indicated by the reg_inzone flag
15:0	EM_ADDMAX	Upper limit for the exchange memory zone indicated by the em_inzone flag

Field Name	Value Symbol	Value Description	Hex Value
REG_ADDMAX	REG_ADDMAX_0		0x0*
EM_ADDMAX	EM_ADDMAX_0		0x0*

9.4.29 BB_DEBUGADDMIN

Bit Field	Field Name	Description
31:16	REG_ADDMIN	Lower limit for the register zone indicated by the reg_inzone flag
15:0	EM_ADDMIN	Lower limit for the exchange memory zone indicated by the em_inzone flag

Field Name	Value Symbol	Value Description	Hex Value
REG_ADDMIN	REG_ADDMIN_0		0x0*
EM_ADDMIN	EM_ADDMIN_0		0x0*

9.4.30 BB_ERRORTYPESTAT

Bit Field	Field Name	Description
19	RAL_UNDERRUN	Indicates Resolving Address List engine Underrun issue, happens when RAL List parsing not finished on time
18	RAL_ERROR	Indicates Resolving Address List engine faced a bad setting
17	CONCEVTIRQ_ERROR	Indicates whether two consecutive and concurrent ble_event_irq have been generated, and not acknowledged in time by the Bluetooth baseband software
16	RXDATA_PTR_ERROR	Indicates whether Rx data buffer pointer value programmed is null (major failure)
15	TXDATA_PTR_ERROR	Indicates whether Tx data buffer pointer value programmed is null during advertising / scanning / initiating events, or during master / slave connections with non-null packet length (major failure)
14	RXDESC_EMPTY_ERROR	Indicates whether Rx descriptor pointer value programmed in register is null (major failure)

Bit Field	Field Name	Description
13	TXDESC_EMPTY_ERROR	Indicates whether Tx descriptor pointer value programmed in control structure is null during advertising / scanning / initiating events (major failure)
12	CSFORMAT_ERROR	Indicates whether CS-FORMAT has been programmed with an invalid value (major failure)
11	LLCHMAP_ERROR	Indicates Link Layer channel map error, happens when actual number of CS-LLCHMAP bits set to one is different from CS-NBCHGOOD at the beginning of frequency hopping process
10	ADV_UNDERRUN	Indicates advertising interval under run
9	IFS_UNDERRUN	Indicates inter frame space under run, occurs if IFS time is not enough to update and read control structure / descriptors, and/or white list parsing is not finished and/or decryption time is too long to be finished on time
8	WHITELIST_ERROR	Indicates white list timeout error, occurs if white list parsing is not finished on time
7	EVT_CNTL_APFM_ERROR	Indicates anticipated pre-fetch mechanism error: happens when 2 consecutive events are programmed, and when the first event is not completely finished while second pre-fetch instant is reached
6	EVT_SCHDL_APFM_ERROR	Indicates anticipated pre-fetch mechanism error: happens when 2 consecutive events are programmed, and when the first event is not completely finished while second pre-fetch instant is reached
5	EVT_SCHDL_ENTRY_ERROR	Indicates event scheduler faced invalid timing programing on two consecutive ET entries (e.g first one with 624us offset and second one with no offset)
4	EVT_SCHDL_EMACC_ERROR	Indicates event scheduler exchange memory access error, happens when exchange memory accesses are not served in time, and blocks the exchange table entry read
3	RADIO_EMACC_ERROR	Indicates radio controller exchange memory access error, happens when exchange memory accesses are not served in time and data are corrupted
2	PKTCNTL_EMACC_ERROR	Indicates packet controller exchange memory access error, happens when exchange memory accesses are not served in time and Tx/Rx data are corrupted
1	RXCRYPT_ERROR	Indicates real time decryption error, happens when AES-CCM decryption is too slow compared to packet controller requests
0	TXCRYPT_ERROR	Indicates real time encryption error, happens when AES-CCM encryption is too slow compared to packet controller requests

Field Name	Value Symbol	Value Description	Hex Value
RAL_UNDERRUN	RAL_UNDERRUN_0	No error	0x0*
	RAL_UNDERRUN_1	Error occurred	0x1
RAL_ERROR	RAL_ERROR_0	No error	0x0*
	RAL_ERROR_1	Error occurred	0x1
CONCEVTIRQ_ERROR	CONCEVTIRQ_ERROR_0	No error	0x0*
	CONCEVTIRQ_ERROR_1	Error occurred	0x1
RXDATA_PTR_ERROR	RXDATA_PTR_ERROR_0	No error	0x0*
	RXDATA_PTR_ERROR_1	Error occurred	0x1
TXDATA_PTR_ERROR	TXDATA_PTR_ERROR_0	No error	0x0*
	TXDATA_PTR_ERROR_1	Error occurred	0x1

Field Name	Value Symbol	Value Description	Hex Value
RXDESC_EMPTY_ERROR	RXDESC_EMPTY_ERROR_0	No error	0x0*
	RXDESC_EMPTY_ERROR_1	Error occurred	0x1
TXDESC_EMPTY_ERROR	TXDESC_EMPTY_ERROR_0	No error	0x0*
	TXDESC_EMPTY_ERROR_1	Error occurred	0x1
CSFORMAT_ERROR	CSFORMAT_ERROR_0	No error	0x0*
	CSFORMAT_ERROR_1	Error occurred	0x1
LLCHMAP_ERROR	LLCHMAP_ERROR_0	No error	0x0*
	LLCHMAP_ERROR_1	Error occurred	0x1
ADV_UNDERRUN	ADV_UNDERRUN_0	No error	0x0*
	ADV_UNDERRUN_1	Error occurred	0x1
IFS_UNDERRUN	IFS_UNDERRUN_0	No error	0x0*
	IFS_UNDERRUN_1	Error occurred	0x1
WHITELIST_ERROR	WHITELIST_ERROR_0	No error	0x0*
	WHITELIST_ERROR_1	Error occurred	0x1
EVT_CNTL_APFM_ERROR	EVT_CNTL_APFM_ERROR_0	No error	0x0*
	EVT_CNTL_APFM_ERROR_1	Error occurred	0x1
EVT_SCHDL_APFM_ERROR	EVT_SCHDL_APFM_ERROR_0	No error	0x0*
	EVT_SCHDL_APFM_ERROR_1	Error occurred	0x1
EVT_SCHDL_ENTRY_ERROR	EVT_SCHDL_ENTRY_ERROR_0	No error	0x0*
	EVT_SCHDL_ENTRY_ERROR_1	Error occurred	0x1
EVT_SCHDL_EMACC_ERROR	EVT_SCHDL_EMACC_ERROR_0	No error	0x0*
	EVT_SCHDL_EMACC_ERROR_1	Error occurred	0x1
RADIO_EMACC_ERROR	RADIO_EMACC_ERROR_0	No error	0x0*
	RADIO_EMACC_ERROR_1	Error occurred	0x1
PKTCNTL_EMACC_ERROR	PKTCNTL_EMACC_ERROR_0	No error	0x0*
	PKTCNTL_EMACC_ERROR_1	Error occurred	0x1
RXCRYPT_ERROR	RXCRYPT_ERROR_0	No error	0x0*
	RXCRYPT_ERROR_1	Error occurred	0x1
TXCRYPT_ERROR	TXCRYPT_ERROR_0	No error	0x0*
	TXCRYPT_ERROR_1	Error occurred	0x1

9.4.31 BB_SWPROFILING

Bit Field	Field Name	Description
31:0	SWPROF	Software profiling register: used by Bluetooth baseband software for profiling purpose

Field Name	Value Symbol	Value Description	Hex Value
SWPROF	SWPROF_0		0x0*

9.4.32 BB_RADIOCNTL0

Bit Field	Field Name	Description
31:16	SPIPTR	Pointer to the buffer containing data to be transferred to or received from the SPI port
5:4	SPIFREQ	SPI clock frequency
1	SPICOMP	SPI transfer status
0	SPIGO	Start SPI transfer when writing a 1

Field Name	Value Symbol	Value Description	Hex Value
SPIPTR	SPIPTR_0	SPI pointer	0x0*
SPIFREQ	SPIFREQ_0	SPI clock is master1_gclk / 3	0x0*
	SPIFREQ_1	NA	0x1
	SPIFREQ_2	NA	0x2
	SPIFREQ_3	NA	0x3
SPICOMP	SPICOMP_0	Indicates SPI transfer in progress	0x0
	SPICOMP_1	Indicates SPI transfer is completed. Bluetooth baseband core is ready to start a new transfer	0x1*
SPIGO	SPIGO_0		0x0*
	SPIGO_1	Triggers the SPI transfer	0x1

9.4.33 BB_RADIOCNTL1

Bit Field	Field Name	Description
31	FORCEAGC_EN	Control ATLAS/Ripple AGC force mode based on radio BB_RADIOCNTL1_FORCEAGC_LENGTH value
30	FORCEBLEIQ	Control Ripple modulation mode in between FM and I and Q
27:16	FORCEAGC_LENGTH	Control ATLAS/Ripple AGC force mode based on radio BB_RADIOCNTL1_FORCEAGC_LENGTH value
15	SYNC_PULSE_MODE	Define whether the SYNC_P pulse is generated as pulse or level
13	DPCORR_EN	Enable the use of delayed DC compensated data path in radio correlator block
12	JEF_SELECT	Selects Jitter Elimination FIFO
8:4	XRFSEL	Extended radio selection field
3:0	SUBVERSION	CSEM RF Sub-version selection

Field Name	Value Symbol	Value Description	Hex Value
FORCEAGC_EN	FORCEAGC_EN_0	Disable	0x0*
	FORCEAGC_EN_1	Enable	0x1
FORCEBLEIQ	FORCEBLEIQ_0	FM modulation mode	0x0*
	FORCEBLEIQ_1	I and Q modulation mode	0x1
FORCEAGC_LENGTH	FORCEAGC_LENGTH_0		0x0*

Field Name	Value Symbol	Value Description	Hex Value
SYNC_PULSE_MODE	SYNC_PULSE_MODE_0	SYNC_P generated as pulse	0x0*
	SYNC_PULSE_MODE_1	SYNC_P generated as level	0x1
DPCORR_EN	DPCORR_EN_0	Disable	0x0*
	DPCORR_EN_1	Enable	0x1
JEF_SELECT	JEF_SELECT_0		0x0*
	JEF_SELECT_1		0x1
XRFSEL	XRFSEL_0	No radio selected	0x0*
	XRFSEL_3	Integrated radio (Bluetooth low energy technology)	0x3
SUBVERSION	SUBVERSION_3	Current RFFE revision	0x3

9.4.34 BB_RADIOCNTL2

Bit Field	Field Name	Description
15:0	FREQTABLE_PTR	Frequency table pointer

Field Name	Value Symbol	Value Description	Hex Value
FREQTABLE_PTR	FREQTABLE_PTR_64		0x40*

9.4.35 BB_RADIOPWRUPDN0

Bit Field	Field Name	Description
23:16	RXPWRUP0	This register holds the length in us of the Rx power up phase for the current radio device
12:8	TXPWRDN0	This register extends the length in us of the Tx power down phase for the current radio device
7:0	TXPWRUP0	This register holds the length in us of the Tx power up phase for the current radio device

Field Name	Value Symbol	Value Description	Hex Value
RXPWRUP0	RXPWRUP0_0		0x0*
TXPWRDN0	TXPWRDN0_0		0x0*
TXPWRUP0	TXPWRUP0_0		0x0*

9.4.36 BB_RADIOPWRUPDN1

Bit Field	Field Name	Description
23:16	RXPWRUP1	This register holds the length in us of the Rx power up phase for the current radio device
12:8	TXPWRDN1	This register extends the length in us of the Tx power down phase for the current radio device
7:0	TXPWRUP1	This register holds the length in us of the Tx power up phase for the current radio device

Field Name	Value Symbol	Value Description	Hex Value
RXPWRUP1	RXPWRUP1_0		0x0*
TXPWRDN1	TXPWRDN1_0		0x0*
TXPWRUP1	TXPWRUP1_0		0x0*

9.4.37 BB_RADIOTXRXTIM0

Bit Field	Field Name	Description
31:24	TXPATHDLY0	
20:16	RXPATHDLY0	
14:8	RFRXTMDA0	
6:0	SYNC_POSITION0	

Field Name	Value Symbol	Value Description	Hex Value
TXPATHDLY0	TXPATHDLY0_0		0x0*
RXPATHDLY0	RXPATHDLY0_0		0x0*
RFRXTMDA0	RFRXTMDA0_0		0x0*
SYNC_POSITION0	SYNC_POSITION0_0		0x0*

9.4.38 BB_RADIOTXRXTIM1

Bit Field	Field Name	Description
31:24	TXPATHDLY1	
20:16	RXPATHDLY1	
14:8	RFRXTMDA1	
6:0	SYNC_POSITION1	

Field Name	Value Symbol	Value Description	Hex Value
TXPATHDLY1	TXPATHDLY1_0		0x0*
RXPATHDLY1	RXPATHDLY1_0		0x0*

Field Name	Value Symbol	Value Description	Hex Value
RFRXTMDA1	RFRXTMDA1_0		0x0*
SYNC_POSITION1	SYNC_POSITION1_0		0x0*

9.4.39 BB_SPIPTRCNTL0

Bit Field	Field Name	Description
31:16	TXOFFPTR	Pointer to the TxOFF sequence address section
15:0	TXONPTR	Pointer to the TxON sequence address section

Field Name	Value Symbol	Value Description	Hex Value
TXOFFPTR	TXOFFPTR_0		0x0*
TXONPTR	TXONPTR_0		0x0*

9.4.40 BB_SPIPTRCNTL1

Bit Field	Field Name	Description
31:16	RXOFFPTR	Pointer to the RxOFF sequence address section
15:0	RXONPTR	Pointer to the RxON sequence address section

Field Name	Value Symbol	Value Description	Hex Value
RXOFFPTR	RXOFFPTR_0		0x0*
RXONPTR	RXONPTR_0		0x0*

9.4.41 BB_SPIPTRCNTL2

Bit Field	Field Name	Description
15:0	RSSIPTR	Pointer to the RSSI read sequence address section

Field Name	Value Symbol	Value Description	Hex Value
RSSIPTR	RSSIPTR_0		0x0*

9.4.42 BB_ADVCHMAP

Bit Field	Field Name	Description
2:0	ADVCHMAP	Advertising channel map, defined as per the advertising connection settings. Contains advertising channels index 37 to 39

Field Name	Value Symbol	Value Description	Hex Value
ADVCHMAP	ADVCHMAP_7		0x7*

9.4.43 BB_ADVTIM

Bit Field	Field Name	Description
13:0	ADVINT	Advertising packet interval defines the time interval in between two ADV_xxx packet sent (value in us)

Field Name	Value Symbol	Value Description	Hex Value
ADVINT	ADVINT_0		0x0*

9.4.44 BB_ACTSCANSTAT

Bit Field	Field Name	Description
24:16	BACKOFF	Active scan mode back-off counter initialization value
8:0	UPPERLIMIT	Active scan mode upper limit counter value

Field Name	Value Symbol	Value Description	Hex Value
BACKOFF	BACKOFF_1		0x1*
UPPERLIMIT	UPPERLIMIT_1		0x1*

9.4.45 BB_WLPUBADDPTR

Bit Field	Field Name	Description
15:0	WLPUBADDPTR	Start address pointer of the public devices white list

Field Name	Value Symbol	Value Description	Hex Value
WLPUBADDPTR	WLPUBADDPTR_0		0x0*

9.4.46 BB_WLPRIVADDPTR

Bit Field	Field Name	Description
15:0	WLPRIVADDPTR	Start address pointer of the private devices white list

Field Name	Value Symbol	Value Description	Hex Value
WLPRIVADDPTR	WLPRIVADDPTR_0		0x0*

9.4.47 BB_WLNBDEV

Bit Field	Field Name	Description
15:8	NBPRIVDEV	Number of private devices in the white list
7:0	NBPUBDEV	Number of public devices in the white list

Field Name	Value Symbol	Value Description	Hex Value
NBPRIVDEV	NBPRIVDEV_0		0x0*
NBPUBDEV	NBPUBDEV_0		0x0*

9.4.48 BB_AESCNTL

Bit Field	Field Name	Description
1	AES_MODE	Cipher mode control
0	AES_START	Starts AES-128 ciphering process

Field Name	Value Symbol	Value Description	Hex Value
AES_MODE	AES_MODE_0	Cipher mode	0x0*
	AES_MODE_1	Not implemented	0x1
AES_START	AES_START_0		0x0*
	AES_START_1	Starts AES-128 ciphering process (the bit is reset once the process is finished)	0x1

9.4.49 BB_AESKEY31_0

Bit Field	Field Name	Description
31:0	AESKEY31_0	AES encryption 128-bit key (bits 31 down to 0)

Field Name	Value Symbol	Value Description	Hex Value
AESKEY31_0	AESKEY31_0_0		0x0*

9.4.50 BB_AESKEY63_32

Bit Field	Field Name	Description
31:0	AESKEY63_32	AES encryption 128-bit key (bits 63 down to 32)

Field Name	Value Symbol	Value Description	Hex Value
AESKEY63_32	AESKEY63_32_0		0x0*

9.4.51 BB_AESKEY95_64

Bit Field	Field Name	Description
31:0	AESKEY95_64	AES encryption 128-bit key (bits 95 down to 64)

Field Name	Value Symbol	Value Description	Hex Value
AESKEY95_64	AESKEY95_64_0		0x0*

9.4.52 BB_AESKEY127_96

Bit Field	Field Name	Description
31:0	AESKEY127_96	AES encryption 128-bit key (bits 127 down to 96)

Field Name	Value Symbol	Value Description	Hex Value
AESKEY127_96	AESKEY127_96_0		0x0*

9.4.53 BB_AESPTR

Bit Field	Field Name	Description
15:0	AESPTR	Pointer to the memory zone where the block to cipher using AES-128 is stored.

Field Name	Value Symbol	Value Description	Hex Value
AESPTR	AESPTR_0		0x0*

9.4.54 BB_TXMICVAL

Bit Field	Field Name	Description
31:0	TXMICVAL	AES-CCM plain MIC value. Valid on when MIC has been calculated (in Tx)

Field Name	Value Symbol	Value Description	Hex Value
TXMICVAL	TXMICVAL_0		0x0*

9.4.55 BB_RXMICVAL

Bit Field	Field Name	Description
31:0	RXMICVAL	AES-CCM plain MIC value. Valid on once MIC has been extracted from Rx packet

Field Name	Value Symbol	Value Description	Hex Value
RXMICVAL	RXMICVAL_0		0x0*

9.4.56 BB_RFTESTCNTL

Bit Field	Field Name	Description
31	INFINITERX	Applicable in RF Test Mode only
27	RXPKTCNTEN	Applicable in RF Test Mode only
15	INFINITETX	Applicable in RF Test Mode only
14	TXLENGTHSRC	Applicable only in Tx/Rx RF Test Mode
13	PRBSTYPE	Applicable only in Tx/Rx RF Test Mode
12	TXPLDSRC	Applicable only in Tx/Rx RF Test Mode
11	TXPKTCNTEN	Applicable in RF Test Mode only
8:0	TXLENGTH	Tx packet length in number of bytes

Field Name	Value Symbol	Value Description	Hex Value
INFINITERX	INFINITERX_0	Normal mode of operation	0x0*
	INFINITERX_1	Infinite Rx window	0x1
RXPKTCNTEN	RXPKTCNTEN_0	Rx packet count disabled	0x0*
	RXPKTCNTEN_1	Rx packet count enabled, and reported in CS-RXCCMPKTCNT and RFTESTRXSTAT-RXPKTCNT on RF abort command	0x1
INFINITETX	INFINITETX_0	Normal mode of operation	0x0*
	INFINITETX_1	Infinite Tx packet / Normal start of a packet but endless payload	0x1
TXLENGTHSRC	TXLENGTHSRC_0	Normal mode of operation: TxDESC-TXADVLEN controls the Tx packet payload size	0x0*
	TXLENGTHSRC_1	Uses RFTESTCTRL-TXLENGTH packet length (can support up to 512 bytes transmit)	0x1
PRBSTYPE	PRBSTYPE_0	Tx packet payload are PRBS9 type	0x0*
	PRBSTYPE_1	Tx packet payload are PRBS15 type	0x1
TXPLDSRC	TXPLDSRC_0	Tx packet payload source is the control structure	0x0*
	TXPLDSRC_1	Tx packet payload are PRBS generator	0x1
TXPKTCNTEN	TXPKTCNTEN_0	Tx packet count disabled	0x0
	TXPKTCNTEN_1	Tx packet count enabled, and reported in CS-TXCCMPKTCNT and RFTESTTXSTAT-TXPKTCNT on RF abort command	0x1*
TXLENGTH	TXLENGTH_0		0x0*

9.4.57 BB_RFTESTTXSTAT

Bit Field	Field Name	Description
31:0	TXPKTCNT	Reports number of transmitted packets during test modes

Field Name	Value Symbol	Value Description	Hex Value
TXPKTCNT	TXPKTCNT_00		0x0*

9.4.58 BB_RFTESTRXSTAT

Bit Field	Field Name	Description
31:0	RXPKTCNT	Reports number of correctly received packets during test modes

Field Name	Value Symbol	Value Description	Hex Value
RXPKTCNT	RXPKTCNTX_0		0x0*

9.4.59 BB_TIMGENCNTL

Bit Field	Field Name	Description
31	APFM_EN	Controls the anticipated pre-fetch abort mechanism
25:16	PREFETCHABORT_TIME	Defines the instant in us at which immediate abort is required after anticipated pre-fetch abort
8:0	PREFETCH_TIME	Defines exchange table pre-fetch instant in us

Field Name	Value Symbol	Value Description	Hex Value
APFM_EN	APFM_EN_0	Disabled	0x0
	APFM_EN_1	Enabled	0x1*
PREFETCHABORT_TIME	PREFETCHABORT_TIME_254		0x1FE*
PREFETCH_TIME	PREFETCH_TIME_150		0x96*

9.4.60 BB_GROSSTIMTGT

Bit Field	Field Name	Description
22:0	GROSSTARGET	Gross timer target value on which a ble_grosstgtim_irq must be generated (precision of 10ms)

Field Name	Value Symbol	Value Description	Hex Value
GROSSTARGET	GROSSTARGET_0		0x0*

9.4.61 BB_FINETIMTGT

Bit Field	Field Name	Description
26:0	FINETARGET	Fine timer target value on which a ble_finetgtim_irq must be generated (precision of 625us)

Field Name	Value Symbol	Value Description	Hex Value
FINETARGET	FINETARGET_0		0x0*

9.4.62 BB_COEXIFCNTL0

Bit Field	Field Name	Description
25:24	MWSSCANFREQMSK	Determines how mws_scan_frequency impacts Bluetooth low energy technology Tx and Rx
21:20	WLCRXPRIOMODE	Defines Bluetooth low energy technology packet ble_rx mode behavior
17:16	WLCTXPRIOMODE	Defines Bluetooth low energy technology packet ble_tx mode behavior
15:14	MWSTXFREQMSK	Determines how MWS Tx Frequency impacts Bluetooth low energy technology Tx and Rx
13:12	MWSRXFREQMSK	Determines how MWS Rx Frequency impacts Bluetooth low energy technology Tx and Rx
11:10	MWSTXMSK	Determines how mws_tx impacts Bluetooth low energy technology Tx and Rx
9:8	MWSRXMSK	Determines how mws_rx impacts Bluetooth low energy technology Tx and Rx
7:6	TXMSK	Determines how tx impacts Bluetooth low energy technology Tx and Rx
5:4	RXMSK	Determines how rx impacts Bluetooth low energy technology Tx and Rx
3	MWSWCI_EN	Enable / Disable control of the WCI MWS Coexistence interface / Valid in Dual Mode only
2	MWSCOEX_EN	Enable / Disable control of the MWS Coexistence control / Valid in Dual Mode only
1	SYNCGEN_EN	Determines whether ble_sync is generated or not
0	COEX_EN	Enable / disable control of the MWS/RF coexistence control

Field Name	Value Symbol	Value Description	Hex Value
MWSSCANFREQMSK	MWSSCANFREQMSK_0	mws_scan_frequency has no impact	0x0*
	MWSSCANFREQMSK_1	mws_scan_frequency can stop Bluetooth low energy technology Tx, no impact on Bluetooth low energy technology Rx	0x1
	MWSSCANFREQMSK_2	mws_scan_frequency can stop Bluetooth low energy technology Rx, no impact on Bluetooth low energy technology Tx	0x2
	MWSSCANFREQMSK_3	mws_scan_frequency can stop both Bluetooth low energy technology Tx and Bluetooth low energy technology Rx	0x3
WLCRXPRIOMODE	WLCRXPRIOMODE_0	Rx indication excluding Rx power up delay (starts when correlator is enabled)	0x0*
	WLCRXPRIOMODE_1	Rx indication including Rx power up delay	0x1
	WLCRXPRIOMODE_2	Rx High priority indicator	0x2
	WLCRXPRIOMODE_3	NA	0x3

Field Name	Value Symbol	Value Description	Hex Value
WLCTXPRIOMODE	WLCTXPRIOMODE_0	Tx indication excluding Tx power up delay	0x0*
	WLCTXPRIOMODE_1	Tx indication including Tx power up delay	0x1
	WLCTXPRIOMODE_2	Tx High priority indicator	0x2
	WLCTXPRIOMODE_3	NA	0x3
MWSTXFREQMSK	MWSTXFREQMSK_0	mws Tx Frequency has no impact	0x0*
	MWSTXFREQMSK_1	mws Tx Frequency can stop Bluetooth low energy technology Tx, no impact on Bluetooth low energy technology Rx	0x1
	MWSTXFREQMSK_2	mws Tx Frequency can stop Bluetooth low energy technology Rx, no impact on Bluetooth low energy technology Tx	0x2
	MWSTXFREQMSK_3	mws Tx Frequency can stop both Bluetooth low energy technology Tx and Bluetooth low energy technology Rx	0x3
MWSRXFREQMSK	MWSRXFREQMSK_0	mws Tx Frequency has no impact	0x0
	MWSRXFREQMSK_1	mws Tx Frequency can stop Bluetooth low energy technology Tx, no impact on Bluetooth low energy technology Rx	0x1*
	MWSRXFREQMSK_2	mws Tx Frequency can stop Bluetooth low energy technology Rx, no impact on Bluetooth low energy technology Tx	0x2
	MWSRXFREQMSK_3	mws Tx Frequency can stop both Bluetooth low energy technology Tx and Bluetooth low energy technology Rx	0x3
MWSTXMSK	MWSTXMSK_0	mws_tx has no impact	0x0*
	MWSTXMSK_1	mws_tx can stopBluetooth low energy technology Tx, no impact on Bluetooth low energy technology Rx	0x1
	MWSTXMSK_2	mws_tx can stop Bluetooth low energy technology Rx, no impact on Bluetooth low energy technology Tx	0x2
	MWSTXMSK_3	mws_tx can stop both Bluetooth low energy technology Tx and Bluetooth low energy technology Rx	0x3
MWSRXMSK	MWSRXMSK_0	mws_tx has no impact	0x0
	MWSRXMSK_1	mws_tx can stop Bluetooth low energy technology Tx, no impact on Bluetooth low energy technology Rx	0x1*
	MWSRXMSK_2	mws_tx can stop Bluetooth low energy technology Rx, no impact on Bluetooth low energy technology Tx	0x2
	MWSRXMSK_3	mws_tx can stop both Bluetooth low energy technology Tx and Bluetooth low energy technology Rx	0x3

Field Name	Value Symbol	Value Description	Hex Value
TXMSK	TXMSK_0	tx has no impact	0x0*
	TXMSK_1	tx can stop Bluetooth low energy technology Tx, no impact on Bluetooth low energy technology Rx	0x1
	TXMSK_2	tx can stop Bluetooth low energy technology Rx, no impact on Bluetooth low energy technology Tx	0x2
	TXMSK_3	tx can stop both Bluetooth low energy technology Tx and Bluetooth low energy technology Rx	0x3
RXMSK	RXMSK_0	rx has no impact	0x0
	RXMSK_1	rx can stop Bluetooth low energy technology Tx, no impact on Bluetooth low energy technology Rx	0x1*
	RXMSK_2	rx can stop Bluetooth low energy technology Rx, no impact on Bluetooth low energy technology Tx	0x2
	RXMSK_3	rx can stop both Bluetooth low energy technology Tx and Bluetooth low energy technology Rx	0x3
MWSWCI_EN	MWSWCI_EN_0	MWS WCI Interface disabled	0x0*
	MWSWCI_EN_1	MWS WCI Interface enabled	0x1
MWSCOEX_EN	MWSCOEX_EN_0	MWS Coexistence interface disabled	0x0*
	MWSCOEX_EN_1	MWS Coexistence interface enabled	0x1
SYNCGEN_EN	SYNCGEN_EN_0	ble_sync pulse not generated	0x0*
	SYNCGEN_EN_1	ble_sync pulse generated	0x1
COEX_EN	COEX_EN_0	Coexistence interface disabled	0x0*
	COEX_EN_1	Coexistence interface enabled	0x1

9.4.63 BB_COEXIFCNTL1

Bit Field	Field Name	Description
28:24	WLCPRXTHR	Determines the threshold for Rx priority setting (applies on ble_rx if WLCRXPRIOMODE equals "10")
20:16	WLCPTXTHR	Determines the threshold for priority setting (applies on ble_tx if WLCTXPRIOMODE equals "10")
14:8	WLCPDURATION	Determines how many us the priority information must be maintained (applies on ble_tx and ble_rx if WLCTXPRIOMODE equals "10")
6:0	WLCPDELAY	Determines the delay (in us) in Tx/Rx enables rises the time Bluetooth low energy technology Tx/Rx priority has to be provided (applies on ble_tx and ble_rx if WLCTXPRIOMODE equals "10")

Field Name	Value Symbol	Value Description	Hex Value
WLCPRXTHR	WLCPRXTHR_0	If ble_pti[3:0] output value is greater than WLCPRXTHR, then Rx Bluetooth low energy technology priority is considered as high, and must be provided to the RF coexistence interface	0x0*
WLCPTXTHR	WLCPTXTHR_0	If ble_pti[3:0] output value is greater than WLCPTXTHR, then Tx Bluetooth low energy technology priority is considered as high, and must be provided to the RF coexistence interface	0x0*
WLCPDURATION	WLCPDURATION_0	Note that if WLCPDURATION = 0x00, then Tx/Rx priority levels are maintained till Tx/Rx EN are de-asserted.	0x0*
WLCPDELAY	WLCPDELAY_0		0x0*

9.4.64 BB_COEXIFCNTL2

Bit Field	Field Name	Description
11:8	RX_ANT_DELAY	Time (in us) by which is anticipated bt_rx to be provided before effective Radio receipt operation
3:0	TX_ANT_DELAY	Time (in us) by which is anticipated bt_tx to be provided before effective Radio transmit operation

Field Name	Value Symbol	Value Description	Hex Value
RX_ANT_DELAY	RX_ANT_DELAY_0		0x0*
TX_ANT_DELAY	TX_ANT_DELAY_0		0x0*

9.4.65 BB_BBMPRIO0

Bit Field	Field Name	Description
31:28	BLEM7	Set priority value for passive scanning
27:24	BLEM6	Set priority value for non-connectable advertising
23:20	BLEM5	Set priority value for connectable advertising Bluetooth low energy technology message
19:16	BLEM4	Set priority value for active scanning Bluetooth low energy technology message
15:12	BLEM3	Set priority value for initiating (scanning) Bluetooth low energy technology message
11:8	BLEM2	Set priority value for data channel transmission Bluetooth low energy technology message
7:4	BLEM1	Set priority value for LLCP Bluetooth low energy technology message
3:0	BLEM0	Set priority value for initiating (connection request response) Bluetooth low energy technology message

Field Name	Value Symbol	Value Description	Hex Value
BLEM7	BLEM7_3		0x3*
BLEM6	BLEM6_4		0x4*
BLEM5	BLEM5_8		0x8*
BLEM4	BLEM4_9		0x9*
BLEM3	BLEM3_10		0xA*
BLEM2	BLEM2_13		0xD*
BLEM1	BLEM1_14		0xE*
BLEM0	BLEM0_15		0xF*

9.4.66 BB_BBMPRIO1

Bit Field	Field Name	Description
31:28	BLEMDEFAULT	Set default priority value for Bluetooth low energy technology message other than those defined above
7:4	BLEM9	Set default priority value for ISO Channel first Tx/Rx attempt
3:0	BLEM8	Set default priority value for ISO Channel subsequent Tx/Rx attempt

Field Name	Value Symbol	Value Description	Hex Value
BLEMDEFAULT	BLEMDEFAULT_3		0x3*
BLEM9	BLEM7_13		0xD*
BLEM8	BLEM6_12		0xC*

9.4.67 BB_RALPTR

Bit Field	Field Name	Description
15:0	RALPTR	Start address pointer of the RAL structure

Field Name	Value Symbol	Value Description	Hex Value
RALPTR	RALPTR_0		0x0*

9.4.68 BB_RALNBDEV

Bit Field	Field Name	Description
7:0	RALNBDEV	Number of devices in RAL Structure

Field Name	Value Symbol	Value Description	Hex Value
RALNBDEV	RALNBDEV_0		0x0*

9.4.69 BB_RAL_LOCAL_RND

Bit Field	Field Name	Description
31	LRND_INIT	Writing a 1 initializes of local RPA random number generation LFSR
21:0	LRND_VAL	Initialization value for local RPA random generation when LRDN_INIT is set to 1, else reports the current Local RPA random number LFSR value

Field Name	Value Symbol	Value Description	Hex Value
LRND_INIT	LRND_INIT_0		0x0*
LRND_VAL	LRND_VAL_4132623		0x3F0F0F*

9.4.70 BB_RAL_PEER_RND

Bit Field	Field Name	Description
31	PRND_INIT	Writing a 1 initializes peer RPA random number generation LFSR
21:0	PRND_VAL	Initialization value for peer RPA random generation when LRDN_INIT is set to 1, else reports the current Local RPA random number LFSR value

Field Name	Value Symbol	Value Description	Hex Value
PRND_INIT	PRND_INIT_0		0x0*
PRND_VAL	PRND_VAL_3207408		0x30F0F0*

9.4.71 BB_ISOCHANCNTL0

Bit Field	Field Name	Description
4	RETXACKEN0	Generate Tx ACK
3	SYNCGEN0	Enable audio syn_p generation
2	ISOCHANEN0	Enable ISO channel
1:0	ISOTYPE0	ISO Channel Type

Field Name	Value Symbol	Value Description	Hex Value
RETXACKEN0	RETXACKEN0_0	No Tx ACK generation in re-Tx	0x0*
	RETXACKEN0_1	Tx ACK generation in re-Tx	0x1
SYNCGEN0	SYNCGEN0_0	Disable audio0_syn_p generation	0x0*
	SYNCGEN0_1	Enable audio0_syn_p generation	0x1
ISOCHANEN0	ISOCHANEN0_0	Disable ISO Channel (LLID=0 invalid)	0x0*
	ISOCHANEN0_1	Enable ISO Channel (LLID=0 valid)	0x1
ISOTYPE0	ISOTYPE0_0	Audio Mode 0	0x0*
	ISOTYPE0_1	Reserved	0x1
	ISOTYPE0_2	Reserved	0x2
	ISOTYPE0_3	Reserved	0x3

9.4.72 BB_ISOMUTECNTL0

Bit Field	Field Name	Description
31	TOGO0	Indicates which buffer is in use (direct copy of ET-ISOBUFSEL)
19	MUTE_SINK0	HW mute control
18	MUTE_SOURCE0	HW mute control
17	INVL0_1	SW mute status for ISO buffer 1 (i.e updated when ET-ISOBUFSEL = 0)
16	INVL0_0	SW mute status for ISO buffer 0 (i.e updated when ET-ISOBUFSEL = 1)
7:0	MUTE_PATTERN0	Value of the ISO channel 0 Mute Pattern to be used when HW muting is enabled

Field Name	Value Symbol	Value Description	Hex Value
TOGO0	TOGO0_0	Use buffer 0	0x0*
	TOGO0_1	Use buffer 1	0x1
MUTE_SINK0	MUTE_SINK0_0	Do not mute on bad reception of an ISO packet	0x0*
	MUTE_SINK0_1	Mute after data or bad reception, with the pattern stored in MUTE_PATTERN0	0x1
MUTE_SOURCE0	MUTE_SOURCE0_0	Provides Source buffer to the Packet Controller for Tx operations	0x0*
	MUTE_SOURCE0_1	Forces null length packet to be sent as a replacement of ISO Packets	0x1
INVL0_1	INVL0_1_0	Do not mute current ISO buffer	0x0
	INVL0_1_1	Current ISO buffer is invalid, mute	0x1*
INVL0_0	INVL0_0_0	Do not mute current ISO buffer	0x0
	INVL0_0_1	Current ISO buffer is invalid, mute	0x1*
MUTE_PATTERN0	MUTE_PATTERN0_0	Value of the ISO channel 0 Mute Pattern to be used when HW muting is enabled	0x0*

9.4.73 BB_ISOCURRENTTXPTR0

Bit Field	Field Name	Description
31:16	ISO0TXPTR0	Tx ISO Buffer pointer 0 of ISO Channel 0
15:0	ISO0TXPTR1	Tx ISO Buffer pointer 1 of ISO Channel 0

Field Name	Value Symbol	Value Description	Hex Value
ISO0TXPTR0	ISO0TXPTR0_0	Tx ISO Buffer pointer 0 of ISO Channel 0	0x0*
ISO0TXPTR1	ISO0TXPTR1_0	Tx ISO Buffer pointer 1 of ISO Channel 0	0x0*

9.4.74 BB_ISOCURRENTRXPTR0

Bit Field	Field Name	Description
31:16	ISO0RXPTR0	Rx ISO Buffer pointer 0 of ISO Channel 0
15:0	ISO0RXPTR1	Rx ISO Buffer pointer 1 of ISO Channel 0

Field Name	Value Symbol	Value Description	Hex Value
ISO0RXPTR0	ISO0RXPTR0_0	Rx ISO Buffer pointer 0 of ISO Channel 0	0x0*
ISO0RXPTR1	ISO0RXPTR1_0	TR ISO Buffer pointer 1 of ISO Channel 0	0x0*

9.4.75 BB_ISOTRCNL0

Bit Field	Field Name	Description
23:16	ISO0RXLEN	Negotiated, maximum expected number of bytes for ISO Channel 0 Rx payloads
7:0	ISO0TXLEN	Negotiated, number of bytes for ISO Channel 0 Tx payloads

Field Name	Value Symbol	Value Description	Hex Value
ISO0RXLEN	ISO0RXLEN_0		0x0*
ISO0TXLEN	ISO0TXLEN_0		0x0*

9.4.76 BB_ISOEVTCNTLOFFSETL0

Bit Field	Field Name	Description
31:0	EVT_CNT_OFFSETL0	LSB part of EVT_CNT_OFFSET0[39:0] field

Field Name	Value Symbol	Value Description	Hex Value
EVT_CNT_OFFSETL0	EVT_CNT_OFFSETL0_0		0x0*

9.4.77 BB_ISOEVTCNTLOFFSETU0

Bit Field	Field Name	Description
6:0	EVT_CNT_OFFSETU0	MSB part of EVT_CNT_OFFSET0[39:0] field

Field Name	Value Symbol	Value Description	Hex Value
EVT_CNT_OFFSETU0	EVT_CNT_OFFSETU0_0		0x0*

9.4.78 BB_ISOCHANCNTL1

Bit Field	Field Name	Description
4	RETXACKEN1	Generate Tx ACK
3	SYNCGEN1	Enable audio syn_p generation
2	ISOCHANEN1	Enable ISO channel
1:0	ISOTYPE1	ISO Channel Type

Field Name	Value Symbol	Value Description	Hex Value
RETXACKEN1	RETXACKEN1_0	No Tx ACK generation in re-Tx	0x0*
	RETXACKEN1_1	Tx ACK generation in re-Tx	0x1
SYNCGEN1	SYNCGEN1_0	Disable audio0_syn_p generation	0x0*
	SYNCGEN1_1	Enable audio0_syn_p generation	0x1
ISOCHANEN1	ISOCHANEN1_0	Disable ISO Channel (LLID=0 invalid)	0x0*
	ISOCHANEN1_1	Enable ISO Channel (LLID=0 valid)	0x1
ISOTYPE1	ISOTYPE1_0	Audio Mode 0	0x0*
	ISOTYPE1_1	Reserved	0x1
	ISOTYPE1_2	Reserved	0x2
	ISOTYPE1_3	Reserved	0x3

9.4.79 BB_ISOMUTECNTL1

Bit Field	Field Name	Description
31	TOGO1	Indicates which buffer is in use (direct copy of ET-ISOBUFSEL)
19	MUTE_SINK1	HW mute control
18	MUTE_SOURCE1	HW mute control
17	INVL1_1	SW mute status for ISO buffer 1 (i.e updated when ET-ISOBUFSEL = 0)
16	INVL1_0	SW mute status for ISO buffer 0 (i.e updated when ET-ISOBUFSEL = 1)
7:0	MUTE_PATTERN1	Value of the ISO channel 0 Mute Pattern to be used when HW muting is enabled

Field Name	Value Symbol	Value Description	Hex Value
TOGO1	TOGO1_0	Use buffer 0	0x0*
	TOGO1_1	Use buffer 1	0x1
MUTE_SINK1	MUTE_SINK1_0	Do not mute on bad reception of an ISO packet	0x0*
	MUTE_SINK1_1	Mute after data or bad reception, with the pattern stored in MUTE_PATTERN0	0x1
MUTE_SOURCE1	MUTE_SOURCE1_0	Provides Source buffer to the Packet Controller for Tx operations	0x0*
	MUTE_SOURCE1_1	Forces null length packet to be sent as a replacement for ISO Packets	0x1

Field Name	Value Symbol	Value Description	Hex Value
INVL1_1	INVL1_1_0	Do not mute current ISO buffer	0x0
	INVL1_1_1	Current ISO buffer is invalid, mute	0x1*
INVL1_0	INVL1_0_0	Do not mute current ISO buffer	0x0
	INVL1_0_1	Current ISO buffer is invalid, mute	0x1*
MUTE_PATTERN1	MUTE_PATTERN1_0	Value of the ISO channel 0 Mute Pattern to be used when HW muting is enabled	0x0*

9.4.80 BB_ISOCURRENTTXPTR1

Bit Field	Field Name	Description
31:16	ISO1TXPTR0	Tx ISO Buffer pointer 0 of ISO Channel 1
15:0	ISO1TXPTR1	Tx ISO Buffer pointer 1 of ISO Channel 1

Field Name	Value Symbol	Value Description	Hex Value
ISO1TXPTR0	ISO1TXPTR0_0	Tx ISO Buffer pointer 0 of ISO Channel 1	0x0*
ISO1TXPTR1	ISO1TXPTR1_0	Tx ISO Buffer pointer 1 of ISO Channel 1	0x0*

9.4.81 BB_ISOCURRENTRXPTR1

Bit Field	Field Name	Description
31:16	ISO1RXPTR0	Rx ISO Buffer pointer 0 of ISO Channel 1
15:0	ISO1RXPTR1	Rx ISO Buffer pointer 1 of ISO Channel 1

Field Name	Value Symbol	Value Description	Hex Value
ISO1RXPTR0	ISO1RXPTR0_0	Rx ISO Buffer pointer 0 of ISO Channel 1	0x0*
ISO1RXPTR1	ISO1RXPTR1_0	TR ISO Buffer pointer 1 of ISO Channel 1	0x0*

9.4.82 BB_ISOTRCNL1

Bit Field	Field Name	Description
23:16	ISO1RXLEN	Negotiated, maximum expected number of bytes for ISO Channel 0 Rx payloads
7:0	ISO1TXLEN	Negotiated, number of bytes for ISO Channel 0 Tx payloads

Field Name	Value Symbol	Value Description	Hex Value
ISO1RXLEN	ISO1RXLEN_0		0x0*
ISO1TXLEN	ISO1TXLEN_0		0x0*

9.4.83 BB_ISOEVTCNTLOFFSETL1

Bit Field	Field Name	Description
31:0	EVT_CNT_OFFSETL1	LSB part of EVT_CNT_OFFSET0[39:0] field

Field Name	Value Symbol	Value Description	Hex Value
EVT_CNT_OFFSETL1	EVT_CNT_OFFSETL1_0		0x0*

9.4.84 BB_ISOEVTCNTLOFFSETU1

Bit Field	Field Name	Description
6:0	EVT_CNT_OFFSETU1	MSB part of EVT_CNT_OFFSET0[39:0] field

Field Name	Value Symbol	Value Description	Hex Value
EVT_CNT_OFFSETU1	EVT_CNT_OFFSETU1_0		0x0*

9.4.85 BB_ISOCHANCNTL2

Bit Field	Field Name	Description
4	RETXACKEN2	Generate Tx ACK
3	SYNCGEN2	Enable audio syn_p generation
2	ISOCHANEN2	Enable ISO channel
1:0	ISOTYPE2	ISO Channel Type

Field Name	Value Symbol	Value Description	Hex Value
RETXACKEN2	RETXACKEN2_0	No Tx ACK generation in re-Tx	0x0*
	RETXACKEN2_1	Tx ACK generation in re-Tx	0x1
SYNCGEN2	SYNCGEN2_0	Disable audio2_syn_p generation	0x0*
	SYNCGEN2_1	Enable audio2_syn_p generation	0x1
ISOCHANEN2	ISOCHANEN2_0	Disable ISO Channel (LLID=0 invalid)	0x0*
	ISOCHANEN2_1	Enable ISO Channel (LLID=0 valid)	0x1
ISOTYPE2	ISOTYPE2_0	Audio Mode 0	0x0*
	ISOTYPE2_1	Reserved	0x1
	ISOTYPE2_2	Reserved	0x2
	ISOTYPE2_3	Reserved	0x3

9.4.86 BB_ISOMUTECNTL2

Bit Field	Field Name	Description
31	TOGO2	Indicates which buffer is in use (direct copy of ET-ISOBUFSEL)
19	MUTE_SINK2	HW mute control
18	MUTE_SOURCE2	HW mute control
17	INVL2_1	SW mute status for ISO buffer 1 (i.e updated when ET-ISOBUFSEL = 0)
16	INVL2_0	SW mute status for ISO buffer 0 (i.e updated when ET-ISOBUFSEL = 1)
7:0	MUTE_PATTERN2	Value of the ISO channel 0 Mute Pattern to be used when HW muting is enabled

Field Name	Value Symbol	Value Description	Hex Value
TOGO2	TOGO2_0	Use buffer 0	0x0*
	TOGO2_1	Use buffer 1	0x1
MUTE_SINK2	MUTE_SINK2_0	Do not mute on bad reception of an ISO packet	0x0*
	MUTE_SINK2_1	Mute after data or bad reception, with the pattern stored in MUTE_PATTERN0	0x1
MUTE_SOURCE2	MUTE_SOURCE2_0	Provides Source buffer to the Packet Controller for Tx operations	0x0*
	MUTE_SOURCE2_1	Forces null length packet to be sent as a replacement for ISO Packets	0x1
INVL2_1	INVL2_1_0	Do not mute current ISO buffer	0x0
	INVL2_1_1	Current ISO buffer is invalid, mute	0x1*
INVL2_0	INVL2_0_0	Do not mute current ISO buffer	0x0
	INVL2_0_1	Current ISO buffer is invalid, mute	0x1*
MUTE_PATTERN2	MUTE_PATTERN2_0	Value of the ISO channel 0 Mute Pattern to be used when HW muting is enabled	0x0*

9.4.87 BB_ISOCURRENTTXPTR2

Bit Field	Field Name	Description
31:16	ISO2TXPTR0	Tx ISO Buffer pointer 0 of ISO Channel 2
15:0	ISO2TXPTR1	Tx ISO Buffer pointer 1 of ISO Channel 2

Field Name	Value Symbol	Value Description	Hex Value
ISO2TXPTR0	ISO2TXPTR0_0	Tx ISO Buffer pointer 0 of ISO Channel 2	0x0*
ISO2TXPTR1	ISO2TXPTR1_0	Tx ISO Buffer pointer 1 of ISO Channel 2	0x0*

9.4.88 BB_ISOCURRENTRXPTR2

Bit Field	Field Name	Description
31:16	ISO2RXPTR0	Rx ISO Buffer pointer 0 of ISO Channel 2
15:0	ISO2RXPTR1	Rx ISO Buffer pointer 1 of ISO Channel 2

Field Name	Value Symbol	Value Description	Hex Value
ISO2RXPTR0	ISO2RXPTR0_0	Rx ISO Buffer pointer 0 of ISO Channel 2	0x0*
ISO2RXPTR1	ISO2RXPTR1_0	TR ISO Buffer pointer 1 of ISO Channel 2	0x0*

9.4.89 BB_ISOTRCNL2

Bit Field	Field Name	Description
23:16	ISO2RXLEN	Negotiated, maximum expected number of bytes for ISO Channel 2 Rx payloads
7:0	ISO2TXLEN	Negotiated, number of bytes for ISO Channel 2 Tx payloads

Field Name	Value Symbol	Value Description	Hex Value
ISO2RXLEN	ISO2RXLEN_0		0x0*
ISO2TXLEN	ISO2TXLEN_0		0x0*

9.4.90 BB_ISOEVTCNTLOFFSETL2

Bit Field	Field Name	Description
31:0	EVT_CNT_OFFSETL2	LSB part of EVT_CNT_OFFSET2[39:0] field

Field Name	Value Symbol	Value Description	Hex Value
EVT_CNT_OFFSETL2	EVT_CNT_OFFSETL2_0		0x0*

9.4.91 BB_ISOEVTCNTLOFFSETU2

Bit Field	Field Name	Description
6:0	EVT_CNT_OFFSETU2	MSB part of EVT_CNT_OFFSET2[39:0] field

Field Name	Value Symbol	Value Description	Hex Value
EVT_CNT_OFFSETU2	EVT_CNT_OFFSETU2_0		0x0*

9.4.92 BB_BBPRIOSCHARB

Bit Field	Field Name	Description
15	BLEPRIOMODE	Determine Bluetooth low energy technology priority scheduling arbitration mode
7:0	BLEMARGIN	Determine the decision instant margin for priority scheduling arbitration

Field Name	Value Symbol	Value Description	Hex Value
BLEPRIOMODE	BLEPRIOMODE_0	Bluetooth low energy technology decision instant not used	0x0*
	BLEPRIOMODE_1	Bluetooth low energy technology decision instant used	0x1
BLEMARGIN	BLEMARGIN_0		0x0*

CHAPTER 10

Digital Input/Output

10.1 Overview

The RSL10 system contains 16 digital input/output (DIO) pads that can be configured:

- To support the external interfaces, output clocks, and other I/Os
- As general-purpose I/Os controllable from the core

DIOs support all digital inputs and output functions that are not supported directly by a dedicated I/O. For more information about the functional configuration of DIO pads, see Section 10.2, "Functional Configuration" below.

Dedicated I/Os are supplied for the following pads which are not part of the DIO pad set:

- The wake up pad (WAKEUP)—for more information, see Section 5.4, "Power Modes" on page 50.
- The external clock input pad (EXTCLK)—for more information, see Section 6.2.5, "External Clock Input (EXTCLK)" on page 75.
- The JTCK and JTMS (also used as SWCLK and SWDIO) pads for the standard SWJ-DP debug port included with the Arm Cortex-M3 core — for more information, see Section 3.2, "Debug Port" on page 27.
- A reset pad (NRESET) - for more information, see Section 5.5, "Resets" on page 63.

The DIO pads support a variety of physical configuration parameters that might be required to properly interface with external components, including:

- Pull-up and pull-down resistors
- Low-pass input filtering
- Output drive strength

A complete description of the physical configuration of DIOs can be found in Section 10.3, "Physical Configuration" on page 263.

All of the DIO pads are powered from the VDDO power supply. For more information about this power supply and its configuration, see Section 5.2.2, "Digital Output Supply Voltage (VDDO)" on page 39.

10.2 Functional Configuration

The DIO pads can be configured using the `DIO_CFG_IO_MODE` bit field from the `DIO_CFG_*` registers:

- For a variety of digital output modes
- For a general-purpose digital input mode, with the input function configured by the `DIO_`**`interface`**`_SRC` registers, where **`interface`** is one of PCM, SPI, UART, I2C, audio sink, NMI, baseband Rx, baseband SPI, RF front-end SPI, RF front-end GPIO, DMIC, and LPDSP32 JTAG. For the SPI interfaces, these registers have a numeric postfix to indicate which of the interfaces is being configured.

Table 19 contains a list of the functional modes a DIO can be configured for. Table 20 contains a list of the input sources that a given DIO can be assigned to supply.

NOTE: ADC, wakeup and STANDBYCLK functions are only available on DIO[0:3]. See Section 11.2, "Analog-to-Digital Converters (ADCs)" on page 301, Section 5.4, "Power Modes" on page 50, and Section 6.3.2, "Standby Clock (STANDBYCLK)" on page 79, respectively, for information on configuring these functions.

In addition to standard digital functional configuration, certain DIOs can be configured for special modes. These special modes are described in Section 10.2.1, "Special Functional Configurations" on page 262.

CAUTION: While a DIO can be configured to be both an output and an input, it is the user application's responsibility to ensure that the DIO is not driving an output to a pad that is also being driven externally. If a DIO pad signal has two drivers, the physical values and inputs that are read from this pad are considered undefined.

Table 19. DIO Multiplexed Functionality

Mode	Setting	Description
0x00 0x01	`DIO_MODE_GPIO_IN_0` `DIO_MODE_GPIO_IN_1`	GPIO input mode; bit 0 is the input data value (both `DIO_MODE_GPIO_IN_0` and `DIO_MODE_GPIO_IN_1` configure the DIO for GPIO input mode)
0x02 0x03	`DIO_MODE_GPIO_OUT_0` `DIO_MODE_GPIO_OUT_1`	GPIO output mode; bit 0 is the output data value (`GPIO_OUT_0` provides an output low and `GPIO_OUT_1` provides an output high)
0x04	`DIO_MODE_USRCLK`	User clock (UCLK) output
0x05	`DIO_MODE_SLOWCLK`	Slow clock (SLOWCLK) output
0x06	`DIO_MODE_SYSCLK`	System clock (SYSCLK) output
0x07	`DIO_MODE_PCM_SERO`	PCM interface serial output signal
0x08	`DIO_MODE_PCM_FRAME`	PCM interface frame output (PCM master mode only)
0x09	`DIO_MODE_SPI0_SERO`	SPI0 serial output signal
0x0A	`DIO_MODE_SPI0_CS`	SPI0 chip select output (SPI master mode only)
0x0B	`DIO_MODE_SPI0_CLK`	SPI0 clock output (SPI master mode only)
0x0C	`DIO_MODE_SPI1_SERO`	SPI1 serial output signal
0x0D	`DIO_MODE_SPI1_CS`	SPI1 chip select output (SPI master mode only)
0x0E	`DIO_MODE_SPI1_CLK`	SPI1 clock output (SPI master mode only)
0x0F	`DIO_MODE_UART_TX`	UART transmit output
0x10	`DIO_MODE_SCL`	I^2C clock output (open collector)
0x11	`DIO_MODE_SDA`	I^2C data output (open collector)
0x12	`DIO_MODE_PWM0`	PWM interface output 0
0x13	`DIO_MODE_PWM0_INV`	Inverted PWM interface output 0
0x14	`DIO_MODE_PWM1`	PWM interface output 1
0x15	`DIO_MODE_PWM1_INV`	Inverted PWM interface output 1
0x16	`DIO_MODE_LPDSP32_TDO`	LPDSP32 JTAG test data out
0x17	`DIO_MODE_RFCLK`	RF clock output
0x18	`DIO_MODE_RCCLK`	RC clock output
0x19	`DIO_MODE_JTCK_DIV`	Divided JTAG clock output
0x1A	`DIO_MODE_EXTCLK_DIV`	Divided EXTCLK output
0x1B	`DIO_MODE_STANDBYCLK`	Standby clock (STANDBYCLK) output
0x1C	`DIO_MODE_BB_TX_DATA`	Baseband transmit data output
0x1D	`DIO_MODE_BB_TX_DATA_VALID`	Baseband transmit data valid indicator output
0x1E	`DIO_MODE_BB_SYNC_P`	Baseband synchronization signal

Table 19. DIO Multiplexed Functionality (Continued)

Mode	Setting	Description
0x1F	DIO_MODE_BB_AUDIO0_SYNC_P	Output baseband controller BLE AUDIO0 synchronization signal
0x20	DIO_MODE_BB_AUDIO1_SYNC_P	Output baseband controller BLE AUDIO1 synchronization signal
0x21	DIO_MODE_BB_AUDIO2_SYNC_P	Output baseband controller BLE AUDIO2 synchronization signal
0x22	DIO_MODE_BB_SPI_CSN	Output baseband controller SPI_CSN signal
0x23	DIO_MODE_BB_SPI_CLK	Output baseband controller SPI_CLK signal
0x24	DIO_MODE_BB_SPI_MOSI	Output baseband controller SPI_MOSI signal
0x25	DIO_MODE_BB_DBG0_0	Output baseband controller diagnostic port 0 (bit 0) signal
0x26	DIO_MODE_BB_DBG0_1	Output baseband controller diagnostic port 0 (bit 1) signal
0x27	DIO_MODE_BB_DBG0_2	Output baseband controller diagnostic port 0 (bit 2) signal
0x28	DIO_MODE_BB_DBG0_3	Output baseband controller diagnostic port 0 (bit 3) signal
0x29	DIO_MODE_BB_DBG0_4	Output baseband controller diagnostic port 0 (bit 4) signal
0x2A	DIO_MODE_BB_DBG0_5	Output baseband controller diagnostic port 0 (bit 5) signal
0x2B	DIO_MODE_BB_DBG0_6	Output baseband controller diagnostic port 0 (bit 6) signal
0x2C	DIO_MODE_BB_DBG0_7	Output baseband controller diagnostic port 0 (bit 7) signal
0x2D	DIO_MODE_RF_SPI_MISO	Output RF front-end SPI_MISO interface signal
0x2E	DIO_MODE_RF_GPIO0	Output RF front-end GPIO0 output (RX_DATA) signal
0x2F	DIO_MODE_RF_GPIO1	Output RF front-end GPIO1 output (RX_CLK) signal
0x30	DIO_MODE_RF_GPIO2	Output RF front-end GPIO2 output signal
0x31	DIO_MODE_RF_GPIO3	Output RF front-end GPIO3 output signal
0x32	DIO_MODE_RF_GPIO4	Output RF front-end GPIO4 output signal
0x33	DIO_MODE_RF_GPIO5	Output RF front-end GPIO5 output signal
0x34	DIO_MODE_RF_GPIO6	Output RF front-end GPIO6 output signal
0x35	DIO_MODE_RF_GPIO7	Output RF front-end GPIO7 output signal
0x36	DIO_MODE_RF_GPIO8	Output RF front-end GPIO8 output signal
0x37	DIO_MODE_RF_GPIO9	Output RF front-end GPIO9 output signal
0x38	DIO_MODE_AUDIOCLK	Output the AUDIOCLK (audio clock) signal
0x39	DIO_MODE_AUDIOSLOWCLK	Output the AUDIOSLOWCLK (slow audio clock) signal
0x3A	DIO_MODE_OD_P	Output OD + signal
0x3B	DIO_MODE_OD_N	Output OD - signal
0x3C	DIO_MODE_AUDIO_SYNC_PULSE	Output audio synchronization pulse
0x3D	DIO_MODE_AUDIO_SYNC_MISSED	Output audio synchronization missed pulse
0x3E	DIO_MODE_INPUT	Input mode
0x3F	DIO_MODE_DISABLE	Disabled

Table 20. DIO Input Functionality

Configuration Register	Signal	Description
DIO_PCM_SRC	PCM_SERI	PCM interface serial input signal
	PCM_FRAME	PCM interface frame input (PCM slave mode only)
	PCM_CLK	PCM interface clock input
DIO_SPI_SRC	SPI_SERI	Serial input signal for the specified SPI interface
	SPI_CS	Chip select input (SPI slave mode only) for the specified SPI interface
	SPI_CLK	Clock input (SPI slave mode only) for the specified SPI interface
DIO_UART_SRC	UART_RX	Receive signal for the specified UART interface
DIO_I2C_SRC	I2C_SCL	I^2C clock input (open collector)
	I2C_SDA	I^2C data input (open collector)
DIO_AUDIOSINK_SRC	AUDIOSINK_CLK	Audio sink clock input selection
DIO_NMI_SRC	NMI	Non-maskable interrupt input
DIO_BB_RX_SRC	BB_RF_SYNC_P	Baseband controller interface RF_SYNC_P input selection
	BB_RX_CLK	Baseband controller Rx clock input selection
	BB_RX_DATA	Baseband controller Rx data input selection
DIO_BB_SPI_SRC	BB_SPI_MISO	Baseband controller SPI_MISO input selection
DIO_RF_SPI_SRC	RF_SPI_MOSI	RF front-end SPI_MOSI input selection
	RF_SPI_CSN	RF front-end SPI_CSN input selection
	RF_SPI_CLK	RF front-end SPI_CLK input selection
DIO_RF_GPIO03_SRC	RF_GPIO3	RF front-end GPIO3 input selection
	RF_GPIO2	RF front-end GPIO2 input selection
	RF_GPIO1	RF front-end GPIO1 input selection
	RF_GPIO0	RF front-end GPIO0 input selection
DIO_RF_GPIO47_SRC	RF_GPIO7	RF front-end GPIO7 input selection
	RF_GPIO6	RF front-end GPIO6 input selection
	RF_GPIO5	RF front-end GPIO5 input selection
	RF_GPIO4	RE front-end GPIO4 input selection
DIO_RF_GPIO89_SRC	RF_GPIO9	RF front-end GPIO9 input selection
	RF_GPIO8	RF front-end GPIO8 input selection
DIO_DMIC_SRC	DMIC_CLK	DMIC clock input selection
	DMIC_DATA	DMIC data input selection
DIO_LPDSP32_JTAG_SRC	LPDSP32_JTAG_TDI	LPDSP32 JTAG test data input selection
	LPDSP32_JTAG_TMS	LPDSP32 JTAG Test Mode select input selection
	LPDSP32_JTAG_TCK	LPDSP32 JTAG test clock selection

10.2.1 Special Functional Configurations

RSL10 contains two special functional configurations that affect a subset of the DIOs. These special functional configurations are described in Table 21.

Table 21. Special DIO Functional Configurations

Mode	Affected DIOs	Description
Analog Input	0, 1, 2, and/or 3	Four of the DIOs support a special configuration as analog inputs. These DIOs can be configured independently from one another using the `DIO_CFG_IO_MODE` bit field from their respective `DIO_CFG_*` registers. NOTE: For a DIO that is used in ADC mode, the pad must be disconnected from both the digital input sensor and digital output driver. For DIOs 0 to 3, the pad might be connected to the analog input. For all other DIOs, the pad is completely disconnected. For more information about DIO configuration when using ADC, see Section 11.2.1, "ADC Input Configuration" on page 302. For more information about analog inputs, see Section 11.2, "Analog-to-Digital Converters (ADCs)" on page 301.
STANDBYCLK		The STANDBYCLK function is only available on DIO[3:0]. For more information on STANDBYCLK configuration, see Section 6.3.2, "Standby Clock (STANDBYCLK)" on page 79.
Wakeup		The wakeup function is only available on DIO[3:0]. For more information on wakeup configuration, see Section 5.4, "Power Modes" on page 50.
Arm Cortex-M3 Processor SWJ-DP JTAG	13 14 and 15	When configured for JTAG mode, the SWJ-DP debug port interface uses the following pads to implement a 4 or 5-wire JTAG interface: • DIO 13 configured as JNTRST (5-wire JTAG interface only) • DIO 14 configured as JTDI • DIO 15 configured as JTDO The `DIO_CFG_IO_MODE` bit field is overridden for DIO 13, when `CM3_JTAG_TRST_ENABLED` is set using the `CM3_JTAG_TRST_EN` bit from the `DIO_JTAG_SW_PAD_CFG` register. The `DIO_CFG_IO_MODE` bit field is overridden for DIOs 14 and 15, when `CM3_JTAG_DATA_ENABLED` is set using the `CM3_JTAG_DATA_EN` bit from the `DIO_JTAG_SW_PAD_CFG` register.

10.3 Physical Configuration

The RSL10 system includes physical configuration parameters for each DIO. These parameters are set using configuration bits from the appropriate `DIO_CFG_*` register.

If the DIO is configured as an input pad, it has the following configuration options:

- The `DIO_CFG_PULL_CTRL` bit field is used to configure the pad to use a pull-up or pull-down resistor. Options include:
  - No pull resistor
  - A weak (250 kΩ) pull-up resistor
  - A weak (250 kΩ) pull-down resistor
  - A strong (1 kΩ) pull-up resistor

- In the reset state while the NRESET pad is driven low, the DIO output drive is disabled and a weak (250 kΩ) pull-down resistor is enabled. After reset is released, the default DIO configuration is applied.

- The `DIO_CFG_LPF` bit enables or disables a low-pass filter that can be used to clean up the DIO's received input signal.

IMPORTANT: For optimal noise performance, we recommend enabling the low-pass filters provided for DIO inputs for input signals received at 1 MHz or less. For signals that use a frequency that exceeds 1 MHz, the DIO low-pass filters should be disabled.

In addition to the configurable physical parameters, all DIO pads contain Schmitt triggers to filter out noise observed at the inputs.

If the DIO is configured as an output pad, it has the following configuration option:

- The `DIO_CFG_DRIVE` bit allows you to select the drive strength for the DIO output.

NOTE: The `DIO_PAD_CFG_DRIVE` bit-field from the `DIO_PAD_CFG` register can be used to increase the drive strength of all outputs by 50% or more if needed for a user application.

10.4 DIO Registers

Register Name	Register Description	Address
`DIO_CFG`	Digital IOs Configuration (ADC function only for pads 0 to 3)	0x40000700
`DIO_DATA`	Digital IOs Data Access Register	0x40000740
`DIO_DIR`	Digital IOs Direction State	0x40000744
`DIO_MODE`	Digital IOs Mode State	0x40000748
`DIO_INT_CFG`	DIO Interrupt Configuration	0x4000074C
`DIO_INT_DEBOUNCE`	DIO Interrupt Button Debounce Filter Time Configuration	0x4000075C
`DIO_PCM_SRC`	PCM Input Selection	0x40000760
`DIO_SPI_SRC`	SPI[1:0] Interface Input Selection	0x40000764
`DIO_UART_SRC`	UART Interface Input Selection	0x4000076C
`DIO_I2C_SRC`	I2C Input Selection	0x40000770
`DIO_AUDIOSINK_SRC`	Audio Sink Input Selection	0x40000774
`DIO_NMI_SRC`	NMI Input Selection	0x40000778
`DIO_BB_RX_SRC`	Baseband controller Rx data and clock input selection	0x4000077C
`DIO_BB_SPI_SRC`	Baseband controller SPI input selection	0x40000780
`DIO_RF_SPI_SRC`	RF front-end SPI input selection	0x40000784
`DIO_RF_GPIO03_SRC`	RF front-end GPIOs 0-3 input selection	0x40000788
`DIO_RF_GPIO47_SRC`	RF front-end GPIOs 4-7 input selection	0x4000078C
`DIO_RF_GPIO89_SRC`	RF front-end GPIOs 8-9 input selection	0x40000790
`DIO_DMIC_SRC`	DMIC data input selection	0x40000794
`DIO_LPDSP32_JTAG_SRC`	LPDSP32 JTAG Configuration Register	0x40000798
`DIO_JTAG_SW_PAD_CFG`	JTAG / SW Pad Configuration Register	0x4000079C
`DIO_EXTCLK_CFG`	External Clock Pad Configuration Register	0x400007A0
`DIO_PAD_CFG`	Global Pads Configuration Register	0x400007A4

10.4.1 DIO_CFG

Bit Field	Field Name	Description
13:12	DRIVE	Drive strength configuration
10	LPF	Low Pass Filter enable
9:8	PULL_CTRL	Pull selection
5:0	IO_MODE	IO mode selection

Field Name	Value Symbol	Value Description	Hex Value
DRIVE	DIO_2X_DRIVE	2x drive strength	0x0
	DIO_3X_DRIVE	3x drive strength	0x1
	DIO_5X_DRIVE	5x drive strength	0x2
	DIO_6X_DRIVE	6x drive strength	0x3*
LPF	DIO_LPF_DISABLE	Disable low pass filter	0x0*
	DIO_LPF_ENABLE	Enable low pass filter	0x1
PULL_CTRL	DIO_NO_PULL	No pull selected	0x0
	DIO_WEAK_PULL_UP	Weak pull-up selected	0x1*
	DIO_WEAK_PULL_DOWN	Weak pull-down selected	0x2
	DIO_STRONG_PULL_UP	Strong pull-up selected	0x3

Field Name	Value Symbol	Value Description	Hex Value
IO_MODE	DIO_MODE_GPIO_IN_0	Input (GPIO input mode)	0x0
	DIO_MODE_GPIO_IN_1	Input (GPIO input mode)	0x1
	DIO_MODE_GPIO_OUT_0	Output low (GPIO output mode)	0x2
	DIO_MODE_GPIO_OUT_1	Output high (GPIO output mode)	0x3
	DIO_MODE_USRCLK	Output USRCLK (user clock) signal	0x4
	DIO_MODE_SLOWCLK	Output SLOWCLK (slow clock) signal	0x5
	DIO_MODE_SYSCLK	Output SYSCLK (system clock) signal	0x6
	DIO_MODE_PCM_SERO	Output PCM_SERO interface signal	0x7
	DIO_MODE_PCM_FRAME	Output PCM_FRAME interface signal	0x8
	DIO_MODE_SPI0_SERO	Output SPI0_SERO interface signal	0x9
	DIO_MODE_SPI0_CS	Output SPI0_CS interface signal	0xA
	DIO_MODE_SPI0_CLK	Output SPI0_CLK interface signal	0xB
	DIO_MODE_SPI1_SERO	Output SPI1_SERO interface signal	0xC
	DIO_MODE_SPI1_CS	Output SPI1_CS interface signal	0xD
	DIO_MODE_SPI1_CLK	Output SPI1_CLK interface signal	0xE
	DIO_MODE_UART_TX	Output UART_TX interface signal	0xF
	DIO_MODE_SCL	Output SCL interface signal (open collector)	0x10
	DIO_MODE_SDA	Output SDA interface signal (open collector)	0x11
	DIO_MODE_PWM0	Output PWM0 interface signal	0x12
	DIO_MODE_PWM0_INV	Output PWM0 interface signal inverted	0x13
	DIO_MODE_PWM1	Output PWM1 interface signal	0x14
	DIO_MODE_PWM1_INV	Output PWM1 interface signal inverted	0x15
	DIO_MODE_LPDSP32_TDO	Output LPDSP32-TDO interface signal	0x16
	DIO_MODE_RFCLK	Output RFCLK signal	0x17
	DIO_MODE_RCCLK	Output RCCLK signal	0x18
	DIO_MODE_JTCK_DIV	Output of JTCK divider signal	0x19
	DIO_MODE_EXTCLK_DIV	Output of EXTCLK divider signal	0x1A

Field Name	Value Symbol	Value Description	Hex Value
IO_MODE (continued)	DIO_MODE_STANDBYCLK	Output STANDBYCLK signal	0x1B
	DIO_MODE_BB_TX_DATA	Output baseband controller TX data signal	0x1C
	DIO_MODE_BB_TX_DATA_VALID	Output baseband controller TX data valid signal	0x1D
	DIO_MODE_BB_SYNC_P	Output baseband controller BLE synchronization signal	0x1E
	DIO_MODE_BB_AUDIO0_SYNC_P	Output baseband controller BLE AUDIO0 synchronization signal	0x1F
	DIO_MODE_BB_AUDIO1_SYNC_P	Output baseband controller BLE AUDIO1 synchronization signal	0x20
	DIO_MODE_BB_AUDIO2_SYNC_P	Output baseband controller BLE AUDIO2 synchronization signal	0x21
	DIO_MODE_BB_SPI_CSN	Output baseband controller SPI_CSN signal	0x22
	DIO_MODE_BB_SPI_CLK	Output baseband controller SPI_CLK signal	0x23
	DIO_MODE_BB_SPI_MOSI	Output baseband controller SPI_MOSI signal	0x24
	DIO_MODE_BB_DBG0_0	Output baseband controller diagnostic port 0 (bit 0) signal	0x25
	DIO_MODE_BB_DBG0_1	Output baseband controller diagnostic port 0 (bit 1) signal	0x26
	DIO_MODE_BB_DBG0_2	Output baseband controller diagnostic port 0 (bit 2) signal	0x27
	DIO_MODE_BB_DBG0_3	Output baseband controller diagnostic port 0 (bit 3) signal	0x28
	DIO_MODE_BB_DBG0_4	Output baseband controller diagnostic port 0 (bit 4) signal	0x29
	DIO_MODE_BB_DBG0_5	Output baseband controller diagnostic port 0 (bit 5) signal	0x2A
	DIO_MODE_BB_DBG0_6	Output baseband controller diagnostic port 0 (bit 6) signal	0x2B
	DIO_MODE_BB_DBG0_7	Output baseband controller diagnostic port 0 (bit 7) signal	0x2C
	DIO_MODE_RF_SPI_MISO	Output RF front-end SPI_MISO interface signal	0x2D
	DIO_MODE_RF_GPIO0	Output RF front-end GPIO0 output (RX_DATA) signal	0x2E
	DIO_MODE_RF_GPIO1	Output RF front-end GPIO1 output (RX_CLK) signal	0x2F
	DIO_MODE_RF_GPIO2	Output RF front-end GPIO2 output signal	0x30
	DIO_MODE_RF_GPIO3	Output RF front-end GPIO3 output signal	0x31
	DIO_MODE_RF_GPIO4	Output RF front-end GPIO4 output signal	0x32

Field Name	Value Symbol	Value Description	Hex Value
IO_MODE (continued)	DIO_MODE_RF_GPIO5	Output RF front-end GPIO5 output signal	0x33
	DIO_MODE_RF_GPIO6	Output RF front-end GPIO6 output signal	0x34
	DIO_MODE_RF_GPIO7	Output RF front-end GPIO7 output signal	0x35
	DIO_MODE_RF_GPIO8	Output RF front-end GPIO8 output signal	0x36
	DIO_MODE_RF_GPIO9	Output RF front-end GPIO9 output signal	0x37
	DIO_MODE_AUDIOCLK	Output the AUDIOCLK (audio clock) signal	0x38
	DIO_MODE_AUDIOSLOWCLK	Output the AUDIOSLOWCLK (slow audio clock) signal	0x39
	DIO_MODE_OD_P	Output OD + signal	0x3A
	DIO_MODE_OD_N	Output OD - signal	0x3B
	DIO_MODE_AUDIO_SYNC_PULSE	Output audio synchronization pulse	0x3C
	DIO_MODE_AUDIO_SYNC_MISSED	Output audio synchronization missed pulse	0x3D
	DIO_MODE_INPUT	Input mode	0x3E
	DIO_MODE_DISABLE	Disabled	0x3F*

10.4.2 DIO_DATA

Bit Field	Field Name	Description
15:0	DIO	DIO[15:0] read data
15:0	GPIO	GPIO[15:0] write data (updates output data of DIOs only for pads with IO_MODE 0b0000XX)

Field Name	Value Symbol	Value Description	Hex Value
DIO	DIO0_LOW	DIO pad is low	0x0*
	DIO1_LOW	DIO pad is low	0x0*
	DIO2_LOW	DIO pad is low	0x0*
	DIO3_LOW	DIO pad is low	0x0*
	DIO4_LOW	DIO pad is low	0x0*
	DIO5_LOW	DIO pad is low	0x0*
	DIO6_LOW	DIO pad is low	0x0*
	DIO7_LOW	DIO pad is low	0x0*
	DIO8_LOW	DIO pad is low	0x0*
	DIO9_LOW	DIO pad is low	0x0*
	DIO10_LOW	DIO pad is low	0x0*
	DIO11_LOW	DIO pad is low	0x0*
	DIO12_LOW	DIO pad is low	0x0*
	DIO13_LOW	DIO pad is low	0x0*
	DIO14_LOW	DIO pad is low	0x0*
	DIO15_LOW	DIO pad is low	0x0*
	DIO0_HIGH	DIO pad is high	0x1
	DIO1_HIGH	DIO pad is high	0x2
	DIO2_HIGH	DIO pad is high	0x4
	DIO3_HIGH	DIO pad is high	0x8
	DIO4_HIGH	DIO pad is high	0x10
	DIO5_HIGH	DIO pad is high	0x20
	DIO6_HIGH	DIO pad is high	0x40
	DIO7_HIGH	DIO pad is high	0x80
	DIO8_HIGH	DIO pad is high	0x100
	DIO9_HIGH	DIO pad is high	0x200
	DIO10_HIGH	DIO pad is high	0x400
	DIO11_HIGH	DIO pad is high	0x800
	DIO12_HIGH	DIO pad is high	0x1000
	DIO13_HIGH	DIO pad is high	0x2000
	DIO14_HIGH	DIO pad is high	0x4000
	DIO15_HIGH	DIO pad is high	0x8000
GPIO	GPIO0_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO1_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO2_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO3_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0

Field Name	Value Symbol	Value Description	Hex Value
GPIO (continued)	GPIO4_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO5_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO6_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO7_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO8_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO9_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO10_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO11_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO12_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO13_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO14_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO15_LOW	Set the DIO pad to low if IO_MODE is 0b0000XX	0x0
	GPIO0_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x1
	GPIO1_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x2
	GPIO2_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x4
	GPIO3_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x8
	GPIO4_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x10
	GPIO5_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x20
	GPIO6_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x40
	GPIO7_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x80
	GPIO8_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x100
	GPIO9_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x200
	GPIO10_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x400
	GPIO11_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x800

Field Name	Value Symbol	Value Description	Hex Value
GPIO (continued)	GPIO12_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x1000
	GPIO13_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x2000
	GPIO14_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x4000
	GPIO15_HIGH	Set the DIO pad to high if IO_MODE is 0b0000XX	0x8000

10.4.3 DIO_DIR

Bit Field	Field Name	Description
15:0	DIO	Get DIO[15:0] direction
15:0	GPIO	Set DIO[15:0] GPIO direction (only in IO_MODE is 0b0000XX)

Field Name	Value Symbol	Value Description	Hex Value
DIO	DIO0_INPUT	DIO is an input	0x0*
	DIO1_INPUT	DIO is an input	0x0*
	DIO2_INPUT	DIO is an input	0x0*
	DIO3_INPUT	DIO is an input	0x0*
	DIO4_INPUT	DIO is an input	0x0*
	DIO5_INPUT	DIO is an input	0x0*
	DIO6_INPUT	DIO is an input	0x0*
	DIO7_INPUT	DIO is an input	0x0*
	DIO8_INPUT	DIO is an input	0x0*
	DIO9_INPUT	DIO is an input	0x0*
	DIO10_INPUT	DIO is an input	0x0*
	DIO11_INPUT	DIO is an input	0x0*
	DIO12_INPUT	DIO is an input	0x0*
	DIO13_INPUT	DIO is an input	0x0*
	DIO14_INPUT	DIO is an input	0x0*
	DIO15_INPUT	DIO is an input	0x0*
	DIO0_OUTPUT	DIO is an output	0x1
	DIO1_OUTPUT	DIO is an output	0x2
	DIO2_OUTPUT	DIO is an output	0x4
	DIO3_OUTPUT	DIO is an output	0x8
	DIO4_OUTPUT	DIO is an output	0x10
	DIO5_OUTPUT	DIO is an output	0x20
	DIO6_OUTPUT	DIO is an output	0x40
	DIO7_OUTPUT	DIO is an output	0x80
	DIO8_OUTPUT	DIO is an output	0x100
	DIO9_OUTPUT	DIO is an output	0x200
	DIO10_OUTPUT	DIO is an output	0x400
	DIO11_OUTPUT	DIO is an output	0x800
	DIO12_OUTPUT	DIO is an output	0x1000
	DIO13_OUTPUT	DIO is an output	0x2000
	DIO14_OUTPUT	DIO is an output	0x4000
	DIO15_OUTPUT	DIO is an output	0x8000
GPIO	GPIO0_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO1_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO2_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO3_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0

Field Name	Value Symbol	Value Description	Hex Value
GPIO (continued)	GPIO4_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO5_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO6_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO7_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO8_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO9_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO10_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO11_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO12_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO13_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO14_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO15_INPUT	Set DIO to input if IO_MODE is 0b0000XX	0x0
	GPIO0_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x1
	GPIO1_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x2
	GPIO2_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x4
	GPIO3_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x8
	GPIO4_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x10
	GPIO5_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x20
	GPIO6_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x40
	GPIO7_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x80
	GPIO8_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x100
	GPIO9_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x200
	GPIO10_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x400
	GPIO11_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x800

Field Name	Value Symbol	Value Description	Hex Value
GPIO (continued)	GPIO12_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x1000
	GPIO13_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x2000
	GPIO14_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x4000
	GPIO15_OUTPUT	Set DIO to output if IO_MODE is 0b0000XX	0x8000

10.4.4 DIO_MODE

Bit Field	Field Name	Description
15:0	GPIO	DIO[15:0] mode

Field Name	Value Symbol	Value Description	Hex Value
GPIO	DIO0_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO1_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO2_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO3_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO4_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO5_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO6_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO7_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO8_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO9_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO10_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO11_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO12_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO13_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO14_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*

Field Name	Value Symbol	Value Description	Hex Value
GPIO (continued)	DIO15_IS_NOT_GPIO	This DIO is not configured as a Arm Cortex-M3 core controlled GPIO	0x0*
	DIO0_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x1
	DIO1_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x2
	DIO2_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x4
	DIO3_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x8
	DIO4_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x10
	DIO5_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x20
	DIO6_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x40
	DIO7_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x80
	DIO8_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x100
	DIO9_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x200
	DIO10_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x400
	DIO11_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x800
	DIO12_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x1000
	DIO13_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x2000
	DIO14_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x4000
	DIO15_IS_GPIO	This DIO is configured as a Arm Cortex-M3 core controlled GPIO	0x8000

10.4.5 DIO_INT_CFG

Bit Field	Field Name	Description
11	DEBOUNCE_ENABLE	Interrupt button debounce filter enable/disable
10:8	EVENT	Interrupt event configuration
3:0	SRC	Interrupt input selection

Field Name	Value Symbol	Value Description	Hex Value
DEBOUNCE_ENABLE	DIO_DEBOUNCE_DISABLE	Button debounce filter disabled	0x0*
	DIO_DEBOUNCE_ENABLE	Button debounce filter enabled	0x1

Field Name	Value Symbol	Value Description	Hex Value
EVENT	DIO_EVENT_NONE	Interrupt not triggered	0x0*
	DIO_EVENT_HIGH_LEVEL	Interrupt triggered on high state	0x1
	DIO_EVENT_LOW_LEVEL	Interrupt triggered on low state	0x2
	DIO_EVENT_RISING_EDGE	Interrupt triggered on rising edge	0x3
	DIO_EVENT_FALLING_EDGE	Interrupt triggered on falling edge	0x4
	DIO_EVENT_TRANSITION	Interrupt triggered on any edge	0x5
SRC	DIO_SRC_DIO_0	Select DIO[0] as source	0x0*
	DIO_SRC_DIO_1	Select DIO[1] as source	0x1
	DIO_SRC_DIO_2	Select DIO[2] as source	0x2
	DIO_SRC_DIO_3	Select DIO[3] as source	0x3
	DIO_SRC_DIO_4	Select DIO[4] as source	0x4
	DIO_SRC_DIO_5	Select DIO[5] as source	0x5
	DIO_SRC_DIO_6	Select DIO[6] as source	0x6
	DIO_SRC_DIO_7	Select DIO[7] as source	0x7
	DIO_SRC_DIO_8	Select DIO[8] as source	0x8
	DIO_SRC_DIO_9	Select DIO[9] as source	0x9
	DIO_SRC_DIO_10	Select DIO[10] as source	0xA
	DIO_SRC_DIO_11	Select DIO[11] as source	0xB
	DIO_SRC_DIO_12	Select DIO[12] as source	0xC
	DIO_SRC_DIO_13	Select DIO[13] as source	0xD
	DIO_SRC_DIO_14	Select DIO[14] as source	0xE
	DIO_SRC_DIO_15	Select DIO[15] as source	0xF

10.4.6 DIO_INT_DEBOUNCE

Bit Field	Field Name	Description
8	DEBOUNCE_CLK	Interrupt button debounce filter clock
7:0	DEBOUNCE_COUNT	Interrupt button debounce filter count

Field Name	Value Symbol	Value Description	Hex Value
DEBOUNCE_CLK	DIO_DEBOUNCE_SLOWCLK_DIV32	Button debounce filter runs on SLOWCLK divided by 32	0x0*
	DIO_DEBOUNCE_SLOWCLK_DIV1024	Button debounce filter runs on SLOWCLK divided by 1024	0x1

10.4.7 DIO_PCM_SRC

Bit Field	Field Name	Description
20:16	SERI	PCM_SERI input selection

Bit Field	Field Name	Description
12:8	FRAME	PCM_FRAME input selection
4:0	CLK	PCM_CLK input selection

Field Name	Value Symbol	Value Description	Hex Value
SERI	PCM_SERI_SRC_DIO_0	Select DIO[0] as source	0x0
	PCM_SERI_SRC_DIO_1	Select DIO[1] as source	0x1
	PCM_SERI_SRC_DIO_2	Select DIO[2] as source	0x2
	PCM_SERI_SRC_DIO_3	Select DIO[3] as source	0x3
	PCM_SERI_SRC_DIO_4	Select DIO[4] as source	0x4
	PCM_SERI_SRC_DIO_5	Select DIO[5] as source	0x5
	PCM_SERI_SRC_DIO_6	Select DIO[6] as source	0x6
	PCM_SERI_SRC_DIO_7	Select DIO[7] as source	0x7
	PCM_SERI_SRC_DIO_8	Select DIO[8] as source	0x8
	PCM_SERI_SRC_DIO_9	Select DIO[9] as source	0x9
	PCM_SERI_SRC_DIO_10	Select DIO[10] as source	0xA
	PCM_SERI_SRC_DIO_11	Select DIO[11] as source	0xB
	PCM_SERI_SRC_DIO_12	Select DIO[12] as source	0xC
	PCM_SERI_SRC_DIO_13	Select DIO[13] as source	0xD
	PCM_SERI_SRC_DIO_14	Select DIO[14] as source	0xE
	PCM_SERI_SRC_DIO_15	Select DIO[15] as source	0xF
	PCM_SERI_SRC_CONST_LOW	Select constant low as source	0x10
	PCM_SERI_SRC_CONST_HIGH	Select constant high as source	0x11*

Field Name	Value Symbol	Value Description	Hex Value
FRAME	PCM_FRAME_SRC_DIO_0	Select DIO[0] as source	0x0
	PCM_FRAME_SRC_DIO_1	Select DIO[1] as source	0x1
	PCM_FRAME_SRC_DIO_2	Select DIO[2] as source	0x2
	PCM_FRAME_SRC_DIO_3	Select DIO[3] as source	0x3
	PCM_FRAME_SRC_DIO_4	Select DIO[4] as source	0x4
	PCM_FRAME_SRC_DIO_5	Select DIO[5] as source	0x5
	PCM_FRAME_SRC_DIO_6	Select DIO[6] as source	0x6
	PCM_FRAME_SRC_DIO_7	Select DIO[7] as source	0x7
	PCM_FRAME_SRC_DIO_8	Select DIO[8] as source	0x8
	PCM_FRAME_SRC_DIO_9	Select DIO[9] as source	0x9
	PCM_FRAME_SRC_DIO_10	Select DIO[10] as source	0xA
	PCM_FRAME_SRC_DIO_11	Select DIO[11] as source	0xB
	PCM_FRAME_SRC_DIO_12	Select DIO[12] as source	0xC
	PCM_FRAME_SRC_DIO_13	Select DIO[13] as source	0xD
	PCM_FRAME_SRC_DIO_14	Select DIO[14] as source	0xE
	PCM_FRAME_SRC_DIO_15	Select DIO[15] as source	0xF
	PCM_FRAME_SRC_CONST_LOW	Select constant low as source	0x10
	PCM_FRAME_SRC_CONST_HIGH	Select constant high as source	0x11*
CLK	PCM_CLK_SRC_DIO_0	Select DIO[0] as source	0x0
	PCM_CLK_SRC_DIO_1	Select DIO[1] as source	0x1
	PCM_CLK_SRC_DIO_2	Select DIO[2] as source	0x2
	PCM_CLK_SRC_DIO_3	Select DIO[3] as source	0x3
	PCM_CLK_SRC_DIO_4	Select DIO[4] as source	0x4
	PCM_CLK_SRC_DIO_5	Select DIO[5] as source	0x5
	PCM_CLK_SRC_DIO_6	Select DIO[6] as source	0x6
	PCM_CLK_SRC_DIO_7	Select DIO[7] as source	0x7
	PCM_CLK_SRC_DIO_8	Select DIO[8] as source	0x8
	PCM_CLK_SRC_DIO_9	Select DIO[9] as source	0x9
	PCM_CLK_SRC_DIO_10	Select DIO[10] as source	0xA
	PCM_CLK_SRC_DIO_11	Select DIO[11] as source	0xB
	PCM_CLK_SRC_DIO_12	Select DIO[12] as source	0xC
	PCM_CLK_SRC_DIO_13	Select DIO[13] as source	0xD
	PCM_CLK_SRC_DIO_14	Select DIO[14] as source	0xE
	PCM_CLK_SRC_DIO_15	Select DIO[15] as source	0xF
	PCM_CLK_SRC_CONST_LOW	Select constant low as source	0x10
	PCM_CLK_SRC_CONST_HIGH	Select constant high as source	0x11*

10.4.8 DIO_SPI_SRC

Bit Field	Field Name	Description
20:16	SERI	SPI_SERI input selection
12:8	CS	SPI_CS input selection
4:0	CLK	SPI_CLK input selection

Field Name	Value Symbol	Value Description	Hex Value
SERI	SPI_SERI_SRC_DIO_0	Select DIO[0] as source	0x0
	SPI_SERI_SRC_DIO_1	Select DIO[1] as source	0x1
	SPI_SERI_SRC_DIO_2	Select DIO[2] as source	0x2
	SPI_SERI_SRC_DIO_3	Select DIO[3] as source	0x3
	SPI_SERI_SRC_DIO_4	Select DIO[4] as source	0x4
	SPI_SERI_SRC_DIO_5	Select DIO[5] as source	0x5
	SPI_SERI_SRC_DIO_6	Select DIO[6] as source	0x6
	SPI_SERI_SRC_DIO_7	Select DIO[7] as source	0x7
	SPI_SERI_SRC_DIO_8	Select DIO[8] as source	0x8
	SPI_SERI_SRC_DIO_9	Select DIO[9] as source	0x9
	SPI_SERI_SRC_DIO_10	Select DIO[10] as source	0xA
	SPI_SERI_SRC_DIO_11	Select DIO[11] as source	0xB
	SPI_SERI_SRC_DIO_12	Select DIO[12] as source	0xC
	SPI_SERI_SRC_DIO_13	Select DIO[13] as source	0xD
	SPI_SERI_SRC_DIO_14	Select DIO[14] as source	0xE
	SPI_SERI_SRC_DIO_15	Select DIO[15] as source	0xF
	SPI_SERI_SRC_CONST_LOW	Select constant low as source	0x10
	SPI_SERI_SRC_CONST_HIGH	Select constant high as source	0x11*

Field Name	Value Symbol	Value Description	Hex Value
CS	SPI_CS_SRC_DIO_0	Select DIO[0] as source	0x0
	SPI_CS_SRC_DIO_1	Select DIO[1] as source	0x1
	SPI_CS_SRC_DIO_2	Select DIO[2] as source	0x2
	SPI_CS_SRC_DIO_3	Select DIO[3] as source	0x3
	SPI_CS_SRC_DIO_4	Select DIO[4] as source	0x4
	SPI_CS_SRC_DIO_5	Select DIO[5] as source	0x5
	SPI_CS_SRC_DIO_6	Select DIO[6] as source	0x6
	SPI_CS_SRC_DIO_7	Select DIO[7] as source	0x7
	SPI_CS_SRC_DIO_8	Select DIO[8] as source	0x8
	SPI_CS_SRC_DIO_9	Select DIO[9] as source	0x9
	SPI_CS_SRC_DIO_10	Select DIO[10] as source	0xA
	SPI_CS_SRC_DIO_11	Select DIO[11] as source	0xB
	SPI_CS_SRC_DIO_12	Select DIO[12] as source	0xC
	SPI_CS_SRC_DIO_13	Select DIO[13] as source	0xD
	SPI_CS_SRC_DIO_14	Select DIO[14] as source	0xE
	SPI_CS_SRC_DIO_15	Select DIO[15] as source	0xF
	SPI_CS_SRC_CONST_LOW	Select constant low as source	0x10
	SPI_CS_SRC_CONST_HIGH	Select constant high as source	0x11*
CLK	SPI_CLK_SRC_DIO_0	Select DIO[0] as source	0x0
	SPI_CLK_SRC_DIO_1	Select DIO[1] as source	0x1
	SPI_CLK_SRC_DIO_2	Select DIO[2] as source	0x2
	SPI_CLK_SRC_DIO_3	Select DIO[3] as source	0x3
	SPI_CLK_SRC_DIO_4	Select DIO[4] as source	0x4
	SPI_CLK_SRC_DIO_5	Select DIO[5] as source	0x5
	SPI_CLK_SRC_DIO_6	Select DIO[6] as source	0x6
	SPI_CLK_SRC_DIO_7	Select DIO[7] as source	0x7
	SPI_CLK_SRC_DIO_8	Select DIO[8] as source	0x8
	SPI_CLK_SRC_DIO_9	Select DIO[9] as source	0x9
	SPI_CLK_SRC_DIO_10	Select DIO[10] as source	0xA
	SPI_CLK_SRC_DIO_11	Select DIO[11] as source	0xB
	SPI_CLK_SRC_DIO_12	Select DIO[12] as source	0xC
	SPI_CLK_SRC_DIO_13	Select DIO[13] as source	0xD
	SPI_CLK_SRC_DIO_14	Select DIO[14] as source	0xE
	SPI_CLK_SRC_DIO_15	Select DIO[15] as source	0xF
	SPI_CLK_SRC_CONST_LOW	Select constant low as source	0x10
	SPI_CLK_SRC_CONST_HIGH	Select constant high as source	0x11*

10.4.9 DIO_UART_SRC

Bit Field	Field Name	Description
4:0	Rx	UART_RX input selection

Field Name	Value Symbol	Value Description	Hex Value
Rx	UART_RX_SRC_DIO_0	Select DIO[0] as source	0x0
	UART_RX_SRC_DIO_1	Select DIO[1] as source	0x1
	UART_RX_SRC_DIO_2	Select DIO[2] as source	0x2
	UART_RX_SRC_DIO_3	Select DIO[3] as source	0x3
	UART_RX_SRC_DIO_4	Select DIO[4] as source	0x4
	UART_RX_SRC_DIO_5	Select DIO[5] as source	0x5
	UART_RX_SRC_DIO_6	Select DIO[6] as source	0x6
	UART_RX_SRC_DIO_7	Select DIO[7] as source	0x7
	UART_RX_SRC_DIO_8	Select DIO[8] as source	0x8
	UART_RX_SRC_DIO_9	Select DIO[9] as source	0x9
	UART_RX_SRC_DIO_10	Select DIO[10] as source	0xA
	UART_RX_SRC_DIO_11	Select DIO[11] as source	0xB
	UART_RX_SRC_DIO_12	Select DIO[12] as source	0xC
	UART_RX_SRC_DIO_13	Select DIO[13] as source	0xD
	UART_RX_SRC_DIO_14	Select DIO[14] as source	0xE
	UART_RX_SRC_DIO_15	Select DIO[15] as source	0xF
	UART_RX_SRC_CONST_LOW	Select constant low as source	0x10
	UART_RX_SRC_CONST_HIGH	Select constant high as source	0x11*

10.4.10 DIO_I2C_SRC

Bit Field	Field Name	Description
12:8	SDA	SDA input selection
4:0	SCL	SCL input selection

Field Name	Value Symbol	Value Description	Hex Value
SDA	SDA_SRC_DIO_0	Select DIO[0] as source	0x0
	SDA_SRC_DIO_1	Select DIO[1] as source	0x1
	SDA_SRC_DIO_2	Select DIO[2] as source	0x2
	SDA_SRC_DIO_3	Select DIO[3] as source	0x3
	SDA_SRC_DIO_4	Select DIO[4] as source	0x4
	SDA_SRC_DIO_5	Select DIO[5] as source	0x5
	SDA_SRC_DIO_6	Select DIO[6] as source	0x6
	SDA_SRC_DIO_7	Select DIO[7] as source	0x7
	SDA_SRC_DIO_8	Select DIO[8] as source	0x8
	SDA_SRC_DIO_9	Select DIO[9] as source	0x9
	SDA_SRC_DIO_10	Select DIO[10] as source	0xA
	SDA_SRC_DIO_11	Select DIO[11] as source	0xB
	SDA_SRC_DIO_12	Select DIO[12] as source	0xC
	SDA_SRC_DIO_13	Select DIO[13] as source	0xD
	SDA_SRC_DIO_14	Select DIO[14] as source	0xE
	SDA_SRC_DIO_15	Select DIO[15] as source	0xF
	SDA_SRC_CONST_LOW	Select constant low as source	0x10
	SDA_SRC_CONST_HIGH	Select constant high as source	0x11*
SCL	SCL_SRC_DIO_0	Select DIO[0] as source	0x0
	SCL_SRC_DIO_1	Select DIO[1] as source	0x1
	SCL_SRC_DIO_2	Select DIO[2] as source	0x2
	SCL_SRC_DIO_3	Select DIO[3] as source	0x3
	SCL_SRC_DIO_4	Select DIO[4] as source	0x4
	SCL_SRC_DIO_5	Select DIO[5] as source	0x5
	SCL_SRC_DIO_6	Select DIO[6] as source	0x6
	SCL_SRC_DIO_7	Select DIO[7] as source	0x7
	SCL_SRC_DIO_8	Select DIO[8] as source	0x8
	SCL_SRC_DIO_9	Select DIO[9] as source	0x9
	SCL_SRC_DIO_10	Select DIO[10] as source	0xA
	SCL_SRC_DIO_11	Select DIO[11] as source	0xB
	SCL_SRC_DIO_12	Select DIO[12] as source	0xC
	SCL_SRC_DIO_13	Select DIO[13] as source	0xD
	SCL_SRC_DIO_14	Select DIO[14] as source	0xE
	SCL_SRC_DIO_15	Select DIO[15] as source	0xF
	SCL_SRC_CONST_LOW	Select constant low as source	0x10
	SCL_SRC_CONST_HIGH	Select constant high as source	0x11*

10.4.11 DIO_AUDIOSINK_SRC

Bit Field	Field Name	Description
4:0	CLK	Audio sink clock input selection

Field Name	Value Symbol	Value Description	Hex Value
CLK	AUDIOSINK_CLK_SRC_DIO_0	Select DIO[0] as source	0x0
	AUDIOSINK_CLK_SRC_DIO_1	Select DIO[1] as source	0x1
	AUDIOSINK_CLK_SRC_DIO_2	Select DIO[2] as source	0x2
	AUDIOSINK_CLK_SRC_DIO_3	Select DIO[3] as source	0x3
	AUDIOSINK_CLK_SRC_DIO_4	Select DIO[4] as source	0x4
	AUDIOSINK_CLK_SRC_DIO_5	Select DIO[5] as source	0x5
	AUDIOSINK_CLK_SRC_DIO_6	Select DIO[6] as source	0x6
	AUDIOSINK_CLK_SRC_DIO_7	Select DIO[7] as source	0x7
	AUDIOSINK_CLK_SRC_DIO_8	Select DIO[8] as source	0x8
	AUDIOSINK_CLK_SRC_DIO_9	Select DIO[9] as source	0x9
	AUDIOSINK_CLK_SRC_DIO_10	Select DIO[10] as source	0xA
	AUDIOSINK_CLK_SRC_DIO_11	Select DIO[11] as source	0xB
	AUDIOSINK_CLK_SRC_DIO_12	Select DIO[12] as source	0xC
	AUDIOSINK_CLK_SRC_DIO_13	Select DIO[13] as source	0xD
	AUDIOSINK_CLK_SRC_DIO_14	Select DIO[14] as source	0xE
	AUDIOSINK_CLK_SRC_DIO_15	Select DIO[15] as source	0xF
	AUDIOSINK_CLK_SRC_CONST_LOW	Select constant low as source	0x10
	AUDIOSINK_CLK_SRC_CONST_HIGH	Select constant high as source	0x11*
	AUDIOSINK_CLK_SRC_STANDBYCLK	Select STANDBYCLK as source	0x12

10.4.12 DIO_NMI_SRC

Bit Field	Field Name	Description
5	NMI_POLARITY	NMI polarity
4:0	NMI	NMI input selection

Field Name	Value Symbol	Value Description	Hex Value
NMI_POLARITY	NMI_ACTIVE_LOW	NMI active low	0x0
	NMI_ACTIVE_HIGH	NMI active high	0x1*

Field Name	Value Symbol	Value Description	Hex Value
NMI	NMI_SRC_DIO_0	Select DIO[0] as source	0x0
	NMI_SRC_DIO_1	Select DIO[1] as source	0x1
	NMI_SRC_DIO_2	Select DIO[2] as source	0x2
	NMI_SRC_DIO_3	Select DIO[3] as source	0x3
	NMI_SRC_DIO_4	Select DIO[4] as source	0x4
	NMI_SRC_DIO_5	Select DIO[5] as source	0x5
	NMI_SRC_DIO_6	Select DIO[6] as source	0x6
	NMI_SRC_DIO_7	Select DIO[7] as source	0x7
	NMI_SRC_DIO_8	Select DIO[8] as source	0x8
	NMI_SRC_DIO_9	Select DIO[9] as source	0x9
	NMI_SRC_DIO_10	Select DIO[10] as source	0xA
	NMI_SRC_DIO_11	Select DIO[11] as source	0xB
	NMI_SRC_DIO_12	Select DIO[12] as source	0xC
	NMI_SRC_DIO_13	Select DIO[13] as source	0xD
	NMI_SRC_DIO_14	Select DIO[14] as source	0xE
	NMI_SRC_DIO_15	Select DIO[15] as source	0xF
	NMI_SRC_CONST_LOW	Select constant low as source	0x10*
	NMI_SRC_CONST_HIGH	Select constant high as source	0x11

10.4.13 DIO_BB_RX_SRC

Bit Field	Field Name	Description
20:16	RF_SYNC_P	Baseband controller interface RF_SYNC_P input selection
12:8	CLK	Baseband controller Rx clock input selection
4:0	DATA	Baseband controller Rx data input selection

Field Name	Value Symbol	Value Description	Hex Value
RF_SYNC_P	BB_RF_SYNC_P_SRC_DIO_0	Select DIO[0] as source	0x0
	BB_RF_SYNC_P_SRC_DIO_1	Select DIO[1] as source	0x1
	BB_RF_SYNC_P_SRC_DIO_2	Select DIO[5] as source	0x2
	BB_RF_SYNC_P_SRC_DIO_3	Select DIO[3] as source	0x3
	BB_RF_SYNC_P_SRC_DIO_4	Select DIO[4] as source	0x4
	BB_RF_SYNC_P_SRC_DIO_5	Select DIO[5] as source	0x5
	BB_RF_SYNC_P_SRC_DIO_6	Select DIO[6] as source	0x6
	BB_RF_SYNC_P_SRC_DIO_7	Select DIO[7] as source	0x7
	BB_RF_SYNC_P_SRC_DIO_8	Select DIO[8] as source	0x8
	BB_RF_SYNC_P_SRC_DIO_9	Select DIO[9] as source	0x9
	BB_RF_SYNC_P_SRC_DIO_10	Select DIO[10] as source	0xA
	BB_RF_SYNC_P_SRC_DIO_11	Select DIO[11] as source	0xB
	BB_RF_SYNC_P_SRC_DIO_12	Select DIO[15] as source	0xC
	BB_RF_SYNC_P_SRC_DIO_13	Select DIO[13] as source	0xD
	BB_RF_SYNC_P_SRC_DIO_14	Select DIO[14] as source	0xE
	BB_RF_SYNC_P_SRC_DIO_15	Select DIO[15] as source	0xF
	BB_RF_SYNC_P_SRC_CONST_LOW	Select constant low as source	0x10
	BB_RF_SYNC_P_SRC_CONST_HIGH	Select constant high as source	0x11
	BB_RF_SYNC_P_SRC_RF	Select RF_SYNC_P from RF front-end as source	0x12*
CLK	BB_RX_CLK_SRC_DIO_0	Select DIO[0] as source	0x0
	BB_RX_CLK_SRC_DIO_1	Select DIO[1] as source	0x1
	BB_RX_CLK_SRC_DIO_2	Select DIO[2] as source	0x2
	BB_RX_CLK_SRC_DIO_3	Select DIO[3] as source	0x3
	BB_RX_CLK_SRC_DIO_4	Select DIO[4] as source	0x4
	BB_RX_CLK_SRC_DIO_5	Select DIO[5] as source	0x5
	BB_RX_CLK_SRC_DIO_6	Select DIO[6] as source	0x6
	BB_RX_CLK_SRC_DIO_7	Select DIO[7] as source	0x7
	BB_RX_CLK_SRC_DIO_8	Select DIO[8] as source	0x8
	BB_RX_CLK_SRC_DIO_9	Select DIO[9] as source	0x9
	BB_RX_CLK_SRC_DIO_10	Select DIO[10] as source	0xA
	BB_RX_CLK_SRC_DIO_11	Select DIO[11] as source	0xB
	BB_RX_CLK_SRC_DIO_12	Select DIO[12] as source	0xC
	BB_RX_CLK_SRC_DIO_13	Select DIO[13] as source	0xD
	BB_RX_CLK_SRC_DIO_14	Select DIO[14] as source	0xE
	BB_RX_CLK_SRC_DIO_15	Select DIO[15] as source	0xF
	BB_RX_CLK_SRC_CONST_LOW	Select constant low as source	0x10
	BB_RX_CLK_SRC_CONST_HIGH	Select constant high as source	0x11
	BB_RX_CLK_SRC_RF_GPIO1	Select RF front-end GPIO1 output (RX_CLK) as source	0x12*

Field Name	Value Symbol	Value Description	Hex Value
DATA	BB_RX_DATA_SRC_DIO_0	Select DIO[0] as source	0x0
	BB_RX_DATA_SRC_DIO_1	Select DIO[1] as source	0x1
	BB_RX_DATA_SRC_DIO_2	Select DIO[2] as source	0x2
	BB_RX_DATA_SRC_DIO_3	Select DIO[3] as source	0x3
	BB_RX_DATA_SRC_DIO_4	Select DIO[4] as source	0x4
	BB_RX_DATA_SRC_DIO_5	Select DIO[5] as source	0x5
	BB_RX_DATA_SRC_DIO_6	Select DIO[6] as source	0x6
	BB_RX_DATA_SRC_DIO_7	Select DIO[7] as source	0x7
	BB_RX_DATA_SRC_DIO_8	Select DIO[8] as source	0x8
	BB_RX_DATA_SRC_DIO_9	Select DIO[9] as source	0x9
	BB_RX_DATA_SRC_DIO_10	Select DIO[10] as source	0xA
	BB_RX_DATA_SRC_DIO_11	Select DIO[11] as source	0xB
	BB_RX_DATA_SRC_DIO_12	Select DIO[12] as source	0xC
	BB_RX_DATA_SRC_DIO_13	Select DIO[13] as source	0xD
	BB_RX_DATA_SRC_DIO_14	Select DIO[14] as source	0xE
	BB_RX_DATA_SRC_DIO_15	Select DIO[15] as source	0xF
	BB_RX_DATA_SRC_CONST_LOW	Select constant low as source	0x10
	BB_RX_DATA_SRC_CONST_HIGH	Select constant high as source	0x11
	BB_RX_DATA_SRC_RF_GPIO0	Select RF front-end GPIO0 output (RX_DATA) as source	0x12*

10.4.14 DIO_BB_SPI_SRC

Bit Field	Field Name	Description
4:0	MISO	Baseband controller SPI_MISO input selection

Field Name	Value Symbol	Value Description	Hex Value
MISO	BB_SPI_MISO_SRC_DIO_0	Select DIO[0] as source	0x0
	BB_SPI_MISO_SRC_DIO_1	Select DIO[1] as source	0x1
	BB_SPI_MISO_SRC_DIO_2	Select DIO[2] as source	0x2
	BB_SPI_MISO_SRC_DIO_3	Select DIO[3] as source	0x3
	BB_SPI_MISO_SRC_DIO_4	Select DIO[4] as source	0x4
	BB_SPI_MISO_SRC_DIO_5	Select DIO[5] as source	0x5
	BB_SPI_MISO_SRC_DIO_6	Select DIO[6] as source	0x6
	BB_SPI_MISO_SRC_DIO_7	Select DIO[7] as source	0x7
	BB_SPI_MISO_SRC_DIO_8	Select DIO[8] as source	0x8
	BB_SPI_MISO_SRC_DIO_9	Select DIO[9] as source	0x9
	BB_SPI_MISO_SRC_DIO_10	Select DIO[10] as source	0xA
	BB_SPI_MISO_SRC_DIO_11	Select DIO[11] as source	0xB
	BB_SPI_MISO_SRC_DIO_12	Select DIO[12] as source	0xC
	BB_SPI_MISO_SRC_DIO_13	Select DIO[13] as source	0xD
	BB_SPI_MISO_SRC_DIO_14	Select DIO[14] as source	0xE
	BB_SPI_MISO_SRC_DIO_15	Select DIO[15] as source	0xF
	BB_SPI_MISO_SRC_CONST_LOW	Select constant low as source	0x10
	BB_SPI_MISO_SRC_CONST_HIGH	Select constant high as source	0x11
	BB_SPI_MISO_SRC_RF_SPI_MISO	Select RF front-end SPI_MISO as source	0x12*

10.4.15 DIO_RF_SPI_SRC

Bit Field	Field Name	Description
20:16	MOSI	RF front-end SPI_MOSI input selection
12:8	CSN	RF front-end SPI_CSN input selection
4:0	CLK	RF front-end SPI_CLK input selection

Field Name	Value Symbol	Value Description	Hex Value
MOSI	RF_SPI_MOSI_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_SPI_MOSI_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_SPI_MOSI_SRC_DIO_2	Select DIO[2] as source	0x2
	RF_SPI_MOSI_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_SPI_MOSI_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_SPI_MOSI_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_SPI_MOSI_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_SPI_MOSI_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_SPI_MOSI_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_SPI_MOSI_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_SPI_MOSI_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_SPI_MOSI_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_SPI_MOSI_SRC_DIO_12	Select DIO[12] as source	0xC
	RF_SPI_MOSI_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_SPI_MOSI_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_SPI_MOSI_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_SPI_MOSI_SRC_CONST_LOW	Select constant low as source	0x10
	RF_SPI_MOSI_SRC_CONST_HIGH	Select constant high as source	0x11
	RF_SPI_MOSI_SRC_BB_SPI_MOSI	Select baseband controller SPI_MOSI as source	0x12*
CSN	RF_SPI_CSN_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_SPI_CSN_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_SPI_CSN_SRC_DIO_2	Select DIO[2] as source	0x2
	RF_SPI_CSN_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_SPI_CSN_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_SPI_CSN_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_SPI_CSN_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_SPI_CSN_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_SPI_CSN_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_SPI_CSN_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_SPI_CSN_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_SPI_CSN_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_SPI_CSN_SRC_DIO_12	Select DIO[12] as source	0xC
	RF_SPI_CSN_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_SPI_CSN_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_SPI_CSN_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_SPI_CSN_SRC_CONST_LOW	Select constant low as source	0x10
	RF_SPI_CSN_SRC_CONST_HIGH	Select constant high as source	0x11
	RF_SPI_CSN_SRC_BB_SPI_CSN	Select baseband controller SPI_CSN as source	0x12*

Field Name	Value Symbol	Value Description	Hex Value
CLK	RF_SPI_CLK_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_SPI_CLK_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_SPI_CLK_SRC_DIO_2	Select DIO[2] as source	0x2
	RF_SPI_CLK_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_SPI_CLK_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_SPI_CLK_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_SPI_CLK_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_SPI_CLK_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_SPI_CLK_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_SPI_CLK_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_SPI_CLK_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_SPI_CLK_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_SPI_CLK_SRC_DIO_12	Select DIO[12] as source	0xC
	RF_SPI_CLK_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_SPI_CLK_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_SPI_CLK_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_SPI_CLK_SRC_CONST_LOW	Select constant low as source	0x10
	RF_SPI_CLK_SRC_CONST_HIGH	Select constant high as source	0x11
	RF_SPI_CLK_SRC_BB_SPI_CLK	Select baseband controller SPI_CLK as source	0x12*

10.4.16 DIO_RF_GPIO03_SRC

Bit Field	Field Name	Description
28:24	GPIO3	RF front-end GPIO3 input selection
20:16	GPIO2	RF front-end GPIO2 input selection
12:8	GPIO1	RF front-end GPIO1 input selection
4:0	GPIO0	RF front-end GPIO0 input selection

Field Name	Value Symbol	Value Description	Hex Value
GPIO3	RF_GPIO3_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_GPIO3_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_GPIO3_SRC_DIO_2	Select DIO[3] as source	0x2
	RF_GPIO3_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_GPIO3_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_GPIO3_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_GPIO3_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_GPIO3_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_GPIO3_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_GPIO3_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_GPIO3_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_GPIO3_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_GPIO3_SRC_DIO_12	Select DIO[13] as source	0xC
	RF_GPIO3_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_GPIO3_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_GPIO3_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_GPIO3_SRC_CONST_LOW	Select constant low as source	0x10
	RF_GPIO3_SRC_CONST_HIGH	Select constant high as source	0x11
	RF_GPIO3_SRC_BB_TX_DATA	Select baseband controller TX_DATA as source	0x12*
GPIO2	RF_GPIO2_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_GPIO2_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_GPIO2_SRC_DIO_2	Select DIO[2] as source	0x2
	RF_GPIO2_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_GPIO2_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_GPIO2_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_GPIO2_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_GPIO2_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_GPIO2_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_GPIO2_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_GPIO2_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_GPIO2_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_GPIO2_SRC_DIO_12	Select DIO[12] as source	0xC
	RF_GPIO2_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_GPIO2_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_GPIO2_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_GPIO2_SRC_CONST_LOW	Select constant low as source	0x10
	RF_GPIO2_SRC_CONST_HIGH	Select constant high as source	0x11
	RF_GPIO2_SRC_BB_SYNC_P	Select baseband controller SYNC_P as source	0x12*

Field Name	Value Symbol	Value Description	Hex Value
GPIO1	RF_GPIO1_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_GPIO1_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_GPIO1_SRC_DIO_2	Select DIO[1] as source	0x2
	RF_GPIO1_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_GPIO1_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_GPIO1_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_GPIO1_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_GPIO1_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_GPIO1_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_GPIO1_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_GPIO1_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_GPIO1_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_GPIO1_SRC_DIO_12	Select DIO[11] as source	0xC
	RF_GPIO1_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_GPIO1_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_GPIO1_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_GPIO1_SRC_CONST_LOW	Select constant low as source	0x10*
	RF_GPIO1_SRC_CONST_HIGH	Select constant high as source	0x11
GPIO0	RF_GPIO0_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_GPIO0_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_GPIO0_SRC_DIO_2	Select DIO[0] as source	0x2
	RF_GPIO0_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_GPIO0_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_GPIO0_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_GPIO0_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_GPIO0_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_GPIO0_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_GPIO0_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_GPIO0_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_GPIO0_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_GPIO0_SRC_DIO_12	Select DIO[10] as source	0xC
	RF_GPIO0_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_GPIO0_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_GPIO0_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_GPIO0_SRC_CONST_LOW	Select constant low as source	0x10*
	RF_GPIO0_SRC_CONST_HIGH	Select constant high as source	0x11

10.4.17 DIO_RF_GPIO47_SRC

Bit Field	Field Name	Description
28:24	GPIO7	RF front-end GPIO7 input selection
20:16	GPIO6	RF front-end GPIO6 input selection
12:8	GPIO5	RF front-end GPIO5 input selection
4:0	GPIO4	RE front-end GPIO4 input selection

Field Name	Value Symbol	Value Description	Hex Value
GPIO7	RF_GPIO7_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_GPIO7_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_GPIO7_SRC_DIO_2	Select DIO[5] as source	0x2
	RF_GPIO7_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_GPIO7_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_GPIO7_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_GPIO7_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_GPIO7_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_GPIO7_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_GPIO7_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_GPIO7_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_GPIO7_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_GPIO7_SRC_DIO_12	Select DIO[15] as source	0xC
	RF_GPIO7_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_GPIO7_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_GPIO7_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_GPIO7_SRC_CONST_LOW	Select constant low as source	0x10*
	RF_GPIO7_SRC_CONST_HIGH	Select constant high as source	0x11

Field Name	Value Symbol	Value Description	Hex Value
GPIO6	RF_GPIO6_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_GPIO6_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_GPIO6_SRC_DIO_2	Select DIO[5] as source	0x2
	RF_GPIO6_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_GPIO6_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_GPIO6_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_GPIO6_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_GPIO6_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_GPIO6_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_GPIO6_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_GPIO6_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_GPIO6_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_GPIO6_SRC_DIO_12	Select DIO[15] as source	0xC
	RF_GPIO6_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_GPIO6_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_GPIO6_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_GPIO6_SRC_CONST_LOW	Select constant low as source	0x10*
	RF_GPIO6_SRC_CONST_HIGH	Select constant high as source	0x11
GPIO5	RF_GPIO5_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_GPIO5_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_GPIO5_SRC_DIO_2	Select DIO[5] as source	0x2
	RF_GPIO5_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_GPIO5_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_GPIO5_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_GPIO5_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_GPIO5_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_GPIO5_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_GPIO5_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_GPIO5_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_GPIO5_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_GPIO5_SRC_DIO_12	Select DIO[15] as source	0xC
	RF_GPIO5_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_GPIO5_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_GPIO5_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_GPIO5_SRC_CONST_LOW	Select constant low as source	0x10*
	RF_GPIO5_SRC_CONST_HIGH	Select constant high as source	0x11

Field Name	Value Symbol	Value Description	Hex Value
GPIO4	RF_GPIO4_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_GPIO4_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_GPIO4_SRC_DIO_2	Select DIO[4] as source	0x2
	RF_GPIO4_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_GPIO4_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_GPIO4_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_GPIO4_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_GPIO4_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_GPIO4_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_GPIO4_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_GPIO4_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_GPIO4_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_GPIO4_SRC_DIO_12	Select DIO[14] as source	0xC
	RF_GPIO4_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_GPIO4_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_GPIO4_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_GPIO4_SRC_CONST_LOW	Select constant low as source	0x10
	RF_GPIO4_SRC_CONST_HIGH	Select constant high as source	0x11
	RF_GPIO4_SRC_BB_TX_DATA_VALID	Select baseband controller TX_DATA_VALID as source	0x12*

10.4.18 DIO_RF_GPIO89_SRC

Bit Field	Field Name	Description
12:8	GPIO9	RF front-end GPIO9 input selection
4:0	GPIO8	RF front-end GPIO8 input selection

Field Name	Value Symbol	Value Description	Hex Value
GPIO9	RF_GPIO9_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_GPIO9_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_GPIO9_SRC_DIO_2	Select DIO[5] as source	0x2
	RF_GPIO9_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_GPIO9_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_GPIO9_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_GPIO9_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_GPIO9_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_GPIO9_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_GPIO9_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_GPIO9_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_GPIO9_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_GPIO9_SRC_DIO_12	Select DIO[15] as source	0xC
	RF_GPIO9_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_GPIO9_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_GPIO9_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_GPIO9_SRC_CONST_LOW	Select constant low as source	0x10*
	RF_GPIO9_SRC_CONST_HIGH	Select constant high as source	0x11
GPIO8	RF_GPIO8_SRC_DIO_0	Select DIO[0] as source	0x0
	RF_GPIO8_SRC_DIO_1	Select DIO[1] as source	0x1
	RF_GPIO8_SRC_DIO_2	Select DIO[5] as source	0x2
	RF_GPIO8_SRC_DIO_3	Select DIO[3] as source	0x3
	RF_GPIO8_SRC_DIO_4	Select DIO[4] as source	0x4
	RF_GPIO8_SRC_DIO_5	Select DIO[5] as source	0x5
	RF_GPIO8_SRC_DIO_6	Select DIO[6] as source	0x6
	RF_GPIO8_SRC_DIO_7	Select DIO[7] as source	0x7
	RF_GPIO8_SRC_DIO_8	Select DIO[8] as source	0x8
	RF_GPIO8_SRC_DIO_9	Select DIO[9] as source	0x9
	RF_GPIO8_SRC_DIO_10	Select DIO[10] as source	0xA
	RF_GPIO8_SRC_DIO_11	Select DIO[11] as source	0xB
	RF_GPIO8_SRC_DIO_12	Select DIO[15] as source	0xC
	RF_GPIO8_SRC_DIO_13	Select DIO[13] as source	0xD
	RF_GPIO8_SRC_DIO_14	Select DIO[14] as source	0xE
	RF_GPIO8_SRC_DIO_15	Select DIO[15] as source	0xF
	RF_GPIO8_SRC_CONST_LOW	Select constant low as source	0x10*
	RF_GPIO8_SRC_CONST_HIGH	Select constant high as source	0x11

10.4.19 DIO_DMIC_SRC

Bit Field	Field Name	Description
12:8	CLK	DMIC clock input selection
4:0	DATA	DMIC data input selection

Field Name	Value Symbol	Value Description	Hex Value
CLK	DMIC_CLK_SRC_DIO_0	Select DIO[0] as source	0x0
	DMIC_CLK_SRC_DIO_1	Select DIO[1] as source	0x1
	DMIC_CLK_SRC_DIO_2	Select DIO[2] as source	0x2
	DMIC_CLK_SRC_DIO_3	Select DIO[3] as source	0x3
	DMIC_CLK_SRC_DIO_4	Select DIO[4] as source	0x4
	DMIC_CLK_SRC_DIO_5	Select DIO[5] as source	0x5
	DMIC_CLK_SRC_DIO_6	Select DIO[6] as source	0x6
	DMIC_CLK_SRC_DIO_7	Select DIO[7] as source	0x7
	DMIC_CLK_SRC_DIO_8	Select DIO[8] as source	0x8
	DMIC_CLK_SRC_DIO_9	Select DIO[9] as source	0x9
	DMIC_CLK_SRC_DIO_10	Select DIO[10] as source	0xA
	DMIC_CLK_SRC_DIO_11	Select DIO[11] as source	0xB
	DMIC_CLK_SRC_DIO_12	Select DIO[12] as source	0xC
	DMIC_CLK_SRC_DIO_13	Select DIO[13] as source	0xD
	DMIC_CLK_SRC_DIO_14	Select DIO[14] as source	0xE
	DMIC_CLK_SRC_DIO_15	Select DIO[15] as source	0xF
	DMIC_CLK_SRC_CONST_LOW	Select constant low as source	0x10
	DMIC_CLK_SRC_CONST_HIGH	Select constant high as source	0x11*

Field Name	Value Symbol	Value Description	Hex Value
DATA	DMIC_DATA_SRC_DIO_0	Select DIO[0] as source	0x0
	DMIC_DATA_SRC_DIO_1	Select DIO[1] as source	0x1
	DMIC_DATA_SRC_DIO_2	Select DIO[2] as source	0x2
	DMIC_DATA_SRC_DIO_3	Select DIO[3] as source	0x3
	DMIC_DATA_SRC_DIO_4	Select DIO[4] as source	0x4
	DMIC_DATA_SRC_DIO_5	Select DIO[5] as source	0x5
	DMIC_DATA_SRC_DIO_6	Select DIO[6] as source	0x6
	DMIC_DATA_SRC_DIO_7	Select DIO[7] as source	0x7
	DMIC_DATA_SRC_DIO_8	Select DIO[8] as source	0x8
	DMIC_DATA_SRC_DIO_9	Select DIO[9] as source	0x9
	DMIC_DATA_SRC_DIO_10	Select DIO[10] as source	0xA
	DMIC_DATA_SRC_DIO_11	Select DIO[11] as source	0xB
	DMIC_DATA_SRC_DIO_12	Select DIO[12] as source	0xC
	DMIC_DATA_SRC_DIO_13	Select DIO[13] as source	0xD
	DMIC_DATA_SRC_DIO_14	Select DIO[14] as source	0xE
	DMIC_DATA_SRC_DIO_15	Select DIO[15] as source	0xF
	DMIC_DATA_SRC_CONST_LOW	Select constant low as source	0x10
	DMIC_DATA_SRC_CONST_HIGH	Select constant high as source	0x11*

10.4.20 DIO_LPDSP32_JTAG_SRC

Bit Field	Field Name	Description
20:16	TDI	LPDSP32_TDI input selection
12:8	TMS	LPDSP32_TMS input selection
4:0	TCK	LPDSP32_TCK input selection

Field Name	Value Symbol	Value Description	Hex Value
TDI	LPDSP32_TDI_SRC_DIO_0	Select DIO[0] as source	0x0
	LPDSP32_TDI_SRC_DIO_1	Select DIO[1] as source	0x1
	LPDSP32_TDI_SRC_DIO_2	Select DIO[2] as source	0x2
	LPDSP32_TDI_SRC_DIO_3	Select DIO[3] as source	0x3
	LPDSP32_TDI_SRC_DIO_4	Select DIO[4] as source	0x4
	LPDSP32_TDI_SRC_DIO_5	Select DIO[5] as source	0x5
	LPDSP32_TDI_SRC_DIO_6	Select DIO[6] as source	0x6
	LPDSP32_TDI_SRC_DIO_7	Select DIO[7] as source	0x7
	LPDSP32_TDI_SRC_DIO_8	Select DIO[8] as source	0x8
	LPDSP32_TDI_SRC_DIO_9	Select DIO[9] as source	0x9
	LPDSP32_TDI_SRC_DIO_10	Select DIO[10] as source	0xA
	LPDSP32_TDI_SRC_DIO_11	Select DIO[11] as source	0xB
	LPDSP32_TDI_SRC_DIO_12	Select DIO[12] as source	0xC
	LPDSP32_TDI_SRC_DIO_13	Select DIO[13] as source	0xD
	LPDSP32_TDI_SRC_DIO_14	Select DIO[14] as source	0xE
	LPDSP32_TDI_SRC_DIO_15	Select DIO[15] as source	0xF
	LPDSP32_TDI_SRC_CONST_LOW	Select constant low as source	0x10
	LPDSP32_TDI_SRC_CONST_HIGH	Select constant high as source	0x11*
TMS	LPDSP32_TMS_SRC_DIO_0	Select DIO[0] as source	0x0
	LPDSP32_TMS_SRC_DIO_1	Select DIO[1] as source	0x1
	LPDSP32_TMS_SRC_DIO_2	Select DIO[2] as source	0x2
	LPDSP32_TMS_SRC_DIO_3	Select DIO[3] as source	0x3
	LPDSP32_TMS_SRC_DIO_4	Select DIO[4] as source	0x4
	LPDSP32_TMS_SRC_DIO_5	Select DIO[5] as source	0x5
	LPDSP32_TMS_SRC_DIO_6	Select DIO[6] as source	0x6
	LPDSP32_TMS_SRC_DIO_7	Select DIO[7] as source	0x7
	LPDSP32_TMS_SRC_DIO_8	Select DIO[8] as source	0x8
	LPDSP32_TMS_SRC_DIO_9	Select DIO[9] as source	0x9
	LPDSP32_TMS_SRC_DIO_10	Select DIO[10] as source	0xA
	LPDSP32_TMS_SRC_DIO_11	Select DIO[11] as source	0xB
	LPDSP32_TMS_SRC_DIO_12	Select DIO[12] as source	0xC
	LPDSP32_TMS_SRC_DIO_13	Select DIO[13] as source	0xD
	LPDSP32_TMS_SRC_DIO_14	Select DIO[14] as source	0xE
	LPDSP32_TMS_SRC_DIO_15	Select DIO[15] as source	0xF
	LPDSP32_TMS_SRC_CONST_LOW	Select constant low as source	0x10
	LPDSP32_TMS_SRC_CONST_HIGH	Select constant high as source	0x11*

Field Name	Value Symbol	Value Description	Hex Value
TCK	LPDSP32_TCK_SRC_DIO_0	Select DIO[0] as source	0x0
	LPDSP32_TCK_SRC_DIO_1	Select DIO[1] as source	0x1
	LPDSP32_TCK_SRC_DIO_2	Select DIO[2] as source	0x2
	LPDSP32_TCK_SRC_DIO_3	Select DIO[3] as source	0x3
	LPDSP32_TCK_SRC_DIO_4	Select DIO[4] as source	0x4
	LPDSP32_TCK_SRC_DIO_5	Select DIO[5] as source	0x5
	LPDSP32_TCK_SRC_DIO_6	Select DIO[6] as source	0x6
	LPDSP32_TCK_SRC_DIO_7	Select DIO[7] as source	0x7
	LPDSP32_TCK_SRC_DIO_8	Select DIO[8] as source	0x8
	LPDSP32_TCK_SRC_DIO_9	Select DIO[9] as source	0x9
	LPDSP32_TCK_SRC_DIO_10	Select DIO[10] as source	0xA
	LPDSP32_TCK_SRC_DIO_11	Select DIO[11] as source	0xB
	LPDSP32_TCK_SRC_DIO_12	Select DIO[12] as source	0xC
	LPDSP32_TCK_SRC_DIO_13	Select DIO[13] as source	0xD
	LPDSP32_TCK_SRC_DIO_14	Select DIO[14] as source	0xE
	LPDSP32_TCK_SRC_DIO_15	Select DIO[15] as source	0xF
	LPDSP32_TCK_SRC_CONST_LOW	Select constant low as source	0x10
	LPDSP32_TCK_SRC_CONST_HIGH	Select constant high as source	0x11*

10.4.21 DIO_JTAG_SW_PAD_CFG

Bit Field	Field Name	Description
9	JTCK_LPF	JTCK Low-Pass-Filter enable / disable
8	JTMS_LPF	JTMS Low-Pass-Filter enable / disable
7	CM3_JTAG_DATA_EN	CM3 JTAG on DIO[14:15]
6	CM3_JTAG_TRST_EN	CM3 JTAG TRST on DIO13
5:4	JTCK_PULL	JTCK pull-up enable / disable
3:2	JTMS_DRIVE	JTMS drive strength
1:0	JTMS_PULL	JTMS pull-up enable / disable

Field Name	Value Symbol	Value Description	Hex Value
JTCK_LPF	JTCK_LPF_DISABLED	Disable low pass filter	0x0*
	JTCK_LPF_ENABLED	Enable low pass filter	0x1
JTMS_LPF	JTMS_LPF_DISABLED	Disable low pass filter	0x0*
	JTMS_LPF_ENABLED	Enable low pass filter	0x1
CM3_JTAG_DATA_EN	CM3_JTAG_DATA_DISABLED	CM3 JTAG data (TDI and TDO) not available on DIO[14:15]	0x0
	CM3_JTAG_DATA_ENABLED	CM3 JTAG data (TDI and TDO) connected through DIO[14:15]	0x1*

Field Name	Value Symbol	Value Description	Hex Value
CM3_JTAG_TRST_EN	CM3_JTAG_TRST_DISABLED	CM3 JTAG TRST not available on DIO13	0x0
	CM3_JTAG_TRST_ENABLED	CM3 JTAG TRST connected through DIO13	0x1*
JTCK_PULL	JTCK_NO_PULL	No pull selected	0x0
	JTCK_WEAK_PULL_UP	Weak pull-up selected	0x1*
	JTCK_WEAK_PULL_DOWN	Weak pull-down selected	0x2
	JTCK_STRONG_PULL_UP	Strong pull-up selected	0x3
JTMS_DRIVE	JTMS_2X_DRIVE	2x drive strength	0x0
	JTMS_3X_DRIVE	3x drive strength	0x1
	JTMS_5X_DRIVE	5x drive strength	0x2
	JTMS_6X_DRIVE	6x drive strength	0x3*
JTMS_PULL	JTMS_NO_PULL	No pull selected	0x0
	JTMS_WEAK_PULL_UP	Weak pull-up selected	0x1*
	JTMS_WEAK_PULL_DOWN	Weak pull-down selected	0x2
	JTMS_STRONG_PULL_UP	Strong pull-up selected	0x3

10.4.22 DIO_EXTCLK_CFG

Bit Field	Field Name	Description
2	LPF	Low Pass Filter enable
1:0	PULL_CTRL	Pull selection

Field Name	Value Symbol	Value Description	Hex Value
LPF	EXTCLK_LPF_DISABLE	Disable low pass filter	0x0*
	EXTCLK_LPF_ENABLE	Enable low pass filter	0x1
PULL_CTRL	EXTCLK_NO_PULL	No pull selected	0x0
	EXTCLK_WEAK_PULL_UP	Weak pull-up selected	0x1*
	EXTCLK_WEAK_PULL_DOWN	Weak pull-down selected	0x2
	EXTCLK_STRONG_PULL_UP	Strong pull-down selected	0x3

10.4.23 DIO_PAD_CFG

Bit Field	Field Name	Description
0	DRIVE	Drive strength configuration (scales the individual drive strengths)

Field Name	Value Symbol	Value Description	Hex Value
DRIVE	PAD_LOW_DRIVE	All pad regular drive strengths	0x0
	PAD_HIGH_DRIVE	All pad drive strengths increased by ~50%	0x1*

CHAPTER 11

External Digital Interfaces

11.1 Introduction

This chapter provides information about the interfaces available to the Arm Cortex-M3 processor that can be used to pass data between RSL10 and external components. These interfaces are:

Analog-to-Digital Converter (ADC)	The ADC can be used to convert external or internal analog signals for monitoring or as required by the user application, with a choice of two different resolutions.
General-Purpose I/O (GPIO)	Any of the DIO pads can be configured as software-controlled GPIO pads, and used for any general-purpose input or output as defined by a user application.
I^{2}C	A subset of the Philips I^{2}C interface, this external interfaces supports both master and slave mode transfers.
Pulse Code Modulation (PCM)	Used for streaming control, configuration or signal data into and out of the microcontroller in PCM format.
Pulse Width Modulation (PWM)	Two pulse-width modulator (PWM) drivers can be configured to generate a single output signal with a specified period and duty cycle. Each PWM driver can be used independently as a simple D/A converter.
Serial Peripheral Interface (SPI)	Allows the system to communicate with external components including external analog front ends, external controllers, and non-volatile memories (NVM).
Universal Asynchronous Receiver-Transmitter (UART)	Provides support for communicating with devices that use the standard UART transmission protocol.
Support	Several internal connections that link the RF front-end and the Bluetooth Low Energy technology baseband hardware.

11.2 Analog-to-Digital Converters (ADCs)

The analog-to-digital converters (ADCs) provide an analog to digital conversion of up to eight differential combinations of four internal signals and four external signals. Each conversion is a differential measurement with a configurable resolution for the converter of 8 or 14 bits of precision.

IMPORTANT: For accurate ADC measurements across operating conditions, the VDDC supply voltage level must be a minimum of 1.00 V. For more information about VDDC configuration, see Section 5.3.4, “Digital Supply Voltages” on page 46, and the “Manufacturing Calibrated Settings” from the *RSL10 Firmware Reference*.

11.2.1 ADC Input Configuration

The purpose of the ADCs is to sample analog signals that are relevant to the user's application use cases—for example, the voltage associated with a potentiometer-based volume control, or a supply voltage for battery monitoring applications using the Bluetooth low energy battery service (BAS).

The signals measured by the ADC block are configured using the `NEG_INPUT_SEL` and `POS_INPUT_SEL` bit-fields from the `ADC_INPUT_SEL` register set. The negative and positive signals used for each differential measurement are selected from:

DIOs 0, 1, 2, 3
: To use ADC functionality, the DIO must be configured as follows:

1. Set the `IO_MODE` field of the `DIO_CFG` register to 0x3F.
2. Set the `PULL_CTRL` field of the `DIO_CFG` register to 0x0.

For more information about DIO configuration for ADCs, see Chapter 10, "Digital Input/ Output" on page 259.

AOUT
: The analog test output signal. The signal provided to AOUT can be configured using the `ACS_AOUT_CTRL_TEST_AOUT` bit field from the `ACS_AOUT_CTRL` register, which allows the ADCs to additionally measure a number of other internal power supply outputs and status flags. For more information about the internal power supplies, see Section 5.3, "Internal Power Supply Voltages" on page 39.

VDDC
: For more information about VDDC and its configuration, see Section 5.3.4, "Digital Supply Voltages" on page 46.

VBAT/2
: A divided form of the battery supply voltage, measured through a fixed resistive divider which ensures that the measured value is in the expected range of the ADC for accurate measurement. To save power, the resistive divider can be configured to only be enabled when a conversion is taking place by setting the `ADC_CFG_DUTY_DIVIDER` bit-field from the `ADC_CFG` register to `ADC_VBAT_DIV2_DUTY`. For additional information about VBAT, see Section 5.2.1, "Battery Supply Voltage (VBAT)" on page 38.

GND
: Ground (`VSS`) supply reference.

11.2.2 ADC Sampling Configuration

The ADC signals are sampled at a sampling rate derived from SLOWCLK, and configured using the `ADC_CFG_FREQ` bit-field from the `ADC_CFG` register. Configuration of SLOWCLK is described in Section 6.3.3, "Slow Clock (SLOWCLK)" on page 80, with SLOWCLK typically initialized to 1 MHz. ADC measurements using this divided clock can be configured for two sampling modes:

Low-Frequency Mode
: SLOWCLK is first prescaled by a fixed factor of 10, with a maximum sampling rate of 5 kHz. ADC measurement results have a resolution of 14 bits.

High-Frequency Mode
: SLOWCLK is prescaled by a factor of 2, with sampling rates of up to 25 kHz where ADC measurement results have a resolution of 14 bits, or up to 50 kHz where ADC measurement results have a resolution of 8 bits.

In addition to selecting the sampling mode, the `ADC_CFG_FREQ` bit-field defines the frequency at which measurements are taken, and the native voltage range of measurements. Lower reference measurement rates provide a larger native voltage range, and in the case of High-Frequency Mode, provide a higher resolution ADC measurement.

The ADC block samples data at the specified frequency, with data samples read from all eight channels in Normal Mode, and from only one channel sampled in continuous mode. This results in a lower effective sample rate for Normal Mode operations.

- If the ADC block is configured in Normal Mode, each ADC channel is sampled in sequence and an ADC interrupt occurs once every eighth sample. The channel number that triggers the interrupt (or phase offset) can be defined as part of the `ADC_INT_CH_NUM` bit field in the `ADC_BATMON_INT_ENABLE` register. Between each interrupt, the data value for each ADC channel is updated to a new, valid sample.
- If the ADC is configured in Continuous Mode, only one ADC channel is sampled and an interrupt occurs every sample. The sampled channel is defined in the `ADC_INT_CH_NUM` bit field as part of the `ADC_BATMON_INT_ENABLE` register.

The resulting sample rates per channel as a function of SLOWCLK are shown in Table 22.

Table 22. ADC Sample Rate Configuration

Value	Setting	Fixed Divider	Configurable Division	Native Voltage Range (V)	Effective Division of SLOWCLK (Normal Mode)	Effective Division of SLOWCLK (Continuous Mode)
0x0	`ADC_DISABLE`	N/A	ADC Disabled	N/A	ADC Disabled	ADC Disabled
0x1	`ADC_PRESCALE_200`	10	Divide by 20	-0.125 to 2.125	Divide by 1600	Divide by 200
0x2	`ADC_PRESCALE_400`	10	Divide by 40	-0.063 to 2.063	Divide by 3200	Divide by 400
0x3	`ADC_PRESCALE_640`	10	Divide by 64	-0.031 to 2.031	Divide by 5120	Divide by 640
0x4	`ADC_PRESCALE_800`	10	Divide by 80	-0.016 to 2.016	Divide by 6400	Divide by 800
0x5	`ADC_PRESCALE_1600`	10	Divide by 160	-0.008 to 2.008	Divide by 12800	Divide by 1600
0x6	`ADC_PRESCALE_3200`	10	Divide by 320	-0.004 to 2.004	Divide by 25600	Divide by 3200
0x7	`ADC_PRESCALE_6400`	10	Divide by 640	-0.002 to 2.002	Divide by 51200	Divide by 6400
0x8	`ADC_PRESCALE_20H`	2	Divide by 10	-1.0 to 3.0	Divide by 160	Divide by 20
0x9	`ADC_PRESCALE_40H`	2	Divide by 20	-0.125 to 2.125	Divide by 320	Divide by 40
0xA	`ADC_PRESCALE_80H`	2	Divide by 40	-0.063 to 2.063	Divide by 512	Divide by 64
0xB	`ADC_PRESCALE_128H`	2	Divide by 64	-0.031 to 2.031	Divide by 640	Divide by 80
0xC	`ADC_PRESCALE_160H`	2	Divide by 80	-0.016 to 2.016	Divide by 1280	Divide by 160
0xD	`ADC_PRESCALE_320H`	2	Divide by 160	-0.008 to 2.008	Divide by 2560	Divide by 320
0xE	`ADC_PRESCALE_640H`	2	Divide by 320	-0.004 to 2.004	Divide by 5120	Divide by 640
0xF	`ADC_PRESCALE_1280H`	2	Divide by 640	-0.002 to 2.002	Divide by 10240	Divide by 1280

NOTE: For a typical SLOWCLK frequency of 1.00 MHz, the ADC samples all eight channels in Normal mode at a configurable rate between 6.25 kHz and 19.5 Hz.

11.2.3 ADC Output Data

The converted output from the most recent conversion of each channel can be found in the `ADC_DATA_TRIM_CH*` and `ADC_DATA_AUDIO_CH*` registers. Data read from both registers that refer to each channel are equivalent, and only the interpretation of the underlying measurement is different.

`ADC_DATA_TRIM_CH*`
: The output data value from the ADC is represented as trimmer data providing a 14-bit unsigned value scaled between 0x000 and 0x3FFF to represent a signal in the range from 0 to 2.0 V. Any measurement outside of the range from 0 to 2.0 V is saturated.

`ADC_DATA_AUDIO_CH*`
: The output data value from the ADC is represented as audio data providing a 14-bit signed value scaled between 0xFFFFE000 and 0x00001FFF to represent a signal in the range from 0 to 2.0 V. Measurements outside of the range from 0 to 2.0 V can extend this range if the ADC sampling configuration extends the range of measured values.

For improved accuracy in the ADC data, the RSL10 SoC provides an offset correction factor in the `ADC_OFFSET` register that automatically applies a compensation value to the measured ADC data. This value normally should be the difference between the expected and measured ADC output when the input is 0 V. The value of this register is automatically updated with a measured compensation value that corrects the measurements when an ADC channel selects GND as the source for both the positive and negative sources for the channel. If no channels are configured in this way, the offset register can be configured to provide any desired compensation (including selecting no compensation by setting this register to 0x00000000).

11.2.4 Power Supply Monitoring

The ADC block additionally provides resources that can be used to monitor the power supply through VBAT/2 (useful if the system is supplied directly from VBAT), or VCC (useful if the system is being supplied through the DC-DC converter). From the information obtained from these supplies, you can detect when the battery is getting close to the end of life.

The supply threshold level can be defined between 0 V and 2 V using 256 steps of approximately 7.8 mV. This is defined in the `ADC_BATMON_CFG_SUPPLY_THRESHOLD` bit field. Select either VBAT/2 or VCC as the source to monitor by setting the `ADC_BATMON_CFG_SUPPLY_SRC` bit. Both of these values are part of the `ADC_BATMON_CFG` register.

When the monitored supply is measured as being below the specified threshold, the value of the `ADC_BATMON_COUNT_VAL` register is incremented. This counter register is reset when read. The value of this register is compared with the value stored to the `ADC_BATMON_CFG_ALARM_COUNT_VALUE` bit-field from the `ADC_BATMON_CFG` register, and if the counter value ever exceeds the alarm value, an alarm is generated.

11.2.5 ADC and Power Supply Monitoring Interrupt

A single interrupt line is shared between the ADC and the power supply monitoring circuitry. This interrupt can be independently configured to trigger when one or both of the following conditions are met:

- Triggering when a specified ADC channel is sampled. To enable this trigger condition, use the `ADC_INT_ENABLE` bit from the `ADC_BATMON_INT_ENABLE` register. To specify the ADC channel, use the `ADC_INT_CH_NUM` bit-field from this same register.
- Triggering when a power supply monitor alarm occurs. To enable this trigger condition, use the `BATMON_ALARM_INT_ENABLE` bit from the `ADC_BATMON_INT_ENABLE` register.

NOTE: To configure a single ADC and/or battery monitoring interrupt, set the ADC_INT_CH_NUM field to the selected battery monitoring source.

The source of the interrupt can always be determined using fields within the ADC_BATMON_STATUS register. The BATMON_ALARM_STAT bit indicates an alarm condition within the battery monitoring circuit. The ADC_READY_STAT bit indicates that a new set of samples are available in the ADC_xxx_DATA registers. The ADC_OVERRUN_STAT bit indicates that the ADC_xxx_DATA registers were not read between successive samples, and one or more samples were overwritten.

NOTE: All three flags in the ADC_BATMON_STATUS register are sticky, and each must be separately cleared by setting the appropriate ADC_BATMON_STATUS_*_CLEAR bit from the ADC_BATMON_STATUS register.

11.2.6 ADC Registers

Register Name	Register Description	Address
ADC_DATA_TRIM_CH	ADC conversion result for channel 0 to 7 in trimmer mode	0x40001200
ADC_DATA_AUDIO_CH	ADC conversion result for channel 0 to 7 in audio mode (signed)	0x40001220
ADC_INPUT_SEL	ADC input selection for channel 0 to 7	0x40001240
ADC_CFG	ADC Configuration Register	0x40001260
ADC_OFFSET	ADC conversion result for ADC GND	0x40001264
ADC_BATMON_CFG	Battery Monitoring Configuration Register	0x40001270
ADC_BATMON_INT_ENABLE	ADC / BATMON Interrupt Mask Register	0x40001274
ADC_BATMON_COUNT_VAL	Battery Monitoring Status Register	0x40001278
ADC_BATMON_STATUS	ADC / BATMON Status Register	0x4000127C

11.2.6.1 ADC_DATA_TRIM_CH

Bit Field	Field Name	Description
13:0	DATA	14-bit ADC conversion result

11.2.6.2 ADC_DATA_AUDIO_CH

Bit Field	Field Name	Description
31:0	DATA	14-bit ADC conversion result (sign extended to 32 bits)

11.2.6.3 ADC_INPUT_SEL

Bit Field	Field Name	Description
6:4	POS_INPUT_SEL	Positive input selection
2:0	NEG_INPUT_SEL	Negative input selection

Field Name	Value Symbol	Value Description	Hex Value
POS_INPUT_SEL	ADC_POS_INPUT_DIO0	Select DIO0 as positive input	0x0
	ADC_POS_INPUT_DIO1	Select DIO1 as positive input	0x1
	ADC_POS_INPUT_DIO2	Select DIO2 as positive input	0x2
	ADC_POS_INPUT_DIO3	Select DIO3 as positive input	0x3
	ADC_POS_INPUT_AOUT	Select AOUT as positive input	0x4
	ADC_POS_INPUT_VDDC	Select VDDC as positive input	0x5
	ADC_POS_INPUT_VBAT_DIV2	Select VBAT/2 as positive input	0x6*
	ADC_POS_INPUT_GND	Select ADC internal ground as positive input	0x7
NEG_INPUT_SEL	ADC_NEG_INPUT_DIO0	Select DIO0 as negative input	0x0
	ADC_NEG_INPUT_DIO1	Select DIO1 as negative input	0x1
	ADC_NEG_INPUT_DIO2	Select DIO2 as negative input	0x2
	ADC_NEG_INPUT_DIO3	Select DIO3 as negative input	0x3
	ADC_NEG_INPUT_AOUT	Select AOUT as negative input	0x4
	ADC_NEG_INPUT_VDDC	Select VDDC as negative input	0x5
	ADC_NEG_INPUT_VBAT_DIV2	Select VBAT/2 as negative input	0x6
	ADC_NEG_INPUT_GND	Select ADC internal ground as negative input	0x7*

11.2.6.4 ADC_CFG

Bit Field	Field Name	Description
5	DUTY_DIVIDER	Duty cycling VDD divider
4	CONTINUOUS_MODE	ADC continuously sampling the channel selected as interrupt source (ADC_INT_CH_NUM)
3:0	FREQ	Defines the sampling frequency of the ADC channels

Field Name	Value Symbol	Value Description	Hex Value
DUTY_DIVIDER	ADC_VBAT_DIV2_NORMAL	Normal mode: VBAT divider is always enabled	0x0*
	ADC_VBAT_DIV2_DUTY	Duty cycling VBAT divider (enabled only during ADC conversion)	0x1
CONTINUOUS_MODE	ADC_NORMAL	Normal mode, all 8 channels sampled	0x0*
	ADC_CONTINUOUS	Continuous mode: only one channel sampled (for test purpose)	0x1

Field Name	Value Symbol	Value Description	Hex Value
FREQ	ADC_DISABLE	ADC disabled	0x0*
	ADC_PRESCALE_200	Sample rate is SLOWCLK/200 (low frequency mode)	0x1
	ADC_PRESCALE_400	Sample rate is SLOWCLK/400 (low frequency mode)	0x2
	ADC_PRESCALE_640	Sample rate is SLOWCLK/640 (low frequency mode)	0x3
	ADC_PRESCALE_800	Sample rate is SLOWCLK/800 (low frequency mode)	0x4
	ADC_PRESCALE_1600	Sample rate is SLOWCLK/1600 (low frequency mode)	0x5
	ADC_PRESCALE_3200	Sample rate is SLOWCLK/3200 (low frequency mode)	0x6
	ADC_PRESCALE_6400	Sample rate is SLOWCLK/6400 (low frequency mode)	0x7
	ADC_PRESCALE_20H	Sample rate is SLOWCLK/20 (high frequency mode)	0x8
	ADC_PRESCALE_40H	Sample rate is SLOWCLK/40 (high frequency mode)	0x9
	ADC_PRESCALE_80H	Sample rate is SLOWCLK/80 (high frequency mode)	0xA
	ADC_PRESCALE_128H	Sample rate is SLOWCLK/128 (high frequency mode)	0xB
	ADC_PRESCALE_160H	Sample rate is SLOWCLK/160 (high frequency mode)	0xC
	ADC_PRESCALE_320H	Sample rate is SLOWCLK/320 (high frequency mode)	0xD
	ADC_PRESCALE_640H	Sample rate is SLOWCLK/640 (high frequency mode)	0xE
	ADC_PRESCALE_1280H	Sample rate is SLOWCLK/1280 (high frequency mode)	0xF

11.2.6.5 ADC_OFFSET

Bit Field	Field Name	Description
14:0	DATA	15-bit ADC signed offset

11.2.6.6 ADC_BATMON_CFG

Bit Field	Field Name	Description
23:16	ALARM_COUNT_VALUE	An Alarm Status bit gets set when SUPPLY_COUNT_VALUE= ALARM_COUNT_VALUE
15:8	SUPPLY_THRESHOLD	Low voltage detection threshold (7.8 mV steps)
0	SUPPLY_SRC	Selects the power supply voltage source to be monitored

Field Name	Value Symbol	Value Description	Hex Value
ALARM_COUNT_VALUE	BATMON_ALARM_NONE	No Alarm is triggered	0x0*
	BATMON_ALARM_COUNT1	Alarm count value is 1	0x1
	BATMON_ALARM_COUNT255	Alarm count value is 255	0xFF
SUPPLY_THRESHOLD	SUPPLY_THRESHOLD_LOW	Lowest voltage threshold: 7.8 mV	0x0
	SUPPLY_THRESHOLD_MID	Mid voltage threshold: 1 V	0x80*
	SUPPLY_THRESHOLD_HIGH	Highest voltage threshold: ~2.0 V	0xFF
SUPPLY_SRC	BATMON_CH6	Channel 6 (typically VBAT divided by 2) is monitored	0x0*
	BATMON_CH7	Channel 7 (typically VDDC) is monitored	0x1

11.2.6.7 ADC_BATMON_INT_ENABLE

Bit Field	Field Name	Description
4	BATMON_ALARM_INT_ENABLE	The BATMON Alarm interrupt mask
3:1	ADC_INT_CH_NUM	Channel number triggering the ADC interrupt
0	ADC_INT_ENABLE	The ADC new sample ready interrupt mask

Field Name	Value Symbol	Value Description	Hex Value
BATMON_ALARM_INT_ENABLE	INT_DIS_BATMON_ALARM	This source cannot set an interrupt	0x0*
	INT_EBL_BATMON_ALARM	This source can set the ADC interrupt line	0x1
ADC_INT_CH_NUM	ADC_INT_CH0	The ADC interrupt is triggered when the ADC_DATA_CH0 register is updated	0x0*
	ADC_INT_CH1	The ADC interrupt is triggered when the ADC_DATA_CH1 register is updated	0x1
	ADC_INT_CH2	The ADC interrupt is triggered when the ADC_DATA_CH2 register is updated	0x2
	ADC_INT_CH3	The ADC interrupt is triggered when the ADC_DATA_CH3 register is updated	0x3
	ADC_INT_CH4	The ADC interrupt is triggered when the ADC_DATA_CH4 register is updated	0x4
	ADC_INT_CH5	The ADC interrupt is triggered when the ADC_DATA_CH5 register is updated	0x5
	ADC_INT_CH6	The ADC interrupt is triggered when the ADC_DATA_CH6 register is updated	0x6
	ADC_INT_CH7	The ADC interrupt is triggered when the ADC_DATA_CH7 register is updated	0x7

Field Name	Value Symbol	Value Description	Hex Value
ADC_INT_ENABLE	INT_DIS_ADC	This source cannot set an interrupt	0x0*
	INT_EBL_ADC	This source can set the ADC interrupt line	0x1

11.2.6.8 ADC_BATMON_COUNT_VAL

Bit Field	Field Name	Description
7:0	SUPPLY_COUNT_VALUE	Number of times the battery voltage has fallen below the battery monitor voltage threshold. The counter is reset when read.

11.2.6.9 ADC_BATMON_STATUS

Bit Field	Field Name	Description
12	BATMON_ALARM_CLEAR	Battery monitoring alarm status bit
9	ADC_OVERRUN_CLEAR	ADC Overrun condition
8	ADC_READY_CLEAR	ADC new sample Ready status bit
4	BATMON_ALARM_STAT	Battery monitoring alarm status bit
1	ADC_OVERRUN_STAT	ADC Overrun condition
0	ADC_READY_STAT	ADC new sample Ready status bit

Field Name	Value Symbol	Value Description	Hex Value
BATMON_ALARM_CLEAR	BATMON_ALARM_CLEAR	Writing a 1 clears the BATMON Alarm status bit	0x1
ADC_OVERRUN_CLEAR	ADC_OVERRUN_CLEAR	Writing a 1 clears the ADC Overrun status bit	0x1
ADC_READY_CLEAR	ADC_READY_CLEAR	Writing a 1 clears the ADC Ready status bit	0x1
BATMON_ALARM_STAT	BATMON_ALARM_FALSE	BATMON Alarm flag not set	0x0*
	BATMON_ALARM_TRUE	BATMON Alarm has been triggered	0x1
ADC_OVERRUN_STAT	ADC_OVERRUN_FALSE	No ADC Overrun detected	0x0*
	ADC_OVERRUN_TRUE	ADC Overrun detected	0x1
ADC_READY_STAT	ADC_READY_FALSE	No new ADC samples available	0x0*
	ADC_READY_TRUE	New ADC samples are ready	0x1

11.3 GENERAL-PURPOSE I/O (GPIO) INTERFACE

RSL10 can configure any of the DIO pads as software-controlled general-purpose DIO (GPIO) pads. The function of these GPIO pads is defined by a user application, which can use them for any general-purpose input or output.

The DIO_MODE register indicates which DIO pads have been configured for GPIO functionality; bits in this register are set (for example, DIO_IS_GPIO) if they are configured as GPIOs, and cleared otherwise (for example, DIO_IS_NOT_GPIO). For any pads that are defined as a GPIO, the DIO_DIR register can be written to set the input/output direction for these pads.

The value observed at the digital input pads can be read from the `DIO_DATA` register. This value is read as 0 for all pads configured as ADC inputs (see Section 11.2, "Analog-to-Digital Converters (ADCs)"), because the digital input is not enabled in this mode. For all other modes, the physical value of the pad is directly measured. The output value for the digital output pads can also be set using the `DIO_DATA` register for any pads that are configured as GPIO outputs.

NOTE: If a DIO is configured for a GPIO mode, bit 0 of the `DIO_CFG_*` register for a DIO can be used to detect or set the data value for the GPIO (hence the `DIO_MODE_GPIO_IN_0`, `DIO_MODE_GPIO_IN_1` and `DIO_MODE_GPIO_OUT_0`, `DIO_MODE_GPIO_OUT_1` bit setting pairs).

11.3.1 GPIO Interrupts

The GPIO interface provides a set of four configurable interrupts which, when enabled, signal the occurrence of an event or a condition on a specified GPIO pad. See Section 14.1, "Nested Vectored Interrupt Controller (NVIC)" on page 400 for information regarding interrupt configuration and handling.

Each of the GPIO interrupts support triggering an interrupt from any of the 16 DIOs as an input source. Each of the GPIO interrupts also supports triggering on one of five possible GPIO events. The source and event trigger for each interrupt can be configured using one of the `DIO_INT_CFG_*` registers. For each of the four interrupts:

- To select the GPIO pad to use as a trigger for the interrupt, set the `INT_CFG_SRC` bit field.
- To select the event to use as a trigger for the interrupt, use the `INT_CFG` bit field. A list of the possible triggering events (including a description of each event) is listed in Table 23.

Table 23. GPIO Interrupt Events

Event	Event Description
Disabled	An interrupt is never generated.
High level	An interrupt is generated whenever a logical 1 is detected on the GPIO pin. Interrupts continue to be generated while this condition remains true.
Low level	An interrupt is generated whenever a logical 0 is detected on the GPIO pin. Interrupts continue to be generated while this condition remains true.
Rising Edge	An interrupt is generated when a transition from a logical 0 to a logical 1 is detected.
Falling Edge	An interrupt is generated when a transition from a logical 1 to a logical 0 is detected.
Transition	An interrupt is generated when any transition between a logical 0 and a logical 1 is detected.

11.4 I^{2}C INTERFACE

The external interfaces include an I^{2}C interface supporting both master and slave mode transfers. This interface implements a subset of the Philips I^{2}C interface as described in the *Philips I^{2}C Bus Specification* document.

The I^{2}C interface uses the rising edges of a serial clock signal (SCL) to clock in data from a serial data signal (SDA) to communicate between devices. The I^{2}C interface is designed to handle bus traffic operating at up to 1 MHz; however, for communications to proceed on the I^{2}C interface, the following relation between the system clock (which is used internally to clock the I^{2}C interface) and the SCL must hold:

$$F_{SYSCLK} \geq 3.125 \times F_{SCL}$$

The example timing diagram shown in Figure 28 provides some information about the important elements in an I^{2}C transaction, which are described in further detail in the previously mentioned bus specification. These elements are:

Start Condition

The SDA transitions from the idle high state to the low state while the SCL remains high. This can also happen during a transmission as a repeated start condition, which indicates that the transaction is starting again without an intermediate stop condition.

Address Bits

During the first byte transmitted, the first seven bits on the SDA provide the address of the device with which the master device wants to communicate. A device that is properly addressed must acknowledge this transaction if communications are to conform to the I^{2}C standard.

Read/Write Direction Bit

During the first byte transmitted, the eighth bit on the SDA indicates the direction of the transaction (except for a general call, which is always a write transaction). A zero for this bit indicates a write by the master; a one indicates a read by the master. If the master device has requested a read, the slave device controls the SDA line on subsequent bytes to transmit data.

During any transaction, the `I2C_STATUS_READ_WRITE` bit from the `I2C_STATUS` register indicates whether the current I^{2}C transaction is a read or write transaction.

Data Byte

All other bytes, excluding the addressing and read/write byte, that are transmitted on the SDA are considered data bytes for the transaction.

Acknowledge or Not Acknowledge Bit

The acknowledge bit is used to tell the sender that the byte has been received. The device receiving data should acknowledge each data byte, including the address byte. During this time, the bus device that is sending data on the data bus stops driving the SDA and allows the line to be pulled high. To not acknowledge a byte, the receiving device does not need to do anything. To acknowledge a byte, the receiving device needs to pull the SDA low.

A receiving slave device should not acknowledge a byte if the slave device is not the device that was addressed, or if the device cannot handle the byte received. A master device should not acknowledge a byte if the master is receiving and wants to end the transaction. If a not acknowledge is encountered, the master device should generate a stop condition.

Stop Condition

The SDA transitions from a low state to high state while the SCL remains high. This ends an I^{2}C transaction.

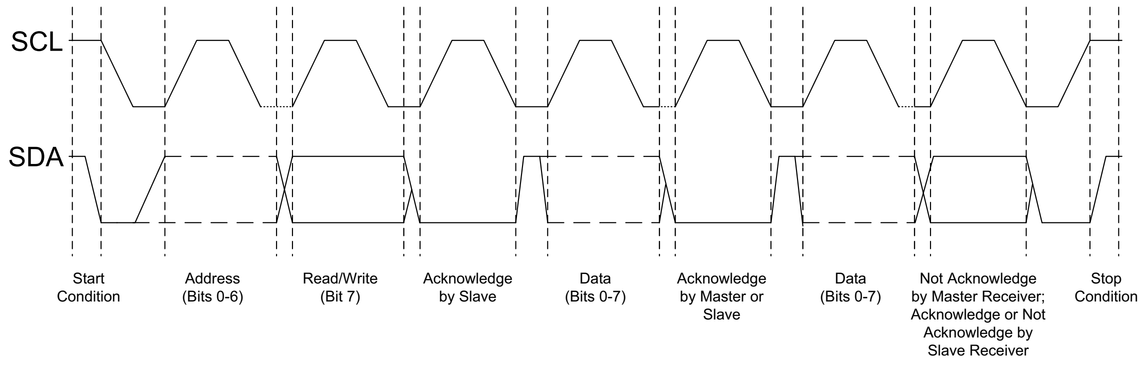


Figure 28. I^{2}C Signal Timing Diagram

For information about configuring the DIO to create an I^{2}C bus that is needed to support the I^{2}C interface, see Section 10.2, “Functional Configuration” on page 259.

IMPORTANT: I^{2}C interfaces are defined using open-collector pads for the I^{2}C bus. Because these pads do not drive a high signal for the bus signals—relying on the bus’s pull-up resistors for proper operation—user applications using this interface must use either the internal pull-up resistors implemented for the DIO pads, or a set of external pull-up resistors on the DIOs assigned to be the I^{2}C bus lines.

The `I2C_CTRL1` register is the interface control register that contains write-only control bits. Each of these bits is used to trigger an event and these bits are summarized in Table 24.

Table 24. I^{2}C Control Bits

Bit Field	Field Name	Field Description
5	`I2C_CTRL1_RESET`	Abort any current transaction and reset the I^{2}C interface.
4	`I2C_CTRL1_LAST_DATA`	Indicate that a byte is the last byte in a transaction, and will automatically terminate an I^{2}C transfer.
3	`I2C_CTRL1_STOP_CMD`	Issue a stop condition (master mode only).
1	`I2C_CTRL1_NACK`	Negative acknowledgement of a sent/received byte.
0	`I2C_CTRL1_ACK`	Acknowledgement of a sent/received byte.

CAUTION: When using the I^{2}C control bits to control a transfer in master mode with auto acknowledgements, it is not possible to use the `I2C_CTRL1_LAST_DATA` bit to terminate a transfer after reading only one byte. This transfer is not supported in this specific mode since events only occur after an acknowledgement, and the first acknowledgement by a master device using auto acknowledgements is after the first data byte. Even if a read transfer in this mode should consist of only one read byte, a second byte will be read due to the automatic acknowledgement of the first byte.

The `I2C_STATUS` register contains bit fields that are defined as either event bits or state bits. The event bits are used to indicate that an event has occurred, and the state bits are used to indicate an interface operating state. These status bits are summarized in Table 25.

Table 25. I^{2}C Event and State Status Bits

Bit Field	Field Name	Bit Type	Field Description
15	`I2C_STATUS_ERROR_S`	Event	Indicate if an error (I^{2}C bus or I^{2}C watchdog) has occurred. `I2C_STATUS_ERROR_S` is 0 when `I2C_ERROR_CNT` contains 0 and `I2C_STATUS_BUS_ERROR_S` is not 1.
14	`I2C_STATUS_BUS_ERROR_STICKY`	Event	Indicate if an I^{2}C bus error has occurred. Automatically cleared when the `I2C_STATUS` register is read, unless the bus error is still present. The bus error disappears when the I^{2}C interface detects a stop condition.
13	`I2C_STATUS_START_PENDING`	State	Indicate if an I^{2}C master transfer is pending. Transition out of this state occurs when the pending I^{2}C master transfer has started.
12	`I2C_STATUS_MASTER_MODE`	State	Indicate if the I^{2}C interface is currently operating in master mode. Transition out of this state occurs when the I^{2}C master transfer has finished, which happens when a stop condition or bus error occurs.
11	`I2C_STATUS_DMA_REQ`	Event	Indicate if the I2C interface is currently requesting a DMA access to the I2C_DATA register (read or write).
10	`I2C_STATUS_STOP_DETECTED`	Event	Indicate if a stop condition has been detected. Cleared after a new transaction starts, or after reading the `I2C_STATUS` register.
9	`I2C_STATUS_DATA_EVENT`	Event	Indicate if data has been received or is needed for transmission. Cleared after reading data from or writing data to the `I2C_DATA` register.
8	`I2C_STATUS_ERROR`	Event	Indicate if an I^{2}C error has occurred since the last stop condition or if an I^{2}C watchdog error has occurred. Cleared after reading the `I2C_ERROR_CNT` register. `I2C_STATUS_ERROR` is 0 when `I2C_ERROR_CNT` contains 0 and `I2C_STATUS_BUS_ERROR` is not 1.
7	`I2C_STATUS_BUS_ERROR`	Event	Indicate if an I^{2}C bus error has occurred since the last stop condition. Automatically cleared when the bus error disappears, which happens when the I^{2}C interface detects a stop condition.
6	`I2C_STATUS_BUFFER_FULL`	State	Indicate that the I^{2}C buffer is full. Transition out of this state occurs after reading the `I2C_DATA` register or transmitting the next byte.
5	`I2C_STATUS_CLK_STRETCH`	State	Indicate that the SCL line of the I^{2}C bus is being held due to an empty transmit buffer. Transition out of this state occurs after writing the `I2C_DATA` register.
4	`I2C_STATUS_BUS_FREE`	State	Indicate that both I^{2}C bus lines are currently high. Transition out of this state occurs when either SDA or SCL are pulled or driven low.
3	`I2C_STATUS_ADDR_DATA`	State	Indicate that an address has been sent or received. Transition out of this state occurs after a data byte has been sent or received.
2	`I2C_STATUS_READ_WRITE`	State	Indicate the state of the Read/Write direction bit last sent or received. Transition out of this state occurs after a new Read/Write direction bit is sent or received.
1	`I2C_STATUS_GEN_CALL`	State	Indicate that the last address sent or received was the general call address (0x0). Transition out of this state occurs after a non-general call address is sent or received.

IMPORTANT: We recommend using the `I2C_STATUS_ERROR_S` and `I2C_STATUS_BUS_ERROR_S` bits when testing for I^{2}C errors. Only use the `ERROR` and `BUS_ERROR` bits to check if an error condition still exists on the I^{2}C interface.

NOTE: The interface supports both master operation and slave operation. A user application can use the interface in both modes, provided the application ensures that the I^{2}C bus is not currently in use. Check the `I2C_STATUS_BUS_FREE` bit in the `I2C_STATUS` register to verify that the I^{2}C bus is not currently in use when attempting to initialize a transfer as a master device.

IMPORTANT: The RSL10 **I^{2}C interface is not fully compatible with multi-master operation on the I^{2}C bus. When** RSL10 **is operating as a master of the I^{2}C bus and there is a conflict on the I^{2}C bus where** RSL10 **loses arbitration, the** RSL10 **device continues in master mode until a stop occurs instead of immediately switching over to its slave mode as proscribed by the I^{2}C specification. Once a stop condition occurs, the RSL10 interface stops attempting to send data, and the conflicting I^{2}C master receives a NACK. Once it has received this NACK, it can restart the transmission without further conflicts from** RSL10**.**

The I^{2}C interface uses the `I2C_DATA` register to transmit and receive data through a 1-byte internal buffer. As use of the `I2C_DATA` register has side effects that can advance the I^{2}C interface state machine, this register is mirrored without side effects in the `I2C_DATA_M` register. If the internal buffer is full, the `I2C_STATUS_BUFFER_FULL` bit from the `I2C_STATUS` register is set.

Generally each byte of an I^{2}C transaction must be acknowledged or not-acknowledged by a user application. Use the `I2C_CTRL0_AUTO_ACK_ENABLE` bit from the `I2C_CTRL0` register to select whether the application itself handles this acknowledgement manually or the interface handles it automatically. In either case, if the interface does not have the needed data or a place to put data, the transaction is paused by stretching the clock signal until the interface can continue. If the clock is being stretched, the `I2C_STATUS_CLK_STRETCH` bit from the `I2C_STATUS` register is set as an indicator.

Use of the above mentioned data registers and the general behavior of the I^{2}C state machine is described in Section 11.4.3.1, "Operation Using Manual Acknowledgement" on page 316 for manual acknowledgement mode, and in Section 11.4.3.2, "Operation Using Auto Acknowledgement" on page 317 for automatic acknowledgement mode.

NOTE: If at any point, when operating in either master or slave mode, a user application needs to reset the I^{2}C bus lines (for example, after encountering a bus error condition), the application should write the `I2C_CTRL1_RESET` bit from the `I2C_CTRL1` register. This resets both the I^{2}C bus and the internal state machines of the I^{2}C interface to allow the system to cleanly start again from a known state.

11.4.1 Slave Mode Specific Configuration

In slave mode, the device is receiving the serial clock signal from an external master device, which also controls addressing, direction, and the start/stop conditions for each I^{2}C transfer.

When in slave mode, the I^{2}C interface can be held while waiting for data to be read or for new data to be available for transmission. In this case, the SCL line is held low by the slave interface - stretching the clock. The `I2C_CTRL0_SPEED` bit field in the `I2C_CTRL0` register is used to configure the settling time required for the SDA line by selecting the number of SYSCLK cycles between when data is put on the SDA line and the SCL line is released.

To enable a device in slave mode, set the `I2C_CTRL0_SLAVE_ENABLE` bit from the `I2C_CTRL0` register. This allows the device to respond to communications on both the I^2C general call address (0x00) and at a programmable slave address that can be selected using the `I2C_CTRL0_SLAVE_ADDRESS` bit field in the `I2C_CTRL0` register. During a transaction, the `I2C_STATUS_GEN_CALL` bit in the `I2C_STATUS` register can be queried to identify the addresses that were used to address the device.

A device receiving data in a slave mode configuration can use the `I2C_CTRL1_LAST_DATA` bit from the `I2C_CTRL1` register to automatically NACK the last data byte, to indicate to a master device that it should send a stop condition.

NOTE: The `I2C_CTRL1_LAST_DATA` bit is cleared when a new transfer is initiated on the I^2C bus.

11.4.2 Master Mode Specific Configuration

In master mode, the device provides the serial clock signal and controls all transfer information. To configure the clock signal, select a prescaling division by 3 using the `I2C_CTRL0_SPEED` bit field in the `I2C_CTRL0` register. The interface clock frequency for a given configuration can be calculated using:

$$f_{\text{I2C MASTER}} = \frac{f_{\text{SYSCLK}}}{3 \cdot (\text{I2C\_CTRL0\_SPEED} + 1)}$$

A user application must manually control all components of a master mode transaction. To transmit or receive using master mode, a device must:

1. Check that the I^2C bus lines are not currently in use by reading the `I2C_STATUS_BUS_FREE` bit in the `I2C_STATUS` register. The device can only start a transaction if the lines are free.
2. Start a transaction, sending the start condition, address and direction, by loading the appropriate address and direction to the `I2C_ADDR_START` register.
3. Handle data and interrupts as appropriate for the transaction (see Section 11.4.3, “I2C Interrupts”).
4. Complete the transaction by sending a stop condition.

   A stop condition can be generated using either the `I2C_CTRL1_STOP_CMD` bit or the `I2C_CTRL1_LAST_DATA` bit in the `I2C_CTRL1` register.

   When using the `I2C_CTRL1_LAST_DATA` bit, a stop condition is automatically generated after the current data byte transfer is completed. If the interface is receiving data, this bit also automatically generates the required NACK of the last data byte that is received.

NOTE: If the I^2C interface is used for transmitting, the last byte of data to be transmitted must be written to the `I2C_DATA` register before setting the `I2C_CTRL1_LAST_DATA` bit, as this bit triggers a stop condition at the end of the current transfer.

NOTE: As previously noted, the `I2C_CTRL1_LAST_DATA` bit is cleared when a new transfer is initiated on the I^2C bus.

When using the `I2C_CTRL1_STOP_CMD` bit, a stop condition is issued immediately. This bit is normally set when the `I2C_CTRL1_ACK` or `I2C_CTRL1_NACK` bits in the `I2C_CTRL1` register are set.

CAUTION: Using the stop event (STOP) immediately transmits a stop condition as soon as possible. This can result in a terminated transfer, and remote devices on the bus might detect I^{2}C errors.

11.4.3 I^{2}C Interrupts

The I^{2}C interface uses an associated interrupt which, when enabled, signals the receipt of a correct address byte and the completion of each data byte in the transaction.

When enabled, the I^{2}C interrupt signals the stop condition following a transaction for a master transfer. The system can also be configured to receive this interrupt when operating in slave mode by setting the `I2C_CTRL0_STOP_INT_ENABLE` bit from the `I2C_CTRL0` register.

Section 14.1, "Nested Vectored Interrupt Controller (NVIC)" on page 400 for information regarding interrupt configuration and handling.

Several status indicators from the `I2C_STATUS` register can be used with the interrupt to identify the state of the I^{2}C interface:

- If the `I2C_STATUS_ADDR_DATA` bit is set, the interrupt was generated in response to a recognized sequence including a start condition and address on the I^{2}C interface.
- If the `I2C_STATUS_STOP_DETECTED` bit is set, the interrupt was generated in response to a stop interrupt being detected on the I^{2}C interface. This bit is cleared immediately when a new transaction starts. No interrupt is generated for I^{2}C transactions that use an address that was not recognized by the I^{2}C interface.
- If the `I2C_STATUS_DATA_EVENT` bit is set, the interrupt indicates that data has been received and can be read, or that data is needed to continue with the transmission of data.
- If the `I2C_STATUS_BUS_ERROR_S` bit is set, then a bus error has occurred. If the `I2C_STATUS_BUS_ERROR` bit is also set, the bus error has not yet been cleared.
- If the `I2C_STATUS_ERROR_S` bit is set, then either a bus error or an I^{2}C watchdog timeout has occurred. If the `I2C_STATUS_ERROR` bit is also set, the error has not yet been cleared.

11.4.3.1 Operation Using Manual Acknowledgement

If a user application is using manual acknowledgement (i.e., the `I2C_CTRL0_AUTO_ACK_ENABLE` bit from the `I2C_CTRL0` register is disabled), it must use the `I2C_CTRL1_NACK` and `I2C_CTRL1_ACK` bits from the `I2C_CTRL1` register to acknowledge (ACK) or not-acknowledge (NACK) transaction bytes.

When handling a transaction, an interrupt is issued whenever data or an acknowledgement is required (in addition to interrupts triggered by stop conditions).

NOTE: An interrupt is not generated following a NACK for any data condition. However, an interrupt is generated (if operating in master mode or stop interrupts are enabled) on the stop condition that is expected to follow the NACK.

An interrupt is also generated if the interface cannot send data that has been provided because the interface is idle or has already received a NACK from the slave device during the current transaction. As noted above, when the interface is waiting for data, the clock line is automatically stretched until the required data is provided.

A special interrupt is also generated for a master read once the acknowledge bit from the address byte has been received. This scenario is a special case because it does not require data nor acknowledgement to trigger an interrupt, but instead is designed to allow the interface to check whether or not a slave device has acknowledged the transfer at that point. Provided the slave device has responded, the interface can issue either an ACK or a NACK command to continue the transfer.

NOTE: Use of the `I2C_CTRL1_NACK`, `I2C_CTRL1_ACK` and `I2C_CTRL1_STOP_CMD` bits from the `I2C_CTRL1` register is only defined when an acknowledgement is needed during a data transfer. Interface behavior in response to these control bits being set at other times is undefined.

11.4.3.2 Operation Using Auto Acknowledgement

When operating using auto acknowledgement (i.e., the `I2C_CTRL0_AUTO_ACK_ENABLE` bit from the `I2C_CTRL0` register is enabled), transfers are controlled by the availability of bytes for transmission, or a place to put a byte when receiving data. This allows transfers to occur without clock stretching, provided the firmware is able to handle each data byte fast enough.

When the interface handles a transaction, interrupts are issued whenever new data can be written or new data has been received. This allows a user application to maintain a transaction without introducing delay.

NOTE: An interrupt is not generated following a NACK for any data condition. However, an interrupt is generated (if operating in master mode or stop interrupts are enabled) on the stop condition that is expected to follow the NACK.

Additionally, interrupts are generated if data was supplied but cannot be sent because the interface has received a NACK from the slave device, or is currently idle (as the interface would be following a stop condition). If data is required but is not currently available because the buffer is empty, the clock is stretched and an additional interrupt is generated. If new data was received, but the previous data has not been read (resulting in the buffer not being available), the clock is stretched until the buffer becomes free.

IMPORTANT: When receiving data in master mode, the timing of interrupts for auto acknowledgement of data prevents a user application from cleanly terminating a transfer using the LAST_DATA event. As a result, if the interface is running in this mode, a dummy byte must be transferred to cleanly terminate the current transfer with a stop condition following this byte.

11.4.4 I^{2}C Registers

Register Name	Register Description	Address
`I2C_CTRL0`	I2C Interface Configuration and Control	0x40000B00
`I2C_CTRL1`	I2C Interface Status and Control	0x40000B04
`I2C_DATA`	I2C Interface Data	0x40000B08
`I2C_DATA_M`	I2C Interface Data (Mirror)	0x40000B0C
`I2C_ADDR_START`	I2C Master Address and Start	0x40000B10
`I2C_STATUS`	I2C Status	0x40000B14

11.4.4.1 I2C_CTRL0

Bit Field	Field Name	Description
23:16	`SPEED`	In I2C interface master mode this prescaler is used to divide SYSCLK to the correct SCL frequency. SCL is prescaled by (SPEED + 1) * 3. In I2C slave mode, if clock stretching is needed, this setting controls the number of SYSCLK cycles between when data is put on the SDA line and the SCL line is released.
14:8	`SLAVE_ADDRESS`	Set the I2C slave address for this device
4	`CONTROLLER`	Select whether data transfer will be controlled by the CM3 or the DMA for I2C
3	`STOP_INT_ENABLE`	Configure whether stop interrupts will be generated by the I2C interface
2	`AUTO_ACK_ENABLE`	Select whether acknowledgement is automatically generated or not
1	`I2C_SAMPLE_CLK_ENABLE`	Enable/disable the I2C sample clock (mandatory to enable the I2C)
0	`SLAVE_ENABLE`	Select whether the I2C interface will be enabled for slave mode or not

Field Name	Value Symbol	Value Description	Hex Value
SPEED	I2C_MASTER_SPEED_3	Master mode: prescale SCL from SYSCLK by 3.	0x0*
	I2C_MASTER_SPEED_6	Master mode: prescale SCL from SYSCLK by 6.	0x1
	I2C_MASTER_SPEED_9	Master mode: prescale SCL from SYSCLK by 9.	0x2
	I2C_MASTER_SPEED_12	Master mode: prescale SCL from SYSCLK by 12.	0x3
	I2C_MASTER_SPEED_15	Master mode: prescale SCL from SYSCLK by 15.	0x4
	I2C_MASTER_SPEED_18	Master mode: prescale SCL from SYSCLK by 18.	0x5
	I2C_MASTER_SPEED_21	Master mode: prescale SCL from SYSCLK by 21.	0x6
	I2C_MASTER_SPEED_24	Master mode: prescale SCL from SYSCLK by 24.	0x7
	I2C_MASTER_SPEED_27	Master mode: prescale SCL from SYSCLK by 27.	0x8
	I2C_MASTER_SPEED_30	Master mode: prescale SCL from SYSCLK by 30.	0x9
	I2C_MASTER_SPEED_33	Master mode: prescale SCL from SYSCLK by 33.	0xA
	I2C_MASTER_SPEED_36	Master mode: prescale SCL from SYSCLK by 36.	0xB
	I2C_MASTER_SPEED_39	Master mode: prescale SCL from SYSCLK by 39.	0xC
	I2C_MASTER_SPEED_42	Master mode: prescale SCL from SYSCLK by 42.	0xD
	I2C_MASTER_SPEED_45	Master mode: prescale SCL from SYSCLK by 45.	0xE
	I2C_MASTER_SPEED_48	Master mode: prescale SCL from SYSCLK by 48.	0xF
	I2C_MASTER_SPEED_51	Master mode: prescale SCL from SYSCLK by 51.	0x10
	I2C_MASTER_SPEED_54	Master mode: prescale SCL from SYSCLK by 54.	0x11
	I2C_MASTER_SPEED_57	Master mode: prescale SCL from SYSCLK by 57.	0x12
	I2C_MASTER_SPEED_60	Master mode: prescale SCL from SYSCLK by 60.	0x13
	I2C_MASTER_SPEED_120	Master mode: prescale SCL from SYSCLK by 120.	0x27
	I2C_MASTER_SPEED_768	Master mode: prescale SCL from SYSCLK by 768. Slave Mode: 255 SYSCLK period between SDA updated and SCL un-stretched	0xFF

Field Name	Value Symbol	Value Description	Hex Value
SPEED (continued)	I2C_SLAVE_SPEED_1	Slave Standard-mode: at least 250 ns +10% data set-up time with SYSCLK = 3 MHz; Slave Fast-mode: at least 100 ns +10% data set-up time with SYSCLK = 3, 4, 5, 8 MHz; Slave Fast-mode Plus: at least 50 ns +10% data set-up time with SYSCLK = 3, 4, 5, 8, 10, 12, 16 MHz	0x0*
	I2C_SLAVE_SPEED_2	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 4, 5 MHz; Slave Fast-Mode: at least 100 ns +10% data set-up time with SYSCLK = 10, 12, 16 MHz; Slave Fast-mode Plus: at least 50 ns +10% data set-up time with SYSCLK = 20, 24 MHz	0x1
	I2C_SLAVE_SPEED_3	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 8, 10 MHz; Slave Fast-Mode: at least 100 ns +10% data set-up time with SYSCLK = 20, 24 MHz; Slave Fast-mode Plus: at least 50 ns +10% data set-up time with SYSCLK = 48 MHz	0x2
	I2C_SLAVE_SPEED_4	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 12 MHz	0x3
	I2C_SLAVE_SPEED_5	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 16 MHz	0x4
	I2C_SLAVE_SPEED_6	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 20 MHz; Slave Fast-Mode: at least 100 ns +10% data set-up time with SYSCLK = 48 MHz	0x5
	I2C_SLAVE_SPEED_7	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 24 MHz	0x6
	I2C_SLAVE_SPEED_14	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 48 MHz	0xD
CONTROLLER	I2C_CONTROLLER_CM3	The CM3 controls data transfers using I2C	0x0*
	I2C_CONTROLLER_DMA	The DMA controls data transfers using I2C	0x1
STOP_INT_ENABLE	I2C_STOP_INT_DISABLE	Stop interrupts will not be generated	0x0*
	I2C_STOP_INT_ENABLE	A stop interrupt will be generated when a stop condition occurs for an active transaction	0x1
AUTO_ACK_ENABLE	I2C_AUTO_ACK_DISABLE	Require manual acknowledgement of all I2C interface transfers	0x0*
	I2C_AUTO_ACK_ENABLE	Use automatic acknowledgement for I2C interface transfers	0x1

Field Name	Value Symbol	Value Description	Hex Value
I2C_SAMPLE_CLK_ENABLE	I2C_SAMPLE_CLK_DISABLE	Disable the I2C sample clock (I2C is disabled)	0x0*
	I2C_SAMPLE_CLK_ENABLE	Enable the I2C sample clock (I2C is enabled)	0x1
SLAVE_ENABLE	I2C_SLAVE_DISABLE	Disable I2C interface slave mode operation	0x0*
	I2C_SLAVE_ENABLE	Enable I2C interface slave mode operation	0x1

11.4.4.2 I2C_CTRL1

Bit Field	Field Name	Description
5	RESET	Reset the I2C interface
4	LAST_DATA	Indicate that the current data is the last byte of a data transfer
3	STOP	Issue a stop condition on the I2C interface bus
1	NACK	Issue a not acknowledge on the I2C interface bus
0	ACK	Issue an acknowledge on the I2C interface bus

Field Name	Value Symbol	Value Description	Hex Value
RESET	I2C_RESET	Reset the I2C interface	0x1
LAST_DATA	I2C_LAST_DATA	Indicate that the current data is the last byte of a data transfer	0x1
STOP	I2C_STOP	Issue a stop condition on the I2C interface bus	0x1
NACK	I2C_NACK	Issue a not acknowledge on the I2C interface bus	0x1
ACK	I2C_ACK	Issue an acknowledge on the I2C interface bus	0x1

11.4.4.3 I2C_DATA

Bit Field	Field Name	Description
7:0	I2C_DATA	Single byte buffer for data transmitted and received over the I2C interface

11.4.4.4 I2C_DATA_M

Bit Field	Field Name	Description
7:0	I2C_DATA_M	Mirror of the single byte buffer for data transmitted and received over the I2C interface

11.4.4.5 I2C_ADDR_START

Bit Field	Field Name	Description
7:1	ADDRESS	I2C address to use for the transaction
0	READ_WRITE	Select whether a read or a write transaction is started

Field Name	Value Symbol	Value Description	Hex Value
READ_WRITE	I2C_START_WRITE	Start an I2C write transaction	0x0
	I2C_START_READ	Start an I2C read transaction	0x1

11.4.4.6 I2C_STATUS

Bit Field	Field Name	Description
15	ERROR_S	Error status bit (sticky) - This status bit is automatically reset when the I2C_STATUS register is read.
14	BUS_ERROR_S	Bus error status bit (sticky) - This status bit is automatically reset when the I2C_STATUS register is read.
13	START_PENDING	Master frame start pending status bit
12	MASTER_MODE	Master mode status bit
11	DMA_REQ	Indicate if the I2C interface is currently requesting DMA data
10	STOP_DETECT	Indicate if an I2C stop bit has been detected
9	DATA_EVENT	Indicate that the I2C interface either needs data to transmit or has received data
8	ERROR	Indicate if the I2C interface has detected any error (automatically cleared when a stop condition is detected)
7	BUS_ERROR	Indicate if the I2C interface has detected a bus error (automatically cleared when a stop condition is detected)
6	BUFFER_FULL	Indicate if the I2C data buffer is full
5	CLK_STRETCH	Indicate if the I2C interface is holding the clock signal
4	BUS_FREE	Indicate if the I2C interface bus is free
3	ADDR_DATA	Indicate if the I2C data register holds an address or data byte
2	READ_WRITE	Indicate whether the I2C bus transfer is a read or a write
1	GEN_CALL	Indicate whether the I2C bus transfer is using the general call address or another address
0	ACK_STATUS	Indicate whether an acknowledge or a not acknowledge has been received

Field Name	Value Symbol	Value Description	Hex Value
ERROR_S	I2C_NO_ERROR_S	I2C interface is not and has not been in an error state	0x0*
	I2C_ERROR_S	I2C interface is or has been in an error state	0x1
BUS_ERROR_S	I2C_NO_BUS_ERROR_S	I2C interface is not and has not been in the bus error state	0x0*
	I2C_BUS_ERROR_S	I2C interface is or has been in the bus error state	0x1

Field Name	Value Symbol	Value Description	Hex Value
START_PENDING	I2C_START_NOT_PENDING	No pending master start frame	0x0*
	I2C_START_PENDING	A master frame is pending to start (bit is set when I2C_ADDR_START is written)	0x1
MASTER_MODE	I2C_MASTER_INACTIVE	I2C interface is not operating in master mode	0x0*
	I2C_MASTER_ACTIVE	I2C interface is operating in master mode	0x1
DMA_REQ	I2C_NO_DMA_REQUEST	The I2C interface is not requesting a DMA action	0x0*
	I2C_DMA_REQUEST	The I2C interface is requesting a DMA action	0x1
STOP_DETECT	I2C_NO_STOP_DETECTED	No stop condition has been detected on the I2C bus	0x0*
	I2C_STOP_DETECTED	A stop condition has been detected on the I2C bus	0x1
DATA_EVENT	I2C_NON_DATA_EVENT	No I2C data is needed or available	0x0*
	I2C_DATA_EVENT	I2C data is needed or is available	0x1
ERROR	I2C_NO_ERROR	I2C interface is not in an error state	0x0*
	I2C_ERROR	I2C interface is in an error state	0x1
BUS_ERROR	I2C_NO_BUS_ERROR	I2C interface is not in the bus error state	0x0*
	I2C_BUS_ERROR	I2C interface is in the bus error state	0x1
BUFFER_FULL	I2C_BUFFER_EMPTY	The I2C interface buffer is empty	0x0*
	I2C_BUFFER_FULL	The I2C interface buffer is full	0x1
CLK_STRETCH	I2C_CLK_NOT_STRETCHED	The I2C clock line is not being stretched	0x0*
	I2C_CLK_STRETCHED	The I2C SCL line is being held low	0x1
BUS_FREE	I2C_BUS_BUSY	One or both of the I2C bus lines is currently 0	0x0
	I2C_BUS_FREE	Both I2C bus lines are currently free	0x1*
ADDR_DATA	I2C_DATA_IS_DATA	The I2C data register holds data	0x0*
	I2C_DATA_IS_ADDR	The I2C data register holds an address	0x1
READ_WRITE	I2C_IS_WRITE	The current I2C transfer is a write	0x0*
	I2C_IS_READ	The current I2C transfer is a read	0x1
GEN_CALL	I2C_ADDR_OTHER	The address used for the current I2C transfer is not the general call address	0x0*
	I2C_ADDR_GEN_CALL	The address used for the current I2C transfer is the general call address	0x1
ACK_STATUS	I2C_HAS_ACK	Indicate that the last I2C byte was acknowledged	0x0*
	I2C_HAS_NACK	Indicate that the last I2C byte was not acknowledged	0x1

11.5 Pulse Code Modulation (PCM) Interface

RSL10 has access to a highly configurable pulse code modulation (PCM) interface that can be used to stream control, configuration or signal data into and out of the microcontroller.

The PCM interface is multiplexed onto the DIO pads, which can be configured as the input and output signals that form the PCM interface with the necessary physical pad configuration. For more information about configuring the multiplexed DIO functionality, see Chapter 10, "Digital Input/Output" on page 259.

This interface makes use of four external signals in communications. These signals are:

- An input interface clock signal (PCM_CLK)
- A bidirectional frame signal (PCM_FRAME)
- A serial output data signal (PCM_SERO)
- A serial input data signal (PCM_SERI)

CAUTION: Disabling the bus pull-up and pull-down resistors is not recommended for the PCM frame or PCM clock inputs. If these pads are used as outputs, you can disable the pull-up resistors. For all other configurations, attempting to use this interface without the pull resistors can result in unintended interface behavior that would result in the PCM interface transmitting and/or receiving undefined data.

The PCM interface uses the `PCM_CTRL_ENABLE` bit in the `PCM_CTRL` register to enable and disable the PCM interface entirely. When using the PCM interface in any mode, enable this bit and ensure that the proper DIO multiplexing has been selected.

The PCM interface can be configured as a either a master device (controlling the frame signal, and potentially providing the interface PCM_CLK through an internally routed DIO function), or as a slave device (receiving the frame signal and usually not providing the PCM_CLK). To select between these configurations, configure the `PCM_CTRL_SLAVE` bit as appropriate in the `PCM_CTRL` register. In master mode, configure the PCM frame signal as an output; in slave mode, configure it as an input. The PCM clock signal is always an input to the PCM interface; this clock signal can be sourced:

- Internally, using the DIO output mode for the pad used for the PCM clock to route an internal clock signal within the RSL10 system to the same pad
- Externally, using an externally generated clock signal

One of the advantages of this interface is the high communication speed that allows the interface to operate at the system clock frequency. PCM_CLK can operate at any division of SYSCLK, up to the same rate as SYSCLK.

The PCM interface is sensitive to only one edge of the PCM clock. All settings are clocked in and signal updates are captured on the rising or falling edge as specified by the `PCM_CLK_POL` bit from the `PCM_CTRL` register. By default, the PCM interface is only sensitive to falling edges of the PCM clock. For I^2S standard compatibility, we recommend configuring the interface to sample on the rising edges of this clock when using this interface in I^2S mode.

Either the Arm Cortex-M3 processor or the DMA can control the data transferred by the PCM interface. Select the controller by configuring the `PCM_CTRL_CONTROLLER` bit in the `PCM_CTRL` register.

IMPORTANT: The PCM interface configuration and internal status registers are tightly synchronized to the input PCM clock. As a result, the PCM interface is only reset and the internal configuration of the PCM interface is only updated on a PCM clock edge.

11.5.1 PCM Signal Configuration

The PCM interface uses a wide number of configuration options to help to define a set of signals that the system can interpret. For the purpose of clarifying the following explanations, the signals generated by the system in the default PCM configuration are shown in Figure 29.

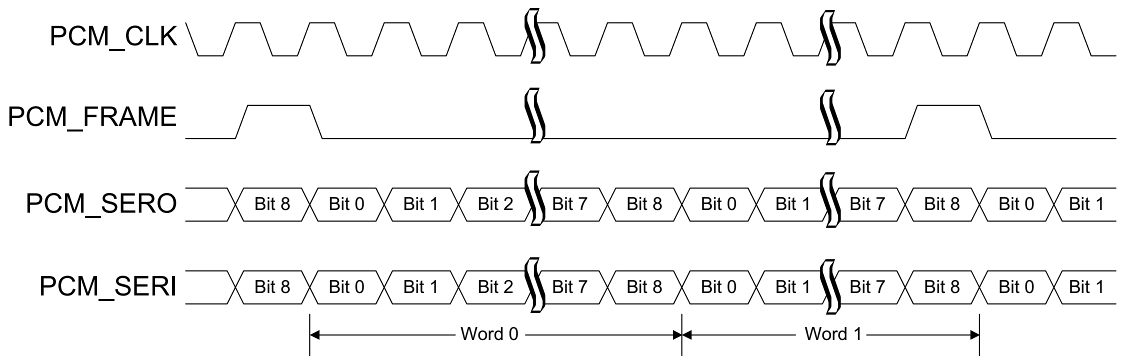


Figure 29. Signal Timing Diagram: Default PCM Configuration

For more information about the required signal configuration and other interface configuration that is needed to allow the interface to be compatible with I^{2}S, including an example timing diagram, see Section 11.5.3, “I2S Configuration and Usage”.

IMPORTANT: If the PCM interface is not idle when changing any configuration settings, changing the PCM configuration during a transaction results in undefined behavior on the associated PCM signal lines.

11.5.1.1 Frame Signal Configuration and Timing

The PCM interface uses a signal, called a frame signal, to realign transmission over the PCM interface between two devices with every data frame transmitted.

The frame signal divides the communications on the PCM interface into data frames (or sub-frames, if the `FRAME_SUBFRAME`S bit of the `PCM_CTRL` register is enabled). Each aspect of the PCM frame signal can be configured as follows (all bits are in the `PCM_CTRL` register):

Interval In the default configuration of the `FRAME_LENGTH` bit field (configured for two words per frame), and the `FRAME_SUBFRAMES` bit (configured to disable a frame signal being generated for each word within a frame), the frame signal is generated once for every two-word frame. Changing the `FRAME_LENGTH` bit field in PCM master mode allows the user to select the number of words per frame, incrementing in multiples of two words, and hence the number of words per data frame. The `FRAME_LENGTH` bit field is not used for PCM slave mode; it should be left in the default configuration (configured for two words per frame), regardless of the actual frame length used, to get the expected behavior. Alternately, if the `FRAME_SUBFRAMES` bit is configured to enable subframes, a frame signal is generated with every word of each data frame.

IMPORTANT: Configuring the `FRAME_LENGTH` bit field to something other than two words per frame while setting the `FRAME_SUBFRAMES` bit, results in a configuration where the user receives a PCM frame signal for every subframe word. Despite the well-defined behavior of the frame signal generated for this configuration, the handling of data and interrupts by the PCM interface might not appear sensible and generally results in undefined or unexpected data.

Shape The frame signal can be configured to take one of two shapes by setting the `FRAME_WIDTH` bit. By default, the frame signal is configured to produce a short width frame signal which produces a single cycle pulse in the frame signal at the beginning of each frame. Alternately, you can configure the signal to produce a long width frame signal which uses a 50% duty cycle square wave and is spaced to align with the specified width between frame signal events.

Alignment You can configure the PCM frame signal by using the `FRAME_ALIGN` bit so that the first bit of sampled data is sampled at the same time as the frame signal, or the first bit of data is sampled one cycle after the frame signal. In this way, the frame signal is aligned with the first bit of the new data frame, or with the last bit of the previous data frame.

NOTE: Regardless of the PCM frame signal alignment configuration, the PCM frame signal occurs at the beginning of each frame, and only one frame signal occurs for each frame received. The PCM frame signal, if aligned with the last bit, occurs one bit before the first data bit transmitted at the start of a transaction, and does not occur on the last bit of the last frame of a transaction. Similarly, if aligned with the first bit, the first frame signal is aligned with the first data bit transmitted at the start of a transaction, and no trailing frame signal occurs for a transaction.

IMPORTANT: To properly terminate a transmission, an expected frame signal pulse must be missed. When the frame signal is configured to be aligned with the first bit of a transmission, this requires one additional PCM clock pulse following the transmission of the last bit of the last frame.

For clarification of the above PCM frame signal timing configuration explanations, several examples of different PCM frame signal traces are shown in Figure 30. In all cases the `FRAME_ALIGN` bit has been set to `PCM_FRAME_ALIGN_FIRST_BIT` (which differs from the default configuration shown in Figure 29 on page 325) to prevent any confusion as to what are the effects of the other frame signal configuration bits. The frame signal configurations shown are:

1. Configured to indicate the subframe words of a frame with a short pulse frame signal
2. Configured to indicate the subframe words of a frame with a long width frame signal
3. Configured to use a 2-word frame with a short pulse frame signal
4. Configured to use a 2-word frame with a long width frame signal
5. Configured to use a 4-word frame with a short pulse frame signal
6. Configured to use a 4-word frame with a long width frame signal

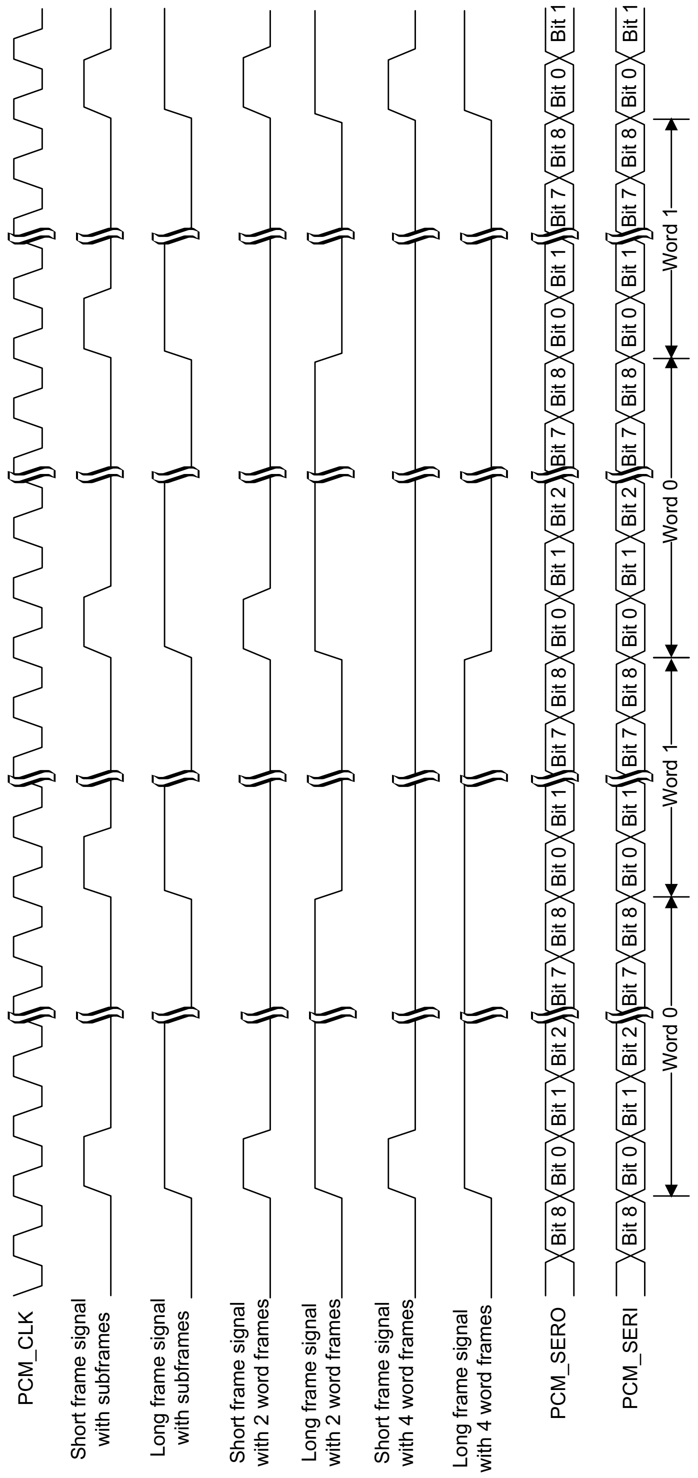

PCM_CLK
Short frame signal with subframes
Long frame signal with subframes
Short frame signal with 2 word frames
Long frame signal with 2 word frames
Short frame signal with 4 word frames
Long frame signal with 4 word frames
PCM_SERO
PCM_SERI
Bit 8
Bit 0
Bit 1
Bit 2
Bit 7
Bit 8
Bit 0
Bit 1
Bit 7
Bit 8
Bit 0
Bit 1
Bit 2
Bit 7
Bit 8
Bit 0
Bit 1
Bit 7
Bit 8
Bit 0
Bit 1
Word 0
Word 1
Word 0
Word 1

Figure 30. Frame Signal Timing Examples

11.5.1.2 Data Serial Input and Output Configuration

The PCM interface allows data bits to be configured for transmission and reception in either an MSB to LSB ordering, or an LSB to MSB ordering. To select between these two configurations, set the BIT_ORDER bit in the PCM_CTRL register.

The length of each word of a PCM transaction can be set to be between one and four bytes (8 and 32 bits) per word by setting the WORD_SIZE bit field in the PCM_CTRL register. When transmitting data that uses less than 32 bits per word, the TX_ALIGN bit from the PCM_CTRL register configures whether the data is loaded from the most significant portion of the data register or from the least significant portion of the data register (both configurations are equivalent when using 32-bit data words).

11.5.2 PCM Interrupt Configuration

The PCM interface uses three associated interrupts that control transmission and reception of PCM data, and report errors that have occurred while transmitting or receiving PCM data. Section 14.1, “Nested Vectored Interrupt Controller (NVIC)” on page 400 for information regarding interrupt configuration and handling.

If a user application is transmitting data from the PCM interface, the PCM_TX interrupt signals:

- When the PCM_TX_DATA register value starts to be transmitted using the PCM interface
- That the application can now load into the PCM_TX_DATA register the next data word (of the specified size) to be transmitted

If a user application is receiving data from the PCM interface, the PCM_RX interrupt signals that a data word of the specified size was successfully received and written to the PCM_RX_DATA register.

The PCM transmit and receive interrupts are generated for every word of data transmitted in a valid PCM frame regardless of the length of the PCM frame.

When the DMA is used to control data transfers over the PCM interface, the PCM_ERROR interrupt signals that either:

- An overrun occurred when the DMA could not read the received data from the PCM_RX_DATA register to the DMA buffer before this data was overwritten.
- An underrun occurred when the DMA could not write the PCM_TX_DATA register between loads of the internal PCM transmission registers.

11.5.3 I^{2}S Configuration and Usage

The PCM interface can be configured to be compatible with the I^{2}S interface standard. Table 26 lists the required signal and data management settings for the PCM interface to enable I^{2}S communication.

Table 26. Required Configuration Settings for I^{2}S Configuration

Configuration Bit or Bit Field	Setting
PCM_CTRL_FRAME_LENGTH	PCM_MULTIWORD_2
PCM_CTRL_FRAME_WIDTH	PCM_FRAME_WIDTH_LONG
PCM_CTRL_FRAME_ALIGN	PCM_FRAME_ALIGN_LAST
PCM_CTRL_BIT_ORDER	PCM_BIT_ORDER_MSB_FIRST

Table 26. Required Configuration Settings for I^{2}S Configuration (Continued)

Configuration Bit or Bit Field	Setting
PCM_CTRL_FRAME_SUBFRAMES	PCM_SUBFRAME_DISABLE
PCM_CTRL_PCM_CLK_POL	PCM_SAMPLE_RISING_EDGE[1]

1. The PCM interface produces a data stream that complies with the I^{2}S standard with either sampling edge. However, to comply completely with the standard, the interface must be configured to sample data on the rising edge of the clock.

In master mode, the I^{2}S configuration for the PCM interface has no special interface behaviors.

In slave mode, the behavior of the PCM interface is slightly different from any other configuration. When configured for slave mode and a long frame width, the PCM interface synchronizes communications with both the rising edge and falling edge of the frame signal (instead of synchronizing with only the rising edge as it does in all other configurations). As shown in the I^{2}S signal timing diagram (Figure 31), after transmitting the LSB of the current data word the device repeatedly transmits the MSB of the next word until a rising or falling edge on the frame signal is detected. Similarly, if a rising or falling edge is detected on the frame signal before the current data word has been completely transmitted, the current data word is truncated and the next data word is sent. In this way, the interface is compatible with I^{2}S signals that transmit a different number of bits than the defined word size.

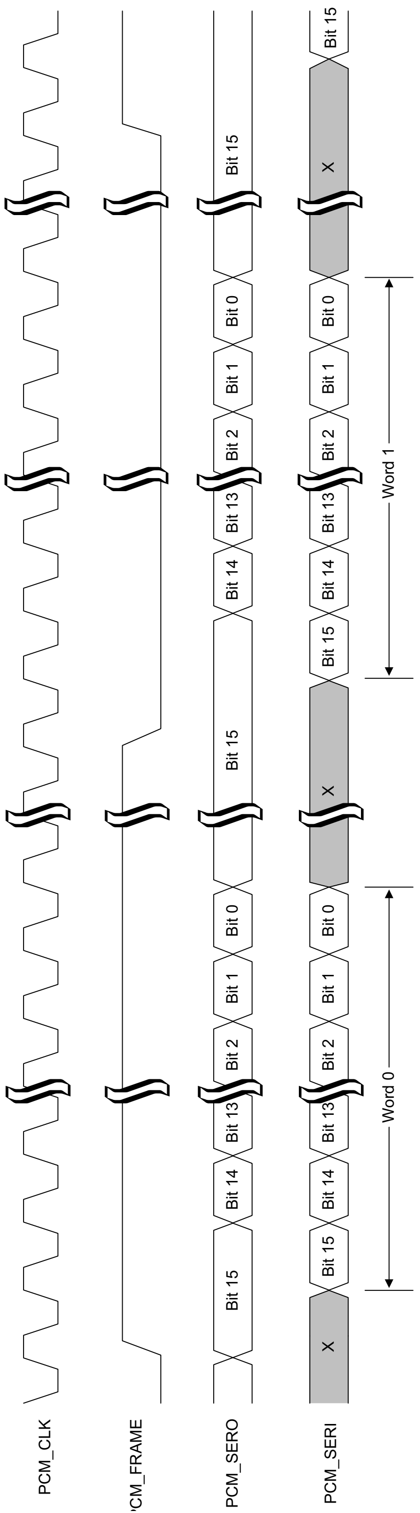


Figure 31. Signal Timing Diagram: I^{2}S Configuration

11.5.4 PCM Registers

Register Name	Register Description	Address
PCM_CTRL	PCM Control	0x40000A00
PCM_TX_DATA	PCM Transmit Data	0x40000A04
PCM_RX_DATA	PCM Receive Data	0x40000A08
PCM_STATUS	PCM Status	0x40000A0C

11.5.4.1 PCM_CTRL

Bit Field	Field Name	Description
14	PCM_CLK_POL	PCM clock polarity
12	BIT_ORDER	Select whether the data will be transmitted starting with the MSB or LSB
11	TX_ALIGN	Select what bits to use for transmitting data
10:9	WORD_SIZE	Select the number of bits per PCM word
8	FRAME_ALIGN	Align the PCM frame signal to the first/last bit
7	FRAME_WIDTH	Use a long/short PCM frame signal
6:4	FRAME_LENGTH	Select the number of words per PCM frame
3	FRAME_SUBFRAMES	Select between a frame signal for every word or one per frame
2	CONTROLLER	Select whether data transfer will be controlled by the Arm Cortex-M3 core or the DMA for PCM
1	ENABLE	Enable/disable the PCM interface
0	SLAVE	Use the PCM interface as a master/slave

Field Name	Value Symbol	Value Description	Hex Value
PCM_CLK_POL	PCM_SAMPLE_FALLING_EDGE	PCM input data sampled on PCM_CLK falling edge	0x0*
	PCM_SAMPLE_RISING_EDGE	PCM input data sampled on PCM_CLK rising edge	0x1
BIT_ORDER	PCM_BIT_ORDER_MSB_FIRST	PCM data is ordered from MSB to LSB	0x0*
	PCM_BIT_ORDER_LSB_FIRST	PCM data is ordered from LSB to MSB	0x1
TX_ALIGN	PCM_TX_ALIGN_MSB	Use MSBs of transmit buffer when word size is less that 32 bits	0x0*
	PCM_TX_ALIGN_LSB	Use LSBs of transmit buffer when word size is less that 32 bits	0x1
WORD_SIZE	PCM_WORD_SIZE_8	Use 8-bits words	0x0
	PCM_WORD_SIZE_16	Use 16-bits words	0x1
	PCM_WORD_SIZE_24	Use 24-bits words	0x2
	PCM_WORD_SIZE_32	Use 32-bits words	0x3*
FRAME_ALIGN	PCM_FRAME_ALIGN_LAST	Align the PCM frame signal to the last bit of the frame	0x0*
	PCM_FRAME_ALIGN_FIRST	Align the PCM frame signal to the first bit of the frame	0x1

Field Name	Value Symbol	Value Description	Hex Value
FRAME_WIDTH	PCM_FRAME_WIDTH_SHORT	The PCM frame is high for one PCM clock period	0x0*
	PCM_FRAME_WIDTH_LONG	The PCM frame is high for half of the frame length	0x1
FRAME_LENGTH	PCM_MULTIWORD_2	PCM frames contain 2 words	0x0*
	PCM_MULTIWORD_4	PCM frames contain 4 words	0x1
	PCM_MULTIWORD_6	PCM frames contain 6 words	0x2
	PCM_MULTIWORD_8	PCM frames contain 8 words	0x3
	PCM_MULTIWORD_10	PCM frames contain 10 words	0x4
	PCM_MULTIWORD_12	PCM frames contain 12 words	0x5
	PCM_MULTIWORD_14	PCM frames contain 14 words	0x6
	PCM_MULTIWORD_16	PCM frames contain 16 words	0x7
FRAME_SUBFRAMES	PCM_SUBFRAME_DISABLE	Generate a frame signal every frame	0x0*
	PCM_SUBFRAME_ENABLE	Generate a frame signal every word	0x1
CONTROLLER	PCM_CONTROLLER_CM3	The Arm Cortex-M3 core controls PCM data transfers	0x0*
	PCM_CONTROLLER_DMA	The DMA controls PCM data transfers	0x1
ENABLE	PCM_DISABLE	Disable the PCM interface	0x0*
	PCM_ENABLE	Enable the PCM interface	0x1
SLAVE	PCM_SELECT_MASTER	Use the PCM interface in master mode	0x0
	PCM_SELECT_SLAVE	Use the PCM interface in slave mode	0x1*

11.5.4.2 PCM_TX_DATA

Bit Field	Field Name	Description
31:0	PCM_TX_DATA	Data to transmit over the PCM interface

11.5.4.3 PCM_RX_DATA

Bit Field	Field Name	Description
31:0	PCM_RX_DATA	Data received from the PCM interface

11.5.4.4 PCM_STATUS

Bit Field	Field Name	Description
3	TRANSMIT_STATUS	Indicate that PCM data has been sent
2	RECEIVE_STATUS	Indicate that PCM data has been received
1	OVERRUN_STATUS	Indicate that an overrun has occurred when receiving data on the PCM interface
0	UNDERRUN_STATUS	Indicate that an underrun has occurred when transmitting data on the PCM interface

Field Name	Value Symbol	Value Description	Hex Value
TRANSMIT_STATUS	PCM_TRANSMIT_FALSE	PCM data transmit flag not set	0x0*
	PCM_TRANSMIT_TRUE	PCM transmit data sent	0x1
	PCM_TRANSMIT_CLEAR	Clear the PCM transmit status bit	0x1
RECEIVE_STATUS	PCM_RECEIVE_FALSE	PCM data receive flag not set	0x0*
	PCM_RECEIVE_TRUE	PCM data received	0x1
	PCM_RECEIVE_CLEAR	Clear the PCM receive status bit	0x1
OVERRUN_STATUS	PCM_OVERRUN_FALSE	No PCM input overrun detected	0x0*
	PCM_OVERRUN_TRUE	PCM input overrun detected	0x1
	PCM_OVERRUN_CLEAR	Clear the PCM overrun bit	0x1
UNDERRUN_STATUS	PCM_UNDERRUN_FALSE	No PCM input underrun detected	0x0*
	PCM_UNDERRUN_TRUE	PCM input underrun detected	0x1
	PCM_UNDERRUN_CLEAR	Clear the PCM underrun bit	0x1

11.6 PULSE WIDTH MODULATION (PWM)

The RSL10 system contains two pulse-width modulator (PWM) drivers that can be configured to generate a single output signal with a specified period and duty cycle. Each PWM driver can be used as independently as a simple D/A converter.

The PWM drivers are multiplexed onto the DIO pads, which can be configured as output signals with the necessary physical pad configuration. The DIOs support output of the PWM signals with the specified period and high-time in each period (DIO_MODE_PWM*) and the inverse of the specified signal (DIO_MODE_PWM*_INV). For more information about configuring the multiplexed DIO functionality, see Chapter 10, “Digital Input/Output” on page 259.

The timing and shape of the signal produced by each PWM is defined by clock signals divided from SLOWCLK and the PWM_CFG_PWM_PERIOD and PWM_CFG_PWM_HIGH bit fields from the PWM_CFG_* register for that PWM.

The PWM drivers are each supported by a clock (PWM*CLK) that is divided from SLOWCLK using the CLK_DIV_CFG1_PWM*CLK_PRESCALE bit fields from the CLK_DIV_CFG1 register. Each PWM clock’s prescaler provides a clock prescaled from SLOWCLK by 1 to 64. For example, the frequency of the clock supplied to PWM0 is defined by:

$$f_{PWM0} = \frac{f_{SLOWCLK}}{(CLK\_DIV\_CFG1\_PWM0CLK\_PRESCALE + 1)}$$

For more information about the clock divisor configuration, see Section 6.3.8, “Interface Clocks” on page 82.

The value written to the PWM_CFG_PWM_PERIOD bit field configures the number of PWM*CLK cycles in one period of that PWM signal. Each period is (PWM_CFG_PWM_PERIOD + 1) PWM*CLK cycles in length, with a maximum length of 256 PWM*CLK cycles.

Similarly, the value written to the PWM_CFG_PWM_HIGH bit field configures the number of PWM*CLK cycles for which the PWM signal is high in each period. The PWM signal is high for the first (PWM_CFG_PWM_HIGH + 1) PWM*CLK cycles of each period, to a maximum equal to the period of that PWM signal. After (PWM_CFG_PWM_HIGH + 1) PWM*CLK cycles, the PWM signal is low for the remainder of the period.

NOTE: If the specified high time is greater than or equal to the specified period for a PWM, the PWM signal does not go low.

Figure 32 illustrates an example PWM configuration for PWM0 where the PWM period is configured for 10 cycles (`PWM_CFG_PWM_PERIOD` set to 9), with a high time of 6 cycles (`PWM_CFG_PWM_HIGH` set to 5). This results in a PWM signal that repeats every 10 PWM0CLK cycles with a duty cycle of 60%.

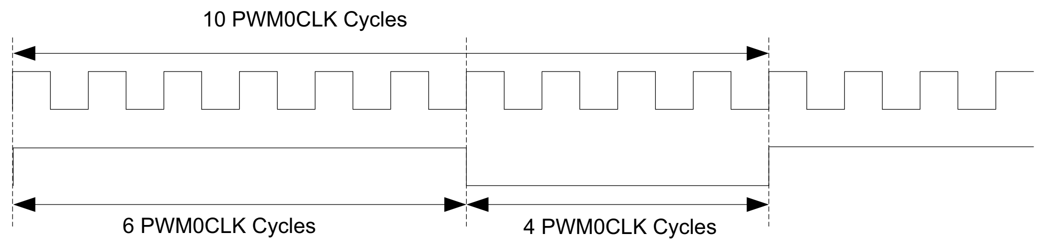


Figure 32. PWM Sample Configuration

Each PWM driver can be independently enabled and disabled by configuring the appropriate `PWM_CTRL_PWM*_ENABLE` bit from the `PWM_CTRL` register.

If the two PWM drivers use the same period and clock divisors, the relative timing between the PWM drivers can be defined. To enable offset configuration, set the `PWM_CTRL_PWM_OFFSET_ENABLE` bit from the `PWM_CTRL` register. If enabled, the number of cycles between the rising edge for PWM0 and the rising edge for PWM1 is equal to the value of the `PWM_CTRL_PWM_OFFSET` bit field from the `PWM_CTRL` register.

NOTE: If the specified offset is greater than or equal to the specified period for the PWM drivers, the behavior is undefined. The behavior is similarly undefined if the two PWM drivers do not share the same period.

11.6.1 PWM Registers

Register Name	Register Description	Address
`PWM_CFG`	PWM Configuration Register	0x40000D00
`PWM_CTRL`	PWM 0 and 1 Control Register	0x40000D08

11.6.1.1 PWM_CFG

Bit Field	Field Name	Description
15:8	`PWM_HIGH`	PWM high duty cycle
7:0	`PWM_PERIOD`	PWM period

11.6.1.2 PWM_CTRL

Bit Field	Field Name	Description
16	`PWM_OFFSET_ENABLE`	Enable/disable the PWM offset function
15:8	`PWM_OFFSET`	PWM0 to PWM1 offset

Bit Field	Field Name	Description
4	PWM1_ENABLE	Enable/disable the PWM1 block
0	PWM0_ENABLE	Enable/disable the PWM0 block

Field Name	Value Symbol	Value Description	Hex Value
PWM_OFFSET_ENABLE	PWM_OFFSET_DISABLE	Disable the PWM offset	0x0*
	PWM_OFFSET_ENABLE	Enable the PWM offset	0x1
PWM1_ENABLE	PWM1_DISABLE	Disable the PWM1 block	0x0*
	PWM1_ENABLE	Enable the PWM1 block	0x1
PWM0_ENABLE	PWM0_DISABLE	Disable the PWM0 block	0x0*
	PWM0_ENABLE	Enable the PWM0 block	0x1

11.7 SERIAL PERIPHERAL INTERFACES (SPI)

The RSL10 system includes two Serial Peripheral Interfaces (SPI) that allow the system to communicate with external components including external analog front ends, external controllers, and non-volatile memories (NVM).

The SPI interfaces are multiplexed onto the DIO pads, which can be configured as the input and output signals that form each SPI interface with the necessary physical pad configuration. For more information about configuring the multiplexed DIO functionality, see Chapter 10, “Digital Input/Output” on page 259.

The SPI interfaces can be enabled or disabled using the `SPI*_CTRL0_ENABLE` bit in the `SPI*_CTRL0` registers.

Data transfers using the SPI interfaces can be controlled directly by the Arm Cortex-M3 processor or indirectly using the DMA. To select the controller for an SPI interface, use the `SPI*_CTRL0_CONTROLLER` bit in the `SPI*_CTRL0` register.

11.7.1 SPI Data Transfers

The SPI interfaces can be configured to operate as an SPI master device or an SPI slave device by configuring the `SPI*_CTRL0_SLAVE` bit in the `SPI*_CTRL0` register. The differences in the SPI pad configurations between master and slave mode are as follows:

*SPI*_CLK*

In master mode, the SPI*_CLK signal is supplied by the RSL10. The SPI*_CLK signal is derived from SYSCLK using a power of two prescaler, configurable via the `SPI*_CTRL0_PRESCALE` bit field from the `SPI*_CTRL0` register, using the following equation:

$$f_{\mathrm{SPI*\_CLK}} = \frac{f_{\mathrm{SYSCLK}}}{2^{(\mathrm{SPI*\_CTRL0\_PRESCALE}+1)}}$$

In slave mode, the SPI*_CLK signal is sourced from a remote SPI master device. For proper operation, the frequency of the SPI_CLK inputs must abide by the following relation:

$$f_{\mathrm{SYSCLK}} \geq 4 \times f_{\mathrm{SPI*\_CLK}}$$

NOTE: If $f_{SYSCLK} < 6 \times f_{SPI*\_CLK}$, the output on the SPI*_SERO pad might be delayed by up to one SYSCLK period.

For both master and slave mode, the `SPI*_CTRL0_CLK_POLARITY` bit in the `SPI*_CTRL0` register is used to control both when data changes and when data is sampled. An SPI interface using normal polarity updates output signals on the falling edge of SPI*_CLK, and samples input signals on the rising edge of SPI*_CLK. If the polarity is inverted, output signals change on the rising edge of SPI*_CLK and input signals are sampled on the falling edge.

*SPI*_CS*

In master mode, the SPI*_CS pad is an output controlled by the `SPI*_CTRL1_CS` bit from the `SPI*_CTRL1` register. The signal from this pad is generally routed to the chip select input of a slave device.

In slave mode, the SPI*_CS pad is an input sourced from a remote SPI master device.

NOTE: For an SPI device, the chip select input is generally interpreted as an active-low signal. As such, if the signal on the SPI*_CS pad is high, the SPI slave ignores all communications using the interface. The minimum delay between a falling edge on the SPI*_CS pad and the first edge on SPI*_CLK is $\frac{1}{2} \cdot \text{SPI*\_CLK Period} + 2 \cdot \text{SYSCLK Periods}$.

*SPI*_SERI*

This pad is a serial input signal that is used to receive data when the following conditions are met:

- SPI*_CLK and SPI*_CS indicate that data should be transferred.
- The `SPI*_CTRL1_RW_CMD` bit field from the `SPI*_CTRL1` register is configured for an SPI read or read-write (full-duplex) transfer.

*SPI*_SERO*

This pad is a serial output signal that is used to transmit data when the following conditions are met:

- SPI*_CLK and SPI*_CS indicate that data should be transferred.
- The `SPI*_CTRL1_RW_CMD` bit field from the `SPI*_CTRL1` register is configured for an SPI write or read-write (full-duplex) transfer.

The SPI interfaces support a configurable word size for each transfer of 1 to 32 bits of data per word, as configured using the `SPI*_CTRL1_WORD_SIZE` bit field in the `SPI*_CTRL1` register. All data transactions using an SPI interface start with the MSB of the word received, and transfer (`SPI*_CTRL1_WORD_SIZE` + 1) bits of data per word.

The `SPI*_CTRL1_START_BUSY` bit in the `SPI*_CTRL1` register indicates if the SPI interface is currently transferring data. When operating in master mode, this same bit can be used to start an SPI transfer.

NOTE: When the interface is disabled, any data currently being transmitted is allowed to complete before the interface shuts down. If the interface is busy, the `SPI*_CTRL*` registers cannot be modified until the interface is idle.

If the `SPI*_CTRL0_MODE_SELECT` bit in the `SPI*_CTRL0` register is set, the SPI interface operates in auto mode to limit the overhead between SPI transfers. This mode works in conjunction with the SPI interface data registers. These registers are:

SPI*_TX_DATA	Shift register containing the data to transmit using the SPI interface.
SPI*_RX_DATA	Shift register containing the most recent data word received from the SPI interface.

When operating as an SPI master in auto mode, these registers can be used to efficiently continue a transfer. In this configuration, if writing to the SPI interface, writing the SPI*_TX_DATA register initiates another write transfer. Similarly, if reading from the SPI interface, reading the SPI*_RX_DATA register initiates another read transfer.

CAUTION: The SPI*_CTRL1_START_BUSY bit is cleared for one cycle between transfers in auto mode. When polling this bit to determine when a transfer completes, ensure that at least one cycle has elapsed after starting the transfer before polling for the completion of a transfer.

In manual mode or when operating as an SPI slave, reading or writing either of these registers has no side effects.

IMPORTANT: When transmitting data as an SPI slave, the SPI interface must load the SPI*_TX_DATA register between SPI*_CLK edges that update the SPI output signals. If no delay occurs between the words of the SPI transfer, new data must be loaded in one SPI*_CLK cycle.

11.7.2 SPI Interrupts

Each of the SPI interfaces uses three interrupts that signal:

- The completion of a data transmission
- The completion of data received
- Any errors that occurred when completing a transfer using the interface

Section 14.1, "Nested Vectored Interrupt Controller (NVIC)" on page 400 for information regarding interrupt configuration and handling.

If a user application is transmitting data using an SPI interface, the SPI*_TX interrupt signals:

- That the interface has started transmitting the data value from the SPI*_TX_DATA register
- That the application can now load the next data word (of the specified size) to be transmitted

If a user application is receiving data from an SPI interface, the SPI*_RX interrupt signals that a data word of the specified size was successfully received and written to the SPI*_RX_DATA register.

To support tracking the status of an SPI transfer, the SPI*_STATUS registers include a pair of status bits that indicate if data was transmitted (SPI*_STATUS_SPI*_TRANSMIT_STATUS) or received (SPI*_STATUS_SPI*_RECEIVE_STATUS) using the SPI interface since the bit was last cleared. The transmit status bit can be cleared by writing SPI*_TRANSMIT_CLEAR to the SPI*_STATUS register. The receive status bits can similarly be cleared by writing SPI*_RECEIVE_CLEAR to the SPI*_STATUS register.

11.7.3 SPI DMA Control

If using the DMA to control data transfers over an SPI interface, transmit events are triggered on the completion of a data transmission and receive events are triggered on the completion of a received data word. There are two methods to ensure that the first DMA word is transferred:

- Configure the SPI interface for write or full-duplex mode before switching from Arm Cortex-M3 processor control of the SPI interface to DMA control of the interface. Using this method generates a DMA TX request.
- Manually prepare the first data word to be transmitted by writing to the `SPI*_TX_DATA` register.

Additionally, while using the DMA to control data transfers over an SPI interface, the `SPI*_ERROR` interrupts indicate when an error has occurred.

- An overrun occurs when the DMA cannot read the received data from the `SPI*_RX_DATA` register to the DMA buffer before this data is overwritten. To monitor for overrun events, set the `SPI*_CTRL0_OVERRUN_INT_ENABLE` bit in the `SPI*_CTRL0` register. If an overrun occurs and this event monitor is enabled, the `SPI*_STATUS_SPI*_OVERRUN_STATUS` bit in the appropriate `SPI*_STATUS` register is set.
- An underrun occurs when the DMA cannot write the `SPI*_TX_DATA` register before a second data transfer starts using this register's previous data. To monitor for underrun events, set the `SPI*_CTRL0_UNDERRUN_INT_ENABLE` bit in the `SPI*_CTRL0` register. If an underrun occurs and this event monitor is enabled, the `SPI*_STATUS_SPI*_UNDERRUN_STATUS` bit in the appropriate `SPI*_STATUS` register is set.

11.7.4 SPI Registers

Register Name	Register Description	Address
SPI0_CTRL0	SPI0 Control and Configuration Register	0x40000800
SPI0_CTRL1	SPI0 Transaction Control Register	0x40000804
SPI0_TX_DATA	SPI0 Transmit Data	0x40000808
SPI0_RX_DATA	SPI0 Received Data	0x4000080C
SPI0_STATUS	SPI0 Status	0x40000810
SPI1_CTRL0	SPI1 Control and Configuration Register	0x40000900
SPI1_CTRL1	SPI1 Transaction Control Register	0x40000904
SPI1_TX_DATA	SPI1 Transmit Data	0x40000908
SPI1_RX_DATA	SPI1 Received Data	0x4000090C
SPI1_STATUS	SPI1 Status	0x40000910

11.7.4.1 SPI0_CTRL0

Bit Field	Field Name	Description
10	SPI0_OVERRUN_INT_ENABLE	Enable/disable SPI overrun interrupts
9	SPI0_UNDERRUN_INT_ENABLE	Enable/disable SPI underrun interrupts
8	SPI0_CONTROLLER	Select whether data transfer will be controlled by the Arm Cortex-M3 core or the DMA for SPI
7	SPI0_SLAVE	Use the SPI interface as master or slave
6	SPI0_CLK_POLARITY	Select the polarity of the SPI clock
5	SPI0_MODE_SELECT	Select between manual and auto transaction handling modes for SPI master transactions
4	SPI0_ENABLE	Enable/disable the SPI interface
3:0	SPI0_PRESCALE	Prescale the SPI interface clock for master transfers

Field Name	Value Symbol	Value Description	Hex Value
SPI0_OVERRUN_INT_ENABLE	SPI0_OVERRUN_INT_DISABLE	No interrupts are raised when an overrun occurs on the SPI interface	0x0*
	SPI0_OVERRUN_INT_ENABLE	An interrupt is raised when an overrun occurs on the SPI interface	0x1
SPI0_UNDERRUN_INT_ENABLE	SPI0_UNDERRUN_INT_DISABLE	No interrupts are raised when an underrun occurs on the SPI interface	0x0*
	SPI0_UNDERRUN_INT_ENABLE	An interrupt is raised when an underrun occurs on the SPI interface	0x1
SPI0_CONTROLLER	SPI0_CONTROLLER_CM3	The Arm Cortex-M3 core controls data transfers using SPI	0x0*
	SPI0_CONTROLLER_DMA	The DMA controls data transfers using SPI	0x1
SPI0_SLAVE	SPI0_SELECT_MASTER	Use the SPI interface in master mode	0x0*
	SPI0_SELECT_SLAVE	Use the SPI interface in slave mode	0x1
SPI0_CLK_POLARITY	SPI0_CLK_POLARITY_NORMAL	In both master and slave modes SERO changes on the falling edge of the SPI clock. The SERI is sampled in slave mode just after and in master mode at the rising edge of the SPI clock	0x0*
	SPI0_CLK_POLARITY_INVERSE	In both master and slave modes SERO changes on the rising edge of the SPI clock. The SERI is sampled in slave mode just after and in master mode at the falling edge of the SPI clock	0x1
SPI0_MODE_SELECT	SPI0_MODE_SELECT_MANUAL	Master transfers using the SPI interface do not automatically continue	0x0*
	SPI0_MODE_SELECT_AUTO	Automatically continue master transfers using the SPI interface	0x1
SPI0_ENABLE	SPI0_DISABLE	Disable the SPI interface	0x0*
	SPI0_ENABLE	Enable the SPI interface	0x1
SPI0_PRESCALE	SPI0_PRESCALE_2	Prescale the SPI interface clock by 2	0x0*
	SPI0_PRESCALE_4	Prescale the SPI interface clock by 4	0x1
	SPI0_PRESCALE_8	Prescale the SPI interface clock by 8	0x2
	SPI0_PRESCALE_16	Prescale the SPI interface clock by 16	0x3
	SPI0_PRESCALE_32	Prescale the SPI interface clock by 32	0x4
	SPI0_PRESCALE_64	Prescale the SPI interface clock by 64	0x5
	SPI0_PRESCALE_128	Prescale the SPI interface clock by 128	0x6
	SPI0_PRESCALE_256	Prescale the SPI interface clock by 256	0x7
	SPI0_PRESCALE_512	Prescale the SPI interface clock by 512	0x8
	SPI0_PRESCALE_1024	Prescale the SPI interface clock by 1024	0x9

11.7.4.2 SPI0_CTRL1

Bit Field	Field Name	Description
8	SPI0_START_BUSY	Start an SPI data transfer and indicate if a transfer is in progress
8	SPI0_BUSY_STATUS	SPI data transfer status read
7:6	SPI0_RW_CMD	Issue a read command or write command to the SPI interface
5	SPI0_CS	Set the chip-select line for SPI (master mode), read the chip-select line for SPI (slave mode)
4:0	SPI0_WORD_SIZE	Select the word size used by the SPI interface (word size = SPI0_WORD_SIZE + 1)

Field Name	Value Symbol	Value Description	Hex Value
SPI0_START_BUSY	SPI0_IDLE	Stop a transfer or indicate that the SPI interface is idle	0x0*
	SPI0_START	Start a transfer on the SPI interface (master mode only)	0x1
SPI0_BUSY_STATUS	SPI0_BUSY	Indicate that the SPI interface is currently transferring data	0x1
SPI0_RW_CMD	SPI0_NOP	No operation	0x0*
	SPI0_WRITE_DATA	Write data using the SPI interface	0x1
	SPI0_READ_DATA	Read data using the SPI interface	0x2
	SPI0_RW_DATA	Read and write data using the SPI interface	0x3
SPI0_CS	SPI0_CS_0	Set the SPI CS signal low	0x0
	SPI0_CS_1	Set the SPI CS signal high	0x1*
SPI0_WORD_SIZE	SPI0_WORD_SIZE_1	SPI transfers use 1-bit words	0x0*
	SPI0_WORD_SIZE_8	SPI transfers use 8-bit words	0x7
	SPI0_WORD_SIZE_16	SPI transfers use 16-bit words	0xF
	SPI0_WORD_SIZE_24	SPI transfers use 24-bit words	0x17
	SPI0_WORD_SIZE_32	SPI transfers use 32-bit words	0x1F

11.7.4.3 SPI0_TX_DATA

Bit Field	Field Name	Description
31:0	SPI0_TX_DATA	Single word buffer for data to be transmitted over the SPI interface

11.7.4.4 SPI0_RX_DATA

Bit Field	Field Name	Description
31:0	SPI0_RX_DATA	Single word buffer for data that has been received over the SPI interface

11.7.4.5 SPI0_STATUS

Bit Field	Field Name	Description
3	SPI0_TRANSMIT_STATUS	Indicate that the transmission of the data is completed
2	SPI0_RECEIVE_STATUS	Indicate that new data has been received
1	SPI0_OVERRUN_STATUS	Indicate that an overrun has occurred when receiving data on the SPI interface
0	SPI0_UNDERRUN_STATUS	Indicate that an underrun has occurred when transmitting data on the SPI interface

Field Name	Value Symbol	Value Description	Hex Value
SPI0_TRANSMIT_STATUS	SPI0_TRANSMIT_FALSE	SPI data transmit flag not set	0x0*
	SPI0_TRANSMIT_TRUE	SPI transmit data sent	0x1
	SPI0_TRANSMIT_CLEAR	Clear the SPI transmit status bit	0x1
SPI0_RECEIVE_STATUS	SPI0_RECEIVE_FALSE	SPI data receive flag not set	0x0*
	SPI0_RECEIVE_TRUE	SPI data received	0x1
	SPI0_RECEIVE_CLEAR	Clear the SPI receive status bit	0x1
SPI0_OVERRUN_STATUS	SPI0_OVERRUN_FALSE	No SPI input overrun detected	0x0*
	SPI0_OVERRUN_TRUE	SPI input overrun detected	0x1
	SPI0_OVERRUN_CLEAR	Clear the SPI overrun bit	0x1
SPI0_UNDERRUN_STATUS	SPI0_UNDERRUN_FALSE	No SPI input underrun detected	0x0*
	SPI0_UNDERRUN_TRUE	SPI input underrun detected	0x1
	SPI0_UNDERRUN_CLEAR	Clear the SPI underrun bit	0x1

11.7.4.6 SPI1_CTRL0

Bit Field	Field Name	Description
10	SPI1_OVERRUN_INT_ENABLE	Enable/disable SPI overrun interrupts
9	SPI1_UNDERRUN_INT_ENABLE	Enable/disable SPI underrun interrupts
8	SPI1_CONTROLLER	Select whether data transfer will be controlled by the Arm Cortex-M3 core or the DMA for SPI
7	SPI1_SLAVE	Use the SPI interface as master or slave
6	SPI1_CLK_POLARITY	Select the polarity of the SPI clock
5	SPI1_MODE_SELECT	Select between manual and auto transaction handling modes for SPI master transactions
4	SPI1_ENABLE	Enable/disable the SPI interface
3:0	SPI1_PRESCALE	Prescale the SPI interface clock for master transfers

Field Name	Value Symbol	Value Description	Hex Value
SPI1_OVERRUN_INT_ENABLE	SPI1_OVERRUN_INT_DISABLE	No interrupts are raised when an overrun occurs on the SPI interface	0x0*
	SPI1_OVERRUN_INT_ENABLE	An interrupt is raised when an overrun occurs on the SPI interface	0x1

Field Name	Value Symbol	Value Description	Hex Value
SPI1_UNDERRUN_INT_ENABLE	SPI1_UNDERRUN_INT_DISABLE	No interrupts are raised when an underrun occurs on the SPI interface	0x0*
	SPI1_UNDERRUN_INT_ENABLE	An interrupt is raised when an underrun occurs on the SPI interface	0x1
SPI1_CONTROLLER	SPI1_CONTROLLER_CM3	The Arm Cortex-M3 core controls data transfers using SPI	0x0*
	SPI1_CONTROLLER_DMA	The DMA controls data transfers using SPI	0x1
SPI1_SLAVE	SPI1_SELECT_MASTER	Use the SPI interface in master mode	0x0*
	SPI1_SELECT_SLAVE	Use the SPI interface in slave mode	0x1
SPI1_CLK_POLARITY	SPI1_CLK_POLARITY_NORMAL	In both master and slave modes SERO changes on the falling edge of the SPI clock. The SERI is sampled in slave mode just after and in master mode at the rising edge of the SPI clock	0x0*
	SPI1_CLK_POLARITY_INVERSE	In both master and slave modes SERO changes on the rising edge of the SPI clock. The SERI is sampled in slave mode just after and in master mode at the falling edge of the SPI clock	0x1
SPI1_MODE_SELECT	SPI1_MODE_SELECT_MANUAL	Master transfers using the SPI interface do not automatically continue	0x0*
	SPI1_MODE_SELECT_AUTO	Automatically continue master transfers using the SPI interface	0x1
SPI1_ENABLE	SPI1_DISABLE	Disable the SPI interface	0x0*
	SPI1_ENABLE	Enable the SPI interface	0x1
SPI1_PRESCALE	SPI1_PRESCALE_2	Prescale the SPI interface clock by 2	0x0*
	SPI1_PRESCALE_4	Prescale the SPI interface clock by 4	0x1
	SPI1_PRESCALE_8	Prescale the SPI interface clock by 8	0x2
	SPI1_PRESCALE_16	Prescale the SPI interface clock by 16	0x3
	SPI1_PRESCALE_32	Prescale the SPI interface clock by 32	0x4
	SPI1_PRESCALE_64	Prescale the SPI interface clock by 64	0x5
	SPI1_PRESCALE_128	Prescale the SPI interface clock by 128	0x6
	SPI1_PRESCALE_256	Prescale the SPI interface clock by 256	0x7
	SPI1_PRESCALE_512	Prescale the SPI interface clock by 512	0x8
	SPI1_PRESCALE_1024	Prescale the SPI interface clock by 1024	0x9

11.7.4.7 SPI1_CTRL1

Bit Field	Field Name	Description
8	SPI1_START_BUSY	Start an SPI data transfer and indicate if a transfer is in progress
7:6	SPI1_RW_CMD	Issue a read command or write command to the SPI interface

Bit Field	Field Name	Description
5	SPI1_CS	Set the chip-select line for SPI (master mode), read the chip-select line for SPI (slave mode)
4:0	SPI1_WORD_SIZE	Select the word size used by the SPI interface (word size = SPI1_WORD_SIZE + 1)

Field Name	Value Symbol	Value Description	Hex Value
SPI1_START_BUSY	SPI1_IDLE	Stop a transfer or indicate that the SPI interface is idle	0x0*
	SPI1_START	Start a transfer on the SPI interface (master mode only)	0x1
	SPI1_BUSY	Indicate that the SPI interface is currently transferring data	0x1
SPI1_RW_CMD	SPI1_NOP	No operation	0x0*
	SPI1_WRITE_DATA	Write data using the SPI interface	0x1
	SPI1_READ_DATA	Read data using the SPI interface	0x2
	SPI1_RW_DATA	Read and write data using the SPI interface	0x3
SPI1_CS	SPI1_CS_0	Set the SPI CS signal low	0x0
	SPI1_CS_1	Set the SPI CS signal high	0x1*
SPI1_WORD_SIZE	SPI1_WORD_SIZE_1	SPI transfers use 1-bit words	0x0*
	SPI1_WORD_SIZE_8	SPI transfers use 8-bit words	0x7
	SPI1_WORD_SIZE_16	SPI transfers use 16-bit words	0xF
	SPI1_WORD_SIZE_24	SPI transfers use 24-bit words	0x17
	SPI1_WORD_SIZE_32	SPI transfers use 32-bit words	0x1F

11.7.4.8 SPI1_TX_DATA

Bit Field	Field Name	Description
31:0	SPI1_TX_DATA	Single word buffer for data to be transmitted over the SPI interface

11.7.4.9 SPI1_RX_DATA

Bit Field	Field Name	Description
31:0	SPI1_RX_DATA	Single word buffer for data that has been received over the SPI interface

11.7.4.10 SPI1_STATUS

Bit Field	Field Name	Description
3	SPI1_TRANSMIT_STATUS	Indicate that the transmission of the data is completed
2	SPI1_RECEIVE_STATUS	Indicate that a new data has been received
1	SPI1_OVERRUN_STATUS	Indicate that an overrun has occurred when receiving data on the SPI interface
0	SPI1_UNDERRUN_STATUS	Indicate that an underrun has occurred when transmitting data on the SPI interface

Field Name	Value Symbol	Value Description	Hex Value
SPI1_TRANSMIT_STATUS	SPI1_TRANSMIT_FALSE	SPI data transmit flag not set	0x0*
	SPI1_TRANSMIT_TRUE	SPI transmit data sent	0x1
	SPI1_TRANSMIT_CLEAR	Clear the SPI transmit status bit	0x1
SPI1_RECEIVE_STATUS	SPI1_RECEIVE_FALSE	SPI data receive flag not set	0x0*
	SPI1_RECEIVE_TRUE	SPI data received	0x1
	SPI1_RECEIVE_CLEAR	Clear the SPI receive status bit	0x1
SPI1_OVERRUN_STATUS	SPI1_OVERRUN_FALSE	No SPI input overrun detected	0x0*
	SPI1_OVERRUN_TRUE	SPI input overrun detected	0x1
	SPI1_OVERRUN_CLEAR	Clear the SPI overrun bit	0x1
SPI1_UNDERRUN_STATUS	SPI1_UNDERRUN_FALSE	No SPI input underrun detected	0x0*
	SPI1_UNDERRUN_TRUE	SPI input underrun detected	0x1
	SPI1_UNDERRUN_CLEAR	Clear the SPI underrun bit	0x1

11.8 Universal Asynchronous Receiver-Transmitter (UART) Interfaces

The general-purpose Universal Asynchronous Receiver-Transmitter (UART) interface provides support for communicating with devices supporting the standard UART transmission protocol.

The UART interface is multiplexed onto the DIO pads, which can be configured as the UART interface's receiver and transmitter signals, with the necessary physical pad configuration (pull-up/pull-down resistor and low pass filtering configuration for the Rx signal, drive strength configuration for the Tx signal). For more information about configuring the multiplexed DIO functionality, see Chapter 10, "Digital Input/Output" on page 259.

The UART interface can be enabled or disabled using the `UART_CFG_ENABLE` bit in the appropriate `UART_CFG` register. When disabled, the UART interface will immediately terminate any ongoing transfer, and setting the UART transmit line high and ignoring any partially received receive data.

The UART interface operates in full-duplex mode using a standard data format of 1 start bit, 8 data bits and 1 stop bit. All data bytes being sent or received are interpreted as starting with the LSB. Figure 33 shows the waveform for a UART transmit or receive transaction.

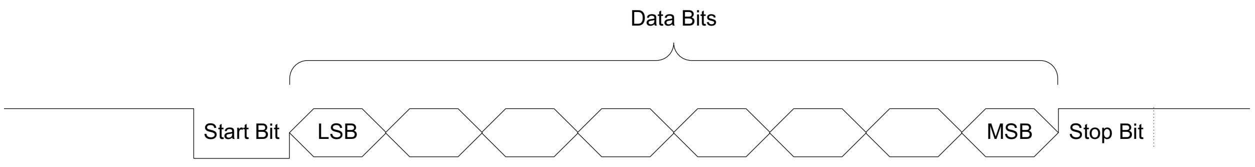


Figure 33. UART Transaction Waveform

Data to be transmitted is written to the `UART_TX_DATA` register. Data that has been received over the UART interface is stored in the `UART_RX_DATA` register. The `UART_TX_DATA` and `UART_RX_DATA` registers are only accessible after the interface has been enabled via the `UART_CFG` register.

NOTE: When the UART interface is disabled using the `UART_CFG_ENABLE` bit, data transmissions that are in progress complete before the interface shuts down.

The baud rates (specified in bits per second) for the UART interfaces are defined in terms of the UARTCLK frequency and several configuration parameters, as seen in the following equation for baud rate calculation:

$$\text{baud rate} = \frac{(\text{UART\_CFG\_PRESCALE} \times \text{UARTCLK})}{(2^{18} \times (1 + 11 \times \text{UART\_CFG\_PRESCALE\_ENABLE}))}$$

The `CLK_DIV_CFG1_UARTCLK_PRESCALE` bit field from the `CLK_DIV_CFG1` register is used to prescale the SYSCLK frequency to a reasonable frequency for UARTCLK. The `UART_CFG_PRESCALE_ENABLE` bit from the `UART_CFG` registers can be used as a coarse divide-by-12 clock prescaler for further reducing the frequency of UARTCLK to the appropriate range for the desired baud rate. The `UART_CFG_PRESCALE` bit field from `UART_CFG` provides the necessary fine adjustments to match an exact baud rate. The configuration supports baud rates up to 1/4 of the UARTCLK frequency. For more information about the clock divisor configuration, see Section 6.3.8, “Interface Clocks” on page 82.

IMPORTANT: For proper functionality, UART interfaces require both sides of a connection to have an absolute clock accuracy error of less than 2.5%. To allow for clock jitter, we recommend that all UART communications use a clock with a maximum of 2% error versus the expected target frequency.

Data transfers using the UART interfaces can be controlled directly by the Arm Cortex-M3 processor or indirectly using the DMA. Controller selection for the UART interface is configured using the `DMA_ENABLE` bit in the `UART_CFG` registers.

11.8.1 UART Interrupts

The UART interface uses three associated interrupts that separately control transmission and reception of UART data and handling of UART transmission errors. Section 14.1, “Nested Vectored Interrupt Controller (NVIC)” on page 400 for information regarding interrupt configuration and handling.

If a user application is transmitting data from the UART interface, the `UART_TX` interrupt signals that transmission of the data value in the `UART_TX_DATA` register has started, and that the application can now load the next byte to be transmitted.

If a user application is receiving data from the UART interface, the `UART_RX` interrupt signals that a data byte has been successfully received and has been written to the `UART_RX_DATA` register.

In both Interrupt Mode and DMA Mode, the `UART_ERROR` interrupt is raised when an RX buffer overrun occurs. When this happens, the `UART_RX_OVERRUN_STATUS` bit from the `UART_STATUS` register is set.

11.8.2 UART Interface Registers

Register Name	Register Description	Address
`UART_CFG`	UART Configuration Register	0x40000C00
`UART_TX_DATA`	UART Transmit Data Register	0x40000C04
`UART_RX_DATA`	UART Receive Data Register	0x40000C08
`UART_STATUS`	UART Status Register	0x40000C0C

11.8.2.1 UART_CFG

Bit Field	Field Name	Description
23:8	PRESCALE	Prescaling multiplier in baud rate calculation
4	PRESCALE_ENABLE	Enable/disable a fixed prescaler by 12
1	DMA_ENABLE	DMA mode enable
0	ENABLE	Enable/disable the UART interface

Field Name	Value Symbol	Value Description	Hex Value
PRESCALE_ENABLE	UART_PRESCALE_DISABLE	Disable prescaling division by 12	0x0*
	UART_PRESCALE_ENABLE	Enable prescaling division by 12	0x1
DMA_ENABLE	UART_DMA_MODE_DISABLE	Disable the DMA mode	0x0*
	UART_DMA_MODE_ENABLE	Enable the DMA mode	0x1
ENABLE	UART_DISABLE	Disable the UART interface	0x0*
	UART_ENABLE	Enable the UART interface	0x1

11.8.2.2 UART_TX_DATA

Bit Field	Field Name	Description
7:0	UART_TX_DATA	UART Transmit data

11.8.2.3 UART_RX_DATA

Bit Field	Field Name	Description
7:0	UART_RX_DATA	UART Received data

11.8.2.4 UART_STATUS

Bit Field	Field Name	Description
0	UART_RX_OVERRUN_STATUS	Indicate that an overrun has occurred when receiving data on the UART interface

Field Name	Value Symbol	Value Description	Hex Value
UART_RX_OVERRUN_STATUS	UART_RX_OVERRUN_FALSE	No UART Rx overrun detected	0x0*
	UART_RX_OVERRUN_TRUE	UART Rx overrun detected	0x1
	UART_RX_OVERRUN_CLEAR	Clear the Rx overrun status bit	0x1

11.9 Support Interfaces

The RSL10 SoC includes several internal connections that link the RF front-end (see Chapter 8, “RF Front-End” on page 131) and the Bluetooth low energy technology baseband hardware (see Chapter 9, “Bluetooth Low Energy Baseband” on page 201). This includes:

- An internal SPI interface bus for control and configuration of the RF front-end

- RF front-end GPIO signals
- Bluetooth low energy technology baseband Tx, Rx, and synchronization control signals
- Bluetooth low energy technology baseband Tx, and Rx data signals

All signals from these support interfaces are accessible through the DIOs for debug and testing purposes. DIO outputs can be configured to provide the signals from the support interfaces using the `DIO_CFG` registers. DIO inputs can be configured to provide alternate sources for support interface inputs using the `BB_*_SRC` and `RF_*_SRC` registers.

If your are using DIO[12:15], in addition to configuring the DIOs in the `DIO_CFG_*` register, disable the JTAG bit-banding signals in the JTAG configuration register `DIO_JTAG_SW_PAD_CFG` as follows:

- Set the `CM3_JTAG_DATA_EN_ALIAS` bit field to `CM3_JTAG_DATA_DISABLED_BITBAND`.
- Set the `CM3_JTAG_TRST_EN_ALIAS` bit field to `CM3_JTAG_TRST_DISABLED_BITBAND`.

For more information on DIO configuration, see Chapter 10, “Digital Input/Output” on page 259. Figure 34 shows how these interfaces (and their DIO connections) are connected internally on the RSL10 SoC, and Table 27 provides a description of these connections.

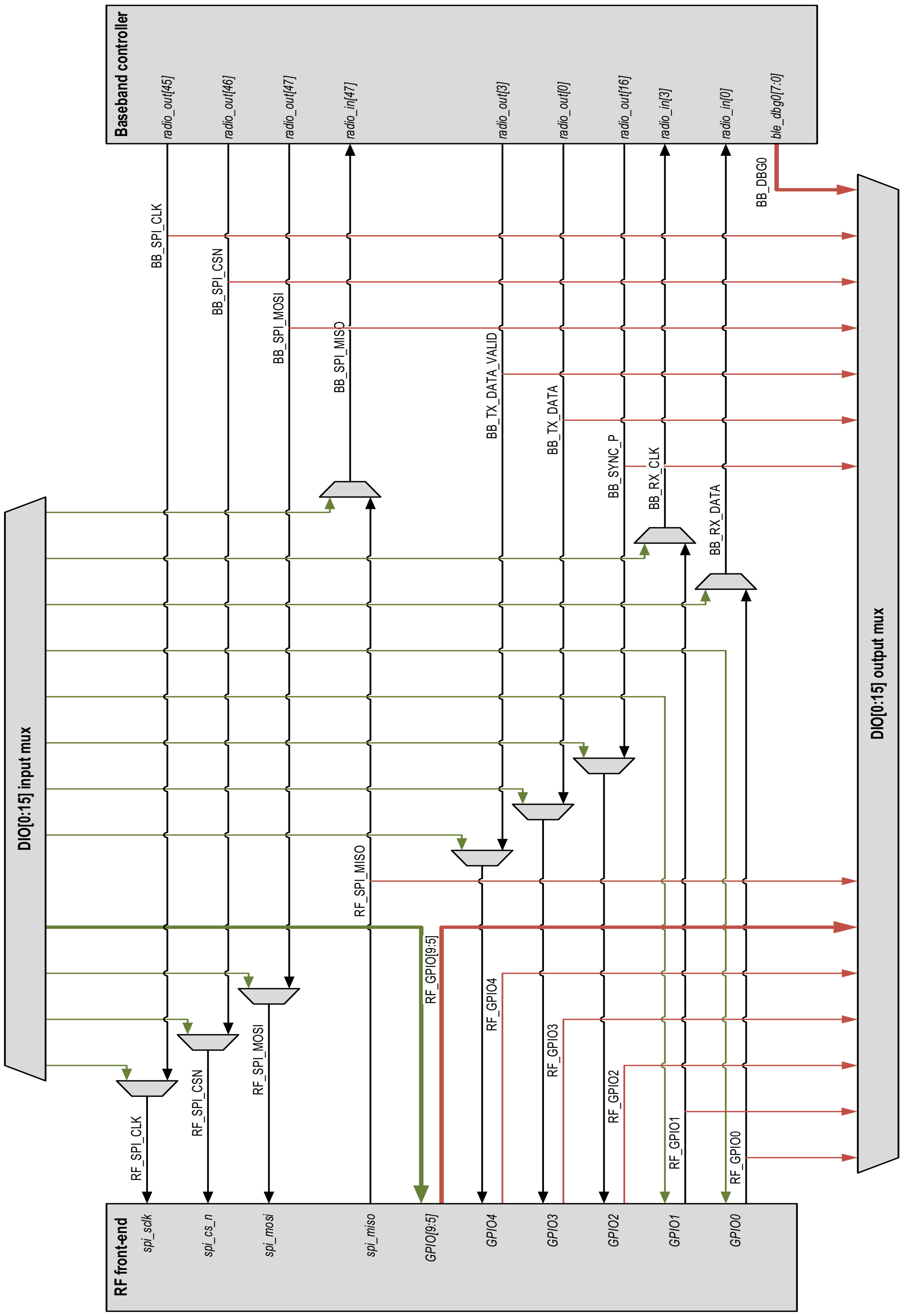


Figure 34. Interface between the Baseband Controller and the RF Front-End

Table 27. Interface Signals Between the Baseband Controller and the RF Front-End

Baseband Signal	RF Front-end Signal	Source	Description
BB_SPI_MISO	RF_SPI_MISO	RF Front-end	SPI data slave to master
BB_SPI_MOSI	RF_SPI_MOSI	Baseband	SPI data master to slave
BB_SPI_CLK	RF_SPI_CLK	Baseband	SPI data clock
BB_SPI_CSN	RF_SPI_CSN	Baseband	SPI slave device select
BB_RX_DATA	RF_GPIO0	RF Front-end	Rx data
BB_RX_CLK	RF_GPIO1	RF Front-end	Rx data clock
BB_TX_DATA	RF_GPIO3	Baseband	Tx data
BB_TX_DATA_VALID	RF_GPIO4	Baseband	Tx data valid indicator
BB_SYNC_P	RF_GPIO2	Baseband	Access address detection

When the RF front-end is isolated or powered down, signals provided by the RF front-end are forced to zero. When the baseband controller is disabled, all signals provided by the Bluetooth baseband except SPI_CSN are forced to zero (SPI_CSN is forced to one).

NOTE: Disabling or removing the clock source from the RF front-end or the baseband controller will stop, and potentially corrupt, any ongoing SPI transaction. As a result, care should be taken to disable the RF front-end and baseband controller only when the baseband-RF front-end interface is idle, as indicated by the `BBIF_SYNC_CFG_RF_ACTIVE` bit from the `BBIF_SYNC_CFG` register.

The RF front-end supports five additional GPIOs that are not connected between the baseband controller and the RF front-end.

All ten of the RF front-end GPIOs can be used to monitor or route internal signals from the RF front-end, and their use is defined by the `RF_REG*_PAD_CONF_*_PAD_*_CONF` bit-fields from the `RF_REG03`, `RF_REG04` registers. Using the RF front-end GPIOs in their non-default configurations (i.e., using a setting other than the specified `PAD_CONF_*_PAD_*_CONF_DEFAULT` bit-setting) is not recommended in any user applications.

CHAPTER 12

Peripherals

12.1 Cyclic Redundancy Check (CRC) Generator

The peripherals to the Arm Cortex-M3 processor include a CRC generator that provides support for two standard cyclic redundancy check (CRC) algorithms (CRC-CCITT and CRC-32, defined by the IEEE 802.3 Ethernet standard). The calculated outputs from this generator can be employed by a user application to ensure data integrity of communications and non-volatile memory information. They do this by guaranteeing that all single-bit errors, two-bit errors, burst errors (i.e., multiple bit errors in a row), and any error containing an odd number of bits can be detected.

NOTE: The integrity of Bluetooth communications is already protected by a 24-bit CRC. The integrity of individual pairs of flash memory words are protected by the flash's integrated error correction code.

The CRC generator can be configured to select the CRC-CCITT algorithm, by clearing the `CRC_CTRL_CRC_TYPE` bit from the `CRC_CTRL` register to the `CRC_CCITT` bit setting. The parameters associated with the CRC-CCITT algorithm implementation are provided in Table 28.

Table 28. CRC-CCITT Algorithm Parameters

CRC Parameter	Parameter Value
Order	16
Polynomial	$x^{16} + x^{12} + x^5 + 1$
Polynomial (hex)	0x1021
Initial Value (hex)	0xFFFF
Final XOR Value (hex)	0x0000

No data manipulation is required for the output CRC generated for the standard CRC-CCITT algorithm (i.e., no data byte reversal, reversal of the final result or other finalization).

The CRC generator can be configured to select the CRC-32 algorithm, by setting the `CRC_CTRL_CRC_TYPE` bit from the `CRC_CTRL` register to the `CRC_32` bit setting. The parameters associated with the CRC-32 algorithm implementation are provided in Table 29.

Table 29. CRC-32 Algorithm Parameters

CRC Parameter	Parameter Value
Order	32
Polynomial	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
Polynomial (hex)	0x4C11DB7
Initial Value (hex)	0xFFFFFFFF
Final XOR Value (hex)	0xFFFFFFFF

The output CRC generated for the standard CRC-32 algorithm requires data byte reversal and reversal of the final result.

The Arm Cortex-M3 processor's CRC generator supports non-standard variants of the CRC-CCITT and CRC-32 standard implementation.

- To use non-standard CRC ordering of data within each data byte, set the `CRC_CTRL_BIT_ORDER` bit from the `CRC_CTRL` register.
- To use non-standard CRC ordering of the final result, set the `CRC_CTRL_FINAL_CRC_REVERSE` bit from the `CRC_CTRL` register.
- To use non-standard CRC XOR of the final result, set the `CRC_CTRL_FINAL_CRC_XOR` bit from the `CRC_CTRL` register. If configured for a non-standard XOR, this uses a final XOR value of 0xFFFF for CRC-CCITT and 0x00000000 for CRC-32.

To use the CRC generator:

1. Write an initial value of 0xFFFF or 0xFFFFFFFF to the `CRC_VALUE` register.
2. Write to the CRC input registers any data that must be included in the final CRC read from the generator; 1-bit, 8-bit, 16-bit, 24-bit and 32-bit data values are supported. Input data to the CRC generator can be interpreted as either little-endian or big-endian data by selecting the appropriate endian byte ordering, using the `CRC_CTRL_BYTE_ORDER` bit from the `CRC_CTRL` register.
3. At any time, you can read the `CRC_FINAL` register to obtain the CRC for the data that has been added using the CRC input registers since the last time the `CRC_VALUE` register was initialized.

12.1.1 CRC Registers

Register Name	Register Description	Address
`CRC_CTRL`	CRC Generator Control	0x40000F00
`CRC_VALUE`	CRC Generator Current Value	0x40000F04
`CRC_ADD_1`	CRC Generator - Add 1 Bit	0x40000F08
`CRC_ADD_8`	CRC Generator - Add 1 Byte	0x40000F0C
`CRC_ADD_16`	CRC Generator - Add 1 Half-word	0x40000F10
`CRC_ADD_24`	CRC Generator - Add 3 Bytes	0x40000F14
`CRC_ADD_32`	CRC Generator - Add 1 Word	0x40000F18
`CRC_FINAL`	CRC Generator Final Value	0x40000F1C

12.1.1.1 CRC_CTRL

Bit Field	Field Name	Description
4	`FINAL_CRC_XOR`	Selects the final CRC XOR mode
3	`FINAL_CRC_REVERSE`	Selects the final CRC reversal mode
2	`BIT_ORDER`	Selects the bit order for bytes added to the CRC
1	`CRC_TYPE`	Selects the CRC type
0	`BYTE_ORDER`	Selects the endianness for bytes added to the CRC

Field Name	Value Symbol	Value Description	Hex Value
`FINAL_CRC_XOR`	`CRC_FINAL_XOR_STANDARD`	Final CRC XOR is done according to the standard (CRC-CCITT: no XOR; CRC-32: XOR with 0xFFFFFFFF)	0x0*
	`CRC_FINAL_XOR_NON_STANDARD`	Final CRC XOR is done in opposite of the standard	0x1

Field Name	Value Symbol	Value Description	Hex Value
FINAL_CRC_REVERSE	CRC_FINAL_REVERSE_STANDARD	Final CRC reversal is done according to the standard (CRC-CCITT: normal; CRC-32 reversed)	0x0*
	CRC_FINAL_REVERSE_NON_STANDARD	Final CRC reversal is done in opposite of the standard	0x1
BIT_ORDER	CRC_BIT_ORDER_STANDARD	Bit order is as defined by the standard (CRC-CCITT: normal; CRC-32 reversed)	0x0*
	CRC_BIT_ORDER_NON_STANDARD	Bit order is opposite of the standard	0x1
CRC_TYPE	CRC_CCITT	CRC-CCITT algorithm selected	0x0*
	CRC_32	CRC-32 (IEEE 802.3) algorithm selected	0x1
BYTE_ORDER	CRC_BIG_ENDIAN	Bytes are added to the CRC in big-endian order	0x0*
	CRC_LITTLE_ENDIAN	Bytes are added to the CRC in little-endian order	0x1

12.1.1.2 CRC_VALUE

Bit Field	Field Name	Description
31:0	CURRENT_CRC	CRC generator value: Write 0xFFFFFFFF (32) or 0xFFFF (CCITT) to initialize the CRC, read provides the current CRC value.

Field Name	Value Symbol	Value Description	Hex Value
CURRENT_CRC	CRC_CCITT_INIT_VALUE	Initial value for the CRC CCITT calculation	0xFFFF*
	CRC_32_INIT_VALUE	Initial value for the CRC 32 calculation	0xFFFFFFFF

12.1.1.3 CRC_ADD_1

Bit Field	Field Name	Description
0	CRC_ADD_1_BIT	Add 1 bit to the CRC calculation

12.1.1.4 CRC_ADD_8

Bit Field	Field Name	Description
7:0	CRC_ADD_8	Add 1 byte (8 bits) to the CRC calculation

12.1.1.5 CRC_ADD_16

Bit Field	Field Name	Description
15:0	CRC_ADD_16	Add 1 half-word (16 bits) to the CRC calculation

12.1.1.6 CRC_ADD_24

Bit Field	Field Name	Description
23:0	`CRC_ADD_24_BITS`	Add 3 bytes (24 bits) to the CRC calculation

12.1.1.7 CRC_ADD_32

Bit Field	Field Name	Description
31:0	`CRC_ADD_32`	Add 1 word (32 bits) to the CRC calculation

12.1.1.8 CRC_FINAL

Bit Field	Field Name	Description
31:0	`FINAL_CRC`	CRC generator final value: After XOR for CCITT or byte reversal for CRC-32

12.2 Direct Memory Access (DMA) Controller

12.2.1 Introduction

The direct memory access (DMA) controller module allows background transfers between peripherals and memory without core intervention. This allows the system core to be used for other computational needs while allowing high-speed sustained transfers to and from the peripherals. The DMA is connected to the Arm Cortex-M3 core, the processor’s peripherals and interfaces, and the processor’s data memory via four independent channels, as shown in Figure 35.

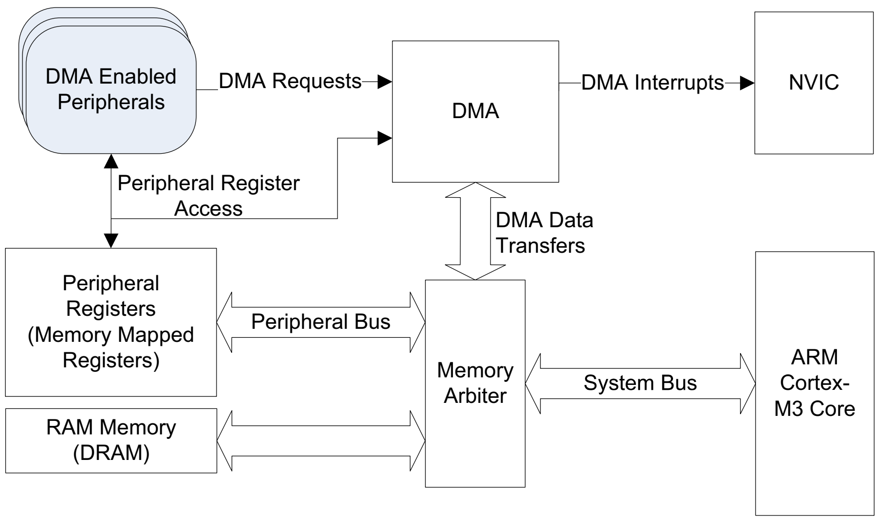


Figure 35. DMA Overview

12.2.2 DMA Channel Configuration

The DMA has eight independently configurable channels. To enable or disable a DMA channel, configure the `DMA_CTRL0_ENABLE` bit from the appropriate `DMA_CTRL0` register. If the DMA channel is in the midst of an operation when it is disabled, the operation is aborted. All pending bus requests are subsequently aborted when the channel is disabled.

Other DMA channel configurations include:

Source Address Configuration

The `DMA_SRC_BASE_ADDR` registers define the base address for source data for a DMA channel; the base address must be word aligned.

The `DMA_NEXT_SRC_ADDR` registers contain calculated values that indicate the address where the next word will be read from for a DMA channel. This pointer is updated with each word transferred, using the base source address, current transfer count of how many words have been read from the DMA channel's source (provided as `DMA_WORD_CNT`), and other source address configurations. Since the base source address is not modified during a transfer, the increment setting only affects the calculation of the next address.

NOTE: When the DMA channel is disabled, the next address is always the base address.

Each transfer can be configured to use a static location for the source data, or to increment the source data pointer after every word transferred. Configure this mode using the `DMA_CTRL0_SRC_ADDR_INC` bit from the appropriate `DMA_CTRL0` register. Transfers that use a peripheral as the source typically have source incrementing disabled. Conversely, transfers that use memory as the source typically have source incrementing enabled.

If the address is incremented, the increment can be configured to be positive or negative, and to use a step size of between one and four 32-bit words, by setting the `SRC_ADDR_STEP_MODE` bit and the `SRC_ADDR_STEP_SIZE` bit field from the appropriate `DMA_CTRL0` register.

Destination Address Configuration

The `DMA_DEST_BASE_ADDR` registers define the base address for destination data for a DMA channel; the base address must be word aligned.

The `DMA_NEXT_DEST_ADDR` registers contain calculated values that indicate the address where the next word will be written for a DMA channel. This pointer is updated with each word transferred, using the base destination address, current transfer count of how many words have been read from the DMA channel's source (provided as `DMA_WORD_CNT`), and other destination address configurations. Since the base destination address is not modified during a transfer, the increment setting only affects the calculation of the next address.

NOTE: When the DMA channel is disabled, the next address is always the base address.

Each transfer can be configured to use a static location for the destination data, or to increment the destination data pointer after every word transferred. Configure this mode using the `DMA_CTRL0_DEST_ADDR_INC` bit from the appropriate `DMA_CTRL0` register. Transfers that use a peripheral as the destination typically have destination incrementing disabled. Conversely, transfers that use memory as the destination typically have destination incrementing enabled.

If the address is incremented, the increment can be configured to be positive or negative, and to use a step size of between one and four 32-bit words, by setting the

DMA_CTRL0_DEST_ADDR_STEP_MODE bit and the DMA_CTRL0_DEST_ADDR_STEP_SIZE bit field from the appropriate DMA_CTRL0 register.

Source and Destination Data

The source data and destination data can be configured to be interpreted as packed into a variety of different word sizes and endian ordering. For more information, see Section 12.2.3, “Word Size, Data Packing and Transfer Length” on page 356.

Transfer Length and Intermediate Transfer Counter Configuration

The DMA channels support variable transfer lengths up to 2^{16} packed words (32 bits per word). The length of a transfer can be set using the DMA_CTRL1_TRANSFER_LENGTH bit field from the appropriate DMA_CTRL1 register, where a setting of 0 indicates a 2^{16}-word transfer.

The transfer can also be configured to indicate when a portion of the transfer has been completed, by setting the DMA_CTRL1_COUNTER_INT_VALUE bit field from the appropriate DMA_CTRL1 register. This counter value can be set for any portion of the transfer length, from one word, to one word less than the overall transfer length. The counter interrupt is never triggered if the counter interrupt value is set to 0, to the transfer length, or to any value that exceeds the transfer length.

Each transfer can be configured to run once (linear mode) and complete, or to run repeatedly (circular mode), producing a circular buffer. When operating in circular mode, the DMA channel’s transfer repeats until explicitly stopped by the user application. Configure this mode using the DMA_CTRL0_ADDR_MODE bit from the appropriate DMA_CTRL0 register.

For more information about transfer length, see Section 12.2.3.3, “Transfer Length” on page 359. For more information about interrupts indicating the transfer length and counter configuration, see Section 12.2.5, “DMA Interrupt Configuration” on page 363.

Transfer Types and Peripherals Used

Each transfer is defined as one of peripheral-to-memory, memory-to-peripheral, memory-to-memory, or peripheral-to-peripheral. The transfer mode used for a DMA channel is selected using the DMA_CTRL0_TRANSFER_TYPE bit field from its DMA_CTRL0 register.

For transfer types that use a source peripheral, select the desired source peripheral using the DMA_CTRL0_SRC_SELECT bit field from the DMA channel’s DMA_CTRL0 register.

For transfer types that use a destination peripheral, select the desired destination peripheral using the DMA_CTRL0_DEST_SELECT bit field from the DMA channel’s DMA_CTRL0 register.

For more information about transfer type behavior and the peripherals that can be used as sources and destinations for DMA transfers, see Section 12.2.4, “DMA Transfer Types” on page 359.

Interrupt Configuration

The interrupts used to coordinate with, and control, a DMA transfer can be defined using the DMA_CTRL0_*_INT_ENABLE bits from the DMA channel’s DMA_CTRL0 register. For more information, see Section 12.2.5, “DMA Interrupt Configuration” on page 363.

Transfer Priority

The relative priority of this DMA channel's transfer. For more information, see Section 12.2.6, "Channel Priority".

IMPORTANT: The DMA uses the current counter value to calculate the next source address and next destination address. The base source address and base destination address are not changed. The starting length is not changed either. Only the counter value is modified during a transfer. This allows the DMA configuration to be reused (either explicitly through firmware or in circular mode) without rewriting the configuration register for multiple transfers.

The DMA controller contains a set of summary status registers (`DMA_STATUS`) indicating the completion status of each DMA channel (idle or complete), and each channel's interrupt status. The firmware can use this to quickly assess the status of a DMA channel.

12.2.3 Word Size, Data Packing and Transfer Length

12.2.3.1 Word Size

Each DMA channel can have different word sizes set for the source and destination, as shown in Table 30. The word sizes can be configured as 4-bit, 8-bit, 16-bit, or 32-bit. The native memory word size is 32-bit; however, the Arm Cortex-M3 processor supports byte addresses. Therefore, smaller word sizes can be used to optimize memory utilization when retrieving or storing data.

Table 30. DMA Word Sizes

Encoded Word Size	Word Size
0x3	4 bits
0x0	8 bits
0x1	16 bits
0x2	32 bits

To set the source word size used, write the correct encoding to the `DMA_CTRL0_DEST_SRC_WORD_SIZE` bit-field in the appropriate `DMA_CTRL0` register. To set the destination word size used, write the correct encoding to the `DMA_CTRL0_DEST_DEST_WORD_SIZE` bit-field in the appropriate `DMA_CTRL0` register.

12.2.3.2 Data Packing

Packing and unpacking help to reduce bus utilization. These operations are performed automatically by the DMA controller when the source and destination have different word sizes. Specifically:

- When the source word size is smaller than the destination word size, packing is used to consolidate data from multiple source word transfers.
- When the destination word size is smaller than the source word size, unpacking is used to split data into multiple destination word transfers.

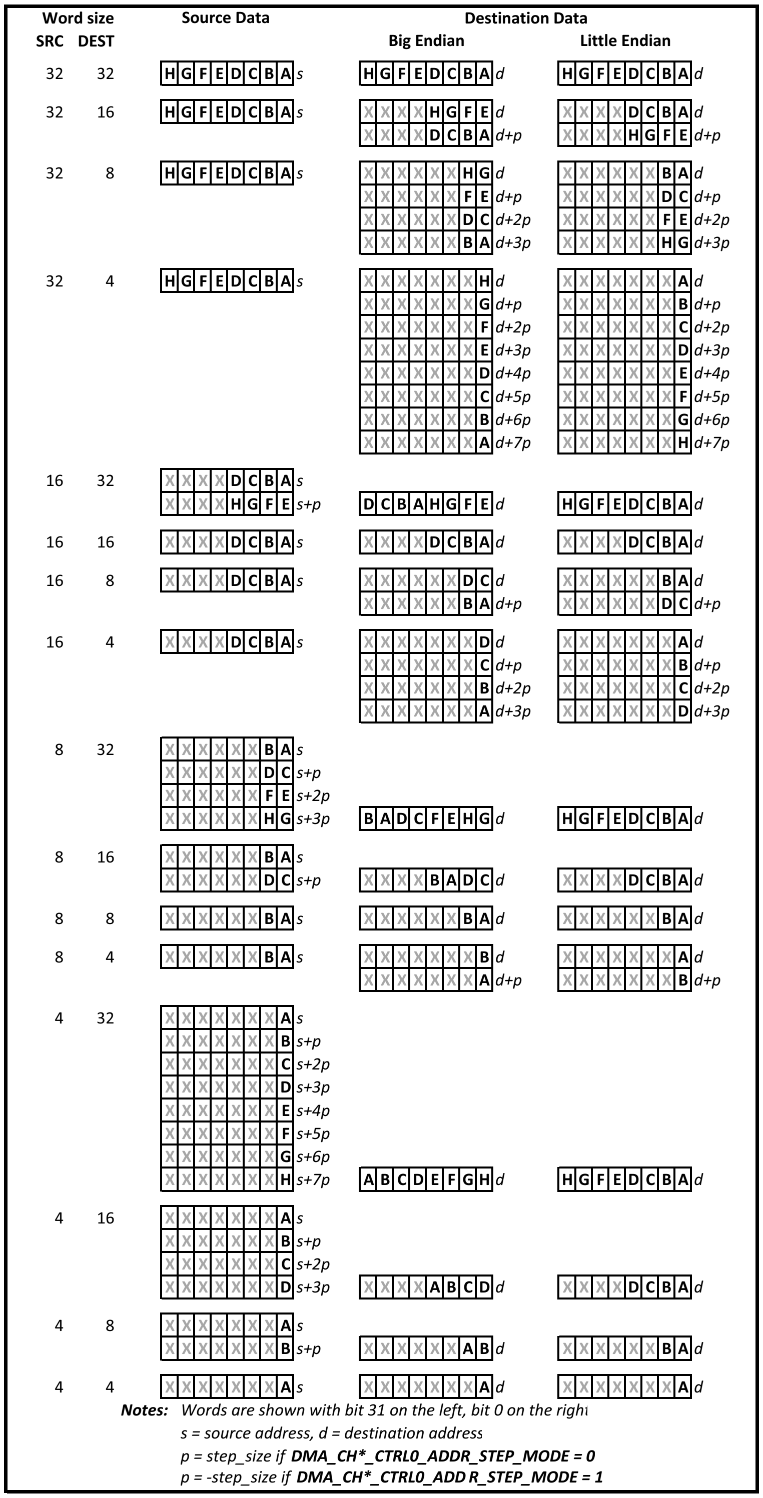
For example, when reading 8-bit bytes from the I^2C interface and storing them to data memory, four 8-bit data words from the I^2C interface are packed before they are written to memory. In this example, it is most efficient to make the destination word size 32 bits to minimize bus utilization and maximize memory use efficiency.

When both the source and destination have the same word size, no packing or unpacking of data is performed.

NOTE: When reading from or writing to memory, the DMA might utilize only a portion of the memory data if the selected word sizes differ.

The packing and unpacking behavior is illustrated in Figure 36. The figure provides a mapping of how data is transferred between source and destination for:

- All source and destination word size combinations
- Both big and little endian transfers



Notes: *Words are shown with bit 31 on the left, bit 0 on the right*
s = source address, d = destination address
p = step_size if ***DMA_CH*_CTRL0_ADDR_STEP_MODE = 0***
p = -step_size if ***DMA_CH*_CTRL0_ADDR_STEP_MODE = 1***

Figure 36. DMA Packing and Unpacking

For memory-to-peripheral transfers, the counter in the DMA channel always operates on the peripheral's word size. For peripheral-to-memory, peripheral-to-peripheral and memory-to-memory transfers, the counter in the DMA channel always operates on the source's word size. The next address for reading or writing can be determined from:

- The configured word size
- The current counter
- The step size and direction
- The base address

This can result in an actual read or write if no previously loaded data is available, or use of the temporary register if sufficient data has already been loaded.

When data is stored to memory, it is always right aligned. When the destination word size is larger than the source word size, the data is stored starting from the LSB. The upper bits are always written as zero (padded mode). If the configured destination word size is smaller than the configured source word size for a transfer to memory, the data is aligned at the LSB and truncated (truncate mode).

In packing or unpacking data, the endianness of the data is important because the order in which data bytes are extracted from a word is different between little and big endian. The DMA supports processing both little endian and big endian data, through the `DMA_CTRL0_BYTE_ORDER` bit from the `DMA_CTRL0` registers.

IMPORTANT: To provide a nibble ordering that is consistent with the byte endianness, the ordering of nibbles must match the byte ordering. This ensures that transfers which use data packing of 4-bit words—for either the source or destination—order data consistently.

NOTE: When packing data into memory for a transfer length that is not a multiple of the destination word size, the final data memory word is automatically zero padded, before being written to data memory at the end of the transfer.

12.2.3.3 Transfer Length

The transfer length and DMA counter depend on the transfer type, source word size, and destination word size.

- For Peripheral-to-Memory, Peripheral-to-Peripheral and Memory-to-Memory transfers, the transfer length equals the number of words of the specified word size at the source.
- For Memory-to-Peripheral transfers, the transfer length equals the number of words of the specified word size at the destination.

12.2.4 DMA Transfer Types

The DMA supports several types of data transfers between the Arm Cortex-M3 processor's memory and the interfaces and peripherals connected to the peripheral bus, including:

- Memory-to-Memory transfers
- Memory-to-Peripheral transfers
- Peripheral-to-Memory transfers
- Peripheral-to-Peripheral transfers

The interfaces and peripherals mapped onto the peripheral bus that are valid sources of data for a DMA data transfer are listed in Table 31.

Table 31. Valid Sources of DMA Data

Source	Encoded Bits
I^2C	0x0
SPI 0	0x1
SPI 1	0x2
PCM	0x3
UART	0x4
ASRC	0x5
PBUS	0x6
DMIC	0x7

Each source interface or peripheral, when configured for DMA operation, asserts its DMA request signal when data can be read from the interface. This signal is cleared automatically when a data value is read from the interface using the peripheral bus.

The interfaces and peripherals mapped onto the peripheral bus that are valid destinations for data from a DMA data transfer are listed in Table 32.

Table 32. Valid Destinations of DMA Data

Destination	Encoded Bits
I^2C	0x0
SPI 0	0x1
SPI 1	0x2
PCM	0x3
UART	0x4
ASRC	0x5
PBUS	0x6
OD	0x7

Each destination interface or peripheral, when configured for DMA operation, asserts its DMA request signal when data can be written to the peripheral bus. This signal is cleared automatically when a data value is written to that interface using the peripheral bus.

The peripheral bus (PBUS) option can be used as an uncontrolled DMA source or destination for transfers. This can be used to source or sink data from any register mapped onto the peripheral bus. When configured for this mode, the PBUS is configured as a peripheral that immediately accepts all DMA requests to the peripheral. An example use case for this configuration includes using the DMA to load a sequence of data into the CRC generator (see Section 12.1, "Cyclic Redundancy Check (CRC) Generator" on page 350).

CAUTION: When the DMA interface is used to control transfers that use an interface or peripheral, all accesses to that interface or peripheral's data registers using the peripheral bus clears the DMA request signals. If a DMA request signal is cleared due to an Arm Cortex-M3 processor access, the underlying DMA transfer becomes corrupted.

IMPORTANT: Due to the structure of the SPI interfaces, when a user application initializes a transmit transfer using an SPI interface controlled by a DMA channel, the user application must preload the SPI interface's transmit data register (`SPI*_TX_DATA`) with the first data word of the transfer. For more information about the SPI interfaces, see Section 11.7, "Serial Peripheral Interfaces (SPI)" on page 335.

NOTE: To ease understanding, throughout the remainder of this section, all interfaces and peripherals used in DMA transfers are called *peripherals* due to their memory-mapping onto the peripheral bus.

12.2.4.1 Memory-to-Memory (MM)

The DMA memory-to-memory mode is used to transfer data from one location in the Arm Cortex-M3 processor's data memory to another location in the same memory. This relieves the processor from using cycles for a simple copy operation. The operation sequence is as follows (assuming 32-bit operation):

1. The DMA requests access to the Arm Cortex-M3 processor's data memory via the Arm Cortex-M3 processor's data memory arbiter.
2. The DMA is granted access to the Arm Cortex-M3 processor's data memory.
3. The DMA reads the word at the next source address and stores the word in a temporary register.
4. The DMA releases the Arm Cortex-M3 processor's data memory.
5. The DMA requests access to the Arm Cortex-M3 processor's data memory via the Arm Cortex-M3 processor's data memory arbiter.
6. The DMA is granted access to the Arm Cortex-M3 processor's data memory.
7. The DMA writes the word at the next destination address.
8. The DMA releases the Arm Cortex-M3 processor's data memory.
9. The DMA channel's counter is incremented.
10. If the transfer length is not reached, the DMA waits one cycle before starting the next transfer. If the transfer length is reached and the DMA is in linear mode, the DMA channel switches to the complete state. Circular mode is not applicable to Memory-to-Memory transfers.
11. Any DMA interrupts that are triggered by this transfer are generated.

IMPORTANT: The maximum transfer rate for memory-to-memory transfers is one transfer every two SYSCLK cycles.

12.2.4.2 Memory-to-Peripheral

The DMA memory-to-peripheral mode is used to transfer data from the Arm Cortex-M3 processor's data memory to a peripheral on the peripheral bus. If the peripheral is configured to operate in DMA mode, the operation sequence is as follows (assuming 32-bit operation):

1. The DMA channel receives a DMA request from the peripheral.
2. The DMA requests access to the Arm Cortex-M3 processor's data memory via the Arm Cortex-M3 processor's data memory arbiter.
3. The DMA is granted access to the Arm Cortex-M3 processor's data memory.
4. The DMA reads the word at the next source address, storing it to a temporary register.
5. The DMA releases the Arm Cortex-M3 processor's data memory.
6. Through the peripheral bus bridge, the DMA requests access to the peripheral bus.
7. The DMA is granted access to the peripheral bus.
8. The DMA writes the word at the next destination address.
9. The DMA channel acknowledges the peripheral DMA request (implied by the previous write operation).
10. The DMA releases the peripheral bus.

11. The DMA channel’s counter is incremented.
12. If the transfer length is not reached, the DMA channel waits for the next peripheral DMA request. If the transfer length is reached and the DMA is in linear mode, the DMA channel switches to the complete state. If the DMA is in circular mode, the DMA resets the counter to 0 and remains enabled.
13. Any DMA interrupts that are triggered by this transfer are generated.

12.2.4.3 Peripheral-to-Memory (PM)

The DMA peripheral-to-memory mode is used to transfer data from a peripheral to the Arm Cortex-M3 processor’s data memory. If the peripheral is configured to operate in DMA mode, the operation sequence is as follows (assuming 32-bit operation):

1. The peripheral generates a new data word.
2. The peripheral asserts a signal on a DMA request line (indicating that the buffer is full).
3. The DMA channel receives a DMA request from the peripheral.
4. Through the peripheral bus bridge, the DMA requests access to the peripheral bus.
5. The DMA is granted access to the peripheral bus.
6. The DMA reads the word at the next source address and stores the word to a temporary register.
7. The DMA releases the peripheral bus.
8. The DMA channel acknowledges the peripheral DMA request (implied by the previous read operation).
9. The DMA requests access to the Arm Cortex-M3 processor’s data memory via the Arm Cortex-M3 processor’s data memory arbiter.
10. The DMA is granted access to the Arm Cortex-M3 processor’s data memory.
11. The DMA writes the word at the next destination address.
12. The DMA releases the Arm Cortex-M3 processor’s data memory.
13. The DMA channel’s counter is incremented.
14. If the transfer length is not reached, the DMA waits for the next peripheral DMA request. If the transfer length is reached and the DMA is in linear mode, the DMA channel switches to the complete state. If the DMA is in circular mode, the DMA resets the counter to 0 and remains enabled.
15. Any DMA interrupts that are triggered by this transfer are generated.

12.2.4.4 Peripheral-to-Peripheral (PP)

The DMA peripheral-to-peripheral mode is used to transfer data from a peripheral to another peripheral. If both peripherals are configured to operate in DMA mode, the operation sequence is as follows (assuming 32-bit operation):

1. The DMA channel receives a DMA request from both the source and destination peripherals in any order. For the source peripheral, the following operation is required to generate a request:
    a. The source peripheral generates a new data word.
    b. The source peripheral asserts a signal on a DMA request line (indicates that the buffer is full).
    c. The DMA channel receives a DMA request from the source peripheral.

    The destination peripheral generates a request when it can receive an additional data word.

2. Through the peripheral bus bridge, the DMA requests access to the peripheral bus.
3. The DMA is granted access to the peripheral bus.
4. The DMA reads the word at the next source address and stores the word to a temporary register.
5. The DMA releases the peripheral bus.
6. The DMA channel acknowledges the source peripheral DMA request (implied by the previous read operation).
7. Through the peripheral bus bridge, the DMA requests access to the peripheral bus.
8. The DMA is granted access to the peripheral bus.
9. The DMA writes the word at the next destination address.

10. The DMA channel acknowledges the destination peripheral DMA request (implied by the previous write operation).
11. The DMA releases the peripheral bus.
12. The DMA channel's counter is incremented.
13. If the transfer length is not reached, the DMA channel waits for the next peripheral DMA request. If the transfer length is reached and the DMA is in linear mode, the DMA channel switches to the complete state. If the DMA is in circular mode, the DMA resets the counter to 0 and remains enabled.
14. Any DMA interrupts triggered by this transfer are generated.

IMPORTANT: To ensure that DMA operations are atomic, the DMA channel operation does not begin until DMA requests are received from both peripherals.

12.2.5 DMA Interrupt Configuration

The DMA has a separate interrupt for each DMA channel. Each DMA channel can be configured using its `DMA_CTRL0` register to assert an interrupt for several independent conditions:

A transfer has started

Set the `DMA_CTRL0_START_INT_ENABLE` bit to enable this interrupt. A transfer is considered started when the first word is read from the DMA channel's source.

A transfer has reached the counter interrupt value

Set the `DMA_CTRL1_COUNTER_INT_VALUE` bit field in the channel's `DMA_CTRL1` register to configure this interrupt, and set the `DMA_CTRL0_COUNTER_INT_ENABLE` bit to enable it. If the counter is set to *n*, the transfer has reached the counter interrupt value when it writes the memory word to the DMA channel's destination that contains the n^{th} value read from the DMA channel's source.

A transfer is complete

Set the `DMA_CTRL0_COMPLETE_INT_ENABLE` bit to enable this interrupt. A transfer is considered complete when the last word is written to the DMA channel's destination.

An error has occurred

Set the `DMA_CTRL0_ERROR_INT_ENABLE` bit to enable this interrupt.

The channel is disabled

Set the `DMA_CTRL0_DISABLE_INT_ENABLE` bit to enable this interrupt.

The DMA channel status register for each channel (`DMA_STATUS`) indicates which interrupts have triggered using the `DMA_*_INT_STATUS` bits. When an interrupt is generated, the application is responsible for reading the status register for that channel to determine which interrupts have been triggered. For all interrupts, the application's interrupt handlers are responsible for clearing the status, and thus clearing the pending interrupt, using the `DMA_*_INT_CLEAR`.

NOTE: A start interrupt is generated each time a transfer starts (including when a circular transfer restarts). Similarly, a complete interrupt is generated each time a transfer is completed. A disable interrupt is triggered whenever the channel is disabled for any reason (including when a linear transfer is completed).

When using a circular DMA transfer, the counter interrupt and complete interrupt can be used in tandem to create a two-page buffer for continuous data transfers. When the first page has transferred, the counter interrupt triggers. When the second page has transferred, the complete interrupt triggers. In this configuration:

For a memory-to-peripheral transfer

The counter interrupt indicates that data in the first page of the source buffer has transferred, and can be replaced while the second page is transferred. Similarly, the complete interrupt indicates that the second page can be replaced while the replacement first page is transferred.

For peripheral-to-memory transfers

The counter interrupt indicates when the first page of the buffer can be processed. The complete interrupt indicates when the second page of the buffer can be processed.

For debug purposes, the `DMA_STATUS` registers also include the `DMA_STATUS_STATE` bit-field which indicates the DMA channel's current state. This can be used to investigate why a DMA transfer has stalled if it does not complete, as it indicates whether the DMA is idle, is waiting on the source or destination, is waiting for a read/write to occur, or is in another intermediate state.

12.2.6 Channel Priority

Only one transfer using the DMA channels can be actively serviced at a time. To coordinate between DMA channels, the DMA contains a channel arbiter that is responsible for determining which DMA channel is active. The relative priority of each channel can be set using the `DMA_CTRL0_CHANNEL_PRIORITY` bit field from the `DMA_CTRL0` registers.

When a DMA channel receives DMA requests from the peripherals it is configured to use, it indicates to the arbiter that it is ready to transfer one word. Each DMA channel request is handled automatically. Several situations where multiple requests might be pending are:

- Two or more DMA channels receive a DMA request during the same clock cycle.
- One or more DMA requests come in during the processing of another DMA request.
- A DMA channel is enabled with multiple requests already pending.

When choosing which DMA channel to activate, the DMA arbiter applies the channel's priority settings as follows:

- When multiple channels are ready, the channel with the highest channel priority setting is activated.
- If more than one channel is ready to be activated and they share the same priority setting, the lowest numbered channel is activated.

When a single channel is ready, the arbiter grants access and the channel begins to service that request immediately.

IMPORTANT: A lower priority DMA channel might never be served if a higher priority DMA channel is generating requests too fast. This type of situation must be avoided by application design.

12.2.7 Data Memory Usage by the DMA and ARM Cortex-M3 Processor

The DMA has direct access to the Arm Cortex-M3 processor data memory and LPDSP32 data memories. This access is limited by a system bus arbiter to avoid memory access conflicts between the DMA, the Arm Cortex-M3 processor, and the LPDSP32 DSP.

When the DMA is granted access to memory, it performs a single operation (read or write) and releases the processor data memory. This ensures that the DMA never blocks access to data memory from the processor for more than a single memory operation.

Configuration of the memory arbiter for each memory instance uses the `SYSCTRL_MEM_ARBITER_CFG` register to select between prioritizing the Arm Cortex-M3 processor, prioritizing LPDSP32, or selecting one of two round-robin arbitration schemes:

- If the Arm Cortex-M3 is given priority access to a memory, access to the memory is given to the Arm Cortex-M3 processor, then to the LPDSP32 DSP if the Arm Cortex-M3 processor is not using it, and finally to the DMA if it would otherwise be idle.
- If the LPDSP32 DSP is given priority access to a memory, access to the memory is given to the LPDSP32 DSP, then to the Arm Cortex-M3 processor if the LPDSP32 DSP is not using it, and finally to the DMA if it would otherwise be idle.
- If the `SYSCTRL_MEM_ARBITER_CFG_ROUND_ROBIN_TOKEN` bit indicates a real-time DMA round robin scheme, the system behaves as though the Arm Cortex-M3 processor has been given priority access unless the DMA has been blocked for 7 consecutive SYSCLK cycles. If the DMA remains blocked after 7 consecutive cycles, the DMA is temporarily given the highest priority until it releases the memory again.
- If the `SYSCTRL_MEM_ARBITER_CFG_ROUND_ROBIN_TOKEN` bit indicates a normal round robin scheme, priority rotates in sequence between the Arm Cortex-M3 processor, the LPDSP32 DSP, and the DMA.

  Assuming that the Arm Cortex-M3 processor is performing only normal (non-bit-banded) memory transfers, the worst-case access time to the Arm Cortex-M3 processor data memory for DMA transfers can be established with high certainty. Typically, if the DMA and the processor are both always requesting data memory access, they can each utilize the memory 50% of the time.

NOTE: If the Arm Cortex-M3 processor is utilizing bit-banding operations, data memory access for the LPDSP32 and DMA might be delayed by an additional cycle because the memory controller uses an additional cycle for these operations.

For more information about memories and memory arbitration, see Chapter 7, “Memory” on page 93.

12.2.8 DMA Registers

Register Name	Register Description	Address
`DMA_CTRL0[0]`	DMA Channel Control and Configuration 0	0x40000600
`DMA_CTRL0[1]`	DMA Channel Control and Configuration 1	0x40000604
`DMA_CTRL0[2]`	DMA Channel Control and Configuration 2	0x40000608
`DMA_CTRL0[3]`	DMA Channel Control and Configuration 3	0x4000060C
`DMA_CTRL0[4]`	DMA Channel Control and Configuration 4	0x40000610
`DMA_CTRL0[5]`	DMA Channel Control and Configuration 5	0x40000614
`DMA_CTRL0[6]`	DMA Channel Control and Configuration 6	0x40000618
`DMA_CTRL0[7]`	DMA Channel Control and Configuration 7	0x4000061C
`DMA_SRC_BASE_ADDR[0]`	DMA Channel Source Base Address 0	0x40000620
`DMA_SRC_BASE_ADDR[1]`	DMA Channel Source Base Address 1	0x40000624
`DMA_SRC_BASE_ADDR[2]`	DMA Channel Source Base Address 2	0x40000628
`DMA_SRC_BASE_ADDR[3]`	DMA Channel Source Base Address 3	0x4000062C

Register Name	Register Description	Address
DMA_SRC_BASE_ADDR[4]	DMA Channel Source Base Address 4	0x40000630
DMA_SRC_BASE_ADDR[5]	DMA Channel Source Base Address 5	0x40000634
DMA_SRC_BASE_ADDR[6]	DMA Channel Source Base Address 6	0x40000638
DMA_SRC_BASE_ADDR[7]	DMA Channel Source Base Address 7	0x4000063C
DMA_DEST_BASE_ADDR[0]	DMA Channel Destination Base Address 0	0x40000640
DMA_DEST_BASE_ADDR[1]	DMA Channel Destination Base Address 1	0x40000644
DMA_DEST_BASE_ADDR[2]	DMA Channel Destination Base Address 2	0x40000648
DMA_DEST_BASE_ADDR[3]	DMA Channel Destination Base Address 3	0x4000064C
DMA_DEST_BASE_ADDR[4]	DMA Channel Destination Base Address 4	0x40000650
DMA_DEST_BASE_ADDR[5]	DMA Channel Destination Base Address 5	0x40000654
DMA_DEST_BASE_ADDR[6]	DMA Channel Destination Base Address 6	0x40000658
DMA_DEST_BASE_ADDR[7]	DMA Channel Destination Base Address 7	0x4000065C
DMA_CTRL1[0]	DMA Channel Transfer Control 0	0x40000660
DMA_CTRL1[1]	DMA Channel Transfer Control 1	0x40000664
DMA_CTRL1[2]	DMA Channel Transfer Control 2	0x40000668
DMA_CTRL1[3]	DMA Channel Transfer Control 3	0x4000066C
DMA_CTRL1[4]	DMA Channel Transfer Control 4	0x40000670
DMA_CTRL1[5]	DMA Channel Transfer Control 5	0x40000674
DMA_CTRL1[6]	DMA Channel Transfer Control 6	0x40000678
DMA_CTRL1[7]	DMA Channel Transfer Control 7	0x4000067C
DMA_NEXT_SRC_ADDR[0]	DMA Channel Next Source Address 0	0x40000680
DMA_NEXT_SRC_ADDR[1]	DMA Channel Next Source Address 1	0x40000684
DMA_NEXT_SRC_ADDR[2]	DMA Channel Next Source Address 2	0x40000688
DMA_NEXT_SRC_ADDR[3]	DMA Channel Next Source Address 3	0x4000068C
DMA_NEXT_SRC_ADDR[4]	DMA Channel Next Source Address 4	0x40000690
DMA_NEXT_SRC_ADDR[5]	DMA Channel Next Source Address 5	0x40000694
DMA_NEXT_SRC_ADDR[6]	DMA Channel Next Source Address 6	0x40000698
DMA_NEXT_SRC_ADDR[7]	DMA Channel Next Source Address 7	0x4000069C
DMA_NEXT_DEST_ADDR[0]	DMA Channel Next Destination Address 0	0x400006A0
DMA_NEXT_DEST_ADDR[1]	DMA Channel Next Destination Address 1	0x400006A4
DMA_NEXT_DEST_ADDR[2]	DMA Channel Next Destination Address 2	0x400006A8
DMA_NEXT_DEST_ADDR[3]	DMA Channel Next Destination Address 3	0x400006AC
DMA_NEXT_DEST_ADDR[4]	DMA Channel Next Destination Address 4	0x400006B0
DMA_NEXT_DEST_ADDR[5]	DMA Channel Next Destination Address 5	0x400006B4
DMA_NEXT_DEST_ADDR[6]	DMA Channel Next Destination Address 6	0x400006B8
DMA_NEXT_DEST_ADDR[7]	DMA Channel Next Destination Address 7	0x400006BC
DMA_WORD_CNT[0]	DMA Channel Word Count 0	0x400006C0
DMA_WORD_CNT[1]	DMA Channel Word Count 1	0x400006C4
DMA_WORD_CNT[2]	DMA Channel Word Count 2	0x400006C8

Register Name	Register Description	Address
DMA_WORD_CNT[3]	DMA Channel Word Count 3	0x400006CC
DMA_WORD_CNT[4]	DMA Channel Word Count 4	0x400006D0
DMA_WORD_CNT[5]	DMA Channel Word Count 5	0x400006D4
DMA_WORD_CNT[6]	DMA Channel Word Count 6	0x400006D8
DMA_WORD_CNT[7]	DMA Channel Word Count 7	0x400006DC
DMA_STATUS[0]	DMA Status channel 0	0x400006E0
DMA_STATUS[1]	DMA Status channel 1	0x400006E4
DMA_STATUS[2]	DMA Status channel 2	0x400006E8
DMA_STATUS[3]	DMA Status channel 3	0x400006EC
DMA_STATUS[4]	DMA Status channel 4	0x400006F0
DMA_STATUS[5]	DMA Status channel 5	0x400006F4
DMA_STATUS[6]	DMA Status channel 6	0x400006F8
DMA_STATUS[7]	DMA Status channel 7	0x400006FC

12.2.8.1 DMA_CTRL0

The following bit fields and field names apply equally to all DMA_CTRL0[*] registers.

Bit Field	Field Name	Description
31:30	DEST_ADDR_STEP_SIZE	Select the destination address step size
29:28	SRC_ADDR_STEP_SIZE	Select the source address step size
27	DEST_ADDR_STEP_MODE	Configure the destination address to either increment or decrement
26	SRC_ADDR_STEP_MODE	Configure the source address to either increment or decrement
25	BYTE_ORDER	Select the byte ordering for the DMA channel
24	DISABLE_INT_ENABLE	Raise an interrupt when the DMA channel is disabled
23	ERROR_INT_ENABLE	Raise an interrupt when a state machine error occurs during a DMA transfer
22	COMPLETE_INT_ENABLE	Raise an interrupt when the DMA transfer completes
21	COUNTER_INT_ENABLE	Raise an interrupt when the DMA transfer reaches the counter value
20	START_INT_ENABLE	Raise an interrupt when the DMA transfer starts
19:18	DEST_WORD_SIZE	Select the destination word size
17:16	SRC_WORD_SIZE	Select the source word size
14:12	DEST_SELECT	Select the request line for the destination
11:9	SRC_SELECT	Select the request line for the source
7:6	CHANNEL_PRIORITY	Select the priority level for this channel
5:4	TRANSFER_TYPE	Select the type of transfer implemented by DMA channel
3	DEST_ADDR_INC	Configure whether the destination address should increment
2	SRC_ADDR_INC	Configure whether the source address should increment
1	ADDR_MODE	Select the addressing mode for this channel
0	ENABLE	Enable DMA Channel

Field Name	Value Symbol	Value Description	Hex Value
DEST_ADDR_STEP_SIZE	DMA_DEST_ADDR_STEP_SIZE_1	Set the step size of DMA channel to 1	0x0*
	DMA_DEST_ADDR_STEP_SIZE_2	Set the step size of DMA channel to 2	0x1
	DMA_DEST_ADDR_STEP_SIZE_3	Set the step size of DMA channel to 3	0x2
	DMA_DEST_ADDR_STEP_SIZE_4	Set the step size of DMA channel to 4	0x3
SRC_ADDR_STEP_SIZE	DMA_SRC_ADDR_STEP_SIZE_1	Set the step size of DMA channel to 1	0x0*
	DMA_SRC_ADDR_STEP_SIZE_2	Set the step size of DMA channel to 2	0x1
	DMA_SRC_ADDR_STEP_SIZE_3	Set the step size of DMA channel to 3	0x2
	DMA_SRC_ADDR_STEP_SIZE_4	Set the step size of DMA channel to 4	0x3
DEST_ADDR_STEP_MODE	DMA_DEST_ADDR_POS	Increment the destination address used by DMA channel	0x0*
	DMA_DEST_ADDR_NEG	Decrement destination address used by DMA channel	0x1
SRC_ADDR_STEP_MODE	DMA_SRC_ADDR_POS	Increment the source address used by DMA channel	0x0*
	DMA_SRC_ADDR_NEG	Decrement source address used by DMA channel	0x1
BYTE_ORDER	DMA_LITTLE_ENDIAN	DMA channel uses little endian mode	0x0*
	DMA_BIG_ENDIAN	DMA channel uses big endian mode	0x1
DISABLE_INT_ENABLE	DMA_DISABLE_INT_DISABLE	Disable channel disable indicator interrupts for DMA channel	0x0*
	DMA_DISABLE_INT_ENABLE	Enable channel disable indicator interrupts for DMA channel	0x1
ERROR_INT_ENABLE	DMA_ERROR_INT_DISABLE	Disable error indicator interrupts for DMA channel	0x0*
	DMA_ERROR_INT_ENABLE	Enable error indicator interrupts for DMA channel	0x1
COMPLETE_INT_ENABLE	DMA_COMPLETE_INT_DISABLE	Disable completion interrupts for DMA channel	0x0*
	DMA_COMPLETE_INT_ENABLE	Enable completion interrupts for DMA channel	0x1
COUNTER_INT_ENABLE	DMA_COUNTER_INT_DISABLE	Disable counter interrupts for DMA channel	0x0*
	DMA_COUNTER_INT_ENABLE	Enable counter interrupts for DMA channel	0x1
START_INT_ENABLE	DMA_START_INT_DISABLE	Disable start interrupts for DMA channel	0x0*
	DMA_START_INT_ENABLE	Enable start interrupts for DMA channel	0x1
DEST_WORD_SIZE	DMA_DEST_WORD_SIZE_8	Destination data uses 8-bit words	0x0*
	DMA_DEST_WORD_SIZE_16	Destination data uses 16-bit words	0x1
	DMA_DEST_WORD_SIZE_32	Destination data uses 32-bit words	0x2
	DMA_DEST_WORD_SIZE_4	Destination data uses 4-bit words	0x3

Field Name	Value Symbol	Value Description	Hex Value
SRC_WORD_SIZE	DMA_SRC_WORD_SIZE_8	Source data uses 8-bit words	0x0*
	DMA_SRC_WORD_SIZE_16	Source data uses 16-bit words	0x1
	DMA_SRC_WORD_SIZE_32	Source data uses 32-bit words	0x2
	DMA_SRC_WORD_SIZE_4	Source data uses 4-bit words	0x3
DEST_SELECT	DMA_DEST_I2C	Data writes are triggered by the I2C request line	0x0*
	DMA_DEST_SPI0	Data writes are triggered by the SPI0 request line	0x1
	DMA_DEST_SPI1	Data writes are triggered by the SPI1 request line	0x2
	DMA_DEST_PCM	Data writes are triggered by the PCM request line	0x3
	DMA_DEST_UART	Data writes are triggered by the UART request line	0x4
	DMA_DEST_ASRC	Data writes are triggered by the ASRC request line	0x5
	DMA_DEST_PBUS	Data writes are triggered by the PBUS request line	0x6
	DMA_DEST_OD	Data writes are triggered by the OD request line	0x7
SRC_SELECT	DMA_SRC_I2C	Data reads are triggered by the I2C request line	0x0*
	DMA_SRC_SPI0	Data reads are triggered by the SPI0 request line	0x1
	DMA_SRC_SPI1	Data reads are triggered by the SPI1 request line	0x2
	DMA_SRC_PCM	Data reads are triggered by the PCM request line	0x3
	DMA_SRC_UART	Data reads are triggered by the UART request line	0x4
	DMA_SRC_ASRC	Data reads are triggered by the ASRC request line	0x5
	DMA_SRC_PBUS	Data reads are triggered by the PBUS request line	0x6
	DMA_SRC_DMIC	Data reads are triggered by the DMIC request line	0x7
CHANNEL_PRIORITY	DMA_PRIORITY_0	Set the priority of DMA channel to 0 (Lowest)	0x0*
	DMA_PRIORITY_1	Set the priority of DMA channel to 1	0x1
	DMA_PRIORITY_2	Set the priority of DMA channel to 2	0x2
	DMA_PRIORITY_3	Set the priority of DMA channel to 3 (Highest)	0x3

Field Name	Value Symbol	Value Description	Hex Value
TRANSFER_TYPE	DMA_TRANSFER_M_TO_M	DMA channel will provide a memory-to-memory data transfer	0x0*
	DMA_TRANSFER_M_TO_P	DMA channel will provide a memory-to-peripheral data transfer	0x1
	DMA_TRANSFER_P_TO_M	DMA channel will provide a peripheral-to-memory data transfer	0x2
	DMA_TRANSFER_P_TO_P	DMA channel will provide a peripheral-to-peripheral data transfer	0x3
DEST_ADDR_INC	DMA_DEST_ADDR_STATIC	Do not increment the destination address used by DMA channel	0x0*
	DMA_DEST_ADDR_INC	Increment destination address used by DMA channel	0x1
SRC_ADDR_INC	DMA_SRC_ADDR_STATIC	Do not increment the source address used by DMA channel	0x0*
	DMA_SRC_ADDR_INC	Increment source address used by DMA channel	0x1
ADDR_MODE	DMA_ADDR_CIRC	Use circular addressing for DMA channel	0x0*
	DMA_ADDR_LIN	Use linear addressing for DMA channel	0x1
ENABLE	DMA_DISABLE	Disable DMA channel (channel will wait on current transfer before stopping)	0x0*
	DMA_ENABLE	Enable DMA channel	0x1

12.2.8.2 DMA_SRC_BASE_ADDR

The following bit fields and field names apply equally to all `DMA_SRC_BASE_ADDR[*]` registers.

Bit Field	Field Name	Description
31:0	DMA_SRC_BASE_ADDR	Base address for the source of data transferred using DMA channel

12.2.8.3 DMA_DEST_BASE_ADDR

The following bit fields and field names apply equally to all `DMA_DEST_BASE_ADDR[*]` registers.

Bit Field	Field Name	Description
31:0	DMA_DEST_BASE_ADDR	Base address for the destination of data transferred using DMA channel

12.2.8.4 DMA_CTRL1

The following bit fields and field names apply equally to all `DMA_CTRL1[*]` registers.

Bit Field	Field Name	Description
31:16	COUNTER_INT_VALUE	Trigger a counter interrupt when the DMA transfer word count reaches this value
15:0	TRANSFER_LENGTH	The length, in words, of each data transfer using DMA channel

12.2.8.5 DMA_NEXT_SRC_ADDR

The following bit fields and field names apply equally to all `DMA_NEXT_SRC_ADDR[*]` registers.

Bit Field	Field Name	Description
31:0	DMA_NEXT_SRC_ADDR	Address of the next data to be transferred using DMA channel

12.2.8.6 DMA_NEXT_DEST_ADDR

The following bit fields and field names apply equally to all `DMA_NEXT_DEST_ADDR[*]` registers.

Bit Field	Field Name	Description
31:0	DMA_NEXT_DEST_ADDR	Address where the next data to be transferred using DMA channel will be stored

12.2.8.7 DMA_WORD_CNT

The following bit fields and field names apply equally to all `DMA_WORD_CNT[*]` registers.

Bit Field	Field Name	Description
15:0	DMA_WORD_CNT	The number of words that have been transferred using DMA channel during the current transfer

12.2.8.8 DMA_STATUS

The following bit fields and field names apply equally to all `DMA_STATUS[*]` registers.

Bit Field	Field Name	Description
12	ERROR_INT_CLEAR	Clear the state machine error interrupt flag
11	COMPLETE_INT_CLEAR	Clear the complete interrupt flag
10	COUNTER_INT_CLEAR	Clear the counter interrupt flag
9	START_INT_CLEAR	Clear the start interrupt flag
8	DISABLE_INT_CLEAR	Clear the channel disable flag
7:5	STATE	DMA channel state
4	ERROR_INT_STATUS	Indicate if a state machine error interrupt has occurred on DMA channel
3	COMPLETE_INT_STATUS	Indicate if a complete interrupt has occurred on DMA channel
2	COUNTER_INT_STATUS	Indicate if a counter interrupt has occurred on DMA channel
1	START_INT_STATUS	Indicate if a start interrupt has occurred on DMA channel
0	DISABLE_INT_STATUS	Indicate if a channel disable interrupt has occurred on DMA channel

Field Name	Value Symbol	Value Description	Hex Value
ERROR_INT_CLEAR	DMA_ERROR_INT_CLEAR	Clear the state machine error interrupt flag	0x1
COMPLETE_INT_CLEAR	DMA_COMPLETE_INT_CLEAR	Clear the complete interrupt flag	0x1
COUNTER_INT_CLEAR	DMA_COUNTER_INT_CLEAR	Clear the counter interrupt flag	0x1

Field Name	Value Symbol	Value Description	Hex Value
START_INT_CLEAR	DMA_START_INT_CLEAR	Clear the start interrupt flag	0x1
DISABLE_INT_CLEAR	DMA_DISABLE_INT_CLEAR	Clear the channel disable flag	0x1
STATE	DMA_IDLE	Indicate that DMA channel is idle	0x0*
	DMA_CHK_SRC_DST	Indicate that the DMA is checking the next source and destination addresses	0x1
	DMA_WAIT_SRC_RDY	Indicate that DMA channel is waiting on the source	0x2
	DMA_WAIT_RD_COMP	Indicate the DMA is waiting for a scheduled read to complete	0x3
	DMA_CHK_DST	Indicate that the DMA is checking the destination address	0x4
	DMA_DIS_CH	Indicate that this DMA channel is transitioning to a disabled state.	0x5
	DMA_WAIT_DST_RDY	Indicate that DMA channel is waiting on the destination	0x6
	DMA_WAIT_WR_COMP	Indicate that the DMA is waiting for a scheduled write to complete	0x7
ERROR_INT_STATUS	DMA_ERROR_INT_STATUS	Indicate that a state machine error interrupt has occurred	0x1
COMPLETE_INT_STATUS	DMA_COMPLETE_INT_STATUS	Indicate that a complete interrupt has occurred	0x1
COUNTER_INT_STATUS	DMA_COUNTER_INT_STATUS	Indicate that a counter interrupt has occurred	0x1
START_INT_STATUS	DMA_START_INT_STATUS	Indicate that a start interrupt has occurred	0x1
DISABLE_INT_STATUS	DMA_DISABLE_INT_STATUS	Indicate that a channel disable interrupt has occurred	0x1

12.3 TIMERS

The RSL10 system provides five timers including:

- The SysTick timer from the Arm Cortex-M3 processor, which is described in Section 14.2, “SysTick” on page 408
- Four general-purpose timers

Each general-purpose timer provides:

- A 24-bit counter
- A pair of configurable prescalers controlling a fixed prescale by 2 or 32, as well as a variable 3-bit prescale factor
- 3 operating modes: single-shot, multiple-shot, and free-run
- A dedicated interrupt that can be used to signal timer expiration
- Dedicated configuration and status registers

The general-purpose timers are clocked from a divided form of slow clock. The divisor for each clock is configurable by a pair of prescaling factors set using the appropriate `TIMER_CFG_*` register:

- The `TIMER_CFG_CLK_SRC` bit is used to select between fixed initial division of slow clock by 2 or by 32

- The TIMER_CFG_PRESCALE bit field is used to increase the scaling factor by a power of two, allowing the selection of a clock that is additionally divided by as much as 128 for each timer.

These prescaling divisions result in a wider range of timing that the timers can achieve; however, the granularity at which the timer can be configured to trigger increases in parallel.

After prescaling slow clock, each timer can be configured to trigger after 1 to 2^{24} cycles of the prescaled clock by setting the TIMER_CFG_TIMEOUT_VALUE bit field in the appropriate TIMER_CFG_* register. The resulting timer delay is equal to:

$$\mathrm{DELAY} = \frac{2^{(\mathrm{TIMER\_CFG\_CLK\_SRC} + \mathrm{TIMER\_CFG\_PRESCALE})} \times (\mathrm{TIMER\_CFG\_TIMEOUT\_VALUE} + 1)}{f_{\mathrm{SLOWCLK}}}$$

12.3.1 Starting or Stopping Timers

The state of a timer can be read from the TIMER_CTRL_TIMER_STATUS bit of its TIMER_CTRL_* register.

If the timer is not running, it can be started by setting the TIMER_CTRL_TIMER_START bit of its TIMER_CTRL_* register. If the timer is running, setting this same bit restarts the timer by reloading the timer value.

Each timer can be stopped at any time by setting the TIMER_CTRL_TIMER_STOP bit of its TIMER_CTRL_* register.

12.3.2 Mode Selection

In Free-Run Mode, the timer loads the initial time-out value and counts down to 0. When it reaches 0, it issues an interrupt, and reloads the time-out value and restarts the countdown timer. The process is repeated indefinitely until the timer is explicitly stopped by writing a 1 to the TIMER_CTRL_TIMER_STOP bit of the TIMER_CTRL_* register.

In Multi-Shot Mode, the timer loads the initial time-out value and counts down to 0. When it reaches 0, it issues an interrupt, and checks the TIMER_CFG_MULTI_COUNT bit field from the TIMER_CFG_* to determine if it must restart the countdown timer. This process repeats (TIMER_CFG_MULTI_COUNT + 1) times before disabling the timer, unless explicitly stopped by the TIMER_STOP bit. Single-shot mode is a special case of Multi-Shot Mode where the timer is configured to trigger an interrupt only one time.

12.3.3 Timer Registers

Register Name	Register Description	Address
TIMER_CFG[0]	Timer configuration register 0	0x40000400
TIMER_CFG[1]	Timer configuration register 1	0x40000404
TIMER_CFG[2]	Timer configuration register 2	0x40000408
TIMER_CFG[3]	Timer configuration register 3	0x4000040C
TIMER_CTRL[0]	General-purpose timer control/status register 0	0x40000410
TIMER_CTRL[1]	General-purpose timer control/status register 1	0x40000414
TIMER_CTRL[2]	General-purpose timer control/status register 2	0x40000418
TIMER_CTRL[3]	General-purpose timer control/status register 3	0x4000041C
TIMER_VAL[0]	Timer current value register 0	0x40000420
TIMER_VAL[1]	Timer current value register 1	0x40000424

Register Name	Register Description	Address
TIMER_VAL[2]	Timer current value register 2	0x40000428
TIMER_VAL[3]	Timer current value register 3	0x4000042C

12.3.3.1 TIMER_CFG

The following bit fields and field names apply equally to all TIMER_CFG[*] registers.

Bit Field	Field Name	Description
31:29	MULTI_COUNT	Multi-count value
28	MODE	Timer mode
27	CLK_SRC	Clock source
26:24	PRESCALE	Prescale value of the timer
23:0	TIMEOUT_VALUE	Number of Timer clock cycles to time-out

Field Name	Value Symbol	Value Description	Hex Value
MULTI_COUNT	TIMER_MULTI_COUNT_1	Stop on 1st Time-out occurrence and issue an interrupt	0x0*
	TIMER_MULTI_COUNT_2	Stop on 2nd Time-out occurrence and issue an interrupt	0x1
	TIMER_MULTI_COUNT_3	Stop on 3rd Time-out occurrence and issue an interrupt	0x2
	TIMER_MULTI_COUNT_4	Stop on 4th Time-out occurrence and issue an interrupt	0x3
	TIMER_MULTI_COUNT_5	Stop on 5th Time-out occurrence and issue an interrupt	0x4
	TIMER_MULTI_COUNT_6	Stop on 6th Time-out occurrence and issue an interrupt	0x5
	TIMER_MULTI_COUNT_7	Stop on 7th Time-out occurrence and issue an interrupt	0x6
	TIMER_MULTI_COUNT_8	Stop on 8th Time-out occurrence and issue an interrupt	0x7
MODE	TIMER_SHOT_MODE	Enable the one shot / multi shot mode	0x0*
	TIMER_FREE_RUN	Enable the Free-Run Mode	0x1
CLK_SRC	TIMER_SLOWCLK_DIV32	Timer clock source is SLOWCLK divided by 32	0x0*
	TIMER_SLOWCLK_DIV2	Timer clock source is SLOWCLK divided by 2	0x1

Field Name	Value Symbol	Value Description	Hex Value
PRESCALE	TIMER_PRESCALE_1	Divide the input clock frequency by 1	0x0*
	TIMER_PRESCALE_2	Divide the input clock frequency by 2	0x1
	TIMER_PRESCALE_4	Divide the input clock frequency by 4	0x2
	TIMER_PRESCALE_8	Divide the input clock frequency by 8	0x3
	TIMER_PRESCALE_16	Divide the input clock frequency by 16	0x4
	TIMER_PRESCALE_32	Divide the input clock frequency by 32	0x5
	TIMER_PRESCALE_64	Divide the input clock frequency by 64	0x6
	TIMER_PRESCALE_128	Divide the input clock frequency by 128	0x7

12.3.3.2 TIMER_CTRL

The following bit fields and field names apply equally to all TIMER_CTRL[*] registers.

Bit Field	Field Name	Description
2	TIMER_STATUS	Indicate if the timer is active or not
1	TIMER_START	Start or restart the timer
0	TIMER_STOP	Stop the timer

Field Name	Value Symbol	Value Description	Hex Value
TIMER_STATUS	TIMER_INACTIVE	The timer is inactive	0x0*
	TIMER_ACTIVE	The timer is active	0x1
TIMER_START	TIMER_START	Writing a 1 will start or restart the timer	0x1
TIMER_STOP	TIMER_STOP	Writing a 1 will stop the timer	0x1

12.3.3.3 TIMER_VAL

The following bit fields and field names apply equally to all `TIMER_VAL[*]` registers.

Bit Field	Field Name	Description
23:0	TIMER_VALUE	Current timer (0, 1, 2 or 3) value

12.4 WATCHDOG TIMER

The watchdog timer is a safety system that resets a system that has malfunctioned. This safety system uses a countdown timer that must be periodically acknowledged by writing `WATCHDOG_REFRESH` to the `WATCHDOG_REFRESH_CTRL` register before it reaches zero. The system assumes that the application's failure to acknowledge this countdown timer before it reaches zero indicates that the system is malfunctioning and must be reset. The countdown timer value for the watchdog timer is not visible to the core.

IMPORTANT: The watchdog timer is disabled when the `DEBUG_HALT_CTRL_C_DEBUGEN` bit in the `DEBUG_HALT_CTRL` register is set. This prevents watchdog timeouts during initial code development for the RSL10.

The watchdog timer runs on a prescaled clock that has been derived from the slow clock using a fixed division of 1024. This clock is used to decrement the value in the watchdog's 13-bit counter.

When the watchdog timer is refreshed, a configurable number of bits in the 13-bit counter are set and the prescaling counter is reset. The `WATCHDOG_CTRL_TIMEOUT` bit field in the `WATCHDOG_CTRL` register selects how many of these bits to set when the timer is refreshed. The number of cycles that must elapse between refresh events to trigger a watchdog timeout event is defined by the following equation:

$$\text{CYCLES} = 1024 \times 2^{(\text{WATCHDOG\_CTRL\_TIMEOUT} + 1)}$$

The watchdog has an associated warning interrupt that is pended if the watchdog timer times out. When this interrupt is pended, the watchdog timeout restarts, and if the watchdog timer still has not been reset and a second watchdog timeout occurs, a hard reset occurs. For more information about resets, see Section 5.5, "Resets" on page 63.

12.4.1 Watchdog Registers

Register Name	Register Description	Address
`WATCHDOG_CFG`	Watchdog timer Configuration Register	0x40000300
`WATCHDOG_CTRL`	Watchdog Refresh Control Register	0x40000304

12.4.1.1 WATCHDOG_CFG

Bit Field	Field Name	Description
3:0	`TIMEOUT_VALUE`	Watchdog timeout period. Values 0xC to 0xF result in the same timeout period as the value 0xB.

Field Name	Value Symbol	Value Description	Hex Value
`TIMEOUT_VALUE`[1]	`WATCHDOG_TIMEOUT_2M048`	2.048 ms	0x0
	`WATCHDOG_TIMEOUT_4M096`	4.096 ms	0x1
	`WATCHDOG_TIMEOUT_8M2`	8.192 ms	0x2
	`WATCHDOG_TIMEOUT_16M4`	16.384 ms	0x3
	`WATCHDOG_TIMEOUT_32M8`	32.768 ms	0x4
	`WATCHDOG_TIMEOUT_65M5`	65.536 ms	0x5
	`WATCHDOG_TIMEOUT_131M1`	131.072 ms	0x6
	`WATCHDOG_TIMEOUT_262M1`	262.144 ms	0x7
	`WATCHDOG_TIMEOUT_524M3`	524.3 ms	0x8
	`WATCHDOG_TIMEOUT_1048M6`	1.048 sec	0x9
	`WATCHDOG_TIMEOUT_2097M1`	2.097 sec	0xA
	`WATCHDOG_TIMEOUT_4194M3`	4.194 sec	0xB*

1. The times are based on a 1.28 MHz clock.

12.4.1.2 WATCHDOG_CTRL

Bit Field	Field Name	Description
31:0	`WATCHDOG_REFRESH`	Write a key to reset the watchdog

Field Name	Value Symbol	Value Description	Hex Value
`WATCHDOG_REFRESH`	`WATCHDOG_REFRESH`	Write 32-bit key to reset the watchdog (other values have no effect)	0x2B1E211

CHAPTER 13

Audio

13.1 DIGITAL MICROPHONE (DMIC) INPUTS

The DMIC block provides a serial audio interface for up to two channels, using a pulse density modulated (PDM) digital output stream. The DMIC input data is decimated in a two-step process:

1. A filter supporting a decimation range between 8 and 36 is used.
2. The data is decimated with a filter providing fixed decimation by eight.

   In addition, an optional DC removal filter is available.

The interface consists of an input data line with the data time-interleaved between the two channels, and an output clock signal. These two signals can be routed from any of the DIOs to the DMIC block. In addition, the `AUDIOCLK` or `AUDIOSLOWCLK` has to be routed from the Clock Generation block to the DIO that is used for the DMIC clock signal. For more information on DIO routing configuration for use as a DMIC interface, see Section 10.2, "Functional Configuration" on page 259.

The two DMIC channels can be independently configured using the `AUDIO_CFG` register. The independent configurations available for the interface are shown in the list that follows:

NOTE: The use of '*' in the register or field name indicates either 0 or 1.

- Enabling the channels using the `AUDIO_CFG_DMIC*_ENABLE` bits.

NOTE: When a particular channel is disabled, all its internal states are reset.

- The bit alignment of the decimated audio data from each of the DMIC inputs is configured with the `AUDIO_CFG_DMIC*_DATA_ALIGN` bit:
  - If the `AUDIO_CFG_DMIC*_DATA_ALIGN` bit is configured for MSB alignment, the data is written to the `AUDIO_DMIC*_DATA` registers as an 18-bit MSB value with the bottom 14 bits set to zero.
  - If the `AUDIO_CFG_DMIC*_DATA_ALIGN` bit is configured for LSB alignment, the data is written to the `AUDIO_DMIC*_DATA` registers as a 16-bit LSB value with sign extension in the upper 16 bits.

NOTE: This also configures the DC removal filter to work with either 16-bit or 18-bit data. The 16-bit data from each of the two channels is written to the `AUDIO_DMIC_DATA` register, with the DMIC0 data placed in the 16 LSBs of this register, and the DMIC1 data placed in the 16 MSBs. When the `AUDIO_DMIC_DATA` register is used, the `AUDIO_CFG_DMIC*_DATA_ALIGN` bits must therefore be set to LSB aligned mode.

- When using the DMIC with interrupts, the `AUDIO_CFG_DMIC*_INT_GEN_EN` bit can be set to trigger the `DMIC_OUT_OD_IN` interrupt when an audio input sample is received.
- When using the DMIC with the DMA, the `AUDIO_CFG_DMIC*_DMA_REQ_EN` bit is set so that DMA requests are triggered whenever an audio input sample is received. For more details concerning the DMA refer to Section 12.2, "Direct Memory Access (DMA) Controller" on page 353.

IMPORTANT: The DMIC interface is supported by only one DMA request line, which produces one read from the DMIC interface per DMA request event. This limits DMA support in dual-channel DMIC configurations to reads using the `AUDIO_DMIC_DATA` register and hence 16-bit samples. Reads from a single channel using the DMA do not have this limitation.

The following two `AUDIO_CFG` register fields configure both channels:

- The AUDIO_CFG_DEC_RATE bit-field defines the decimation rate applied to data input from the DMIC. The decimation factor as a function of the AUDIO_CFG_DEC_RATE register is as follows:
  $\text{Decimation Factor} = (\text{AUDIO\_CFG\_DEC\_RATE} + 8)$.
- The AUDIO_CFG_DMIC_CLK_SRC bit is used to select either clocking the DMIC input signal with the audio clock (AUDIOCLK), or the pre-scaled version of this clock (AUDIOSLOWCLK). For more details concerning the audio clocks, refer to Section 6.3.8, “Interface Clocks” on page 82.

In addition to the AUDIO_CFG register, detailed configuration of the DMIC interface is available through the AUDIO_DMIC_CFG register. This configuration includes:

- The AUDIO_DMIC*_CFG_CLK_EDGE bit, which controls whether the data from the DMIC input is sampled on the rising or falling edge of the DMIC input clock. Refer to Figure 37 for detailed timing.

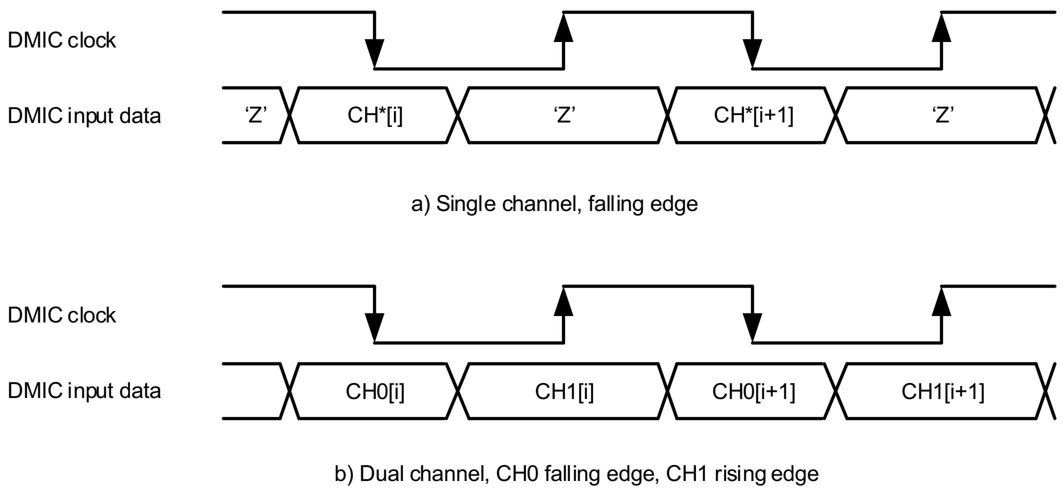


Figure 37. DMIC Timing

- The AUDIO_DMIC*_DCRM bit-field, which can be used to configure the DMIC input’s DC removal filter. This is a high-pass filter that can help in situations when there is a DC offset present in the input signal.
- The DMIC1 input can be delayed relative to the DMIC0 input by up to 1.875 sample periods, in steps of 0.125 sample periods. An additional delay of up to 31 DMIC clock periods can also be added, but is limited to AUDIO_CFG_DEC_RATE + 7 clock periods. These delays are configured with the AUDIO_DMIC_CFG_DMIC1_DELAY and AUDIO_DMIC_CFG_DMIC1_FRAC_DELAY bit-fields respectively.

The AUDIO_STATUS register contains information on the status of the DMIC inputs. This includes:

- The AUDIO_STATUS_DMIC*_DATA_RDY_FLAG status bits, which indicate when a new audio sample is available from the DMIC inputs. These bits are reset when their respective AUDIO_DMIC*_DATA registers are read, and can be used in place of the DMA or interrupt control, if polling is used to control the reading of data from the DMIC input.
- The AUDIO_STATUS_DMIC*_OVERRUN_FLAG status bits, which are flags that indicate if an overrun has been detected. An overrun occurs whenever an audio sample is not read from the AUDIO_DMIC*_DATA register before it is overwritten with a subsequent sample. If an overrun has occurred, this flag remains set until the corresponding AUDIO_STATUS_DMIC*_OVERRUN_FLAG_CLEAR bit is used to clear it.

Data received from the DMIC input samples is decimated into 18-bit samples, using a two-phase decimation filter. This filter includes:

- A configurable 5th-order SINC filter that decimates the input data by a factor in the range from 8 to 36 (as defined by the `AUDIO_CFG_DEC_RATE` bit-field from the `AUDIO_CFG` register). This allows the use of digital microphones that have a sigma-delta ADC up to 4th order.
- A low-pass wave digital filter structure that provides a fixed decimation by 8.

  In addition to filtering, a gain of between 0 and 200% is applied to the audio samples, configurable through the `AUDIO_DMIC*_GAIN` registers. This can be used to calibrate the gain of the individual microphones. If no additional gain needs to be applied, use the `DMIC*_NOMINAL_GAIN` settings. The formula for setting the gain is:

$$\text{Gain} = \frac{\texttt{AUDIO\_DMIC*\_GAIN}}{2048}$$

IMPORTANT: Avoid changing the decimation rate while reading input samples, as this can introduce audio artifacts.

13.1.1 Digital Microphone and Shared Digital Microphone/Output Driver Registers

Register Name	Register Description	Address
AUDIO_CFG	DMIC and OD Configuration Register	0x40000E00
AUDIO_STATUS	DMIC and OD Status Register	0x40000E04
AUDIO_DMIC_CFG	DMIC Configuration Register	0x40000E08
AUDIO_DMIC0_GAIN	DMIC0 Gain Configuration Register	0x40000E0C
AUDIO_DMIC1_GAIN	DMIC1 Gain Configuration Register	0x40000E10
AUDIO_DMIC_DATA	DMIC0 and DMIC1 Data Register	0x40000E14
AUDIO_DMIC0_DATA	DMIC0 Data Register	0x40000E18
AUDIO_DMIC1_DATA	DMIC1 Data Register	0x40000E1C

13.1.1.1 AUDIO_CFG

Bit Field	Field Name	Description
25	OD_CLK_SRC	Output driver clock selection
24	DMIC_CLK_SRC	DMIC clock selection (the same clock must be output to the DMIC_CLK DIO)
20:16	DEC_RATE	DMIC input data decimation rate (also determines the OD interpolation rate in combination with DMIC_CLK_SRC and OD_CLK_SRC configuration bits)
12	OD_UNDERRUN_PROTECT	Enable OD_DATA underrun protection (automatically resets OD_DATA if it hasn't been updated during 16 sample periods)
11	OD_DMA_REQ_EN	Enable the DMA request when a new output driver sample is required
10	OD_INT_GEN_EN	Enable the interrupt generation when a new output driver sample is required
9	OD_DATA_ALIGN	Data alignment in AUDIO_OD_DATA
8	OD_ENABLE	Enable output driver output
7	DMIC1_DMA_REQ_EN	Enable the DMA request when a new DMIC1 sample is ready

Bit Field	Field Name	Description
6	DMIC1_INT_GEN_EN	Enable the interrupt generation when a new DMIC1 sample is ready
5	DMIC1_DATA_ALIGN	Data alignment in AUDIO_DMIC_DATA_1
4	DMIC1_ENABLE	Enable DMIC1 input
3	DMIC0_DMA_REQ_EN	Enable the DMA request when a new DMIC0 sample is ready
2	DMIC0_INT_GEN_EN	Enable the interrupt generation when a new DMIC0 sample is ready
1	DMIC0_DATA_ALIGN	Data alignment in AUDIO_DMIC_DATA_0
0	DMIC0_ENABLE	Enable DMIC0 input

Field Name	Value Symbol	Value Description	Hex Value
OD_CLK_SRC	OD_AUDIOCLK	OD uses AUDIOCLK	0x0*
	OD_AUDIOSLOWCLK	OD uses AUDIOSLOWCLK	0x1
DMIC_CLK_SRC	DMIC_AUDIOCLK	DMIC uses AUDIOCLK	0x0*
	DMIC_AUDIOSLOWCLK	DMIC uses AUDIOSLOWCLK	0x1
DEC_RATE	DECIMATE_BY_64	Decimate the DMIC input data by 64	0x0*
	DECIMATE_BY_72	Decimate the DMIC input data by 72	0x1
	DECIMATE_BY_80	Decimate the DMIC input data by 80	0x2
	DECIMATE_BY_128	Decimate the DMIC input data by 128	0x8
	DECIMATE_BY_136	Decimate the DMIC input data by 136	0x9
	DECIMATE_BY_256	Decimate the DMIC input data by 256	0x16
	DECIMATE_BY_288	Decimate the DMIC input data by 288	0x1C
OD_UNDERRUN_PROTECT	OD_UNDERRUN_PROTECT_DISABLE	OD_DATA underrun protection disabled	0x0*
	OD_UNDERRUN_PROTECT_ENABLE	OD_DATA underrun protection enabled	0x1
OD_DMA_REQ_EN	OD_DMA_REQ_DISABLE	Disable the DMA request when a new OD sample is required	0x0*
	OD_DMA_REQ_ENABLE	Enable the DMA request when a new OD sample is required	0x1
OD_INT_GEN_EN	OD_INT_GEN_DISABLE	Disable the interrupt generation when a new OD sample is required	0x0*
	OD_INT_GEN_ENABLE	Enable the interrupt generation when a new OD sample is required	0x1
OD_DATA_ALIGN	OD_DATA_LSB_ALIGNED	OD_DATA is 16-bit LSB aligned	0x0*
	OD_DATA_MSB_ALIGNED	OD_DATA is 18-bit MSB aligned	0x1
OD_ENABLE	OD_DISABLE	Disable the OD output	0x0*
	OD_ENABLE	Enable the OD output	0x1

Field Name	Value Symbol	Value Description	Hex Value
DMIC1_DMA_REQ_EN	DMIC1_DMA_REQ_DISABLE	Disable the DMA request when a new DMIC1 sample is ready	0x0*
	DMIC1_DMA_REQ_ENABLE	Enable the DMA request when a new DMIC1 sample is ready	0x1
DMIC1_INT_GEN_EN	DMIC1_INT_GEN_DISABLE	Disable the interrupt generation when a new DMIC1 sample is ready	0x0*
	DMIC1_INT_GEN_ENABLE	Enable the interrupt generation when a new DMIC1 sample is ready	0x1
DMIC1_DATA_ALIGN	DMIC1_DATA_LSB_ALIGNED	DMIC1_DATA is 16-bit LSB aligned	0x0*
	DMIC1_DATA_MSB_ALIGNED	DMIC1_DATA is 18-bit MSB aligned	0x1
DMIC1_ENABLE	DMIC1_DISABLE	Disable the DMIC1 input	0x0*
	DMIC1_ENABLE	Enable the DMIC1 input	0x1
DMIC0_DMA_REQ_EN	DMIC0_DMA_REQ_DISABLE	Disable the DMA request when a new DMIC0 sample is ready	0x0*
	DMIC0_DMA_REQ_ENABLE	Enable the DMA request when a new DMIC0 sample is ready	0x1
DMIC0_INT_GEN_EN	DMIC0_INT_GEN_DISABLE	Disable the interrupt generation when a new DMIC0 sample is ready	0x0*
	DMIC0_INT_GEN_ENABLE	Enable the interrupt generation when a new DMIC0 sample is ready	0x1
DMIC0_DATA_ALIGN	DMIC0_DATA_LSB_ALIGNED	DMIC0_DATA is 16-bit LSB aligned	0x0*
	DMIC0_DATA_MSB_ALIGNED	DMIC0_DATA is 18-bit MSB aligned	0x1
DMIC0_ENABLE	DMIC0_DISABLE	Disable the DMIC0 input	0x0*
	DMIC0_ENABLE	Enable the DMIC0 input	0x1

13.1.1.2 AUDIO_STATUS

Bit Field	Field Name	Description
10	OD_UNDERRUN_FLAG_CLEAR	Reset the output driver underrun detection sticky bit
9	OD_UNDERRUN_FLAG	Sticky bit indicating the detection of an output driver underrun
8	OD_DATA_REQ_FLAG	Flag indicating that a new output driver sample is required
6	DMIC1_OVERRUN_FLAG_CLEAR	Reset the DMIC1 overrun detection sticky bit
5	DMIC1_OVERRUN_FLAG	Sticky bit indicating the detection of a DMIC1 overrun
4	DMIC1_DATA_RDY_FLAG	Flag indicating the availability of a new DMIC1 sample
2	DMIC0_OVERRUN_FLAG_CLEAR	Reset the DMIC0 overrun detection sticky bit
1	DMIC0_OVERRUN_FLAG	Sticky bit indicating the detection of a DMIC0 overrun
0	DMIC0_DATA_RDY_FLAG	Flag indicating the availability of a new DMIC0 sample

Field Name	Value Symbol	Value Description	Hex Value
OD_UNDERRUN_FLAG_CLEAR	OD_UNDERRUN_FLAG_CLEAR	Reset the OD underrun detection sticky bit	0x1

Field Name	Value Symbol	Value Description	Hex Value
OD_UNDERRUN_FLAG	OD_UNDERRUN_NOT_DETECTED	Indicates that no OD underrun has been detected	0x0*
	OD_UNDERRUN_DETECTED	Indicates that an OD underrun has been detected	0x1
OD_DATA_REQ_FLAG	OD_DATA_NOT_REQUIRED	Indicates that no new OD sample is required	0x0*
	OD_DATA_REQUIRED	Indicates that a new OD sample is required	0x1
DMIC1_OVERRUN_FLAG_CLEAR	DMIC1_OVERRUN_FLAG_CLEAR	Reset the DMIC1 overrun detection sticky bit	0x1
DMIC1_OVERRUN_FLAG	DMIC1_OVERRUN_NOT_DETECTED	Indicates that no DMIC1 overrun has been detected	0x0*
	DMIC1_OVERRUN_DETECTED	Indicates that a DMIC1 overrun has been detected	0x1
DMIC1_DATA_RDY_FLAG	DMIC1_DATA_NOT_READY	Indicates that no new DMIC1 sample is available	0x0*
	DMIC1_DATA_READY	Indicates that a new DMIC1 sample is available	0x1
DMIC0_OVERRUN_FLAG_CLEAR	DMIC0_OVERRUN_FLAG_CLEAR	Reset the DMIC0 overrun detection sticky bit	0x1
DMIC0_OVERRUN_FLAG	DMIC0_OVERRUN_NOT_DETECTED	Indicates that no DMIC0 overrun has been detected	0x0*
	DMIC0_OVERRUN_DETECTED	Indicates that a DMIC0 overrun has been detected	0x1
DMIC0_DATA_RDY_FLAG	DMIC0_DATA_NOT_READY	Indicates that no new DMIC0 sample is available	0x0*
	DMIC0_DATA_READY	Indicates that a new DMIC0 sample is available	0x1

13.1.1.3 AUDIO_DMIC_CFG

Bit Field	Field Name	Description
28:24	DMIC1_FRAC_DELAY	DMIC1 fractional delay (each step represents a DMIC clock cycle)
19:16	DMIC1_DELAY	DMIC1 delay (0 to 1.875 samples in steps of 0.125 samples)
14:12	DMIC1_DCRM	DMIC1 DC removal filter enable and cut-off frequency
10:8	DMIC0_DCRM	DMIC0 DC removal filter enable and cut-off frequency
1	DMIC1_CLK_EDGE	DMIC1 input clock edge
0	DMIC0_CLK_EDGE	DMIC0 input clock edge

Field Name	Value Symbol	Value Description	Hex Value
DMIC1_DELAY	DMIC1_DELAY_DISABLE	Delay disabled	0x0*
	DMIC1_DELAY_0P125	Delay of 0.125 samples	0x1
	DMIC1_DELAY_0P25	Delay of 0.25 samples	0x2
	DMIC1_DELAY_0P375	Delay of 0.375 samples	0x3
	DMIC1_DELAY_0P5	Delay of 0.5 samples	0x4
	DMIC1_DELAY_0P625	Delay of 0.625 samples	0x5
	DMIC1_DELAY_0P75	Delay of 0.75 samples	0x6
	DMIC1_DELAY_0P875	Delay of 0.875 samples	0x7
	DMIC1_DELAY_1P0	Delay of 1 sample	0x8
	DMIC1_DELAY_1P125	Delay of 1.125 samples	0x9
	DMIC1_DELAY_1P25	Delay of 1.25 samples	0xA
	DMIC1_DELAY_1P375	Delay of 1.375 samples	0xB
	DMIC1_DELAY_1P5	Delay of 1.5 samples	0xC
	DMIC1_DELAY_1P625	Delay of 1.625 samples	0xD
	DMIC1_DELAY_1P75	Delay of 1.75 samples	0xE
	DMIC1_DELAY_1P875	Delay of 1.875 samples	0xF
DMIC1_DCRM	DMIC1_DCRM_CUTOFF_5HZ	Cut-off frequency is Fs/3200 (5 Hz at Fs=16 kHz)	0x0*
	DMIC1_DCRM_CUTOFF_10HZ	Cut-off frequency is Fs/1600 (10 Hz at Fs=16 kHz)	0x1
	DMIC1_DCRM_CUTOFF_20HZ	Cut-off frequency is Fs/800 (20 Hz at Fs=16 kHz)	0x2
	DMIC1_DCRM_CUTOFF_40HZ	Cut-off frequency is Fs/400 (40 Hz at Fs=16 kHz)	0x3
	DMIC1_DCRM_CUTOFF_80HZ	Cut-off frequency is Fs/200 (80 Hz at Fs=16 kHz)	0x4
	DMIC1_DCRM_CUTOFF_160HZ	Cut-off frequency is Fs/100 (160 Hz at Fs=16 kHz)	0x5
	DMIC1_DCRM_CUTOFF_320HZ	Cut-off frequency is Fs/50 (320 Hz at Fs=16 kHz)	0x6
	DMIC1_DCRM_DISABLE	DC removal filter disabled	0x7

Field Name	Value Symbol	Value Description	Hex Value
DMIC0_DCRM	DMIC0_DCRM_CUTOFF_5HZ	Cut-off frequency is Fs/3200 (5 Hz at Fs=16 kHz)	0x0*
	DMIC0_DCRM_CUTOFF_10HZ	Cut-off frequency is Fs/1600 (10 Hz at Fs=16 kHz)	0x1
	DMIC0_DCRM_CUTOFF_20HZ	Cut-off frequency is Fs/800 (20 Hz at Fs=16 kHz)	0x2
	DMIC0_DCRM_CUTOFF_40HZ	Cut-off frequency is Fs/400 (40 Hz at Fs=16 kHz)	0x3
	DMIC0_DCRM_CUTOFF_80HZ	Cut-off frequency is Fs/200 (80 Hz at Fs=16 kHz)	0x4
	DMIC0_DCRM_CUTOFF_160HZ	Cut-off frequency is Fs/100 (160 Hz at Fs=16 kHz)	0x5
	DMIC0_DCRM_CUTOFF_320HZ	Cut-off frequency is Fs/50 (320 Hz at Fs=16 kHz)	0x6
	DMIC0_DCRM_DISABLE	DC removal filter disabled	0x7
DMIC1_CLK_EDGE	DMIC1_FALLING_EDGE	Sample the DMIC1 input data on the falling DMIC clock	0x0*
	DMIC1_RISING_EDGE	Sample the DMIC1 input data on the rising DMIC clock	0x1
DMIC0_CLK_EDGE	DMIC0_FALLING_EDGE	Sample the DMIC0 input data on the falling DMIC clock	0x0*
	DMIC0_RISING_EDGE	Sample the DMIC0 input data on the rising DMIC clock	0x1

13.1.1.4 AUDIO_DMIC0_GAIN

Bit Field	Field Name	Description
11:0	GAIN	DMIC calibration gain (unsigned value from 0 to +2)

Field Name	Value Symbol	Value Description	Hex Value
GAIN	DMIC0_NOMINAL_GAIN	Nominal gain	0x800*

13.1.1.5 AUDIO_DMIC1_GAIN

Bit Field	Field Name	Description
11:0	GAIN	DMIC calibration gain (unsigned value from 0 to +2)

Field Name	Value Symbol	Value Description	Hex Value
GAIN	DMIC1_NOMINAL_GAIN	Nominal gain	0x800*

13.1.1.6 AUDIO_DMIC_DATA

Bit Field	Field Name	Description
31:16	`DMIC1_DATA`	DMIC1 input data (16-bit)
15:0	`DMIC0_DATA`	DMIC0 input data (16-bit)

13.1.1.7 AUDIO_DMIC0_DATA

Bit Field	Field Name	Description
31:0	`DATA`	DMIC0 input data (LSB or MSB aligned according to AUDIO_CFG); data is sign extended from 16-bit to 32-bit when read in LSB aligned mode, or zero padded when read in MSB aligned mode

13.1.1.8 AUDIO_DMIC1_DATA

Bit Field	Field Name	Description
31:0	`DATA`	DMIC1 input data (LSB or MSB aligned according to AUDIO_CFG); data is sign extended from 16-bit to 32-bit when read in LSB aligned mode, or zero padded when read in MSB aligned mode

13.2 Output Driver

The output driver provides a mono digital audio output from the RSL10 system. This output driver can be connected to drive one or more DIO pairs, which are used as the driver for a speaker or receiver.

The output driver consists of four stages:

1. The gain stage applies a gain of between 0 and 200% to the audio samples that are being passed to the next stage, configurable through the `AUDIO_OD_GAIN` register. The gain is set as follows:

$$\text{Gain} = \frac{\texttt{AUDIO\_OD\_GAIN}}{2048}$$

2. The interpolation filter stage upsamples and filters the audio samples that are being passed to the next stage. The interpolation filter uses a low-pass wave digital filter structure that provides a fixed interpolation by 8.

NOTE: If no additional gain is expected, use the `OD_NOMINAL_GAIN` setting.

3. The sigma-delta modulator stage consists of a 4th-order three-level sigma-delta modulator for the output channel, to produce a pulse density modulated (PDM) output signal provided to the next stage in the form of two single bit signals, OD_P and OD_N.
4. The output driver routing stage sends the OD_P and OD_N signals to any DIO pair(s). For more information on DIO configuration for use as an output driver, see Section 10.2, “Functional Configuration” on page 259.

IMPORTANT: If a higher load drive (lower output impedance) is required than is provided by a single DIO pair, multiple DIO pairs can be connected in parallel. When connecting DIOs in parallel, pairs are best selected in such a way that OD_P DIOs are close together on the package and the same applies to OD_N DIOs.

The output driver is enabled by setting the `AUDIO_CFG_OD_ENABLE` bit from the `AUDIO_CFG` register. Other configurations of the output driver from this register include:

- The AUDIO_CFG_OD_DATA_ALIGN bit can be used to select MSB or LSB alignment in the AUDIO_OD_DATA register. When set to LSB alignment, the 16 bottom bits of the register are used. When MSB-aligned, the 32-bit input data is rounded nearest to infinity with saturation to 18 bits.
- When using the output driver with interrupts, the AUDIO_CFG_OD_INT_GEN_EN bit can be set to trigger the DMIC_OUT_OD_IN interrupt when another audio output sample is required.
- When using the output driver with the DMA, the AUDIO_CFG_OD_DMA_REQ_EN bit needs to be set so that DMA requests are triggered when another audio sample is required. For more details concerning the DMA, refer to Section 12.2, “Direct Memory Access (DMA) Controller” on page 353.
- The AUDIO_CFG_OD_UNDERRUN_PROTECT bit can be set to enable the mechanism that clears the AUDIO_OD_DATA register if it has not been written to for 16 sample periods. Use of underrun protection is recommended, as this prevents the OD from potentially driving a large DC value if, for some reason, the system fails to supply new output samples.

CAUTION: A large DC value could burn out the attached speaker/receiver.

- The AUDIO_CFG_OD_CLK_SRC bit is used to select either clocking the output driver with the audio clock (AUDIOCLK), or the pre-scaled version of this clock (AUDIOSLOWCLK). For more details concerning the audio clocks, refer to Section 6.3.8, “Interface Clocks” on page 82.

Detailed configuration of the output driver is available through the AUDIO_OD_CFG register. This configuration includes:

- The AUDIO_OD_CFG_CLK_EDGE bit, which controls whether the output driver updates the output on the rising or the falling edge of the output driver’s clock.
- The AUDIO_OD_CFG_DITHER bit can be used to enable or disable dithering of the output data stream. Use of dithering is recommended, as this avoids idle tones and other artifacts produced by sigma-delta modulation.
- The AUDIO_OD_CFG_DCRM bit can be used to configure the output driver’s DC removal filter. This is a high-pass filter that can help to remove noise artifacts, which can occur in situations where the level of the output signal is very low, and there is a DC offset present in the signal (typically caused by rounding errors that occur in the processing of the audio data). We recommend using the DC removal filter with a cut-off frequency of 20 Hz.

The AUDIO_SDM_CFG register controls internal configuration of the sigma-delta modulator. For normal operation, this register must be set to the SDM_CFG_NORMAL setting.

The AUDIO_STATUS register contains information on the status of the output driver. This includes:

- The AUDIO_STATUS_OD_STATUS bit, which indicates if the output driver is included in this version of RSL10
- The AUDIO_STATUS_OD_DATA_REQ_FLAG bit, which indicates when a new audio sample is required for the output driver. This status bit is reset when the AUDIO_OD_DATA register is written, and can be used in place of the DMA or interrupt control, if polling is used to control the writing of data to the OD output.
- The AUDIO_STATUS_OD_UNDERRUN_FLAG bit, which provides a flag that indicates if an underrun has been detected. An underrun occurs whenever an audio sample is used by the output driver multiple times. If an underrun has occurred, this flag remains set until the AUDIO_STATUS_OD_UNDERRUN_FLAG_CLEAR bit is used to clear it.

The recommended output driver configuration is shown in Table 33.

Table 33. Recommended Output Driver Configuration

Register	Bit Field	Setting	Notes
CLK_DIV1	AUDIOCLK_PRESCALE AUDIOSLOWCLK_PRESCALE	Set supplied clock frequency to between 1 and 2 MHz (1 MHz for lowest power consumption, 2 MHz for lowest high frequency noise).	
AUDIO_CFG	OD_UNDERRUN_PROTECT	OD_UNDERRUN_PROTECT_ENABLE	Prevent the output driver from driving large DC outputs when new samples are not available.
AUDIO_OD_CFG	DITHER	DITHER_ENABLE	Avoids idle tones and other audio artifacts.
	AUDIO_OD_CFG_DCRM	Set to 20 Hz cut-off frequency (actual bit-field value depends on OD clock frequency.	
AUDIO_SDM_CFG	SDM_CFG	SDM_CFG_NORMAL	Use the normal configuration for the sigma-delta modulator.
DIO_PAD_CFG	DRIVE	PAD_HIGH_DRIVE	Set maximum DIO pad driver strength (only allowed if VDDO < 2.7 V).

13.2.1 Output Driver Registers

Register Name	Register Description	Address
AUDIO_OD_CFG	Output Driver Configuration Register	0x40000E20
AUDIO_OD_GAIN	Output Driver Gain Configuration Register	0x40000E24
AUDIO_OD_DATA	Output Driver Data Register	0x40000E28
AUDIO_SDM_CFG	Sigma-Delta Modulator Configuration Register	0x40000E30

13.2.1.1 AUDIO_OD_CFG

Bit Field	Field Name	Description
19:16	DCRM	Output driver DC removal filter enable and cut-off frequency
10	DITHER	Sigma-delta modulator dithering enable
0	CLK_EDGE	Output driver output clock edge

Field Name	Value Symbol	Value Description	Hex Value
DCRM	DCRM_DISABLE	DC removal filter disabled	0x0*
	DCRM_CUTOFF_1P25HZ	Cut-off frequency is OD_CLK/800000 (1.25 Hz at OD_CLK=1 MHz)	0x1
	DCRM_CUTOFF_2P5HZ	Cut-off frequency is OD_CLK/400000 (2.5 Hz at OD_CLK=1 MHz)	0x2
	DCRM_CUTOFF_3P75HZ	Cut-off frequency is OD_CLK/266667 (3.75 Hz at OD_CLK=1 MHz)	0x3
	DCRM_CUTOFF_5HZ	Cut-off frequency is OD_CLK/200000 (5 Hz at OD_CLK=1 MHz)	0x4
	DCRM_CUTOFF_7P5HZ	Cut-off frequency is OD_CLK/133333 (7.5 Hz at OD_CLK=1 MHz)	0x5
	DCRM_CUTOFF_10HZ	Cut-off frequency is OD_CLK/100000 (10 Hz at OD_CLK=1 MHz)	0x6
	DCRM_CUTOFF_15HZ	Cut-off frequency is OD_CLK/66667 (15 Hz at OD_CLK=1 MHz)	0x7
	DCRM_CUTOFF_20HZ	Cut-off frequency is OD_CLK/50000 (20 Hz at OD_CLK=1 MHz)	0x8
	DCRM_CUTOFF_30HZ	Cut-off frequency is OD_CLK/33333 (30 Hz at OD_CLK=1 MHz)	0x9
	DCRM_CUTOFF_40HZ	Cut-off frequency is OD_CLK/25000 (40 Hz at OD_CLK=1 MHz)	0xA
	DCRM_CUTOFF_60HZ	Cut-off frequency is OD_CLK/16667 (60 Hz at OD_CLK=1 MHz)	0xB
	DCRM_CUTOFF_80HZ	Cut-off frequency is OD_CLK/12500 (80 Hz at OD_CLK=1 MHz)	0xC
	DCRM_CUTOFF_120HZ	Cut-off frequency is OD_CLK/8333 (120 Hz at OD_CLK=1 MHz)	0xD
	DCRM_CUTOFF_160HZ	Cut-off frequency is OD_CLK/6250 (160 Hz at OD_CLK=1 MHz)	0xE
	DCRM_CUTOFF_240HZ	Cut-off frequency is OD_CLK/4167 (240 Hz at OD_CLK=1 MHz)	0xF
DITHER	DITHER_DISABLE	Dithering disabled	0x0*
	DITHER_ENABLE	Dithering enabled	0x1
CLK_EDGE	OD_FALLING_EDGE	Output OD data on the falling OD clock	0x0
	OD_RISING_EDGE	Output OD data on the rising OD clock	0x1*

13.2.1.2 AUDIO_OD_GAIN

Bit Field	Field Name	Description
11:0	GAIN	Output driver calibration gain (unsigned value from 0 to +2)

Field Name	Value Symbol	Value Description	Hex Value
GAIN	OD_NOMINAL_GAIN	Nominal gain	0x800*

13.2.1.3 AUDIO_OD_DATA

Bit Field	Field Name	Description
31:0	`DATA`	OD output data (LSB or MSB aligned according to OD_CFG); data is truncated to 16 bits when written in LSB aligned mode, or rounded nearest to infinity with saturation when written in MSB aligned mode
31:0	`DATA_RD`	OD output data (LSB or MSB aligned according to OD_CFG); data is sign extended from 16-bit to 32-bit when read in LSB aligned mode, or zero padded in MSB aligned mode

13.2.1.4 AUDIO_SDM_CFG

Bit Field	Field Name	Description
31:0	`SDM_CFG`	Sigma-Delta modulator internal configuration for testing purposes

Field Name	Value Symbol	Value Description	Hex Value
`SDM_CFG`	`SDM_CFG_NORMAL`	Normal Sigma-Delta modulator configuration	0x50012

13.3 Audio Sink Clock Counters

The audio sink clock counters can be used to measure the timing of the frame periods of a BLE/RF host relative to the internal audio sampling rate or that of a connected external device. In addition, it can be used to measure the frequency of the internal 32 kHz RC oscillator using the 48 MHz crystal oscillator.

The following uses are supported:

- Support an asynchronous sample rate conversion of audio samples being sourced over the radio link, and consumed by a connected device or the output driver. For details on asynchronous sample rate conversion, see Section 13.4, “Asynchronous Sample Rate Converter (ASRC)” on page 394.
- Support an asynchronous sample rate conversion of audio samples being sourced from a connected device or the DMIC inputs, and consumed over the radio link.

NOTE: Although the DMIC audio is the source, its sample clock is considered as the audio sink clock to the RF host throughout this section.

- Generate control information to change the clock frequency of the connected device so that it becomes synchronous with a radio link.
- Measure the internal 32 kHz RC oscillator.
    - An example that uses the 48 MHz crystal oscillator to measure the internal 32 kHz RC oscillator using the audio sink block can be seen in the calibration library, as described in the *RSL10 Firmware Reference.*

IMPORTANT: The audio sink clock being measured is sourced from a DIO or from STANDBYCLK, as specified by the DIO configuration. For more information on DIO configuration for use as the audio sink source, see **Section 10.2, “Functional Configuration” on page 259**.

The audio sink clock being measured is sourced from a DIO or from STANDBYCLK, as specified by the DIO configuration. For more information on DIO configuration for use as the audio sink source, see Section 10.2, “Functional Configuration” on page 259.

The timing diagram in Figure 38 highlights the measurement registers in the Audio Sink module.

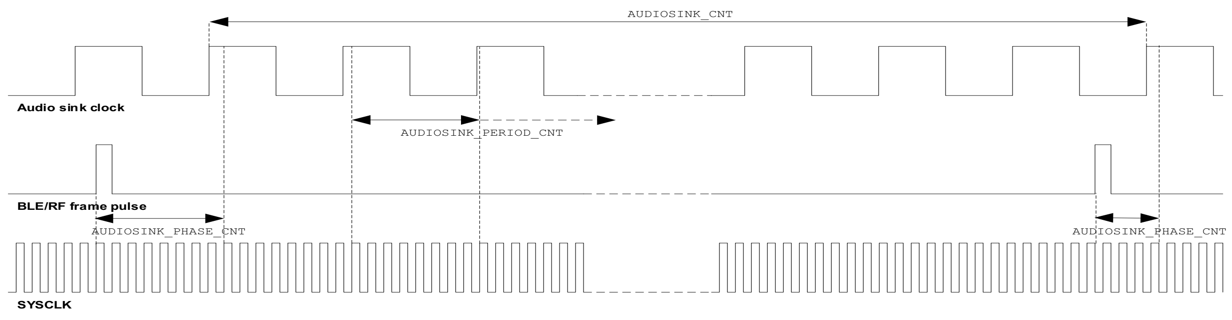


Figure 38. Audio Sink Timing

The measurement registers are defined as follows:

- The AUDIOSINK_CNT register holds the integer number of cycles of the audio sink clock being measured between consecutive BLE/RF frame pulses.
- The AUDIOSINK_PERIOD_CNT register contains the period count of the audio sink clock being measured in terms of SYSCLK cycles, which is typically derived from the 48 MHz crystal oscillator. (See Section 6.3.1, “System Clock (SYSCLK)” on page 78 for more information.) The period can be measured over 1-16 audio sink clock cycles. The number of audio sink clock cycles measured is controlled by the AUDIOSINK_CFG_PERIODS_CFG bit field in the AUDIOSINK_CFG register. By using the value stored in the AUDIOSINK_PERIOD_CNT register, the period can be calculated as:

$$\text{period} = \frac{\text{AUDIOSINK\_PERIOD\_CNT}}{(\text{AUDIOSINK\_CFG\_PERIODS} + 1)} \times \text{period}_{\text{SYSCLK}}$$

NOTE: The accuracy of the period measurement is improved by increasing the AUDIOSINK_CFG_PERIODS_CFG value and/or the SYSCLK frequency.

This period counter is supported by:

- The AUDIOSINK_CTRL_PERIOD_CNT_START bit from the AUDIOSINK_CTRL register, which clears and starts the period counter when a rising edge of the audio sink clock is detected.
- The AUDIOSINK_CTRL_PERIOD_CNT_STOP bit from the AUDIOSINK_CTRL register, which stops the period counter mechanism manually.
- The AUDIOSINK_CTRL_PERIOD_STATUS bit, which indicates whether the period counter mechanism is currently active or idle.
- The AUDIOSINK_PERIOD_CNT register, which contains the number of SYSCLK cycles between when the counter started, and when the configured number of rising edges of the audio sink clock were detected (saturated to 0xFFFF), at which point the counter is stopped automatically.
- The AUDIOSINK_PERIOD interrupt, which is triggered when the audio sink period counter finishes counting the defined number of audio sink clock periods.

- The AUDIOSINK_PHASE_CNT register measures the time from the BLE/RF frame pulse to the first detected rising edge of the audio sink clock, in terms of SYSCLK cycles. This measurement is used to improve the resolution of the calculated number of audio sink clock cycle per BLE/RF frame. This counter is supported by:

- The `AUDIOSINK_CTRL_PHASE_CNT_START` bit, which clears and starts the phase counter the next time a synchronization pulse is generated by the BLE/RF blocks.
- The `AUDIOSINK_CTRL_PHASE_CNT_START_NO_WAIT` bit, which clears and starts the phase counter immediately.
- The `AUDIOSINK_CTRL_PHASE_CNT_STOP` bit, which stops the phase counter mechanism manually.
- The `AUDIOSINK_CTRL_PHASE_STATUS` bit, which indicates whether the phase counter mechanism is currently active or idle.
- The `AUDIOSINK_PHASE_CNT` register, which contains the number of SYSCLK cycles between when the counter started, and when a rising edge has been detected on the audio sink clock (saturated to 0xFFFF), at which point the counter is stopped automatically.
- The `AUDIOSINK_PHASE` interrupt, which is triggered when the phase counter is running and a rising edge occurs on the audio sink clock, or if a synchronization error has been received from the BLE/RF block (the `AUDIOSINK_CTRL_PHASE_CNT_MISSED_STATUS` bit is set if an error occurs, and is cleared otherwise), at which point the counter mechanism is stopped automatically.

Using the values obtained from the `AUDIOSINK_CNT` and `AUDIOSINK_PHASE_CNT` registers plus the previous period and phase count values, you can calculate the number of audio sink clock cycles between the previous (i-1) and the current (i) synchronization pulses as follows:

$$\text{audio sink clock cycles} = \texttt{AUDIOSINK\_CNT[i]} + \frac{\texttt{AUDIOSINK\_PHASE\_CNT[i-1]} - \texttt{AUDIOSINK\_PHASE\_CNT[i]}}{\texttt{AUDIOSINK\_PERIOD\_CNT[i-1]}/(\texttt{AUDIOSINK\_PERIODS\_CFG} + 1}$$

The recommended process for using the measurement registers is as follows:

- Before a synchronization frame pulse will occur, start the phase counter mechanism.
- When the `AUDIOSINK_PHASE` interrupt occurs:
  a. Save the `AUDIOSINK_PHASE_CNT` counter value from the previous frame.
  b. Record the `AUDIOSINK_PHASE_CNT` counter value.
  c. Record the value in the `AUDIOSINK_CNT` register.
  d. Reset the `AUDIOSINK_CNT` register/counter.
  e. Start the period counter.

- When the `AUDIOSINK_PERIOD` interrupt occurs:
  a. Record the `AUDIOSINK_PERIOD_CNT` counter value.
  b. Calculate the number of SYSCLK cycles per audio sink clock period.

13.3.1 Audio Sink Registers

Register Name	Register Description	Address
`AUDIOSINK_CTRL`	Audio Sink Clock Control Register	0x40001000
`AUDIOSINK_CFG`	Audio Sink Clock Configuration Register	0x40001004
`AUDIOSINK_CNT`	Audio Sink Clock Counter Register	0x40001008
`AUDIOSINK_PHASE_CNT`	Audio Sink Clock Phase Counter Register	0x4000100C
`AUDIOSINK_PERIOD_CNT`	Audio Sink Clock Period Counter Register	0x40001010

13.3.1.1 AUDIOSINK_CTRL

Bit Field	Field Name	Description
8	PHASE_CNT_START_NO_WAIT	Start the audio sink clock phase counter mechanism without waiting on a sync pulse
7	PERIOD_CNT_STATUS	Audio sink clock period counter status
6	PERIOD_CNT_STOP	Stop the audio sink clock period counter mechanism
5	PERIOD_CNT_START	Start the audio sink clock period counter mechanism
4	PHASE_CNT_MISSED_STATUS	Audio sink clock phase counter missed status
3	PHASE_CNT_STATUS	Audio sink clock phase counter status
2	PHASE_CNT_STOP	Stop the audio sink clock phase counter mechanism
1	PHASE_CNT_START	Start the audio sink clock phase counter mechanism and wait for sync pulse
0	CNT_RESET	Reset audio sink clock counter

Field Name	Value Symbol	Value Description	Hex Value
PHASE_CNT_START_NO_WAIT	PHASE_CNT_START_NO_WAIT	Reset the AUDIOSINK_PHASE_CNT register and start the audio sink clock phase counter mechanism without waiting on a sync pulse	0x1
PERIOD_CNT_STATUS	PERIOD_CNT_IDLE	The audio sink clock period counter mechanism is idle	0x0*
	PERIOD_CNT_BUSY	The audio sink clock period counter mechanism is busy	0x1
PERIOD_CNT_STOP	PERIOD_CNT_STOP	Stop the audio sink clock period counter mechanism	0x1
PERIOD_CNT_START	PERIOD_CNT_START	Reset the AUDIOSINK_PERIOD_CNT register and start the audio sink clock period counter mechanism	0x1
PHASE_CNT_MISSED_STATUS	PHASE_CNT_NORMAL	The audio sink clock phase counter mechanism was not stopped due to a missed synchronization signal	0x0*
	PHASE_CNT_MISSED	The audio sink clock phase counter mechanism was stopped due to a missed synchronization signal	0x1
PHASE_CNT_STATUS	PHASE_CNT_IDLE	The audio sink clock phase counter mechanism is idle	0x0*
	PHASE_CNT_BUSY	The audio sink clock phase counter mechanism is busy	0x1
PHASE_CNT_STOP	PHASE_CNT_STOP	Stop the audio sink clock phase counter mechanism	0x1
PHASE_CNT_START	PHASE_CNT_START	Reset the AUDIOSINK_PHASE_CNT register and start the audio sink clock phase counter mechanism and wait on a sync pulse	0x1
CNT_RESET	CNT_RESET	Reset the AUDIOSINK_CNT register	0x1

13.3.1.2 AUDIOSINK_CFG

Bit Field	Field Name	Description
3:0	PERIODS_CFG	Defines over how many audio sink clock periods the period counter measures

Field Name	Value Symbol	Value Description	Hex Value
PERIODS_CFG	AUDIO_SINK_PERIODS_1	Measure 1 audio sink clock period	0x0*
	AUDIO_SINK_PERIODS_2	Measure 2 audio sink clock periods	0x1
	AUDIO_SINK_PERIODS_3	Measure 3 audio sink clock periods	0x2
	AUDIO_SINK_PERIODS_4	Measure 4 audio sink clock periods	0x3
	AUDIO_SINK_PERIODS_5	Measure 5 audio sink clock periods	0x4
	AUDIO_SINK_PERIODS_6	Measure 6 audio sink clock periods	0x5
	AUDIO_SINK_PERIODS_7	Measure 7 audio sink clock periods	0x6
	AUDIO_SINK_PERIODS_8	Measure 8 audio sink clock periods	0x7
	AUDIO_SINK_PERIODS_9	Measure 9 audio sink clock periods	0x8
	AUDIO_SINK_PERIODS_10	Measure 10 audio sink clock periods	0x9
	AUDIO_SINK_PERIODS_11	Measure 11 audio sink clock periods	0xA
	AUDIO_SINK_PERIODS_12	Measure 12 audio sink clock periods	0xB
	AUDIO_SINK_PERIODS_13	Measure 13 audio sink clock periods	0xC
	AUDIO_SINK_PERIODS_14	Measure 14 audio sink clock periods	0xD
	AUDIO_SINK_PERIODS_15	Measure 15 audio sink clock periods	0xE
	AUDIO_SINK_PERIODS_16	Measure 16 audio sink clock periods	0xF

13.3.1.3 AUDIOSINK_CNT

Bit Field	Field Name	Description
11:0	CNT	Audio sink clock counter value

13.3.1.4 AUDIOSINK_PHASE_CNT

Bit Field	Field Name	Description
15:0	PHASE_CNT	Audio sink clock phase counter value

13.3.1.5 AUDIOSINK_PERIOD_CNT

Bit Field	Field Name	Description
15:0	PERIOD_CNT	Audio sink clock period counter value

13.4 Asynchronous Sample Rate Converter (ASRC)

The asynchronous sample rate converter (ASRC) block provides a means of synchronizing the audio sample rate between the radio and a local source or sink for the transferred audio data.

The ASRC operates in one of four possible modes. Use of these operating modes is dependent on the relationship between the source (F_{src}) and sink (F_{sink}) frequencies, as shown in Table 34. The four modes are as follows:

- Mode 0 is used when the sink frequency is greater than the source frequency.
- Mode 1 is used when the sink and source frequencies are nearly identical. Plus or minus 25% is included to account for the possibility of frequency drift.
- Mode 2 is used when the sink frequency is guaranteed to be less than the source frequency, and greater than 0.4 times the source frequency.
- Mode 3 is used when the sink frequency is guaranteed to be less than one-half of the source frequency.

The ASRC is configured with the `ASRC_CFG` register. This register configures:

- The ASRC mode of operation as described above, controlled by the `ASRC_CFG_ASRC_MODE` setting.
- The ASRC bandwidth mode, configured with the `WDF_TYPE` bit that can select between:
  - Low delay mode: This mode provides the lowest group delay setting, but at a slightly lower bandwidth.
  - Wide band mode: This mode provides the highest bandwidth setting, but at a higher group delay.

The `ASRC_PHASE_INC` register contains a signed 32-bit value that controls the conversion rate with the ASRC. It is dependent on `ASRC_CFG_ASRC_MODE`, and on the difference between F_{src} and F_{sink}. The formulas for setting `ASRC_PHASE_INC` are listed in Table 34.

Table 34. ASRC Settings

`ASRC_CFG_ASRC_MODE`	**F_{sink} range**		`ASRC_PHASE_INC`	`ASRC_STATE_MEM` **size**	
	Minimum	**Maximum**		**Low delay**	**Wide band**
`ASRC_INT_MODE`	$>F_{src}$	N/A	$2^{29}\left(\frac{F_{src}-F_{sink}}{F_{sink}}\right)$	12	18
`ASRC_DEC_MODE1`	$0.85{*}F_{src}$	$1.125{*}F_{src}$		18	30
`ASRC_DEC_MODE2`	$0.4{*}F_{src}$	F_{src}			
`ASRC_DEC_MODE3`	$0.2{*}F_{src}$	$0.5{*}F_{src}$	$2^{27}\left(\frac{F_{src}-2\bullet F_{sink}}{F_{sink}}\right)$	18	24

The ASRC is controlled by the `ASRC_CTRL` register. This register provides the following functionality:

- The ASRC can be enabled by setting the `ASRC_CTRL_ASRC_ENABLE` bit, and disabled by setting the `ASRC_CTRL_ASRC_DISABLE` bit. Also the ASRC active status can be queried from the `ASRC_CTRL_ASRC_EN_STATUS` bit.
- The `ASRC_CTRL_ASRC_RESET` bit can be used to reset the ASRC state memory (stored in the `ASRC_STATE_MEM` registers).
- The `ASRC_CTRL_PROC_STATUS` bit indicates if the ASRC is currently processing data.
- The `ASRC_CTRL_IN_REQ` bit indicates if the `ASRC_IN` register is in use, or if it is ready for more data.
- The `ASRC_CTRL_OUT_REQ` bit indicates if the `ASRC_OUT` register contains no new data, or if it has new data that can be read.
- The `ASRC_CTRL_*_ERR` bits indicate what errors have been captured by the ASRC block, and the `ASRC_CTRL_*_ERR_CLR` bits can be written to clear these error indication bits.

The ASRC is capable of processing one audio channel at a time. In cases where more than one channel needs to be processed, the ASRC block's internal states need to be saved to memory after a block has been processed, and restored before the next block is processed, as follows:

- ASRC_PHASE_CNT: This register holds the current phase of the polyphase filter.
- ASRC_STATE_MEM: This 30-value register array holds the internal filter states of the polyphase filter. The number of states that need to be stored depends on the ASRC_CFG_ASRC_MODE setting, as outlined in Table 34 on page 395.

IMPORTANT: The ASRC must be disabled when saving or restoring its internal states. Behavior of the ASRC is undefined if the ASRC is active while its state is being updated.

Data is provided to the ASRC through the ASRC_IN register. Data is read from the ASRC through the ASRC_OUT register, and the ASRC_OUT_CNT register indicates how many samples have been generated. Typically the counter is cleared after each block of samples has been completely processed.

The ASRC supports four interrupts, configured using the ASRC_INT_ENABLE register:

1. The ASRC_INT_ENABLE_ASRC_IN_REQ bit can be enabled to trigger an ASRC_IN interrupt when more input data is required. This mode is used when providing audio samples using the ARM Cortex-M3 processor when using interrupts.
2. The ASRC_INT_ENABLE_ASRC_OUT_REQ bit can be enabled to trigger an ASRC_OUT interrupt when more output data is available. This mode is used when reading audio samples using the ARM Cortex-M3 processor when using interrupts.
3. The ASRC_INT_ENABLE_ASRC_IN_ERR bit can be enabled to trigger an ASRC_ERROR interrupt when data is written to the ASRC_IN register before processing of the previously written value is complete.
4. The ASRC_INT_ENABLE_ASRC_UPDATE_ERR bit can be enabled to trigger an ASRC_ERROR interrupt when any of the ASRC_PHASE_CNT, ASRC_PHASE_INC or ASRC_OUTPUT_CNT registers are written while processing of a sample is ongoing.

13.4.1 ASRC Registers

Register Name	Register Description	Address
ASRC_CTRL	ASRC Control Register	0x40001100
ASRC_INT_ENABLE	ASRC Interrupt Mask Register	0x40001104
ASRC_OUT	ASRC Output Data Register	0x40001108
ASRC_IN	ASRC Input Data Register	0x4000110C
ASRC_CFG	ASRC Configuration Register	0x40001110
ASRC_OUTPUT_CNT	ASRC output sample counter	0x40001114
ASRC_PHASE_INC	ASRC phase counter increment value	0x40001118
ASRC_PHASE_CNT	ASRC phase counter	0x4000111C
ASRC_STATE_MEM	ASRC State Memory 0 to 29 (32 bit)	0x40001120

13.4.1.1 ASRC_CTRL

Bit Field	Field Name	Description
14	ASRC_PROC_STATUS	The ASRC processing state
13	ASRC_OUT_REQ	The ASRC_OUT register status
12	ASRC_IN_REQ	The ASRC_IN register status
11	ASRC_UPDATE_ERR	The ASRC state/configuration update error interrupt status
10	ASRC_IN_ERR	The ASRC input interface error interrupt status
9	ASRC_UPDATE_ERR_CLR	Clear the ASRC update/configuration error interrupt status
8	ASRC_IN_ERR_CLR	Clear the ASRC input interface error interrupt
3	ASRC_RESET	Write a 1 to reset ASRC
2	ASRC_EN_STATUS	Enable status of the re-sampler block
1	ASRC_DISABLE	Disable the re-sampler block
0	ASRC_ENABLE	Enable the re-sampler block

Field Name	Value Symbol	Value Description	Hex Value
ASRC_PROC_STATUS	ASRC_IDLE	The ASRC is idle	0x0*
	ASRC_BUSY	The ASRC is busy processing a sample	0x1
ASRC_OUT_REQ	INT_IDLE_ASRC_OUT	The ASRC_OUT register is idle	0x0*
	INT_RDY_ASRC_OUT	The ASRC_OUT register is ready to be read	0x1
ASRC_IN_REQ	INT_IDLE_ASRC_IN	The ASRC_IN register is idle	0x0*
	INT_RDY_ASRC_IN	The ASRC_IN register is requesting an input	0x1
ASRC_UPDATE_ERR	INT_IDLE_ASRC_PH_CNT_ERR	This source has not set an interrupt	0x0*
	INT_PENDING_ASRC_PH_CNT_ERR	The internal states or configuration were updated while the accelerator was busy	0x1
ASRC_IN_ERR	INT_IDLE_ASRC_IN_ERR	This source has not set an interrupt	0x0*
	INT_PENDING_ASRC_IN_ERR	The ASRC input interface register was overwritten before the accelerator finished its processing	0x1
ASRC_UPDATE_ERR_CLR	CLR_ASRC_UPDATE_ERR	Clear the ASRC update error interrupt	0x1
ASRC_IN_ERR_CLR	CLR_ASRC_IN_ERR	Clear the ASRC IN error interrupt	0x1
ASRC_RESET	ASRC_RESET	Reset ASRC	0x1
ASRC_EN_STATUS	ASRC_DISABLED	The re-sampler is disabled	0x0*
	ASRC_ENABLED	The re-sampler is enabled	0x1
ASRC_DISABLE	ASRC_DISABLE	Disable the re-sampler	0x1*
ASRC_ENABLE	ASRC_ENABLE	Enable the re-sampler block	0x1

13.4.1.2 ASRC_INT_ENABLE

Bit Field	Field Name	Description
3	ASRC_UPDATE_ERR	The ASRC state/configuration update error interrupt mask
2	ASRC_IN_ERR	The ASRC input interface error interrupt mask
1	ASRC_OUT_REQ	The ASRC_OUT register interrupt status
0	ASRC_IN_REQ	The ASRC_IN register interrupt status

Field Name	Value Symbol	Value Description	Hex Value
ASRC_UPDATE_ERR	INT_DIS_ASRC_UPDATE_ERR	This source can not set an interrupt	0x0
	INT_EBL_ASRC_UPDATE_ERR	This source can set the interrupt line	0x1*
ASRC_IN_ERR	INT_DIS_ASRC_IN_ERR	This source can not set an interrupt	0x0*
	INT_EBL_ASRC_IN_ERR	This source can set the interrupt line	0x1
ASRC_OUT_REQ	INT_DIS_ASRC_OUT	This source can not set an interrupt	0x0*
	INT_EBL_ASRC_OUT	This source can set the interrupt line	0x1
ASRC_IN_REQ	INT_DIS_ASRC_IN	This source can not set an interrupt	0x0*
	INT_EBL_ASRC_IN	This source can set the interrupt line	0x1

13.4.1.3 ASRC_OUT

Bit Field	Field Name	Description
15:0	ASRC_OUT	Audio sample output

13.4.1.4 ASRC_IN

Bit Field	Field Name	Description
15:0	ASRC_IN	Audio sample input

13.4.1.5 ASRC_CFG

Bit Field	Field Name	Description
2	WDF_TYPE	WDF Type Selection
1:0	ASRC_MODE	ASRC mode

Field Name	Value Symbol	Value Description	Hex Value
WDF_TYPE	LOW_DELAY	Low Delay filter	0x0*
	WIDE_BAND	Wide band response filter	0x1
ASRC_MODE	ASRC_INT_MODE	Interpolation mode	0x0*
	ASRC_DEC_MODE1	Decimation mode 1	0x1
	ASRC_DEC_MODE3	Decimation mode 3	0x3

13.4.1.6 ASRC_OUTPUT_CNT

Bit Field	Field Name	Description
11:0	`ASRC_OUTPUT_CNT`	ASRC output sample counter

13.4.1.7 ASRC_PHASE_INC

Bit Field	Field Name	Description
31:0	`ASRC_STEP`	ASRC phase counter increment step size

13.4.1.8 ASRC_PHASE_CNT

Bit Field	Field Name	Description
31:0	`ASRC_PHASE_CNT`	ASRC phase counter

13.4.1.9 ASRC_STATE_MEM

Bit Field	Field Name	Description
31:0	`ASRC_STATE_MEM`	ASRC State Memory 0 to 29

CHAPTER 14

Private Peripherals

14.1 Nested Vectored Interrupt Controller (NVIC)

The Arm Cortex-M3 processor is closely tied to a nested vectored interrupt controller (NVIC), which is an integral part of the processor and provides the interrupt handling functionality. This block is implemented with the Arm Cortex-M3 processor and is described in the *ARM Cortex-M3 Technical Reference Manual*.

The Arm Cortex-M3 processor as implemented for RSL10 uses pulse interrupts. These interrupts are sampled on the rising edge of SYSCLK. A pulse interrupt can be reasserted during the ISR so that the interrupt can be in the pending state and active at the same time. If another pulse arrives while the interrupt is still pending, the interrupt remains pending and the ISR runs only once.

The NVIC handles a non-maskable interrupt (NMI), several faults, predefined interrupts, and a set of general-purpose interrupts that are external to the Arm Cortex-M3 processor, and linked to its interfaces and peripherals. A list of the interrupts supported by the NVIC for the Arm Cortex-M3 processor is provided in Table 35.

Table 35. Interrupts in the Arm Cortex-M3 Processor

Interrupt Enumeration Define	Enumeration Value	Vector Number	Description
Reset_IRQn	-15	1	Reset vector
NonMaskableInt_IRQn	-14	2	Non-maskable interrupt (NMI)
HardFault_IRQn	-13	3	Hard fault interrupt
MemoryManagement_IRQn	-12	4	Memory management interrupt
BusFault_IRQn	-11	5	Bus fault interrupt
UsageFault_IRQn	-10	6	Usage fault interrupt
SVCall_IRQn	-5	11	SVCall interrupt
DebugMonitor_IRQn	-4	12	Debug monitor interrupt
PendSV_IRQn	-2	14	PendSV interrupt
SysTick_IRQn	-1	15	System Tick interrupt
WAKEUP_IRQn	0	16	Wake-up interrupt
RTC_ALARM_IRQn	1	17	RTC alarm interrupt
RTC_CLOCK_IRQn	2	18	RTC clock interrupt
ADC_BATMON_IRQn	3	19	ADC/Battery monitor interrupt
TIMER0_IRQn	4	20	Timer 0 interrupt
TIMER1_IRQn	5	21	Timer 1 interrupt
TIMER2_IRQn	6	22	Timer 2 interrupt
TIMER3_IRQn	7	23	Timer 3 interrupt
DMA0_IRQn	8	24	DMA channel 0 interrupt
DMA1_IRQn	9	25	DMA channel 1 interrupt
DMA2_IRQn	10	26	DMA channel 2 interrupt
DMA3_IRQn	11	27	DMA channel 3 interrupt
DMA4_IRQn	12	28	DMA channel 4 interrupt
DMA5_IRQn	13	29	DMA channel 5 interrupt
DMA6_IRQn	14	30	DMA channel 6 interrupt
DMA7_IRQn	15	31	DMA channel 7 interrupt

Table 35. Interrupts in the Arm Cortex-M3 Processor (Continued)

Interrupt Enumeration Define	Enumeration Value	Vector Number	Description
DIO0_IRQn	16	32	DIO0 interrupt
DIO1_IRQn	17	33	DIO1 interrupt
DIO2_IRQn	18	34	DIO2 interrupt
DIO3_IRQn	19	35	DIO3 interrupt
WATCHDOG_IRQn	20	36	Watchdog interrupt
SPI0_RX_IRQn	21	37	SPI0 receive interrupt
SPI0_TX_IRQn	22	38	SPI0 transmit interrupt
SPI0_ERROR_IRQn	23	39	SPI0 error interrupt
SPI1_RX_IRQn	24	40	SPI1 receive interrupt
SPI1_TX_IRQn	25	41	SPI1 transmit interrupt
SPI1_ERROR_IRQn	26	42	SPI1 error interrupt
I2C_IRQn	27	43	I^{2}C interrupt
UART_RX_IRQn	28	44	UART receive interrupt
UART_TX_IRQn	29	45	UART transmit interrupt
UART_ERROR_IRQn	30	46	UART error interrupt
DMIC_OUT_OD_IN_IRQn	31	47	DMIC and output driver data ready interrupt
DMIC_OD_ERROR_IRQn	32	48	DMIC overrun and output driver underrun detect interrupt
PCM_RX_IRQn	33	49	PCM receive interrupt
PCM_TX_IRQn	34	50	PCM transmit interrupt
PCM_ERROR_IRQn	35	51	PCM error interrupt
DSP0_IRQn	36	52	DSP event interrupt 0
DSP1_IRQn	37	53	DSP event interrupt 1
DSP2_IRQn	38	54	DSP event interrupt 2
DSP3_IRQn	39	55	DSP event interrupt 3
DSP4_IRQn	40	56	DSP event interrupt 4
DSP5_IRQn	41	57	DSP event interrupt 5
DSP6_IRQn	42	58	DSP event interrupt 6
DSP7_IRQn	43	59	DSP event interrupt 7
BLE_CSCNT_IRQn	44	60	BLE 625 μs time reference interrupt
BLE_SLP_IRQn	45	61	BLE Sleep Mode interrupt
BLE_RX_IRQn	46	62	BLE received packet interrupt
BLE_EVENT_IRQn	47	63	BLE event interrupt
BLE_CRYPT_IRQn	48	64	BLE AES ciphering complete interrupt
BLE_ERROR_IRQn	49	65	BLE baseband error interrupt
BLE_GROSSTGTIM_IRQn	50	66	BLE gross timer interrupt
BLE_FINETGTIM_IRQn	51	67	BLE fine timer interrupt
BLE_SW_IRQn	52	68	BLE SW triggered interrupt
BLE_COEX_RX_TX_IRQn	53	69	RF coexistence Bluetooth low energy technology start/stop Rx or Tx interrupt

Table 35. Interrupts in the Arm Cortex-M3 Processor (Continued)

Interrupt Enumeration Define	Enumeration Value	Vector Number	Description
BLE_COEX_IN_PROCESS_IRQn	54	70	RF coexistence Bluetooth low energy technology in process interrupt
RF_TX_IRQn	55	71	Bluetooth low energy technology transmit interrupt
RF_RXSTOP_IRQn	56	72	Bluetooth low energy technology receive stop interrupt
RF_IRQ_RECEIVED_IRQn	57	73	Bluetooth low energy technology received packet interrupt
RF_SYNC_IRQn	58	74	Bluetooth low energy technology received sync word interrupt
RF_TXFIFO_IRQn	59	75	Tx FIFO near underflow detect interrupt
RF_RXFIFO_IRQn	60	76	Rx FIFO near overflow detect interrupt
ASRC_ERROR_IRQn	61	77	ASRC error interrupt
ASRC_IN_IRQn	62	78	ASRC data input interrupt
ASRC_OUT_IRQn	63	79	ASRC data output interrupt
AUDIOSINK_PHASE_IRQn	64	80	Audio sink clock phase interrupt
AUDIOSINK_PERIOD_IRQn	65	81	Audio sink clock period interrupt
CLKDET_IRQn	66	82	Clock detection interrupt
FLASH_COPY_IRQn	67	83	Flash copy interrupt
FLASH_ECC_IRQn	68	84	Flash ECC event interrupt
MEMORY_ERR_IRQn	69	85	Memory error interrupt
BLE_AUDIO0	70	86	Audio over Bluetooth low energy technology channel 0 interrupt
BLE_AUDIO1	71	87	Audio over Bluetooth low energy technology channel 1 interrupt
BLE_AUDIO2	72	88	Audio over Bluetooth low energy technology channel 2 interrupt

Table 36 lists the NVIC registers. The following subsections describe their bit fields and use by the RSL10 microcontroller.

Table 36. NVIC Control Registers

Register Name	Register Description	Address
`SCnSCB_ICTR`	NVIC Interrupt Controller Type Register	0xE000E004
`NVIC_ISER0`	NVIC External Interrupt Set Enable Register 0	0xE000E100
`NVIC_ISER1`	NVIC External Interrupt Set Enable Register 1	0xE000E104
`NVIC_ISER2`	NVIC External Interrupt Set Enable Register 2	0xE000E108
`NVIC_ICER0`	NVIC External Interrupt Clear Enable Register 0	0xE000E180
`NVIC_ICER1`	NVIC External Interrupt Clear Enable Register 1	0xE000E184
`NVIC_ICER2`	NVIC External Interrupt Clear Enable Register 2	0xE000E188
`NVIC_ISPR0`	NVIC External Interrupt Set Pending Register 0	0xE000E200
`NVIC_ISPR1`	NVIC External Interrupt Set Pending Register 1	0xE000E204

Table 36. NVIC Control Registers (Continued)

Register Name	Register Description	Address
NVIC_ISPR2	NVIC External Interrupt Set Pending Register 2	0xE000E208
NVIC_ICPR0	NVIC External Interrupt Clear Pending Register 0	0xE000E280
NVIC_ICPR1	NVIC External Interrupt Clear Pending Register 1	0xE000E284
NVIC_ICPR2	NVIC External Interrupt Clear Pending Register 2	0xE000E288
NVIC_IABR0	NVIC External Interrupt Active Register 0	0xE000E300
NVIC_IABR1	NVIC External Interrupt Active Register 1	0xE000E304
NVIC_IABR2	NVIC External Interrupt Active Register 2	0xE000E308
NVIC_IP0	NVIC External Interrupt Priority Register 0	0xE000E400
NVIC_IP1	NVIC External Interrupt Priority Register 1	0xE000E404
NVIC_IP2	NVIC External Interrupt Priority Register 2	0xE000E408
NVIC_IP3	NVIC External Interrupt Priority Register 3	0xE000E40C
NVIC_IP4	NVIC External Interrupt Priority Register 4	0xE000E410
NVIC_IP5	NVIC External Interrupt Priority Register 5	0xE000E414
NVIC_IP6	NVIC External Interrupt Priority Register 6	0xE000E418
NVIC_IP7	NVIC External Interrupt Priority Register 7	0xE000E41C
NVIC_IP8	NVIC External Interrupt Priority Register 8	0xE000E420
NVIC_IP9	NVIC External Interrupt Priority Register 9	0xE000E424
NVIC_IP10	NVIC External Interrupt Priority Register 10	0xE000E428
NVIC_IP11	NVIC External Interrupt Priority Register 11	0xE000E432
NVIC_IP12	NVIC External Interrupt Priority Register 12	0xE000E436
NVIC_IP13	NVIC External Interrupt Priority Register 13	0xE000E440
NVIC_IP14	NVIC External Interrupt Priority Register 14	0xE000E444
NVIC_IP15	NVIC External Interrupt Priority Register 15	0xE000E448
NVIC_IP16	NVIC External Interrupt Priority Register 16	0xE000E452
NVIC_IP17	NVIC External Interrupt Priority Register 17	0xE000E456
SCB_CPUID	NVIC CPU ID Base Register	0xE000ED00
SCB_ICSR	NVIC Interrupt Control and State Register	0xE000ED04
SCB_VTOR	NVIC Vector Table Offset Register	0xE000ED08
SCB_AIRCR	NVIC Application Interrupt and Reset Control Register	0xE000ED0C
SCB_SCR	NVIC System Control Register	0xE000ED10
SCB_CCR	NVIC Configuration Control Register	0xE000ED14
SCB_SHP0	NVIC System Interrupt Priority Register 0	0xE000ED18
SCB_SHP1	NVIC System Interrupt Priority Register 1	0xE000ED1C
SCB_SHP2	NVIC System Interrupt Priority Register 2	0xE000ED20
SCB_SHCSR	NVIC System Handler Control and State Register	0xE000ED24
SCB_CFSR	NVIC Fault Status Register	0xE000ED28
SCB_HFSR	NVIC Hard Fault Status Register	0xE000ED2C
SCB_DFSR	NVIC Debug Fault Status Register	0xE000ED30

Table 36. NVIC Control Registers (Continued)

Register Name	Register Description	Address
SCB_MMFAR	NVIC Memory Management Fault Address Register	0xE000ED34
SCB_BFAR	NVIC Bus Fault Address Register	0xE000ED38
NVIC_STIR	NVIC Software Trigger Interrupt Register	0xE000EF00

IMPORTANT: The NVIC is a standard component provided with the Arm Cortex-M3 processor. The registers for this peripheral are defined in *core_cm3.h* and augmented by defines for the bit fields, bit settings and subregisters in *rsl10_hw.h*.

14.1.1 Interrupt Controller Type Register

The Interrupt Controller Type register (`SCnSCB_ICTR`) indicates the number of interrupts supported by the NVIC. This register is from the Arm Cortex-M3 processor's `SCnSCB` register block. Table 37 describes this register.

Table 37. Interrupt Controller Type Register Bit Assignments

Bit Field	Field Name	Description
4:0	INTLINESNUM	Total number of interrupt line groups: RSL10 uses 69 external interrupts (NVIC_INTLINESNUM_65_96, hex value 0x2)

14.1.2 Interrupt Set Enable and Clear Enable Registers

Use the Interrupt Set Enable registers (`NVIC_ISER[0]` to `NVIC_ISER[2]`) to:

- Enable interrupts
- Determine which interrupts are currently enabled

Use the Interrupt Clear-Enable registers (`NVIC_ICER[0]` to `NVIC_ICER[2]`) to:

- Disable interrupts
- Determine which interrupts are currently disabled

These registers contain a bit for each of the external interrupts (vectors 16 to 57) listed in Table 35 on page 400. Setting a bit in an Interrupt Set-Enable register enables the corresponding interrupt. Setting a bit in an Interrupt Clear-Enable register disables the corresponding interrupt.

When the enable bit of a pending interrupt is set, the processor activates the interrupt based on its priority. When the enable bit is cleared, asserting its interrupt signal pends the interrupt, but it is not possible to activate the interrupt, regardless of its priority. Therefore, a disabled interrupt can serve as a latched general-purpose I/O bit that can be read and cleared without invoking an interrupt.

NOTE: Clearing an Interrupt Set-Enable Register bit does not affect currently active interrupts. It only prevents new activations.

These registers are part of the Arm Cortex-M3 processor's `NVIC` register block. Table 38 describes the field of the Interrupt Set-Enable registers. Table 39 describes the field of the Interrupt Clear-Enable registers.

Table 38. Interrupt Set-Enable Register Bit Assignments

Bit Field	Field Name	Description
31:0	SETENA	Interrupt set enable bits. For write operation: • 1 = Enable interrupt • 0 = No effect. For read operation: • 1 = Interrupt enabled • 0 = Interrupt disabled Writing 0 to a SETENA bit has no effect. Reading the bit returns its current enable state. Reset clears the SETENA fields.

Table 39. Interrupt Clear-Enable Register Bit Assignments

Bit Field	Field Name	Description
31:0	CLRENA	Interrupt clear-enable bits. For write operation: • 1 = Disable interrupt • 0 = No effect. For read operation: • 1 = Enable interrupt • 0 = Disable interrupt. Writing 0 to a CLRENA bit has no effect. Reading the bit returns its current enable state.

14.1.3 Interrupt Set-Pending Registers and Interrupt Clear-Pending Registers

Use the Interrupt Set-Pending Registers (`NVIC_ISPR[0]` to `NVIC_ISPR[2]`) to:

- Force interrupts into the pending state
- Determine which interrupts are currently pending

Use the Interrupt Clear-Pending Registers (`NVIC_ICPR[0]` to `NVIC_ICPR[2]`) to:

- Clear pending interrupts
- Determine which interrupts are currently pending

These registers contain a bit for each of the external interrupts (vectors 16 to 57) listed in Table 35 on page 400. Setting a bit in an Interrupt Set-Pending register causes the corresponding interrupt to be pending. Setting a bit in an Interrupt Clear-Pending register puts the interrupt into the non-pending state.

NOTE: Writing to an Interrupt Set-Pending register has no effect on an interrupt that is already pending or is disabled. Similarly, writing to an Interrupt Clear-Pending register has no effect on an interrupt that is active unless it is also pending.

These registers are part of the Arm Cortex-M3 processor's `NVIC` register block. Table 40 describes the Interrupt Set-Pending registers. Table 41 describes the Interrupt Clear-Pending registers.

Table 40. Interrupt Set-Pending Register Bit Assignments

Bit Field	Field Name	Description
31:0	SETPEND	Interrupt set-pending bits: • 1 = Pend the corresponding interrupt • 0 = Corresponding interrupt not pending Writing 0 to a SETPEND bit has no effect. Reading the bit returns its current state.

Table 41. Interrupt Clear-Pending Registers Bit Assignments

Bit Field	Field Name	Description
31:0	CLRPEND	Interrupt clear-pending bits: • 1 = Clear pending interrupt • 0 = Do not clear pending interrupt Writing 0 to a CLRPEND bit has no effect. Reading the bit returns its current state.

14.1.4 Active Bit Register

Read the Active Bit registers (`NVIC_IABR[0]` to `NVIC_IABR[2]`) to determine which interrupts are active. This register contains a flag for each of the external interrupts (vectors 16 to 57) listed in Table 35 on page 400. These registers are part of the Arm Cortex-M3 processor's `NVIC` register block. Table 42 describes the Active Bit register.

Table 42. Active Bit Register Bit Assignments

Bit Field	Field Name	Description
31:0	ACTIVE	Interrupt active flags: • 1 = Interrupt active or pre-empted and stacked • 0 = Interrupt not active nor stacked.

14.1.5 Interrupt Priority Registers

Use the Interrupt Priority Registers (`NVIC_IP[0]` to `NVIC_IP[9]`) to assign a priority to each of the available interrupts. Each byte in an Interrupt Priority can be used to set the priority for one of the external interrupts (vectors 16 to 84) listed in Table 35 on page 400.

NOTE: Configuration of the interrupt priorities for standard Arm Cortex-M3 processor exceptions (vectors 4 to 15) are set using the System Handler Priority registers. For more information, see the *ARMv7M Architecture Reference Manual*.

The NVIC for the Arm Cortex-M3 processor in the RSL10 system has been implemented with three interrupt priority bits per interrupt. These three priority bits are MSB aligned to an eight-bit priority bit field as required by Arm. Generally, the lower the priority value, the higher the priority that interrupt is given.

The `SCB_AIRCR_PRIGROUP` bit field from the Application Interrupt and Reset Control register (see the *ARMv7M Architecture Reference Manual*) is used to divide the interrupt priority settings into interrupt groups, and to prioritize interrupts within those groups. The possible configurations for the division of the priority bit field into pre-emption priority and subgroup priority is shown in Table 43.

Table 43. Division of Priority into Pre-Empt Priority and Subgroup Priority

PRIGROUP Setting	Pre-Empt Priority Field	Subgroup Priority Field
0x0 to 0x4	7:5	-
0x5	7:6	5
0x6	7	6:5
0x7	-	7:5

When choosing which interrupt to activate, the priority settings are applied as follows:

- When multiple interrupts are pending, but no interrupts are active, the interrupt with the lowest priority setting is activated. If more than one pending interrupt shares the lowest priority setting, the interrupt with the lower vector number is activated.
- If an interrupt is currently active, it can be pre-empted by any interrupt that is pended in a lower numbered group. If multiple interrupts that could pre-empt the active interrupt are pending, the interrupt with the lowest priority setting is activated. As before, if more than one pending interrupt shares the lowest priority setting, the interrupt with the lower vector number is activated.

For example, setting the `SCB_AIRCR_PRIGROUP` bit field to 0x6 divides the three interrupt priority bits to use bits [7:6] to assign the interrupt to a group, and bit [5] to assign the interrupt a priority relative to the other interrupts in that group. Suppose the following interrupts are pending:

- `TIMER0` with priority set to 0xC0
- `DMA0` with priority set to 0xB0
- `DIO0` with priority set to 0xB0

In this example, the `DIO0` and `DMA0` interrupts have the lowest priority, and the `DMA0` interrupt is activated because it has a lower vector number (24 versus 32). If the `TIMER0` interrupt is then pended with a priority of 0x80, the `DMA0` interrupt is not pre-empted because the `TIMER0`, `DMA0` and `DIO` interrupts all share the same priority group. If the `WAKEUP` interrupt is then pended with a priority level of 0x20, the `DMA0` interrupt is pre-empted because the `WAKEUP` interrupt belongs to a higher priority interrupt group.

IMPORTANT: The reset, NMI and fault vectors have priority levels of -3, -2, and -1 respectively. As such, these events can always pre-empt interrupts with lower priorities.

These registers are part of the Arm Cortex-M3 processor's `NVIC` register block. Table 44 describes the bit assignments for a single interrupt within the Interrupt Priority registers.

Table 44. Interrupt Priority Register Bit Assignments

Bit Field	Field Name	Description
7:5	PRI_**`n`**	Priority of interrupt *n*

14.1.6 Registers Described by Arm Documentation

The following registers are documented in the *ARMv7M Architecture Reference Manual*:

- Interrupt Control State register
- Vector Table Offset register

- Application Interrupt and Reset Control register
- System Control register
- Configuration Control register
- System Handler Priority registers
- System Handler Control and State register
- Configurable Fault Status register
- Software Trigger Interrupt register

14.2 SysTick

The Arm Cortex-M3 core peripherals include the system tick (SysTick) count-down timer from the Arm Cortex-M3 processor implementation. This block is implemented as part of the NVIC, and is described in the *ARM Cortex-M3 Technical Reference Manual*.

The clock used by the SysTick timer is selected using the `SysTick_CTRL_CLKSOURCE` bit from the `SysTick_CTRL` register. This timer can be clocked from the system clock (SYSCLK) or from the SysTick-specific reference clock (STCLK) that is divided from SLOWCLK by 32. For more information about SYSCLK, see Section 6.3.1, “System Clock (SYSCLK)” on page 78. For more information about SLOWCLK, see Section 6.3.3, “Slow Clock (SLOWCLK)” on page 80. The `SysTick_CALIB` register is configured to define a 10 ms timer period based on STCLK.

The delay provided by the SysTick timer is defined using the selected clock and a reload value loaded to the `SysTick_LOAD` register, as follows:

$$\text{DELAY} = \frac{(\text{SysTick\_LOAD} + 1)}{f_{\text{SYSCLK or STCLK}}}$$

The current value of the SysTick counter can be read at any time from the `SysTick_VAL` register.

The SysTick timer is enabled by setting the `SysTick_CTRL_ENABLE` bit in the `SysTick_CTRL` register. SysTick interrupts are enabled by setting the `SysTick_CTRL_TICKINT` bit in the `SysTick_CTRL` register. The `SysTick_CTRL_COUNTFLAG` bit in the `SysTick_CTRL` register indicates if the SysTick timer has reached zero since the last time this register was read, and is cleared automatically after being read. This bit can be used if an application uses polling instead of interrupting to monitor for SysTick timer events.

IMPORTANT: If the SysTick timer is sourced from `SYSCLK`, and the clock to the Arm Cortex-M3 processor is gated due to the use of a wait-for-interrupt (`WFI`) or wait-for-event (`WFE`) instruction, the SysTick timer will be clocked at a much slower rate while waiting for the interrupt or event to occur. If using these instructions, we recommend using `STCLK` as the source for the SysTick timer or using a general-purpose timer running at `SLOWCLK` divided by 2 if the clock source should be faster.

14.2.1 SysTick Control and Configuration Registers

Register Name	Register Description	Address
`SysTick_CTRL`	SYSTICK Control and Status Register	0xE000E010
`SysTick_LOAD`	SYSTICK Reload Value Register	0xE000E014
`SysTick_VAL`	SYSTICK Current Value Register	0xE000E018
`SysTick_CALIB`	SYSTICK Calibration Register	0xE000E01C

IMPORTANT: The SysTick timer is a standard component provided with the Arm Cortex-M3 processor. The registers for this peripheral are defined in *core_cm3.h* and augmented by defines for the bit fields, bit settings and subregisters in *rsl10_hw.h*.

14.2.1.1 SysTick_CTRL

Bit Field	Field Name	Description
16	`COUNTFLAG`	Reads as 1 if SYSTICK counter has reached 0 since the last time the timer has reached 0. Clears to 0 on read.
2	`CLKSOURCE`	SYSTICK timer clock source
1	`TICKINT`	SYSTICK timer interrupt enable
0	`ENABLE`	SYSTICK timer enable

Field Name	Value Symbol	Value Description	Hex Value
`COUNTFLAG`	`SYSTICK_COUNTFLAG_NOT_ZERO`	SYSTICK counter has not reached zero since last read	0x0*
	`SYSTICK_COUNTFLAG_ZERO`	SYSTICK counter has reached zero since last read	0x1
`CLKSOURCE`	`SYSTICK_CLKSOURCE_EXTREF_CLK`	Use external reference clock (STCLK)	0x0*
	`SYSTICK_CLKSOURCE_CORE_CLK`	Use the core clock	0x1
`TICKINT`	`SYSTICK_TICKINT_DISABLE`	Disable interrupt generation when SYSTICK timer reaches 0	0x0*
	`SYSTICK_TICKINT_ENABLE`	Enable interrupt generation when SYSTICK timer reaches 0	0x1
`ENABLE`	`SYSTICK_DISABLE`	Disable SYSTICK timer	0x0*
	`SYSTICK_ENABLE`	Enable SYSTICK Timer	0x1

14.2.1.2 SysTick_LOAD

Bit Field	Field Name	Description
23:0	`RELOAD`	Counter reload value for the SYSTICK timer when it reaches 0

14.2.1.3 SysTick_VAL

Bit Field	Field Name	Description
23:0	`CURRENT`	Current value of the SYSTICK counter value. Write to clear counter.

14.2.1.4 SysTick_CALIB

Bit Field	Field Name	Description
31	`NOREF`	Indicates if a reference clock is available
30	`SKEW`	Indicates if calibration value is exactly 10 ms or not
23:0	`TENMS`	SYSTICK counter calibration value for 10 ms. A value of 0 means the calibration value is not available

Field Name	Value Symbol	Value Description	Hex Value
NOREF	SYSTICK_NOREF	No external reference clock available	0x1
	SYSTICK_REF	External reference clock available	0x0*
SKEW	SYSTICK_SKEW	Calibration value is not exactly 10 ms	0x1
	SYSTICK_NOSKEW	Calibration value is exactly 10 ms	0x0*

14.3 DEBUG CONTROLLER

14.3.1 Halting Debug Configuration and Status

The Debug Halting Control and Status Register (DHCSR) provides status information on the processor state, enables core debugging, and allows an external system to halt and single-step the core. To write to this register, DEBUG_HALT_KEY must be written to the CoreDebug_DHCSR_DBGKEY bit field.

The DHCSR is used to configure the Arm Cortex-M3 processor for halting debug. To enable halting debug, set the CoreDebug_DHCSR_C_DEBUGEN bit. If halting debug is enabled:

- To halt the core, set the CoreDebug_DHCSR_C_HALT bit.
- To single-step the core, set the CoreDebug_DHCSR_C_STEP bit.
- To mask interrupts while single-stepping, set the CoreDebug_DHCSR_C_MASKINTS bit (the core must be halted to write this bit).
- To break a stalled memory access, where the memory access might be stalled due to a memory conflict with another component of the RSL10 system, set the CoreDebug_DHCSR_C_SNAPALL bit.

The Debug Exception and Monitor Control Register (DEMCR) contains a number of possible exception conditions that the debug tools might want to monitor during debug. When enabled, each of these vector catch configuration bits monitors for the specified fault or reset event. When a fault or event that is being monitored is detected, a core halt request is used to halt the core as soon as the currently executing instruction completes. Supported vector catch events include debug traps that trigger on:

- A core reset
- A memory management fault
- Usage faults for:
  - No coprocessor errors
  - Unaligned accesses or division by 0
  - State errors

- A bus fault
- Errors when handling an interrupt or exception
- A hard fault

The DHCSR also provides a variety of debug related status information, including:

- If the core has been reset or is resetting (CoreDebug_DHCSR_S_RESET_ST); this bit is cleared when read
- If an instruction has completed execution since this register has been last read; this bit is cleared when read
- If the core is in a locked state
- If the core is in Sleep Mode
- If the core has been halted
- If the most recent register read/write has completed; for more information, see Section 14.3.3, “Arm Cortex-M3 Processor Core Register Access”

NOTE: The DHCSR and all DEMCR bits that are not related to the debug monitor are only reset if a POR or similar occurs. These registers are not reset for a core reset. For more information about resets, see Section 5.5, “Resets” on page 63.

CAUTION: We strongly recommend that only the debugger use the DHCSR because accesses to this register from application code can interfere with the debug behavior of the Arm Cortex-M3 processor debug port.

14.3.2 Debug Monitor Configuration

The NVIC from the Arm Cortex-M3 processor contains a debug monitor that can be used to control debug activities. The debug monitor is tied to the debug monitor system interrupt (vector number 12) and is configured using the `DEMCR` register. To enable the debug monitor and debug monitor exception, set the `CoreDebug_DEMCR_MON_EN` bit from the `DEMCR` register. To manually pend the debug monitor exception, set the `CoreDebug_DEMCR_MON_PEND` bit from the `DEMCR` register. To single-step the core using the debug monitor (if the debug monitor is enabled), set the `CoreDebug_DEMCR_MON_STEP` bit from the `DEMCR` register.

The `CoreDebug_DEMCR_MON_REQ` bit from the `DEMCR` register indicates whether a debug monitor event was caused by a manual request or a debug event (including a debug trap).

14.3.3 Arm Cortex-M3 Processor Core Register Access

The Arm Cortex-M3 debug port includes a pair of registers that the debug port uses to provide read and write access to the Arm Cortex-M3 processor’s core registers: the Debug Core Register Selector Register (`DCRSR`) and the Debug Core Register Data Register (`DCRDR`). The `DCRSR` contains the selection of the register to be read or written, and the type of access used. To define the read/write direction, write `REGWNR_READ` or `REGWNR_WRITE` to the `CoreDebug_DCRSR_REGWnR` bit field. To set the register to be read, use the `REGSEL_*` bit settings for the `CoreDebug_DCRSR_REGSEL` bit field from the `DCRSR`.

Data written using the `DCRSR` is copied from the `DCRDR` to the specified core register. Similarly, data read using the `DCRSR` is written to the `DCRDR`, where it can be accessed using debug port memory reads. If the selector register selects the core special registers, the data read or written is interpreted using the bit fields described in Table 45.

Table 45. Debug Core Register Data Register Special Register Mapping

Bit Field	Arm Cortex-M3 Processor Register
31:24	CONTROL
23:16	FAULTMASK
15:8	BASEPRI
7:0	PRIMASK

14.3.4 Debug Fault Status Register

The Debug Fault Status register (`DFSR`) is used to monitor debug events including:

- External debug requests
- Vector catches
- Data watchpoint matches
- `BKPT` instruction execution
- Halt requests

Each flag in the Debug Fault Status register is set independently when its debug condition occurs. The bits in this register are not set unless the event is caught. One of four things occurs if an event is detected:

If halting debug is enabled

Debug events stop the processor by going into debug mode.

If halting debug is disabled and the debug monitor is enabled

Debug events trigger a debug monitor handler call, if priority permits.

If halting debug and the debug monitor are both disabled

Some debug events are interpreted as Hard Faults, setting the `SCB_HFSR_DBGEVT` bit in the Hard Fault status register. This always includes `BKPT` events, and includes VCATCH events (see Section 14.3.5.1, "SCB_DFSR Settings" on page 412 for descriptions of the events) if those events are enabled in the Debug Exception and Monitor Control Register (see Section 14.3.5.4, "Debug Exception and Monitor Control Register" on page 413).

If not interpreted as a hard fault, the debug event is ignored.

This register is part of the Arm Cortex-M3 processor's `SCB` register block.

14.3.5 Arm Cortex-M3 Processor Debug Port Specific Control and Configuration Registers

Register Name	Register Description	Address
`SCB_DFSR`	NVIC Debug Fault Status Register	0xE000ED30
`CoreDebug_DHCSR`	Debug Halting Control and Status Register	0xE000EDF0
`CoreDebug_DCRSR`	Debug Core Register Selector Register	0xE000EDF4
`CoreDebug_DCRDR`	Debug Core Register Data Register	0xE000EDF8
`CoreDebug_DEMCR`	Debug Exception and Monitor Control Register	0xE000EDFC

IMPORTANT: The Arm Cortex-M3 core debug port components are part of a standard component provided with the Arm Cortex-M3 processor. The registers for this peripheral are defined in *core_cm3.h* and augmented by defines for the bit fields, bit settings and subregisters in *rsl10_hw.h*. These registers are only accessible through the Arm Cortex-M3 processor's private peripheral bus (see Section 7.1, "Memory Architecture" on page 93).

14.3.5.1 SCB_DFSR Settings

Bit Field	Field Name	Description
4	EXTERNAL	Indicate if an external debug request has been asserted. The processor stops on the next instruction boundary.
3	VCATCH	Indicate if a vector catch occurred. When the VCATCH flag is set, a flag in one of the local fault status registers is also set to indicate the type of fault. For information about configuring the system to catch specific vectors, see Section 14.3.5.4, "Debug Exception and Monitor Control Register".
2	DWTTRAP	Indicates DWT match occurred

Bit Field	Field Name	Description
1	BKPT	Indicate if a `BKPT` instruction executed. The `BKPT` flag is set by a `BKPT` instruction in code. The PC's return value points to the instruction containing the breakpoint.
0	HALTED	Indicate if a halt was requested by the NVIC (including a halt due to a single-step debug operation).

14.3.5.2 CoreDebug_DHCSR Settings

Bit Field	Field Name	Description
31:16	KEY	Must be written with the DEBUG_HALT_KEY (0xA05F) to write to other parts of this register
25	S_RESET_ST	Status bit indicating if the core was reset or is being reset since this register was last read
24	S_RETIRE_ST	Status bit indicating an instruction completed since this register was last read
19	S_LOCKUP	Status bit indicating if the Arm Cortex-M3 processor is in a locked state
18	S_SLEEP	Status bit indicating if the Arm Cortex-M3 processor is in Sleep Mode
17	S_HALT	Status bit indicating if the Arm Cortex-M3 processor is halted
16	S_REGRDY	Status bit indicating if a register read or write has completed since this register was last read
5	C_SNAPSTALL	Control bit used to break a stalled memory access
3	C_MASKINTS	Control bit used to mask interrupts while single-stepping; this bit can only be modified if the Arm Cortex-M3 processor is halted
2	C_STEP	Control bit used to single-step the Arm Cortex-M3 processor if halt mode debugging is enabled
1	C_HALT	Control bit used to halt the Arm Cortex-M3 processor if halt mode debugging is enabled
0	C_DEBUGEN	Control bit used to enable halt mode debugging (cannot be written by the core itself)

14.3.5.3 CoreDebug_DCRSR Settings

Bit Field	Field Name	Description
16	REGWnR	Write/not-read bit controlling the direction of the register transfer
4:0	REGSEL	Select the core register to read from or write to

14.3.5.4 Debug Exception and Monitor Control Register

Bit Field	Field Name	Description
19	MON_REQ	Indicate if a debug monitor exception has been triggered by a pending manual request or a hardware debug event
18	MON_STEP	Single-step the core; this bit can only be written if the debug monitor exception is enabled
17	MON_PEND	Manually pend a debug monitor exception
16	MON_EN	Enable the debug monitor exception
10	VC_HARDERR	Capture hard fault events
9	VC_INTERR	Capture errors encountered when serving an interrupt

Bit Field	Field Name	Description
8	VC_BUSERR	Capture bus fault events
7	VC_STATERR	Capture usage faults due to a state error
6	VC_CHKERR	Capture usage faults for the optionally enabled usage errors (i.e., due to an unaligned access or division by zero); for information on configuring these optional error sources, see the Configuration Control Register in *ARMv7M Architecture Reference Manual*.
5	VC_NOCPERR	Capture usage faults due to a no-coprocessor error
4	VC_MMERR	Capture memory management faults
0	VC_CORERESET	Capture core reset events

APPENDIX A
Control and Configuration Registers

This appendix lists all the registers that are available. Refer to the appropriate section for information about the control and configuration registers for a block. The sections are:

- Section A.1, “Chip Identification” on page 416
- Section A.2, “System Control” on page 417
- Section A.3, “Clock Generation” on page 422
- Section A.4, “Reset” on page 423
- Section A.5, “Watchdog Timer” on page 424
- Section A.6, “General-Purpose Timers 0, 1, 2 and 3” on page 425
- Section A.7, “Flash Interface Configuration and Control” on page 426
- Section A.8, “DMA Controller Configuration and Control” on page 429
- Section A.9, “DIO Interface and Digital Pad control” on page 431
- Section A.10, “SPI Interface Configuration and Control” on page 433
- Section A.11, “SPI Interface Configuration and Control” on page 434
- Section A.12, “PCM Interface Configuration and Control” on page 435
- Section A.13, “I2C Interface Configuration and Control” on page 436
- Section A.14, “UART Interface Configuration and Control” on page 438
- Section A.15, “PWM 0 and 1 Configuration and Control” on page 439
- Section A.16, “DMIC Input and Output Driver Configuration and Control” on page 440
- Section A.17, “CRC Generator Control” on page 443
- Section A.18, “Audio Sink Clock Counters” on page 444
- Section A.19, “ASRC Configuration and Control” on page 445
- Section A.20, “Analog-to-Digital Converter and Battery Monitoring” on page 446
- Section A.21, “ACS domain (Analog Bridge Access)” on page 447
- Section A.22, “Baseband Controller Interface” on page 452
- Section A.23, “Baseband Controller” on page 453
- Section A.24, “RF Front-End 2.4 GHz” on page 466
- Section A.25, “SYSTICK Timer” on page 493
- Section A.26, “System Control and ID register not in the SCB” on page 494
- Section A.27, “Nested Vector Interrupt Controller” on page 495
- Section A.28, “System Control Block” on page 512
- Section A.29, “Debug Controller” on page 515

A.1 CHIP IDENTIFICATION

Address	Register Name	Register Write	Register Read	Default	Description
0x1FFFFFFC	AHBREGS_CHIP_ID_NUM	-	(31:24) CHIP_FAMILY	0X9	Chip Family number
		-	(23:16) CHIP_VERSION	0X1	Chip Version number
		-	(15:8) CHIP_MAJOR_REVISION	0X1	Chip Major Revision number
		-	(7:3) CHIP_MINOR_REVISION	0X0	Chip Minor Revision number
		-	(2) OD_FEATURE	0X0	OD feature
		-	(1) LPDSP32_FEATURE	0X0	LPDSP32 feature
		-	(0) AOBLE_FEATURE	0X0	AOBLE feature

A.2 SYSTEM CONTROL

Address	Register Name	Register Write	Register Read	Default	Description
0x40000000	SYSCTRL_DSS_CTRL	-	(8) LPDSP32_STATUS	0X1	LPDSP32 feature status
		(5) DSS_CSS_INT_RESET	-	N/A	Write a 1 to reset pending CSS interrupts in the DSS interrupt controller
		(4) DSS_DMA_INT_RESET	-	N/A	Write a 1 to reset pending DMA interrupts in the DSS interrupt controller
		(3) DSS_RESET	-	N/A	Write a 1 to reset DSS
		(2) LPDSP32_PAUSE	-	N/A	Write a 1 to pause LPDSP32
		(1) LPDSP32_RESUME	-	N/A	Write a 1 to run LPDSP32
		-	(0) LPDSP32_RUNNING	0X0	LPDSP32 running status
0x40000004	SYSCTRL_DSS_CMD	(6) DSS_CMD_6	-	N/A	Write a 1 to issue DSS command 6
		(5) DSS_CMD_5	-	N/A	Write a 1 to issue DSS command 5
		(4) DSS_CMD_4	-	N/A	Write a 1 to issue DSS command 4
		(3) DSS_CMD_3	-	N/A	Write a 1 to issue DSS command 3
		(2) DSS_CMD_2	-	N/A	Write a 1 to issue DSS command 2
		(1) DSS_CMD_1	-	N/A	Write a 1 to issue DSS command 1
		(0) DSS_CMD_0	-	N/A	Write a 1 to issue DSS command 0
0x40000008	SYSCTRL_FLASH_OVERLAY_CFG	(7) DSP_PRAM0_OVERLAY_CFG	(7) DSP_PRAM0_OVERLAY_CFG	0X0	DSP_PRAM0 Flash overlay configuration
		(6) DSP_PRAM1_OVERLAY_CFG	(6) DSP_PRAM1_OVERLAY_CFG	0X0	DSP_PRAM1 Flash overlay configuration
		(5) DSP_PRAM2_OVERLAY_CFG	(5) DSP_PRAM2_OVERLAY_CFG	0X0	DSP_PRAM2 Flash overlay configuration
		(4) DSP_PRAM3_OVERLAY_CFG	(4) DSP_PRAM3_OVERLAY_CFG	0X0	DSP_PRAM3 Flash overlay configuration
		(3) PRAM3_OVERLAY_CFG	(3) PRAM3_OVERLAY_CFG	0X0	PRAM3 Flash overlay configuration
		(2) PRAM2_OVERLAY_CFG	(2) PRAM2_OVERLAY_CFG	0X0	PRAM2 Flash overlay configuration
		(1) PRAM1_OVERLAY_CFG	(1) PRAM1_OVERLAY_CFG	0X0	PRAM1 Flash overlay configuration
		(0) PRAM0_OVERLAY_CFG	(0) PRAM0_OVERLAY_CFG	0X0	PRAM0 Flash overlay configuration
0x4000000C	SYSCTRL_CSS_LOOP_CACHE_CFG	(0) CSS_LOOP_CACHE_ENABLE	(0) CSS_LOOP_CACHE_ENABLE	0X0	CSS loop cache enable
0x40000010	SYSCTRL_DSS_LOOP_CACHE_CFG	(0) DSS_LOOP_CACHE_ENABLE	(0) DSS_LOOP_CACHE_ENABLE	0X1	DSS loop cache enable
0x40000014	SYSCTRL_MEM_ERROR	(5) MEM_ERROR_CLEAR	-	N/A	Write a 1 to clear the memory error flags
		-	(4) BB_MEM_ERROR	0X0	Baseband memory error flag
		-	(3) FLASH_COPIER_MEM_ERROR	0X0	Flash copier memory error flag
		-	(2) DMA_MEM_ERROR	0X0	DMA memory error flag
		-	(1) LPDSP32_DMEM_ERROR	0X0	LPDSP32 data memory error flag
		-	(0) LPDSP32_PMEM_ERROR	0X0	LPDSP32 program memory error flag

Address	Register Name	Register Write	Register Read	Default	Description
0x40000018	SYSCTRL_MEM_POWER_CFG	(21) DSP_DRAM5_POWER	(21) DSP_DRAM5_POWER	0X1	DSP PRAM0 power configuration
		(20) DSP_DRAM4_POWER	(20) DSP_DRAM4_POWER	0X1	DSP PRAM0 power configuration
		(19) DSP_DRAM3_POWER	(19) DSP_DRAM3_POWER	0X1	DSP PRAM0 power configuration
		(18) DSP_DRAM2_POWER	(18) DSP_DRAM2_POWER	0X1	DSP PRAM0 power configuration
		(17) DSP_DRAM1_POWER	(17) DSP_DRAM1_POWER	0X1	DSP PRAM0 power configuration
		(16) DSP_DRAM0_POWER	(16) DSP_DRAM0_POWER	0X1	DSP PRAM0 power configuration
		(15) DSP_PRAM3_POWER	(15) DSP_PRAM3_POWER	0X0	DSP PRAM0 power configuration
		(14) DSP_PRAM2_POWER	(14) DSP_PRAM2_POWER	0X0	DSP PRAM0 power configuration
		(13) DSP_PRAM1_POWER	(13) DSP_PRAM1_POWER	0X0	DSP PRAM0 power configuration
		(12) DSP_PRAM0_POWER	(12) DSP_PRAM0_POWER	0X0	DSP PRAM0 power configuration
		(11) BB_DRAM1_POWER	(11) BB_DRAM1_POWER	0X1	Baseband DRAM1 power configuration
		(10) BB_DRAM0_POWER	(10) BB_DRAM0_POWER	0X1	Baseband DRAM0 power configuration
		(8) DRAM2_POWER	(8) DRAM2_POWER	0X1	DRAM2 power configuration
		(7) DRAM1_POWER	(7) DRAM1_POWER	0X1	DRAM1 power configuration
		(6) DRAM0_POWER	(6) DRAM0_POWER	0X1	DRAM0 power configuration
		(5) PRAM3_POWER	(5) PRAM3_POWER	0X1	PRAM3 power configuration
		(4) PRAM2_POWER	(4) PRAM2_POWER	0X1	PRAM2 power configuration
		(3) PRAM1_POWER	(3) PRAM1_POWER	0X1	PRAM1 power configuration
		(2) PRAM0_POWER	(2) PRAM0_POWER	0X1	PRAM0 power configuration
		(1) FLASH_POWER	(1) FLASH_POWER	0X0	Flash power configuration
		(0) PROM_POWER	(0) PROM_POWER	0X1	PROM power configuration

Address	Register Name	Register Write	Register Read	Default	Description
0x4000001C	SYSCTRL_MEM_ACCESS_CFG	(30:24) WAKEUP_ADDR_PACKED	(30:24) WAKEUP_ADDR_PACKED	0X0	Wakeup restore address in packed 7-bit format. When written, SYSCTRL_WAKEUP_ADDR is updated. This field reads back as zero when SYSCTRL_WAKEUP_ADDR does not point to an enabled RAM instance.
		(21) DSP_DRAM5_ACCESS	(21) DSP_DRAM5_ACCESS	0X0	DSP PRAM0 access configuration
		(20) DSP_DRAM4_ACCESS	(20) DSP_DRAM4_ACCESS	0X0	DSP PRAM0 access configuration
		(19) DSP_DRAM3_ACCESS	(19) DSP_DRAM3_ACCESS	0X0	DSP PRAM0 access configuration
		(18) DSP_DRAM2_ACCESS	(18) DSP_DRAM2_ACCESS	0X0	DSP PRAM0 access configuration
		(17) DSP_DRAM1_ACCESS	(17) DSP_DRAM1_ACCESS	0X0	DSP PRAM0 access configuration
		(16) DSP_DRAM0_ACCESS	(16) DSP_DRAM0_ACCESS	0X0	DSP PRAM0 access configuration
		(15) DSP_PRAM3_ACCESS	(15) DSP_PRAM3_ACCESS	0X0	DSP PRAM0 access configuration
		(14) DSP_PRAM2_ACCESS	(14) DSP_PRAM2_ACCESS	0X0	DSP PRAM0 access configuration
		(13) DSP_PRAM1_ACCESS	(13) DSP_PRAM1_ACCESS	0X0	DSP PRAM0 access configuration
		(12) DSP_PRAM0_ACCESS	(12) DSP_PRAM0_ACCESS	0X0	DSP PRAM0 access configuration
		(11) BB_DRAM1_ACCESS	(11) BB_DRAM1_ACCESS	0X0	Baseband DRAM1 access configuration
		(10) BB_DRAM0_ACCESS	(10) BB_DRAM0_ACCESS	0X0	Baseband DRAM0 access configuration
		(8) DRAM2_ACCESS	(8) DRAM2_ACCESS	0X0	DRAM2 access configuration
		(7) DRAM1_ACCESS	(7) DRAM1_ACCESS	0X0	DRAM1 access configuration
		(6) DRAM0_ACCESS	(6) DRAM0_ACCESS	0X1	DRAM0 access configuration
		(5) PRAM3_ACCESS	(5) PRAM3_ACCESS	0X0	PRAM3 access configuration
		(4) PRAM2_ACCESS	(4) PRAM2_ACCESS	0X0	PRAM2 access configuration
		(3) PRAM1_ACCESS	(3) PRAM1_ACCESS	0X0	PRAM1 access configuration
		(2) PRAM0_ACCESS	(2) PRAM0_ACCESS	0X0	PRAM0 access configuration
		(1) FLASH_ACCESS	(1) FLASH_ACCESS	0X0	Flash access configuration
		(0) PROM_ACCESS	(0) PROM_ACCESS	0X1	PROM access configuration
0x40000020	SYSCTRL_WAKEUP_ADDR	(31:0) WAKEUP_ADDR	(31:0) WAKEUP_ADDR	0X0	Wakeup restore address in unpacked 32-bit format. When written, the WAKEUP_ADDR_PACKED field of SYSCTRL_MEM_ACCESS_CFG is updated. Bits 0-12 must be 0x0000 or 0x1FE8 (top or bottom of memory instance). Bits 17-20, 22-28 and 30-31 must be zero. When the WAKEUP_ADDR_PACKED field does not point to memory that is currently accessible, then SYSCTRL_WAKEUP_ADDR reads back as all zeros.

Address	Register Name	Register Write	Register Read	Default	Description
0x40000024	`SYSCTRL_MEM_RETENTION_CFG`	(21) `DSP_DRAM5_RETENTION`	(21) `DSP_DRAM5_RETENTION`	0X1	DSP PRAM0 retention configuration
		(20) `DSP_DRAM4_RETENTION`	(20) `DSP_DRAM4_RETENTION`	0X1	DSP PRAM0 retention configuration
		(19) `DSP_DRAM3_RETENTION`	(19) `DSP_DRAM3_RETENTION`	0X1	DSP PRAM0 retention configuration
		(18) `DSP_DRAM2_RETENTION`	(18) `DSP_DRAM2_RETENTION`	0X1	DSP PRAM0 retention configuration
		(17) `DSP_DRAM1_RETENTION`	(17) `DSP_DRAM1_RETENTION`	0X1	DSP PRAM0 retention configuration
		(16) `DSP_DRAM0_RETENTION`	(16) `DSP_DRAM0_RETENTION`	0X1	DSP PRAM0 retention configuration
		(15) `DSP_PRAM3_RETENTION`	(15) `DSP_PRAM3_RETENTION`	0X0	DSP PRAM0 retention configuration
		(14) `DSP_PRAM2_RETENTION`	(14) `DSP_PRAM2_RETENTION`	0X0	DSP PRAM0 retention configuration
		(13) `DSP_PRAM1_RETENTION`	(13) `DSP_PRAM1_RETENTION`	0X0	DSP PRAM0 retention configuration
		(12) `DSP_PRAM0_RETENTION`	(12) `DSP_PRAM0_RETENTION`	0X0	DSP PRAM0 retention configuration
		(11) `BB_DRAM1_RETENTION`	(11) `BB_DRAM1_RETENTION`	0X1	Baseband DRAM1 retention configuration
		(10) `BB_DRAM0_RETENTION`	(10) `BB_DRAM0_RETENTION`	0X1	Baseband DRAM0 retention configuration
		(8) `DRAM2_RETENTION`	(8) `DRAM2_RETENTION`	0X1	DRAM2 retention configuration
		(7) `DRAM1_RETENTION`	(7) `DRAM1_RETENTION`	0X1	DRAM1 retention configuration
		(6) `DRAM0_RETENTION`	(6) `DRAM0_RETENTION`	0X0	DRAM0 retention configuration
		(5) `PRAM3_RETENTION`	(5) `PRAM3_RETENTION`	0X1	PRAM3 retention configuration
		(4) `PRAM2_RETENTION`	(4) `PRAM2_RETENTION`	0X1	PRAM2 retention configuration
		(3) `PRAM1_RETENTION`	(3) `PRAM1_RETENTION`	0X1	PRAM1 retention configuration
		(2) `PRAM0_RETENTION`	(2) `PRAM0_RETENTION`	0X1	PRAM0 retention configuration
0x40000028	`SYSCTRL_MEM_ARBITER_CFG`	(29:28) `DSP_DRAM45_ARBITER`	(29:28) `DSP_DRAM45_ARBITER`	0X0	DSP DRAM4 and DRAM5 arbiter configuration
		(27:26) `DSP_DRAM23_ARBITER`	(27:26) `DSP_DRAM23_ARBITER`	0X0	DSP DRAM2 and DRAM3 arbiter configuration
		(25:24) `DSP_DRAM01_ARBITER`	(25:24) `DSP_DRAM01_ARBITER`	0X0	DSP DRAM0 and DRAM1 arbiter configuration
		(23:22) `DSP_PRAM3_ARBITER`	(23:22) `DSP_PRAM3_ARBITER`	0X0	DSP PRAM3 arbiter configuration
		(21:20) `DSP_PRAM2_ARBITER`	(21:20) `DSP_PRAM2_ARBITER`	0X0	DSP PRAM2 arbiter configuration
		(19:18) `DSP_PRAM1_ARBITER`	(19:18) `DSP_PRAM1_ARBITER`	0X0	DSP PRAM1 arbiter configuration
		(17:16) `DSP_PRAM0_ARBITER`	(17:16) `DSP_PRAM0_ARBITER`	0X0	DSP PRAM0 arbiter configuration
		(11:10) `BB_DRAM1_ARBITER`	(11:10) `BB_DRAM1_ARBITER`	0X3	Baseband DRAM1 arbiter configuration
		(9:8) `BB_DRAM0_ARBITER`	(9:8) `BB_DRAM0_ARBITER`	0X3	Baseband DRAM0 arbiter configuration
		(5:4) `DRAM12_ARBITER`	(5:4) `DRAM12_ARBITER`	0X0	DRAM1 and DRAM2 arbiter configuration
		(2) `DRAM0_ARBITER`	(2) `DRAM0_ARBITER`	0X0	DRAM0 arbiter configuration
		(1) `PRAM_ARBITER`	(1) `PRAM_ARBITER`	0X0	PRAM0 to PRAM3 arbiter configuration
		(0) `ROUND_ROBIN_TOKEN`	(0) `ROUND_ROBIN_TOKEN`	0X0	Round-robin token generation configuration

Address	Register Name	Register Write	Register Read	Default	Description
0x4000002C	SYSCTRL_MEM_TIMING_CFG	(9:8) DSP_PRAM_EMAW	(9:8) DSP_PRAM_EMAW	0X0	DSP_PRAM extra write margin configuration
		(6:4) DSP_PRAM_EMA	(6:4) DSP_PRAM_EMA	0X2	DSP_PRAM extra margin configuration
		(3) PROM_KEN	(3) PROM_KEN	0X1	PROM bitlines keeper configuration
		(2:0) PROM_EMA	(2:0) PROM_EMA	0X5	PROM extra margin configuration
0x40000030	SYSCTRL_CNT_CTRL	-	(3) CNT_STATUS	0X0	Activity counters status bit
		(2) CNT_CLEAR	-	N/A	Clear activity counters
		(1) CNT_STOP	-	N/A	Stop activity counters
		(0) CNT_START	-	N/A	Start activity counters
0x40000034	SYSCTRL_SYSCLK_CNT	(31:0) SYSCLK_CNT	(31:0) SYSCLK_CNT	0X0	System clock counter value
0x40000038	SYSCTRL_CM3_CNT	(31:0) CM3_CNT	(31:0) CM3_CNT	0X0	CM3 activity counter value
0x4000003C	SYSCTRL_LPDSP32_CNT	(31:0) LPDSP32_CNT	(31:0) LPDSP32_CNT	0X0	LPDSP32 activity counter value
0x40000040	SYSCTRL_FLASH_READ_CNT	(31:0) FLASH_READ_CNT	(31:0) FLASH_READ_CNT	0X0	Flash read access counter value
0x40000048	SYSCTRL_SPEED_MEASURE	(4) SPEED_MEASURE_START	-	N/A	Start critical path speed measurement
		-	(3) SPEED_MEASURE_STATUS	0X0	Critical path speed measurement status
		-	(2:0) SPEED_MEASURE_RESULT	0X0	Critical path speed measurement result
0x4000004C	SYSCTRL_LPDSP32_DEBUG_CFG	(1) LPDSP32_EXIT_POWERDOWN_WHEN_HALTED	(1) LPDSP32_EXIT_POWERDOWN_WHEN_HALTED	0X0	LPDSP32 exit powerdown mode configuration when halted
		(0) LPDSP32_DEBUG_ENABLE	(0) LPDSP32_DEBUG_ENABLE	0X0	LPDSP32 debug port enable
0x40000050	SYSCTRL_RF_POWER_CFG	(0) RF_POWER	(0) RF_POWER	0X0	RF power configuration
0x40000054	SYSCTRL_RF_ACCESS_CFG	(1) RF_IRQ_ACCESS	(1) RF_IRQ_ACCESS	0X0	RF IRQ access configuration
		(0) RF_ACCESS	(0) RF_ACCESS	0X0	RF access configuration
0x40000058	SYSCTRL_WAKEUP_PAD	-	(0) WAKEUP_PAD_VALUE	0X0	WAKEUP pad value
0x400000DC	SYSCTRL_DBG_LOCK	(31:0) DBG_LOCK_WR	-	N/A	Debug port access lock/unlock
		-	(0) DBG_LOCK_RD	0X1	Debug port access state
0x400000E0	SYSCTRL_DBG_LOCK_KEY	(31:0) DBG_LOCK_KEY	(31:0) DBG_LOCK_KEY	0X0	Debug port key

A.3 Clock Generation

Address	Register Name	Register Write	Register Read	Default	Description
0x40000100	CLK_SYS_CFG	(19:16) JTCK_PRESCALE	(19:16) JTCK_PRESCALE	0X0	Prescale value for the input clock from pad JTCK (1 to 16 in steps of 1)
		(11:8) EXTCLK_PRESCALE	(11:8) EXTCLK_PRESCALE	0X0	Prescale value for the input clock from pad EXTCLK (1 to 16 in steps of 1)
		(2:0) SYSCLK_SRC_SEL	(2:0) SYSCLK_SRC_SEL	0X0	Controls the source of the system clock : JTCK, RFCLK, RCCLK, EXTCLK or STANDBYCLK
0x40000104	CLK_DIV_CFG0	(27:16) USRCLK_PRESCALE	(27:16) USRCLK_PRESCALE	0X0	Prescale value for the USR clock (1 to 4096 in steps of 1)
		(10:8) BBCLK_PRESCALE	(10:8) BBCLK_PRESCALE	0X0	Prescale value for the Baseband peripheral clock (1 to 8 in steps of 1)
		(5:0) SLOWCLK_PRESCALE	(5:0) SLOWCLK_PRESCALE	0X2	Prescale value for the SLOWCLK clock (1 to 64 in steps of 1)
0x40000108	CLK_DIV_CFG1	(31:30) AUDIOSLOWCLK_PRESCALE	(31:30) AUDIOSLOWCLK_PRESCALE	0X0	Prescale value for the slow audio clock down from the fast audio clock (1 to 4 in steps of 1)
		(29:24) AUDIOCLK_PRESCALE	(29:24) AUDIOCLK_PRESCALE	0X0	Prescale value for the fast audio clock (1 to 64 in steps of 1)
		(20:16) UARTCLK_PRESCALE	(20:16) UARTCLK_PRESCALE	0X0	Prescale value for the UART peripheral clock (1 to 32 in steps of 1)
		(13:8) PWM1CLK_PRESCALE	(13:8) PWM1CLK_PRESCALE	0X0	Prescale value for the PWM1 peripheral clock (1 to 64 in steps of 1)
		(5:0) PWM0CLK_PRESCALE	(5:0) PWM0CLK_PRESCALE	0X0	Prescale value for the PWM0 peripheral clock (1 to 64 in steps of 1)
0x4000010C	CLK_DIV_CFG2	(15) CPCLK_DISABLE	(15) CPCLK_DISABLE	0X0	Charge pump clock disable
		(13:8) CPCLK_PRESCALE	(13:8) CPCLK_PRESCALE	0X7	Prescale value for the charge pump clock from the SLOWCLK clock (1 to 64 in steps of 1)
		(7) DCCLK_DISABLE	(7) DCCLK_DISABLE	0X0	DC-DC converter clock disable
		(5:0) DCCLK_PRESCALE	(5:0) DCCLK_PRESCALE	0X0	Prescale value for the DC-DC converter clock (1 to 64 in steps of 1)
0x40000110	CLK_DET_CFG	(5) CLK_DET_SEL	(5) CLK_DET_SEL	0X0	Clock detector source selection
		(4:3) CLK_DET_INT_SEL	(4:3) CLK_DET_INT_SEL	0X0	Clock detector interrupt configuration
		(2:1) CLK_DET_DIV	(2:1) CLK_DET_DIV	0X0	Clock detector configuration - Not used when running on standby clock
		(0) CLK_DET_ENABLE	(0) CLK_DET_ENABLE	0X0	Clock detector enable/disable
0x40000114	CLK_DET_STATUS	-	(1) CLK_DET_INT_STATUS	0X0	Clock detector interrupt status (cleared when read)
		-	(0) CLK_DET_STATUS	0X0	Clock detector status

A.4 RESET

Address	Register Name	Register Write	Register Read	Default	Description
0x40000200	DIG_RESET_STATUS	(7) LOCKUP_RESET_FLAG_CLEAR	-	N/A	Reset the sticky LOCKUP flag
		(6) WATCHDOG_RESET_FLAG_CLEAR	-	N/A	Reset the sticky Watchdog time-out reset flag
		(5) CM3_SW_RESET_FLAG_CLEAR	-	N/A	Reset the sticky CM3 software reset flag
		(4) ACS_RESET_FLAG_CLEAR	-	N/A	Reset the sticky ACS reset flag
		-	(3) LOCKUP_FLAG	0X0	Sticky flag that detects that a LOCKUP occurred
		-	(2) WATCHDOG_RESET_FLAG	0X0	Sticky flag that detects that a Watchdog time-out reset occurred
		-	(1) CM3_SW_RESET_FLAG	0X0	Sticky flag that detects that a CM3 software reset occurred
		-	(0) ACS_RESET_FLAG	0X1	Sticky flag that detects that a ACS reset occurred

A.5 WATCHDOG TIMER

Address	Register Name	Register Write	Register Read	Default	Description
0x40000300	`WATCHDOG_CFG`	(3:0) `TIMEOUT_VALUE`	(3:0) `TIMEOUT_VALUE`	0XB	Watchdog timeout period. Values 0xC to 0xF result in the same timeout period as the value 0xB.
0x40000304	`WATCHDOG_CTRL`	(31:0) `WATCHDOG_REFRESH`	-	N/A	Write a key to reset the watchdog

A.6 General-Purpose Timers 0, 1, 2 and 3

Address	Register Name	Register Write	Register Read	Default	Description
0x40000400	TIMER_CFG	(31:29) MULTI_COUNT	(31:29) MULTI_COUNT	0X0	Multi-count value
		(28) MODE	(28) MODE	0X0	Timer mode
		(27) CLK_SRC	(27) CLK_SRC	0X0	Clock source
		(26:24) PRESCALE	(26:24) PRESCALE	0X0	Prescale value of the timer
		(23:0) TIMEOUT_VALUE	(23:0) TIMEOUT_VALUE	0X0	Number of Timer clock cycles to time-out
0x40000410	TIMER_CTRL	-	(2) TIMER_STATUS	0X0	Indicate if the timer is active or not
		(1) TIMER_START	-	N/A	Start or restart the timer
		(0) TIMER_STOP	-	N/A	Stop the timer
0x40000420	TIMER_VAL	-	(23:0) TIMER_VALUE	0X0	Current timer 0 value

A.7 Flash Interface Configuration and Control

Address	Register Name	Register Write	Register Read	Default	Description
0x40000500	FLASH_IF_CTRL	(18) PREFETCH_DBUS	(18) PREFETCH_DBUS	0X0	Pre-fetch on D-Bus control
		(17) PREFETCH_IBUS	(17) PREFETCH_IBUS	0X0	Pre-fetch on I-Bus control
		(16) NOT_LOAD_AUTO	(16) NOT_LOAD_AUTO	0X0	Do not automatically load the configuration registers and the patch information from NVR4 sector after the command WAKEUP is completed.
		(12) VREAD1_MODE	(12) VREAD1_MODE	0X0	Control VREAD1: Read data after erase with more stringent condition than normal read. Changing this bit will execute the CMD_SET_VREAD1 or CMD_UNSET_VREAD1 command.
		(11) VREAD0_MODE	(11) VREAD0_MODE	0X0	Control VREAD0: Read data after program with more stringent condition than normal read. Changing this bit will execute the CMD_SET_VREAD0 or CMD_UNSET_VREAD0 command.
		(10) RECALL	(10) RECALL	0X0	Set the recall pins mode during CMD_READ. Changing this bit will execute the CMD_SET_RECALL or CMD_UNSET_RECALL command.
		(9:8) RETRY	(9:8) RETRY	0X0	Configures the erase retry iteration. This impacts the eFlash endurance time. Also used by Flash programming.
		(0) LP_MODE	(0) LP_MODE	0X0	Set the low power mode. Changing this bit will execute the CMD_SET_LOW_POWER or CMD_UNSET_LOW_POWER command.
0x40000504	FLASH_MAIN_WRITE_UNLOCK	(31:0) UNLOCK_KEY	-	N/A	32-bit key to allow for write accesses into the Flash MAIN Block
0x40000508	FLASH_MAIN_CTRL	(2) MAIN_HIGH_W_EN	(2) MAIN_HIGH_W_EN	0X0	Authorize the write access to the high part of the Flash MAIN block through the FLASH_IF registers.
		(1) MAIN_MIDDLE_W_EN	(1) MAIN_MIDDLE_W_EN	0X0	Authorize the write access to the middle part of the Flash MAIN block through the FLASH_IF registers.
		(0) MAIN_LOW_W_EN	(0) MAIN_LOW_W_EN	0X0	Authorize the write access to the lower part of the Flash MAIN block through the FLASH_IF registers.
0x40000510	FLASH_DELAY_CTRL	(7) READ_MARGIN	(7) READ_MARGIN	0X0	Flash Read access time margin
		(3:0) SYSCLK_FREQ	(3:0) SYSCLK_FREQ	0X2	Configure Flash, memory and RF power-up delays
0x40000534	FLASH_CMD_CTRL	(6:5) CMD_END	-	N/A	Terminates an active Flash command if possible (e.g. sequential programming sequence)
		(4:0) COMMAND	(4:0) COMMAND	0X0	Flash access command only writable when equal to CMD_IDLE

Address	Register Name	Register Write	Register Read	Default	Description
0x40000538	FLASH_IF_STATUS	-	(13) TRIMMED_STATUS	0X0	Flash trimming status
		-	(12) ISOLATE_STATUS	0X1	Flash isolate status
		-	(11) PROG_SEQ_DATA_REQ	0X0	Request new data while in sequential program mode
		-	(10) BUSY	0X0	Flash interface busy status bit
		-	(9) RED2_W_UNLOCK	0X0	Flash RED2 write unlock status bit
		-	(8) RED1_W_UNLOCK	0X0	Flash RED1 write unlock status bit
		-	(6) NVR3_W_UNLOCK	0X0	Flash NVR3 write unlock status bit
		-	(5) NVR2_W_UNLOCK	0X0	Flash NVR2 write unlock status bit
		-	(4) NVR1_W_UNLOCK	0X0	Flash NVR1 write unlock status bit
		-	(2) MAIN_HIGH_W_UNLOCK	0X0	Write unlock status bit of the high part of the Flash MAIN block
		-	(1) MAIN_MIDDLE_W_UNLOCK	0X0	Write unlock status bit of the middle part of the Flash MAIN block
		-	(0) MAIN_LOW_W_UNLOCK	0X0	Write unlock status bit of the lower part of the Flash MAIN block
0x4000053C	FLASH_ADDR	(20:2) FLASH_ADDR	(20:2) FLASH_ADDR	0X0	Flash Byte Address
0x40000540	FLASH_DATA	(31:0) DATA	(31:0) DATA	0X0	32-bit Flash Data
0x40000548	FLASH_NVR_WRITE_UNLOCK	(31:0) UNLOCK_KEY	-	N/A	32-bit key to allow for write accesses NVR sectors of the Flash
0x4000054C	FLASH_NVR_CTRL	(3) NVR3_W_EN	(3) NVR3_W_EN	0X0	Authorize Write access to the Flash NVR3 sector through the FLASH_IF registers.
		(2) NVR2_W_EN	(2) NVR2_W_EN	0X0	Authorize Write access to the Flash NVR2 sector through the FLASH_IF registers.
		(1) NVR1_W_EN	(1) NVR1_W_EN	0X0	Authorize Write access to the Flash NVR1 sector through the FLASH_IF registers.
0x40000568	FLASH_PATCH_ADDR	(20:11) PATCH_ADDR	(20:11) PATCH_ADDR	0XFF800	
0x40000580	FLASH_COPY_CFG	(18) COMP_ADDR_STEP	(18) COMP_ADDR_STEP	0X0	Comparator address increment/decrement by 1 or 2
		(17) COMP_ADDR_DIR	(17) COMP_ADDR_DIR	0X1	Comparator address-up or address-down
		(16) COMP_MODE	(16) COMP_MODE	0X0	Comparator Mode
		(9) COPY_DEST	(9) COPY_DEST	0X0	Destination copier is the CRC or memories
		(8) COPY_MODE	(8) COPY_MODE	0X0	Select copier mode (32-bit or 40-bit)
		(0) MODE	(0) MODE	0X0	Copier or Comparator Mode Configuration
0x400005C8	FLASH_COPY_CTRL	-	(3) ERROR	0X0	Error status
		(2) STOP	-	N/A	Stop the transfer
		(1) START	-	N/A	Start the transfer
		-	(0) BUSY	0X0	Busy status
0x400005D0	FLASH_COPY_SRC_ADDR_PTR	(20:0) COPY_SRC_ADDR_PTR	(20:0) COPY_SRC_ADDR_PTR	0X0	Source address pointer

Address	Register Name	Register Write	Register Read	Default	Description
0x400005D4	FLASH_COPY_DST_ADDR_PTR	(31:2) COPY_DST_ADDR_PTR	(31:2) COPY_DST_ADDR_PTR	0X0	Destination address pointer
0x400005D8	FLASH_COPY_WORD_CNT	(16:0) COPY_WORD_CNT	(16:0) COPY_WORD_CNT	0X0	Number of words to copy / compare
0x400005DC	FLASH_ECC_CTRL	(15:8) ECC_COR_CNT_INT_THRESHOLD	(15:8) ECC_COR_CNT_INT_THRESHOLD	0X1	Select the number of corrected errors before sending a CM3 interrupt
		(3) COPIER_ECC_CTRL	(3) COPIER_ECC_CTRL	0X1	
		(2) CMD_ECC_CTRL	(2) CMD_ECC_CTRL	0X1	
		(0) IDBUS_ECC_CTRL	(0) IDBUS_ECC_CTRL	0X1	Select the operating mode of the Flash ECC
0x400005E0	FLASH_ECC_STATUS	(6) ECC_COR_ERROR_CNT_CLEAR	-	N/A	Reset the Flash corrected errors counter
		(5) ECC_UNCOR_ERROR_CNT_CLEAR	-	N/A	Reset the Flash uncorrected errors counter
		(4) ECC_ERROR_ADDR_CLEAR	-	N/A	Reset the Flash address of the last detected error
		-	(1) ECC_COR_ERROR_CNT_STATUS	0X0	FLASH_ECC_ERROR_COR_CNT status
		-	(0) ECC_UNCOR_ERROR_CNT_STATUS	0X0	FLASH_ECC_ERROR_UNCOR_CNT status
0x400005E4	FLASH_ECC_ERROR_ADDR	-	(20:2) ECC_ERROR_ADDR	0X0	Store the Flash address of the latest Flash ECC error
0x400005E8	FLASH_ECC_UNCOR_ERROR_CNT	-	(7:0) ECC_UNCOR_ERROR_CNT	0X0	Flash ECC uncorrected error counter
0x400005EC	FLASH_ECC_COR_ERROR_CNT	-	(7:0) ECC_COR_ERROR_CNT	0X0	Flash ECC corrected error counter

A.8 DMA Controller Configuration and Control

Address	Register Name	Register Write	Register Read	Default	Description
0x40000600	DMA_CTRL0	(31:30) DEST_ADDR_STEP_SIZE	(31:30) DEST_ADDR_STEP_SIZE	0X0	Select the destination address step size
		(29:28) SRC_ADDR_STEP_SIZE	(29:28) SRC_ADDR_STEP_SIZE	0X0	Select the source address step size
		(27) DEST_ADDR_STEP_MODE	(27) DEST_ADDR_STEP_MODE	0X0	Configure the destination address to either increment or decrement
		(26) SRC_ADDR_STEP_MODE	(26) SRC_ADDR_STEP_MODE	0X0	Configure the source address to either increment or decrement
		(25) BYTE_ORDER	(25) BYTE_ORDER	0X0	Select the byte ordering for the DMA channel
		(24) DISABLE_INT_ENABLE	(24) DISABLE_INT_ENABLE	0X0	Raise an interrupt when the DMA channel is disabled
		(23) ERROR_INT_ENABLE	(23) ERROR_INT_ENABLE	0X0	Raise an interrupt when a state machine error occurs during a DMA transfer
		(22) COMPLETE_INT_ENABLE	(22) COMPLETE_INT_ENABLE	0X0	Raise an interrupt when the DMA transfer completes
		(21) COUNTER_INT_ENABLE	(21) COUNTER_INT_ENABLE	0X0	Raise an interrupt when the DMA transfer reaches the counter value
		(20) START_INT_ENABLE	(20) START_INT_ENABLE	0X0	Raise an interrupt when the DMA transfer starts
		(19:18) DEST_WORD_SIZE	(19:18) DEST_WORD_SIZE	0X0	Select the destination word size
		(17:16) SRC_WORD_SIZE	(17:16) SRC_WORD_SIZE	0X0	Select the source word size
		(14:12) DEST_SELECT	(14:12) DEST_SELECT	0X0	Select the request line for the destination
		(11:9) SRC_SELECT	(11:9) SRC_SELECT	0X0	Select the request line for the source
		(7:6) CHANNEL_PRIORITY	(7:6) CHANNEL_PRIORITY	0X0	Select the priority level for this channel
		(5:4) TRANSFER_TYPE	(5:4) TRANSFER_TYPE	0X0	Select the type of transfer implemented by DMA channel
		(3) DEST_ADDR_INC	(3) DEST_ADDR_INC	0X0	Configure whether the destination address should increment
		(2) SRC_ADDR_INC	(2) SRC_ADDR_INC	0X0	Configure whether the source address should increment
		(1) ADDR_MODE	(1) ADDR_MODE	0X0	Select the addressing mode for this channel
		(0) ENABLE	(0) ENABLE	0X0	Enable DMA Channel
0x40000620	DMA_SRC_BASE_ADDR	(31:0) DMA_SRC_BASE_ADDR	(31:0) DMA_SRC_BASE_ADDR	0X0	Base address for the source of data transferred using DMA channel
0x40000640	DMA_DEST_BASE_ADDR	(31:0) DMA_DEST_BASE_ADDR	(31:0) DMA_DEST_BASE_ADDR	0X0	Base address for the destination of data transferred using DMA channel
0x40000660	DMA_CTRL1	(31:16) COUNTER_INT_VALUE	(31:16) COUNTER_INT_VALUE	0X0	Trigger a counter interrupt when the DMA transfer word count reaches this value
		(15:0) TRANSFER_LENGTH	(15:0) TRANSFER_LENGTH	0X0	The length, in words, of each data transfer using DMA channel
0x40000680	DMA_NEXT_SRC_ADDR	-	(31:0) DMA_NEXT_SRC_ADDR	0X0	Address of the next data to be transferred using DMA channel

Address	Register Name	Register Write	Register Read	Default	Description
0x400006A0	DMA_NEXT_DEST_ADDR	-	(31:0) DMA_NEXT_DEST_ADDR	0X0	Address where the next data to be transferred using DMA channel will be stored
0x400006C0	DMA_WORD_CNT	-	(15:0) DMA_WORD_CNT	0X0	The number of words that have been transferred using DMA channel during the current transfer
0x400006E0	DMA_STATUS	(12) ERROR_INT_CLEAR	-	N/A	Clear the state machine error interrupt flag
		(11) COMPLETE_INT_CLEAR	-	N/A	Clear the complete interrupt flag
		(10) COUNTER_INT_CLEAR	-	N/A	Clear the counter interrupt flag
		(9) START_INT_CLEAR	-	N/A	Clear the start interrupt flag
		(8) DISABLE_INT_CLEAR	-	N/A	Clear the channel disable flag
		-	(7:5) STATE	0X0	DMA channel state
		-	(4) ERROR_INT_STATUS	0X0	Indicate if a state machine error interrupt has occurred on DMA channel
		-	(3) COMPLETE_INT_STATUS	0X0	Indicate if a complete interrupt has occurred on DMA channel
		-	(2) COUNTER_INT_STATUS	0X0	Indicate if a counter interrupt has occurred on DMA channel
		-	(1) START_INT_STATUS	0X0	Indicate if a start interrupt has occurred on DMA channel
		-	(0) DISABLE_INT_STATUS	0X0	Indicate if a channel disable interrupt has occurred on DMA channel

A.9 DIO Interface and Digital Pad control

Address	Register Name	Register Write	Register Read	Default	Description
0x40000700	`DIO_CFG`	(13:12) `DRIVE`	(13:12) `DRIVE`	0X3	Drive strength configuration
		(10) `LPF`	(10) `LPF`	0X0	Low Pass Filter enable
		(9:8) `PULL_CTRL`	(9:8) `PULL_CTRL`	0X1	Pull selection
		(5:0) `IO_MODE`	(5:0) `IO_MODE`	0X3F	IO mode selection
0x40000740	`DIO_DATA`	-	(15:0) `DIO`	0X0	DIO[15:0] read data
		(15:0) `GPIO`	-	N/A	GPIO[15:0] write data (updates output data of DIOs only for pads with IO_MODE 0b000XX)
0x40000744	`DIO_DIR`	-	(15:0) `DIO`	0X8000	Get DIO[15:0] direction
		(15:0) `GPIO`	-	N/A	Set DIO[15:0] GPIO direction (only in IO_MODE is 000XX)
0x40000748	`DIO_MODE`	-	(15:0) `GPIO`	0X0	DIO[15:0] mode
0x4000074C	`DIO_INT_CFG`	(11) `DEBOUNCE_ENABLE`	(11) `DEBOUNCE_ENABLE`	0X0	Interrupt button debounce filter enable/disable
		(10:8) `EVENT`	(10:8) `EVENT`	0X0	Interrupt event configuration
		(4:0) `SRC`	(4:0) `SRC`	0X0	Interrupt input selection
0x4000075C	`DIO_INT_DEBOUNCE`	(8) `DEBOUNCE_CLK`	(8) `DEBOUNCE_CLK`	0X0	Interrupt button debounce filter clock
		(7:0) `DEBOUNCE_COUNT`	(7:0) `DEBOUNCE_COUNT`	0X0	Interrupt button debounce filter count
0x40000760	`DIO_PCM_SRC`	(20:16) `SERI`	(20:16) `SERI`	0X11	PCM_SERI input selection
		(12:8) `FRAME`	(12:8) `FRAME`	0X11	PCM_FRAME input selection
		(4:0) `CLK`	(4:0) `CLK`	0X11	PCM_CLK input selection
0x40000764	`DIO_SPI_SRC`	(20:16) `SERI`	(20:16) `SERI`	0X11	SPI_SERI input selection
		(12:8) `CS`	(12:8) `CS`	0X11	SPI_CS input selection
		(4:0) `CLK`	(4:0) `CLK`	0X11	SPI_CLK input selection
0x4000076C	`DIO_UART_SRC`	(4:0) `RX`	(4:0) `RX`	0X11	UART_RX input selection
0x40000770	`DIO_I2C_SRC`	(12:8) `SDA`	(12:8) `SDA`	0X11	SDA input selection
		(4:0) `SCL`	(4:0) `SCL`	0X11	SCL input selection
0x40000774	`DIO_AUDIOSINK_SRC`	(4:0) `CLK`	(4:0) `CLK`	0X11	Audio sink clock input selection
0x40000778	`DIO_NMI_SRC`	(5) `NMI_POLARITY`	(5) `NMI_POLARITY`	0X1	NMI polarity
		(4:0) `NMI`	(4:0) `NMI`	0X10	NMI input selection
0x4000077C	`DIO_BB_RX_SRC`	(20:16) `RF_SYNC_P`	(20:16) `RF_SYNC_P`	0X12	Baseband controller interface RF_SYNC_P input selection
		(12:8) `CLK`	(12:8) `CLK`	0X12	Baseband controller RX clock input selection
		(4:0) `DATA`	(4:0) `DATA`	0X12	Baseband controller RX data input selection
0x40000780	`DIO_BB_SPI_SRC`	(4:0) `MISO`	(4:0) `MISO`	0X12	Baseband controller SPI_MISO input selection

Address	Register Name	Register Write	Register Read	Default	Description
0x40000784	DIO_RF_SPI_SRC	(20:16) MOSI	(20:16) MOSI	0X12	RF front-end SPI_MOSI input selection
		(12:8) CSN	(12:8) CSN	0X12	RF front-end SPI_CSN input selection
		(4:0) CLK	(4:0) CLK	0X12	RF front-end SPI_CLK input selection
0x40000788	DIO_RF_GPIO03_SRC	(28:24) GPIO3	(28:24) GPIO3	0X12	RF front-end GPIO3 input selection
		(20:16) GPIO2	(20:16) GPIO2	0X12	RF front-end GPIO2 input selection
		(12:8) GPIO1	(12:8) GPIO1	0X10	RF front-end GPIO1 input selection
		(4:0) GPIO0	(4:0) GPIO0	0X10	RF front-end GPIO0 input selection
0x4000078C	DIO_RF_GPIO47_SRC	(28:24) GPIO7	(28:24) GPIO7	0X10	RF front-end GPIO7 input selection
		(20:16) GPIO6	(20:16) GPIO6	0X10	RF front-end GPIO6 input selection
		(12:8) GPIO5	(12:8) GPIO5	0X10	RF front-end GPIO5 input selection
		(4:0) GPIO4	(4:0) GPIO4	0X12	RE front-end GPIO4 input selection
0x40000790	DIO_RF_GPIO89_SRC	(12:8) GPIO9	(12:8) GPIO9	0X10	RF front-end GPIO9 input selection
		(4:0) GPIO8	(4:0) GPIO8	0X10	RF front-end GPIO8 input selection
0x40000794	DIO_DMIC_SRC	(12:8) CLK	(12:8) CLK	0X11	DMIC clock input selection
		(4:0) DATA	(4:0) DATA	0X11	DMIC data input selection
0x40000798	DIO_LPDSP32_JTAG_SRC	(20:16) TDI	(20:16) TDI	0X11	LPDSP32_TDI input selection
		(12:8) TMS	(12:8) TMS	0X11	LPDSP32_TMS input selection
		(4:0) TCK	(4:0) TCK	0X11	LPDSP32_TCK input selection
0x4000079C	DIO_JTAG_SW_PAD_CFG	(9) JTCK_LPF	(9) JTCK_LPF	0X0	JTCK Low-Pass-Filter enable / disable
		(8) JTMS_LPF	(8) JTMS_LPF	0X0	JTMS Low-Pass-Filter enable / disable
		(7) CM3_JTAG_DATA_EN	(7) CM3_JTAG_DATA_EN	0X1	CM3 JTAG on DIO[14:15]
		(6) CM3_JTAG_TRST_EN	(6) CM3_JTAG_TRST_EN	0X1	CM3 JTAG TRST on DIO13
		(5:4) JTCK_PULL	(5:4) JTCK_PULL	0X1	JTCK pull-up enable / disable
		(3:2) JTMS_DRIVE	(3:2) JTMS_DRIVE	0X3	JTMS drive strength
		(1:0) JTMS_PULL	(1:0) JTMS_PULL	0X1	JTMS pull-up enable / disable
0x400007A0	DIO_EXTCLK_CFG	(2) LPF	(2) LPF	0X0	Low Pass Filter enable
		(1:0) PULL_CTRL	(1:0) PULL_CTRL	0X1	Pull Selection
0x400007A4	DIO_PAD_CFG	(0) DRIVE	(0) DRIVE	0X1	Drive strength configuration (scales the individual drive strengths)

A.10 SPI Interface Configuration and Control

Address	Register Name	Register Write	Register Read	Default	Description
0x40000800	`SPI0_CTRL0`	(10) `SPI0_OVERRUN_INT_ENABLE`	(10) `SPI0_OVERRUN_INT_ENABLE`	0X0	Enable/disable SPI overrun interrupts
		(9) `SPI0_UNDERRUN_INT_ENABLE`	(9) `SPI0_UNDERRUN_INT_ENABLE`	0X0	Enable/disable SPI underrun interrupts
		(8) `SPI0_CONTROLLER`	(8) `SPI0_CONTROLLER`	0X0	Select whether data transfer will be controlled by the CM3 or the DMA for SPI
		(7) `SPI0_SLAVE`	(7) `SPI0_SLAVE`	0X0	Use the SPI interface as master or slave
		(6) `SPI0_CLK_POLARITY`	(6) `SPI0_CLK_POLARITY`	0X0	Select the polarity of the SPI clock
		(5) `SPI0_MODE_SELECT`	(5) `SPI0_MODE_SELECT`	0X0	Select between manual and auto transaction handling modes for SPI master transactions
		(4) `SPI0_ENABLE`	(4) `SPI0_ENABLE`	0X0	Enable/disable the SPI interface
		(3:0) `SPI0_PRESCALE`	(3:0) `SPI0_PRESCALE`	0X0	Prescale the SPI interface clock for master transfers
0x40000804	`SPI0_CTRL1`	(8) `SPI0_START_BUSY`	-	N/A	Start an SPI data transfer and indicate if a transfer is in progress
		-	(8) `SPI0_BUSY_STATUS`	0X0	SPI data transfer status read
		(7:6) `SPI0_RW_CMD`	(7:6) `SPI0_RW_CMD`	0X0	Issue a read command or write command to the SPI interface
		(5) `SPI0_CS`	(5) `SPI0_CS`	0X1	Set the chip-select line for SPI (master mode), read the chip-select line for SPI (slave mode)
		(4:0) `SPI0_WORD_SIZE`	(4:0) `SPI0_WORD_SIZE`	0X0	Select the word size used by the SPI interface (word size = SPI0_WORD_SIZE + 1)
0x40000808	`SPI0_TX_DATA`	(31:0) `SPI0_TX_DATA`	(31:0) `SPI0_TX_DATA`	0X0	Single word buffer for data to be transmitted over the SPI interface
0x4000080C	`SPI0_RX_DATA`	-	(31:0) `SPI0_RX_DATA`	0X0	Single word buffer for data that has been received over the SPI interface
0x40000810	`SPI0_STATUS`	(3) `SPI0_TRANSMIT_STATUS`	(3) `SPI0_TRANSMIT_STATUS`	0X0	Indicate that the transmission of the data is completed
		(2) `SPI0_RECEIVE_STATUS`	(2) `SPI0_RECEIVE_STATUS`	0X0	Indicate that new data has been received
		(1) `SPI0_OVERRUN_STATUS`	(1) `SPI0_OVERRUN_STATUS`	0X0	Indicate that an overrun has occurred when receiving data on the SPI interface
		(0) `SPI0_UNDERRUN_STATUS`	(0) `SPI0_UNDERRUN_STATUS`	0X0	Indicate that an underrun has occurred when transmitting data on the SPI interface

A.11 SPI Interface Configuration and Control

Address	Register Name	Register Write	Register Read	Default	Description
0x40000900	`SPI1_CTRL0`	(10) `SPI1_OVERRUN_INT_ENABLE`	(10) `SPI1_OVERRUN_INT_ENABLE`	0X0	Enable/disable SPI overrun interrupts
		(9) `SPI1_UNDERRUN_INT_ENABLE`	(9) `SPI1_UNDERRUN_INT_ENABLE`	0X0	Enable/disable SPI underrun interrupts
		(8) `SPI1_CONTROLLER`	(8) `SPI1_CONTROLLER`	0X0	Select whether data transfer will be controlled by the CM3 or the DMA for SPI
		(7) `SPI1_SLAVE`	(7) `SPI1_SLAVE`	0X0	Use the SPI interface as master or slave
		(6) `SPI1_CLK_POLARITY`	(6) `SPI1_CLK_POLARITY`	0X0	Select the polarity of the SPI clock
		(5) `SPI1_MODE_SELECT`	(5) `SPI1_MODE_SELECT`	0X0	Select between manual and auto transaction handling modes for SPI master transactions
		(4) `SPI1_ENABLE`	(4) `SPI1_ENABLE`	0X0	Enable/disable the SPI interface
		(3:0) `SPI1_PRESCALE`	(3:0) `SPI1_PRESCALE`	0X0	Prescale the SPI interface clock for master transfers
0x40000904	`SPI1_CTRL1`	(8) `SPI1_START_BUSY`	(8) `SPI1_START_BUSY`	0X0	Start an SPI data transfer and indicate if a transfer is in progress
		(7:6) `SPI1_RW_CMD`	(7:6) `SPI1_RW_CMD`	0X0	Issue a read command or write command to the SPI interface
		(5) `SPI1_CS`	(5) `SPI1_CS`	0X1	Set the chip-select line for SPI (master mode), read the chip-select line for SPI (slave mode)
		(4:0) `SPI1_WORD_SIZE`	(4:0) `SPI1_WORD_SIZE`	0X0	Select the word size used by the SPI interface (word size = SPI1_WORD_SIZE + 1)
0x40000908	`SPI1_TX_DATA`	(31:0) `SPI1_TX_DATA`	(31:0) `SPI1_TX_DATA`	0X0	Single word buffer for data to be transmitted over the SPI interface
0x4000090C	`SPI1_RX_DATA`	-	(31:0) `SPI1_RX_DATA`	0X0	Single word buffer for data that has been received over the SPI interface
0x40000910	`SPI1_STATUS`	(3) `SPI1_TRANSMIT_STATUS`	(3) `SPI1_TRANSMIT_STATUS`	0X0	Indicate that the transmission of the data is completed
		(2) `SPI1_RECEIVE_STATUS`	(2) `SPI1_RECEIVE_STATUS`	0X0	Indicate that new data has been received
		(1) `SPI1_OVERRUN_STATUS`	(1) `SPI1_OVERRUN_STATUS`	0X0	Indicate that an overrun has occurred when receiving data on the SPI interface
		(0) `SPI1_UNDERRUN_STATUS`	(0) `SPI1_UNDERRUN_STATUS`	0X0	Indicate that an underrun has occurred when transmitting data on the SPI interface

A.12 PCM Interface Configuration and Control

Address	Register Name	Register Write	Register Read	Default	Description
0x40000A00	`PCM_CTRL`	(14) `PCM_CLK_POL`	(14) `PCM_CLK_POL`	0X0	PCM clock polarity
		(12) `BIT_ORDER`	(12) `BIT_ORDER`	0X0	Select whether the data will be transmitted starting with the MSB or LSB
		(11) `TX_ALIGN`	(11) `TX_ALIGN`	0X0	Select what bits to use for transmit data
		(10:9) `WORD_SIZE`	(10:9) `WORD_SIZE`	0X3	Select the number of bits per PCM word
		(8) `FRAME_ALIGN`	(8) `FRAME_ALIGN`	0X0	Align the PCM frame signal to the first/last bit
		(7) `FRAME_WIDTH`	(7) `FRAME_WIDTH`	0X0	Use a long/short PCM frame signal
		(6:4) `FRAME_LENGTH`	(6:4) `FRAME_LENGTH`	0X0	Select the number of words per PCM frame
		(3) `FRAME_SUBFRAMES`	(3) `FRAME_SUBFRAMES`	0X0	Enable the frame duration for each word
		(2) `CONTROLLER`	(2) `CONTROLLER`	0X0	Select whether data transfer will be controlled by the CM3 or the DMA for PCM
		(1) `ENABLE`	(1) `ENABLE`	0X0	Enable/disable the PCM interface
		(0) `SLAVE`	(0) `SLAVE`	0X1	Use the PCM interface as a master/slave
0x40000A04	`PCM_TX_DATA`	(31:0) `PCM_TX_DATA`	(31:0) `PCM_TX_DATA`	0X0	Data to transmit over the PCM interface
0x40000A08	`PCM_RX_DATA`	-	(31:0) `PCM_RX_DATA`	0X0	Data received from the PCM interface
0x40000A0C	`PCM_STATUS`	(3) `TRANSMIT_STATUS`	(3) `TRANSMIT_STATUS`	0X0	Indicate that PCM data has been sent
		(2) `RECEIVE_STATUS`	(2) `RECEIVE_STATUS`	0X0	Indicate that PCM data has been received
		(1) `OVERRUN_STATUS`	(1) `OVERRUN_STATUS`	0X0	Indicate that an overrun has occurred when receiving data on the PCM interface
		(0) `UNDERRUN_STATUS`	(0) `UNDERRUN_STATUS`	0X0	Indicate that an underrun has occurred when transmitting data on the PCM interface

A.13 I2C INTERFACE CONFIGURATION AND CONTROL

Address	Register Name	Register Write	Register Read	Default	Description
0x40000B00	I2C_CTRL0	(23:16) SPEED	(23:16) SPEED	0X0	Prescaler used to divide SYSCLK to the correct SCL frequency when operating in I2C interface master mode. SCL is prescaled by (SPEED + 1) * 3. In slave mode controls the number of SYSCLK wait cycles in case of clock streching between the moment the data is put on the SDA line and the SCL line is released.
		(14:8) SLAVE_ADDRESS	(14:8) SLAVE_ADDRESS	0X10	Set the I2C slave address for this device
		(4) CONTROLLER	(4) CONTROLLER	0X0	Select whether data transfer will be controlled by the CM3 or the DMA for I2C
		(3) STOP_INT_ENABLE	(3) STOP_INT_ENABLE	0X0	Configure whether stop interrupts will be generated by the I2C interface
		(2) AUTO_ACK_ENABLE	(2) AUTO_ACK_ENABLE	0X0	Select whether acknowledgement is automatically generated or not
		(1) I2C_SAMPLE_CLK_ENABLE	(1) I2C_SAMPLE_CLK_ENABLE	0X0	Enable/disable the I2C sample clock (mandatory to enable the I2C)
		(0) SLAVE_ENABLE	(0) SLAVE_ENABLE	0X0	Select whether the I2C interface will be enabled for slave mode or not
0x40000B04	I2C_CTRL1	(5) RESET	-	N/A	Reset the I2C interface
		(4) LAST_DATA	-	N/A	Indicate that the current data is the last byte of a data transfer
		(3) STOP	-	N/A	Issue a stop condition on the I2C interface bus
		(1) NACK	-	N/A	Issue a not acknowledge on the I2C interface bus
		(0) ACK	-	N/A	Issue an acknowledge on the I2C interface bus
0x40000B08	I2C_DATA	(7:0) I2C_DATA	(7:0) I2C_DATA	0X0	Single byte buffer for data transmitted and received over the I2C interface
0x40000B0C	I2C_DATA_M	(7:0) I2C_DATA_M	(7:0) I2C_DATA_M	0X0	Mirror of the single byte buffer for data transmitted and received over the I2C interface
0x40000B10	I2C_ADDR_START	(7:1) ADDRESS	-	N/A	I2C address to use for the transaction
		(0) READ_WRITE	-	N/A	Select whether a read or a write transaction is started

Address	Register Name	Register Write	Register Read	Default	Description
0x40000B14	I2C_STATUS	-	(15) ERROR_S	0X0	Same as BUS_ERROR_S: Bus error status bit (sticky) - This status bit is automatically reset when the I2C_STATUS register is read.
		-	(14) BUS_ERROR_S	0X0	Bus error status bit (sticky) - This status bit is automatically reset when the I2C_STATUS register is read.
		-	(13) START_PENDING	0X0	Master frame start pending status bit
		-	(12) MASTER_MODE	0X0	Master mode status bit
		-	(11) DMA_REQ	0X0	Indicate if the I2C interface is currently requesting DMA data
		-	(10) STOP_DETECT	0X0	Indicate if an I2C stop bit has been detected
		-	(9) DATA_EVENT	0X0	Indicate that I2C interface either needs data to transmit or has received data
		-	(8) ERROR	0X0	Same as BUS_ERROR: Indicate if the I2C interface has detected a bus error (automatically cleared when a stop condition is detected)
		-	(7) BUS_ERROR	0X0	Indicate if the I2C interface has detected a bus error (automatically cleared when a stop condition is detected)
		-	(6) BUFFER_FULL	0X0	Indicate if the I2C data buffer is full
		-	(5) CLK_STRETCH	0X0	Indicate if the I2C interface is holding the clock signal
		-	(4) BUS_FREE	0X1	Indicate if the I2C interface bus is free
		-	(3) ADDR_DATA	0X0	Indicate if the I2C data register holds an address or data byte
		-	(2) READ_WRITE	0X0	Indicate whether the I2C bus transfer is a read or a write
		-	(1) GEN_CALL	0X0	Indicate whether the I2C bus transfer is using the general call address or another address
		-	(0) ACK_STATUS	0X0	Indicate whether an acknowledge or a not acknowledge has been received

A.14 UART INTERFACE CONFIGURATION AND CONTROL

Address	Register Name	Register Write	Register Read	Default	Description
0x40000C00	UART_CFG	(23:8) PRESCALE	(23:8) PRESCALE	0X0	Prescaling multiplier in baud rate calculation
		(4) PRESCALE_ENABLE	(4) PRESCALE_ENABLE	0X0	Enable/disable a fixed prescaler by 12
		(1) DMA_ENABLE	(1) DMA_ENABLE	0X0	DMA mode enable
		(0) ENABLE	(0) ENABLE	0X0	Enable/disable the UART interface
0x40000C04	UART_TX_DATA	(7:0) UART_TX_DATA	(7:0) UART_TX_DATA	0X0	UART Transmit data
0x40000C08	UART_RX_DATA	-	(7:0) UART_RX_DATA	0X0	UART Received data
0x40000C0C	UART_STATUS	(0) UART_RX_OVERRUN_STATUS	(0) UART_RX_OVERRUN_STATUS	0X0	Indicate that an overrun has occurred when receiving data on the UART interface

A.15 PWM 0 AND 1 CONFIGURATION AND CONTROL

Address	Register Name	Register Write	Register Read	Default	Description
0x40000D00	PWM_CFG	(15:8) PWM_HIGH	(15:8) PWM_HIGH	0X0	PWM high duty cycle
		(7:0) PWM_PERIOD	(7:0) PWM_PERIOD	0X0	PWM period
0x40000D08	PWM_CTRL	(16) PWM_OFFSET_ENABLE	(16) PWM_OFFSET_ENABLE	0X0	Enable/disable the PWM offset function
		(15:8) PWM_OFFSET	(15:8) PWM_OFFSET	0X0	PWM0 to PWM1 offset
		(4) PWM1_ENABLE	(4) PWM1_ENABLE	0X0	Enable/disable the PWM1 block
		(0) PWM0_ENABLE	(0) PWM0_ENABLE	0X0	Enable/disable the PWM0 block

A.16 DMIC Input and Output Driver Configuration and Control

Address	Register Name	Register Write	Register Read	Default	Description
0x40000E00	AUDIO_CFG	(25) OD_CLK_SRC	(25) OD_CLK_SRC	0X0	Output driver clock selection
		(24) DMIC_CLK_SRC	(24) DMIC_CLK_SRC	0X0	DMIC clock selection (the same clock must be output to the DMIC_CLK DIO pad)
		(20:16) DEC_RATE	(20:16) DEC_RATE	0X0	DMIC input data decimation rate (also determines the OD interpolation rate in combination with DMIC_CLK_SRC and OD_CLK_SRC configuration bits)
		(12) OD_UNDERRUN_PROTECT	(12) OD_UNDERRUN_PROTECT	0X0	Enable OD_DATA underrun protection (automatically resets OD_DATA if it hasn't been updated during 16 sample periods)
		(11) OD_DMA_REQ_EN	(11) OD_DMA_REQ_EN	0X0	Enable the DMA request when a new output driver sample is required
		(10) OD_INT_GEN_EN	(10) OD_INT_GEN_EN	0X0	Enable the interrupt generation when a new output driver sample is required
		(9) OD_DATA_ALIGN	(9) OD_DATA_ALIGN	0X0	Data alignment in AUDIO_OD_DATA
		(8) OD_ENABLE	(8) OD_ENABLE	0X0	Enable output driver output
		(7) DMIC1_DMA_REQ_EN	(7) DMIC1_DMA_REQ_EN	0X0	Enable the DMA request when a new DMIC1 sample is ready
		(6) DMIC1_INT_GEN_EN	(6) DMIC1_INT_GEN_EN	0X0	Enable the interrupt generation when a new DMIC1 sample is ready
		(5) DMIC1_DATA_ALIGN	(5) DMIC1_DATA_ALIGN	0X0	Data alignment in AUDIO_DMIC_DATA_1
		(4) DMIC1_ENABLE	(4) DMIC1_ENABLE	0X0	Enable DMIC1 input
		(3) DMIC0_DMA_REQ_EN	(3) DMIC0_DMA_REQ_EN	0X0	Enable the DMA request when a new DMIC0 sample is ready
		(2) DMIC0_INT_GEN_EN	(2) DMIC0_INT_GEN_EN	0X0	Enable the interrupt generation when a new DMIC0 sample is ready
		(1) DMIC0_DATA_ALIGN	(1) DMIC0_DATA_ALIGN	0X0	Data alignment in AUDIO_DMIC_DATA_0
		(0) DMIC0_ENABLE	(0) DMIC0_ENABLE	0X0	Enable DMIC0 input

Address	Register Name	Register Write	Register Read	Default	Description
0x40000E04	AUDIO_STATUS	-	(11) OD_STATUS	0X1	Output driver feature status
		(10) OD_UNDERRUN_FLAG_CLEAR	-	N/A	Reset the output driver underrun detection sticky bit
		-	(9) OD_UNDERRUN_FLAG	0X0	Sticky bit indicating the detection of an output driver underrun
		-	(8) OD_DATA_REQ_FLAG	0X0	Flag indicating that a new output driver sample is required
		(6) DMIC1_OVERRUN_FLAG_CLEAR	-	N/A	Reset the DMIC1 overrun detection sticky bit
		-	(5) DMIC1_OVERRUN_FLAG	0X0	Sticky bit indicating the detection of a DMIC1 overrun
		-	(4) DMIC1_DATA_RDY_FLAG	0X0	Flag indicating the availability of a new DMIC1 sample
		(2) DMIC0_OVERRUN_FLAG_CLEAR	-	N/A	Reset the DMIC0 overrun detection sticky bit
		-	(1) DMIC0_OVERRUN_FLAG	0X0	Sticky bit indicating the detection of a DMIC0 overrun
		-	(0) DMIC0_DATA_RDY_FLAG	0X0	Flag indicating the availability of a new DMIC0 sample
0x40000E08	AUDIO_DMIC_CFG	(28:24) DMIC1_FRAC_DELAY	(28:24) DMIC1_FRAC_DELAY	0X0	DMIC1 fractional delay (each step represents a DMIC clock cycle)
		(19:16) DMIC1_DELAY	(19:16) DMIC1_DELAY	0X0	DMIC1 delay (0 to 1.875 samples in steps of 0.125 samples)
		(14:12) DMIC1_DCRM	(14:12) DMIC1_DCRM	0X0	DMIC1 DC removal filter enable and cut-off frequency
		(10:8) DMIC0_DCRM	(10:8) DMIC0_DCRM	0X0	DMIC0 DC removal filter enable and cut-off frequency
		(1) DMIC1_CLK_EDGE	(1) DMIC1_CLK_EDGE	0X0	DMIC1 input clock edge
		(0) DMIC0_CLK_EDGE	(0) DMIC0_CLK_EDGE	0X0	DMIC0 input clock edge
0x40000E0C	AUDIO_DMIC0_GAIN	(11:0) GAIN	(11:0) GAIN	0X800	DMIC calibration gain (unsigned value from 0 to +2)
0x40000E10	AUDIO_DMIC1_GAIN	(11:0) GAIN	(11:0) GAIN	0X800	DMIC calibration gain (unsigned value from 0 to +2)
0x40000E14	AUDIO_DMIC_DATA	-	(31:16) DMIC1_DATA	0X0	DMIC1 input data (16-bit)
		-	(15:0) DMIC0_DATA	0X0	DMIC0 input data (16-bit)
0x40000E18	AUDIO_DMIC0_DATA	-	(31:0) DATA	0X0	DMIC0 input data (LSB or MSB aligned according to AUDIO_CFG); data is sign extended from 16-bit to 32-bit when read in LSB aligned mode or zero padded when read in MSB aligned mode
0x40000E1C	AUDIO_DMIC1_DATA	-	(31:0) DATA	0X0	DMIC1 input data (LSB or MSB aligned according to AUDIO_CFG); data is sign extended from 16-bit to 32-bit when read in LSB aligned mode or zero padded when read in MSB aligned mode
0x40000E20	AUDIO_OD_CFG	(19:16) DCRM	(19:16) DCRM	0X0	Output driver DC removal filter enable and cut-off frequency
		(10) DITHER	(10) DITHER	0X0	Sigma-delta modulator dithering enable
		(0) CLK_EDGE	(0) CLK_EDGE	0X1	Output driver output clock edge

Address	Register Name	Register Write	Register Read	Default	Description
0x40000E24	AUDIO_OD_GAIN	(11:0) GAIN	(11:0) GAIN	0X800	Output driver calibration gain (unsigned value from 0 to +2)
0x40000E28	AUDIO_OD_DATA	(31:0) DATA	-	N/A	OD output data (LSB or MSB aligned according to OD_CFG); data is truncated to 16 bits when written in LSB aligned mode or rounded symmetrically with saturation when written in MSB aligned mode
		-	(31:0) DATA_RD	0X0	OD output data (LSB or MSB aligned according to OD_CFG); data is sign extended from 16-bit to 32-bit when read in LSB aligned mode or zero padded in MSB aligned mode
0x40000E30	AUDIO_SDM_CFG	(31:0) SDM_CFG	(31:0) SDM_CFG	0X0	Sigma-Delta modulator internal configuration for test purposes

A.17 CRC GENERATOR CONTROL

Address	Register Name	Register Write	Register Read	Default	Description
0x40000F00	CRC_CTRL	(4) FINAL_CRC_XOR	(4) FINAL_CRC_XOR	0X0	Selects the final CRC XOR mode
		(3) FINAL_CRC_REVERSE	(3) FINAL_CRC_REVERSE	0X0	Selects the final CRC reversal mode
		(2) BIT_ORDER	(2) BIT_ORDER	0X0	Selects the bit order for bytes added to the CRC
		(1) CRC_TYPE	(1) CRC_TYPE	0X0	Selects the CRC type
		(0) BYTE_ORDER	(0) BYTE_ORDER	0X0	Selects the endianness for bytes added to the CRC
0x40000F04	CRC_VALUE	(31:0) CURRENT_CRC	(31:0) CURRENT_CRC	0XFFFF	CRC generator value: Write 0xFFFFFFFF (32) or 0xFFFF (CCITT) to initialize the CRC, read provides the current CRC value.
0x40000F08	CRC_ADD_1	(0) CRC_ADD_1_BIT	-	N/A	Add 1 bit to the CRC calculation
0x40000F0C	CRC_ADD_8	(7:0) CRC_ADD_8	-	N/A	Add 1 byte (8 bits) to the CRC calculation
0x40000F10	CRC_ADD_16	(15:0) CRC_ADD_16	-	N/A	Add 1 half-word (16 bits) to the CRC calculation
0x40000F14	CRC_ADD_24	(23:0) CRC_ADD_24_BITS	-	N/A	Add 3 bytes (24 bits) to the CRC calculation
0x40000F18	CRC_ADD_32	(31:0) CRC_ADD_32	-	N/A	Add 1 word (32 bits) to the CRC calculation
0x40000F1C	CRC_FINAL	-	(31:0) FINAL_CRC	0X0	CRC generator final value: After XOR for CCITT or byte reversal for CRC-32

A.18 Audio Sink Clock Counters

Address	Register Name	Register Write	Register Read	Default	Description
0x40001000	AUDIOSINK_CTRL	(8) PHASE_CNT_START_NO_WAIT	-	N/A	Start the audio sink clock phase counter mechanism without waiting on a sync pulse
		-	(7) PERIOD_CNT_STATUS	0X0	Audio sink clock period counter status
		(6) PERIOD_CNT_STOP	-	N/A	Stop the audio sink clock period counter mechanism
		(5) PERIOD_CNT_START	-	N/A	Start the audio sink clock period counter mechanism
		-	(4) PHASE_CNT_MISSED_STATUS	0X0	Audio sink clock phase counter missed status
		-	(3) PHASE_CNT_STATUS	0X0	Audio sink clock phase counter status
		(2) PHASE_CNT_STOP	-	N/A	Stop the audio sink clock phase counter mechanism
		(1) PHASE_CNT_START	-	N/A	Start the audio sink clock PHASE counter mechanism and wait for sync pulse
		(0) CNT_RESET	-	N/A	Reset audio sink clock counter
0x40001004	AUDIOSINK_CFG	(3:0) PERIODS_CFG	(3:0) PERIODS_CFG	0X0	Defines how over how many audio sink clock periods the period counter measures
0x40001008	AUDIOSINK_CNT	-	(11:0) CNT	0X0	Sink clock counter value
0x4000100C	AUDIOSINK_PHASE_CNT	(15:0) PHASE_CNT	(15:0) PHASE_CNT	0X0	Sink clock phase counter value
0x40001010	AUDIOSINK_PERIOD_CNT	(15:0) PERIOD_CNT	(15:0) PERIOD_CNT	0X0	Sink clock period counter value

A.19 ASRC Configuration and Control

Address	Register Name	Register Write	Register Read	Default	Description
0x40001100	ASRC_CTRL	-	(14) ASRC_PROC_STATUS	0X0	The ASRC processing state
		-	(13) ASRC_OUT_REQ	0X0	The ASRC_OUT register status
		-	(12) ASRC_IN_REQ	0X0	The ASRC_IN register status
		-	(11) ASRC_UPDATE_ERR	0X0	The ASRC state/configuration update error interrupt status
		-	(10) ASRC_IN_ERR	0X0	The ASRC input interface error interrupt status
		(9) ASRC_UPDATE_ERR_CLR	-	N/A	Clear the ASRC update/configuration error interrupt status
		(8) ASRC_IN_ERR_CLR	-	N/A	Clear the ASRC input interface error interrupt
		(3) ASRC_RESET	-	N/A	Write a 1 to reset ASRC
		-	(2) ASRC_EN_STATUS	0X0	Enable status of the re-sampler block
		(1) ASRC_DISABLE	-	N/A	Disable the re-sampler block
		(0) ASRC_ENABLE	-	N/A	Enable the re-sampler block
0x40001104	ASRC_INT_ENABLE	(3) ASRC_UPDATE_ERR	(3) ASRC_UPDATE_ERR	0X1	The ASRC state/configuration update error interrupt mask
		(2) ASRC_IN_ERR	(2) ASRC_IN_ERR	0X0	The ASRC input interface error interrupt mask
		(1) ASRC_OUT_REQ	(1) ASRC_OUT_REQ	0X0	The ASRC_OUT register interrupt status
		(0) ASRC_IN_REQ	(0) ASRC_IN_REQ	0X0	The ASRC_IN register interrupt status
0x40001108	ASRC_OUT	(15:0) ASRC_OUT	(15:0) ASRC_OUT	0X0	Audio sample output
0x4000110C	ASRC_IN	(15:0) ASRC_IN	(15:0) ASRC_IN	0X0	Audio sample input
0x40001110	ASRC_CFG	(2) WDF_TYPE	(2) WDF_TYPE	0X0	WDF Type Selection
		(1:0) ASRC_MODE	(1:0) ASRC_MODE	0X0	ASRC mode
0x40001114	ASRC_OUTPUT_CNT	(11:0) ASRC_OUTPUT_CNT	(11:0) ASRC_OUTPUT_CNT	0X0	ASRC output sample counter
0x40001118	ASRC_PHASE_INC	(31:0) ASRC_STEP	(31:0) ASRC_STEP	0X0	ASRC_PHASE_INC
0x4000111C	ASRC_PHASE_CNT	(31:0) ASRC_PHASE_CNT	(31:0) ASRC_PHASE_CNT	0X0	ASRC phase counter
0x40001120	ASRC_STATE_MEM	(31:0) ASRC_STATE_MEM	(31:0) ASRC_STATE_MEM	0X0	ASRC State Memory 0 to 29

A.20 Analog-to-Digital Converter and Battery Monitoring

Address	Register Name	Register Write	Register Read	Default	Description
0x40001200	ADC_DATA_TRIM_CH	-	(13:0) DATA	0X0	14-bit ADC conversion result
0x40001220	ADC_DATA_AUDIO_CH	-	(31:0) DATA	0X0	14-bit ADC conversion result (sign extended to 32 bits)
0x40001240	ADC_INPUT_SEL	(6:4) POS_INPUT_SEL	(6:4) POS_INPUT_SEL	0X6	Positive input selection
		(2:0) NEG_INPUT_SEL	(2:0) NEG_INPUT_SEL	0X7	Negative input selection
0x40001260	ADC_CFG	(5) DUTY_DIVIDER	(5) DUTY_DIVIDER	0X0	Duty cycling VDD divider
		(4) CONTINUOUS_MODE	(4) CONTINUOUS_MODE	0X0	ADC continuously sampling the channel selected as interrupt source (ADC_INT_CH_NUM)
		(3:0) FREQ	(3:0) FREQ	0X0	Defines the sampling frequency of the ADC channels
0x40001264	ADC_OFFSET	(14:0) DATA	(14:0) DATA	0X0	15-bit ADC signed offset
0x40001270	ADC_BATMON_CFG	(23:16) AL))_COUNT_VALUE	(23:16) ALARM_COUNT_VALUE	0X0	An Alarm Status bit gets set when SUPPLY_COUNT_VALUE= ALARM_COUNT_VALUE
		(15:8) SUPPLY_THRESHOLD	(15:8) SUPPLY_THRESHOLD	0X80	Low voltage detection threshold (7.8 mV steps)
		(0) SUPPLY_SRC	(0) SUPPLY_SRC	0X0	Selects the power supply voltage source to be monitored
0x40001274	ADC_BATMON_INT_ENABLE	(4) BATMON_ALARM_INT_ENABLE	(4) BATMON_ALARM_INT_ENABLE	0X0	The BATMON Alarm interrupt mask
		(3:1) ADC_INT_CH_NUM	(3:1) ADC_INT_CH_NUM	0X0	Channel number triggering the ADC interrupt
		(0) ADC_INT_ENABLE	(0) ADC_INT_ENABLE	0X0	The ADC new sample ready interrupt mask
0x40001278	ADC_BATMON_COUNT_VAL	-	(7:0) SUPPLY_COUNT_VALUE	0X0	Number of times the battery voltage has fallen below the battery monitor voltage threshold. The counter is reset when read.
0x4000127C	ADC_BATMON_STATUS	(12) BATMON_ALARM_CLEAR	-	N/A	Battery monitoring alarm status bit
		(9) ADC_OVERRUN_CLEAR	-	N/A	ADC Overrun condition
		(8) ADC_READY_CLEAR	-	N/A	ADC new sample Ready status bit
		-	(4) BATMON_ALARM_STAT	0X0	Battery monitoring alarm status bit
		-	(1) ADC_OVERRUN_STAT	0X0	ADC Overrun condition
		-	(0) ADC_READY_STAT	0X0	ADC new sample Ready status bit

A.21 ACS DOMAIN (ANALOG BRIDGE ACCESS)

Address	Register Name	Register Write	Register Read	Default	Description
0x40001300	ACS_BG_CTRL	(12:8) SLOPE_TRIM	(12:8) SLOPE_TRIM	0XB	Temperature coefficient trimming
		(5:0) VTRIM	(5:0) VTRIM	0X1E	Reference voltage trimming (2.5 mV steps)
0x40001304	ACS_VCC_CTRL	(19:16) ICH_TRIM	(19:16) ICH_TRIM	0X4	Inductor charge current trimming
		(11) CCM_ENABLE	(11) CCM_ENABLE	0X0	Enable CCM mode
		(10) PULSE_CTRL	(10) PULSE_CTRL	0X0	Pulse mode control
		(9) CHARGE_CTRL	(9) CHARGE_CTRL	0X1	Charge mode control
		(8) BUCK_ENABLE	(8) BUCK_ENABLE	0X0	Enable buck converter mode
		(4:0) VTRIM	(4:0) VTRIM	0XA	Output voltage trimming configuration in 10 mV steps
0x40001308	ACS_VDDA_CP_CTRL	(1:0) PTRIM	(1:0) PTRIM	0X0	Output power trimming
0x4000130C	ACS_VDDC_CTRL	(21:16) STANDBY_VTRIM	(21:16) STANDBY_VTRIM	0X23	VDDC standby voltage trimming (10 mV steps)
		(13) ENABLE_LOW_BIAS	(13) ENABLE_LOW_BIAS	0X0	Low power mode control
		(12) SLEEP_CLAMP	(12) SLEEP_CLAMP	0X0	Sleep mode clamp control
		(5:0) VTRIM	(5:0) VTRIM	0X23	Output voltage trimming configuration in 10 mV steps
0x40001310	ACS_VDDM_CTRL	(21:16) STANDBY_VTRIM	(21:16) STANDBY_VTRIM	0X23	VDDM standby voltage trimming (10 mV steps)
		(13) ENABLE_LOW_BIAS	(13) ENABLE_LOW_BIAS	0X0	Low power mode control
		(12) SLEEP_CLAMP	(12) SLEEP_CLAMP	0X0	Sleep mode clamp control
		(5:0) VTRIM	(5:0) VTRIM	0X23	Output voltage trimming configuration in 10 mV steps
0x40001314	ACS_VDDRF_CTRL	-	(24) READY	0X0	Supply ready
		(12) CLAMP	(12) CLAMP	0X0	Disable mode clamp control
		(8) ENABLE	(8) ENABLE	0X0	Enable control
		(5:0) VTRIM	(5:0) VTRIM	0X23	Output voltage trimming configuration in 10 mV steps
0x40001318	ACS_VDDPA_CTRL	(12) VDDPA_SW_CTRL	(12) VDDPA_SW_CTRL	0X0	Power amplifier supply control
		(9) ENABLE_ISENSE	(9) ENABLE_ISENSE	0X0	Enable current sensing circuit
		(8) ENABLE	(8) ENABLE	0X0	Enable control
		(5:0) VTRIM	(5:0) VTRIM	0X37	Output voltage trimming configuration in 10 mV steps
0x4000131C	ACS_VDDRET_CTRL	(18:17) VDDMRET_VTRIM	(18:17) VDDMRET_VTRIM	0X3	VDDMRET retention regulator voltage trimming
		(16) VDDMRET_ENABLE	(16) VDDMRET_ENABLE	0X0	Enable/Disable the VDDMRET retention regulator
		(10:9) VDDTRET_VTRIM	(10:9) VDDTRET_VTRIM	0X3	VDDTRET retention regulator voltage trimming
		(8) VDDTRET_ENABLE	(8) VDDTRET_ENABLE	0X0	Enable/Disable the VDDTRET retention regulator
		(2:1) VDDCRET_VTRIM	(2:1) VDDCRET_VTRIM	0X3	VDDCRET retention regulator voltage trimming
		(0) VDDCRET_ENABLE	(0) VDDCRET_ENABLE	0X0	Enable/Disable the VDDCRET retention regulator

Address	Register Name	Register Write	Register Read	Default	Description
0x40001320	ACS_RCOSC_CTRL	(18) CLOCK_MULT	(18) CLOCK_MULT	0X0	Enable 12 MHz mode of startup oscillator
		(16) RC_OSC_EN	(16) RC_OSC_EN	0X0	Enable/Disable the 32 kHz RC Oscillator
		(15) FTRIM_FLAG	(15) FTRIM_FLAG	0X0	Trimming flag
		(13:8) FTRIM_START	(13:8) FTRIM_START	0X20	Start RC oscillator frequency trimming
		(6) FTRIM_32K_ADJ	(6) FTRIM_32K_ADJ	0X0	Adjust 32 kHz oscillator frequency range
		(5:0) FTRIM_32K	(5:0) FTRIM_32K	0X20	32 kHz RC oscillator frequency trimming
0x40001324	ACS_XTAL32K_CTRL	-	(24) READY	0X0	XTAL ready status
		(18) XIN_CAP_BYPASS_EN	(18) XIN_CAP_BYPASS_EN	0X0	Switch to bypass the added XIN serial cap to reduce the leakage
		(17) EN_AMPL_CTRL	(17) EN_AMPL_CTRL	0X0	XTAL enable amplitude control (regulation)
		(16) FORCE_READY	(16) FORCE_READY	0X0	XTAL bypass the ready detector
		(13:8) CLOAD_TRIM	(13:8) CLOAD_TRIM	0X9	XTAL load capacitance configuration
		(7:4) ITRIM	(7:4) ITRIM	0X7	XTAL current trimming
		(1) IBOOST	(1) IBOOST	0X0	XTAL current boosting (4x)
		(0) ENABLE	(0) ENABLE	0X0	Enable the XTAL 32 kHz oscillator
0x40001328	ACS_BB_TIMER_CTRL	(9:8) BB_CLK_PRESCALE	(9:8) BB_CLK_PRESCALE	0X0	Prescale value for the baseband timer clock
		(0) BB_TIMER_NRESET	(0) BB_TIMER_NRESET	0X0	nReset signal for the baseband timer
0x4000132C	ACS_CLK_DET_CTRL	-	(8) CLOCK_PRESENT	0X1	Clock present flag
		(1) RESET_IGNORE	(1) RESET_IGNORE	0X0	Clock detector reset condition ignore
		(0) ENABLE	(0) ENABLE	0X1	Clock detector enable
0x40001330	ACS_RTC_CFG	(31:0) START_VALUE	(31:0) START_VALUE	0X7FFF	Start value for the RTC timer counter (counts from start_value down to 0)
0x40001334	ACS_RTC_COUNT	-	(31:0) VALUE	0X0	RTC timer current value
0x40001338	ACS_RTC_CTRL	(25) FORCE_CLOCK	-	N/A	Force a clock on RTC timer (Test Purpose)
		(24) RESET	-	N/A	Reset the RTC timer
		(7:4) ALARM_CFG	(7:4) ALARM_CFG	0X0	Configure RTC timer alarm
		(3:1) CLK_SRC_SEL	(3:1) CLK_SRC_SEL	0X0	Select the RTC, standby and bb timer clock source
		(0) ENABLE	(0) ENABLE	0X0	Enable counter and RTC interrupt every 1s
0x40001340	ACS_PWR_MODES_CTRL	(31:0) POWER_MODE	-	N/A	32-bit key to enter RUN, STANDBY or SLEEP mode

Address	Register Name	Register Write	Register Read	Default	Description
0x40001344	ACS_WAKEUP_CTRL	(24) PADS_RETENTION_EN	(24) PADS_RETENTION_EN	0X0	Enable / Disable the retention mode of the pads
		(20) BOOT_FLASH_APP_REBOOT	(20) BOOT_FLASH_APP_REBOOT	0X0	Boot mode flag
		-	(19) RC_CLOCK_MULT	0X0	RC oscillator clock multiplier read only flag (mirror of CLOCK_MULT of ACS_RCOSC_CTRL register)
		-	(18) RC_FTRIM_FLAG	0X0	RC oscillator trimming read only flag (mirror of FTRIM_FLAG of ACS_RCOSC_CTRL register
		(17:16) BOOT_SELECT	(17:16) BOOT_SELECT	0X0	Boot selection to indicate boot source
		-	(15) DCDC_OVERLOAD_WAKEUP	0X0	
		-	(14) WAKEUP_PAD_WAKEUP	0X0	
		-	(13) RTC_ALARM_WAKEUP	0X0	
		-	(12) BB_TIMER_WAKEUP	0X0	
		-	(11) DIO3_WAKEUP	0X0	
		-	(10) DIO2_WAKEUP	0X0	
		-	(9) DIO1_WAKEUP	0X0	
		-	(8) DIO0_WAKEUP	0X0	
		(7) DCDC_OVERLOAD_CLEAR	-	N/A	
		(6) WAKEUP_PAD_WAKEUP_CLEAR	-	N/A	
		(5) RTC_ALARM_WAKEUP_CLEAR	-	N/A	
		(4) BB_TIMER_WAKEUP_CLEAR	-	N/A	
		(3) DIO3_WAKEUP_CLEAR	-	N/A	
		(2) DIO2_WAKEUP_CLEAR	-	N/A	
		(1) DIO1_WAKEUP_CLEAR	-	N/A	
		(0) DIO0_WAKEUP_CLEAR	-	N/A	

Address	Register Name	Register Write	Register Read	Default	Description
0x40001348	ACS_WAKEUP_CFG	(18:16) DELAY	(18:16) DELAY	0X5	Delay from VDDC ready to digital clock enable (power of 2)
		(9) DCDC_OVERLOAD_EN	(9) DCDC_OVERLOAD_EN	0X0	Enable / Disable the Wake-up functionality on the DCDC overload flag
		(8) WAKEUP_PAD_POL	(8) WAKEUP_PAD_POL	0X0	Wake-up polarity on the WAKEUP pad
		(7) DIO3_POL	(7) DIO3_POL	0X0	Wake-up polarity on the DIO3 pad
		(6) DIO2_POL	(6) DIO2_POL	0X0	Wake-up polarity on the DIO2 pad
		(5) DIO1_POL	(5) DIO1_POL	0X0	Wake-up polarity on the DIO1 pad
		(4) DIO0_POL	(4) DIO0_POL	0X0	Wake-up polarity on the DIO0 pad
		(3) DIO3_EN	(3) DIO3_EN	0X0	Enable / Disable the Wake-up functionality on the DIO3 pad
		(2) DIO2_EN	(2) DIO2_EN	0X0	Enable / Disable the Wake-up functionality on the DIO2 pad
		(1) DIO1_EN	(1) DIO1_EN	0X0	Enable / Disable the Wake-up functionality on the DIO1 pad
		(0) DIO0_EN	(0) DIO0_EN	0X0	Enable / Disable the Wake-up functionality on the DIO0 pad
0x4000134C	ACS_WAKEUP_STATE	-	(18:16) WAKEUP_SRC	0X0	Status register indicates the last wake-up source
		-	(7:0) RTC_VALUE	0X0	RTC counter value captured at wakeup event (only 8 LSBs, corresponds to 7.8 ms)
0x40001350	ACS_WAKEUP_GP_DATA	(31:0) GP_DATA	(31:0) GP_DATA	0X0	32-bit General-Purpose RW Data

Address	Register Name	Register Write	Register Read	Default	Description
0x40001354	`ACS_RESET_STATUS`	-	(14) `TIMEOUT_RESET_FLAG`	0X0	Sticky flag that detects that a timeout in the power up sequence
		-	(13) `CLK_DET_RESET_FLAG`	0X0	Sticky flag that detects that a clock detector reset occurred
		-	(12) `VDDA_RESET_FLAG`	0X1	Sticky flag that detects that a VDDA reset occurred (triggered by vdda_ready = 0)
		-	(11) `VDDM_RESET_FLAG`	0X1	Sticky flag that detects that a VDDM reset occurred (triggered by vddm_ready = 0)
		-	(10) `VDDC_RESET_FLAG`	0X1	Sticky flag that detects that a VDDC reset occurred (triggered by vddc_ready = 0)
		-	(9) `PAD_RESET_FLAG`	0X0	Sticky flag that detects that a reset occurred due to pad NRESET
		-	(8) `POR_RESET_FLAG`	0X1	Sticky flag that detects that a POR reset occurred
		(6) `TIMEOUT_RESET_FLAG_CLEAR`	-	N/A	Reset the sticky TIMEOUT_RESET flag.
		(5) `CLK_DET_RESET_FLAG_CLEAR`	-	N/A	Reset the sticky CLK_DET_RESET flag.
		(4) `VDDA_RESET_FLAG_CLEAR`	-	N/A	Reset the sticky VDDA_RESET flag.
		(3) `VDDM_RESET_FLAG_CLEAR`	-	N/A	Reset the sticky VDDM_RESET flag.
		(2) `VDDC_RESET_FLAG_CLEAR`	-	N/A	Reset the sticky VDDC_RESET flag.
		(1) `PAD_RESET_FLAG_CLEAR`	-	N/A	Reset the sticky PAD_RESET flag.
		(0) `POR_RESET_FLAG_CLEAR`	-	N/A	Reset the sticky POR_RESET flag.
0x40001358	`ACS_AOUT_CTRL`	(13) `RTC_CLOCK_DIO0_STOP_EDGE`	(13) `RTC_CLOCK_DIO0_STOP_EDGE`	0X0	Stop edge for RTC clock output on AOUT
		(12:11) `RTC_CLOCK_DIO0_STOP_SRC`	(12:11) `RTC_CLOCK_DIO0_STOP_SRC`	0X0	Stop source for RTC clock output on AOUT
		(10:8) `RTC_CLOCK_DIO0_START`	(10:8) `RTC_CLOCK_DIO0_START`	0X0	Start event for RTC clock output on AOUT (RTC prescaler and counter need to be enabled)
		(4:0) `TEST_AOUT`	(4:0) `TEST_AOUT`	0X0	AOUT test signal selection
0x4000135C	`ACS_JIC_READ`	-	(7:0) `BYTE0_RO`	0XFF	JIC read only register bits (returning signals from analog part: tied to 1)

A.22 BASEBAND CONTROLLER INTERFACE

Address	Register Name	Register Write	Register Read	Default	Description
0x40001400	BBIF_CTRL	(16) WAKEUP_REQ	(16) WAKEUP_REQ	0X0	External wake up request used to sort-out sleep modes
		(9:4) CLK_SEL	(9:4) CLK_SEL	0X8	Configure the internal baseband controller clock divider in order to provide a 1MHz reference clock
		(0) CLK_ENABLE	(0) CLK_ENABLE	0X0	Enable the baseband controller clocks generation
0x40001404	BBIF_STATUS	-	(15:11) LINK_FORMAT	0X0	BLE link format
		-	(8:4) LINK_LABEL	0X0	BLE link label
		-	(3) AOBLE_STATUS	0X1	Audio over BLE feature status
		-	(2) CLK_STATUS	0X0	Clock status defining the current active clock in use
		-	(1) OSC_EN	0X0	Oscillator front-end enabling
		-	(0) RADIO_EN	0X0	RF front-end enabling
0x40001408	BBIF_COEX_CTRL	(4) TX	(4) TX	0X0	Indicates if the RF front-end performs a non-BLE Tx activity
		(0) RX	(0) RX	0X0	Indicates if the RF front-end performs a non-BLE Rx activity
0x4000140C	BBIF_COEX_STATUS	-	(15:12) BLE_PTI	0X0	Indicates the priority level of the current RW-BLE core activity
		-	(8) BLE_IN_PROCESS	0X0	Indicate if the RW-BLE core has an event in process, active high.
		-	(4) BLE_TX	0X0	Indicates if the RW-BLE core is busy and performs Tx activity, active high.
		-	(0) BLE_RX	0X0	Indicates if the RW-BLE core is busy and performs Rx activity, active high
0x40001410	BBIF_COEX_INT_CFG	(9:8) BLE_IN_PROCESS_EVENT	(9:8) BLE_IN_PROCESS_EVENT	0X0	BLE_IN_PROCESS event interrupt configuration
		(5:4) BLE_TX_EVENT	(5:4) BLE_TX_EVENT	0X0	BLE_TX event interrupt configuration
		(1:0) BLE_RX_EVENT	(1:0) BLE_RX_EVENT	0X0	BLE_RX event interrupt configuration
0x40001414	BBIF_COEX_INT_STATUS	-	(4) BLE_TX_EVENT_FLAG	0X0	Indicates if a BLE_TX_EVENT interrupt has been generated
		-	(0) BLE_RX_EVENT_FLAG	0X0	Indicates if a BLE_RX_EVENT interrupt has been generated
0x40001418	BBIF_SYNC_CFG	(17) RF_RX	(17) RF_RX	0X0	Specify if the RF front-end is currently receiving the audio link
		(16) RF_ACTIVE	(16) RF_ACTIVE	0X0	Specify if the RF front-end is currently processing the audio link
		(15:11) LINK_FORMAT	(15:11) LINK_FORMAT	0X0	Configure the BLE link format for synchronization
		(8:4) LINK_LABEL	(8:4) LINK_LABEL	0X0	Configure the BLE link label for synchronization
		(3:1) SOURCE	(3:1) SOURCE	0X0	Select the BLE/RF link synchronization source
		(0) ENABLE	(0) ENABLE	0X0	Enable the frame synchronization pulse filter

A.23 BASEBAND CONTROLLER

Address	Register Name	Register Write	Register Read	Default	Description
0x40001500	BB_RWBBCNTL	(31) MASTER_SOFT_RST	(31) MASTER_SOFT_RST	0X0	Reset the complete system except registers and timing generator
		(30) MASTER_TGSOFT_RST	(30) MASTER_TGSOFT_RST	0X0	Reset the timing generator
		(29) REG_SOFT_RST	(29) REG_SOFT_RST	0X0	Reset the complete register block
		(28) SWINT_REQ	(28) SWINT_REQ	0X0	Forces the generation of ble_sw_irq
		(26) RFTEST_ABORT	(26) RFTEST_ABORT	0X0	Abort the current RF testing defined as per CS-FORMAT
		(25) ADVERT_ABORT	(25) ADVERT_ABORT	0X0	Abort the current scan window
		(24) SCAN_ABORT	(24) SCAN_ABORT	0X0	Abort the current advertising event
		(22) MD_DSB	(22) MD_DSB	0X0	Allow a single Tx/Rx exchange whatever the MD bits are
		(21) SN_DSB	(21) SN_DSB	0X0	Disable sequence number management
		(20) NESN_DSB	(20) NESN_DSB	0X0	Disable acknowledge scheme
		(19) CRYPT_DSB	(19) CRYPT_DSB	0X0	Disable encryption / decryption
		(18) WHIT_DSB	(18) WHIT_DSB	0X0	Disable whitening
		(17) CRC_DSB	(17) CRC_DSB	0X0	Disable CRC stripping
		(16) HOP_REMAP_DSB	(16) HOP_REMAP_DSB	0X0	Disable frequency hopping remapping algorithm
		(9) ADVERTFILT_EN	(9) ADVERTFILT_EN	0X0	Advertising channels error filtering enable control
		(8) RWBLE_EN	(8) RWBLE_EN	0X0	Enable RW-BLE core exchange table pre-fetch mechanism
		(7:4) RXWINSZDEF	(7:4) RXWINSZDEF	0X0	Default Rx Window size in us (used when device is master connected or performs its second receipt)
		(2:0) SYNCERR	(2:0) SYNCERR	0X0	Indicates the maximum number of errors allowed to recognize the synchronization word
0x40001504	BB_VERSION	-	(31:24) TYP	0X8	RW-BLE core type (BLE v4.2)
		-	(23:16) REL	0X0	RW-BLE core version - major release number
		-	(15:8) UPG	0X9	RW-BLE core upgrade - upgrade number
		-	(7:0) BUILD	0X1	RW-BLE core build - build number

Address	Register Name	Register Write	Register Read	Default	Description
0x40001508	BB_RWBLEBCONF	-	(31) DMMODE	0X0	RW-BLE core dual mode
		-	(25:24) ISOPORTNB	0X3	Number of supported isochronous channels
		-	(23) DECIPHER	0X0	AES deciphering present
		-	(21) COEX	0X1	Coexistence mechanism
		-	(20:16) RFIF	0X3	Support of the RF front-end
		-	(15) USEDBG	0X1	Diagnostic port
		-	(14) USECRYPT	0X1	AES-CCM encryption
		-	(13:8) CLK_SEL	0X8	Operating frequency (in MHz)
		-	(7) INTMODE	0X0	Interruption mode
		-	(6) BUSTYPE	0X1	Processor bus type
		-	(5) DATA_WIDTH	0X1	Processor bus width
		-	(4:0) ADD_WIDTH	0XE	Value of the RW_BLE_ADDRESS_WIDTH parameter concerted into binary
0x4000150C	BB_INTCNTL	(15) CSCNTDEVMSK	(15) CSCNTDEVMSK	0X1	CSCNT interrupt mask during event allowing to enable CSCNT interrupt generation during events
		(12) AUDIOINT2MSK	(12) AUDIOINT2MSK	0X0	Audio channel 2 interrupt mask
		(11) AUDIOINT1MSK	(11) AUDIOINT1MSK	0X0	Audio channel 1 interrupt mask
		(10) AUDIOINT0MSK	(10) AUDIOINT0MSK	0X0	Audio channel 0 interrupt mask
		(9) SWINTMSK	(9) SWINTMSK	0X0	SW triggered interrupt mask
		(8) EVENTAPFAINTMSK	(8) EVENTAPFAINTMSK	0X1	End of event / anticipated pre-fetch abort interrupt mask
		(7) FINETGTIMINTMSK	(7) FINETGTIMINTMSK	0X0	Fine target timer mask
		(6) GROSSTGTIMINTMSK	(6) GROSSTGTIMINTMSK	0X0	Gross target timer mask
		(5) ERRORINTMSK	(5) ERRORINTMSK	0X0	Error interrupt mask
		(4) CRYPTINTMSK	(4) CRYPTINTMSK	0X1	Encryption engine interrupt mask
		(3) EVENTINTMSK	(3) EVENTINTMSK	0X1	End of event interrupt mask
		(2) SLPINTMSK	(2) SLPINTMSK	0X1	Sleep mode interrupt mask
		(1) RXINTMSK	(1) RXINTMSK	0X1	Rx interrupt mask
		(0) CSCNTINTMSK	(0) CSCNTINTMSK	0X1	625us base time interrupt mask

Address	Register Name	Register Write	Register Read	Default	Description
0x40001510	BB_INTSTAT	-	(12) AUDIOINT2STAT	0X0	Audio channel 2 interrupt status
		-	(11) AUDIOINT1STAT	0X0	Audio channel 1 interrupt status
		-	(10) AUDIOINT0STAT	0X0	Audio channel 0 interrupt status
		-	(9) SWINTSTAT	0X0	SW triggered interrupt status
		-	(8) EVENTAPFAINTSTAT	0X0	End of event / anticipated pre-fetch abort interrupt status
		-	(7) FINETGTIMINTSTAT	0X0	Masked fine target timer error interrupt status
		-	(6) GROSSTGTIMINTSTAT	0X0	Masked gross target timer interrupt status
		-	(5) ERRORINTSTAT	0X0	Masked error interrupt status
		-	(4) CRYPTINTSTAT	0X0	Masked encryption engine interrupt status
		-	(3) EVENTINTSTAT	0X0	Masked end of event interrupt status
		-	(2) SLPINTSTAT	0X0	Masked sleep interrupt status
		-	(1) RXINTSTAT	0X0	Masked packet reception interrupt status
		-	(0) CSCNTINTSTAT	0X0	Masked 625us base time reference interrupt status
0x40001514	BB_INTRAWSTAT	-	(12) AUDIOINT2RAWSTAT	0X0	Audio channel 2 interrupt raw status
		-	(11) AUDIOINT1RAWSTAT	0X0	Audio channel 1 interrupt raw status
		-	(10) AUDIOINT0RAWSTAT	0X0	Audio channel 0 interrupt raw status
		-	(9) SWINTRAWSTAT	0X0	SW triggered interrupt raw status
		-	(8) EVENTAPFAINTRAWSTAT	0X0	End of event / anticipated pre-fetch abort interrupt raw status
		-	(7) FINETGTIMINTRAWSTAT	0X0	Masked fine target timer error interrupt raw status
		-	(6) GROSSTGTIMINTRAWSTAT	0X0	Masked gross target timer interrupt raw status
		-	(5) ERRORINTRAWSTAT	0X0	Masked error interrupt raw status
		-	(4) CRYPTINTRAWSTAT	0X0	Masked encryption engine interrupt raw status
		-	(3) EVENTINTRAWSTAT	0X0	Masked end of event interrupt raw status
		-	(2) SLPINTRAWSTAT	0X0	Masked sleep interrupt raw status
		-	(1) RXINTRAWSTAT	0X0	Masked packet reception interrupt raw status
		-	(0) CSCNTINTRAWSTAT	0X0	Masked 625us base time reference interrupt raw status

Address	Register Name	Register Write	Register Read	Default	Description
0x40001518	BB_INTACK	-	(12) AUDIOINT2ACK	0X0	Audio channel 2 interrupt acknowledgement bit
		-	(11) AUDIOINT1ACK	0X0	Audio channel 1 interrupt acknowledgement bit
		-	(10) AUDIOINT0ACK	0X0	Audio channel 0 interrupt acknowledgement bit
		(9) SWINTACK	-	N/A	SW triggered interrupt acknowledgement bit
		(8) EVENTAPFAINTACK	-	N/A	End of event / anticipated pre-fetch abort interrupt acknowledgement bit
		(7) FINETGTIMINTACK	-	N/A	Fine target timer interrupt acknowledgement bit
		(6) GROSSTGTIMINTACK	-	N/A	Gross target timer interrupt acknowledgement bit
		(5) ERRORINTACK	-	N/A	Error interrupt acknowledgement bit
		(4) CRYPTINTACK	-	N/A	Encryption engine interrupt acknowledgement bit
		(3) EVENTINTACK	-	N/A	End of event interrupt acknowledgment bit
		(2) SLPINTACK	-	N/A	End of deep sleep interrupt acknowledgment bit
		(1) RXINTACK	-	N/A	Packet reception interrupt acknowledgment bit
		(0) CSCNTINTACK	-	N/A	625us base time reference interrupt acknowledgment bit
0x4000151C	BB_BASETIMECNT	(31) SAMP	(31) SAMP	0X0	Sample the base time counter
		-	(26:0) BASETIMECNT	0X0	Value of the 625us base time reference counter
0x40001520	BB_FINETIMECNT	-	(9:0) FINECNT	0X0	Value of the current us fine time reference counter
0x40001524	BB_BDADDRL	(31:0) BDADDRL	(31:0) BDADDRL	0X0	BLE device address (LSB part)
0x40001528	BB_BDADDRU	(16) PRIV_NPUB	(16) PRIV_NPUB	0X0	BLE device address privacy indicator
		(15:0) BDADDRU	(15:0) BDADDRU	0X0	BLE device address (MSB part)
0x4000152C	BB_ET_CURRENTRXDESCPTR	(31:16) ETPTR	(31:16) ETPTR	0X0	Exchange table pointer that determines the starting point of the exchange table
		(14:0) CURRENTRXDESCPTR	(14:0) CURRENTRXDESCPTR	0X0	Rx descriptor pointer that determines the starting point of the receive buffer chained list
0x40001530	BB_DEEPSLCNTL	(31) EXTWKUPDSB	(31) EXTWKUPDSB	0X0	External wake-up disable
		-	(15) DEEP_SLEEP_STAT	0X0	Indicator of current deep sleep clock mux status
		(4) SOFT_WAKEUP_REQ	(4) SOFT_WAKEUP_REQ	0X0	Wake up request from RW-BLE software applying when system is in deep sleep mode
		(3) DEEP_SLEEP_CORR_EN	(3) DEEP_SLEEP_CORR_EN	0X0	625us base time reference integer and fractional part correction applying when system has been woken-up from deep sleep mode
		(2) DEEP_SLEEP_ON	(2) DEEP_SLEEP_ON	0X0	RW-BLE core power mode control
		(1) RADIO_SLEEP_EN	(1) RADIO_SLEEP_EN	0X0	Controls the radio module
		(0) OSC_SLEEP_EN	(0) OSC_SLEEP_EN	0X0	Controls the RF high frequency crystal oscillator
0x40001534	BB_DEEPSLWKUP	(31:0) DEEPSLTIME	(31:0) DEEPSLTIME	0X0	Determines the time in low_power_clk clock cycles to spend in deep sleep mode before waking-up the device

Address	Register Name	Register Write	Register Read	Default	Description
0x40001538	BB_DEEPSLSTAT	-	(31:0) DEEPSLDUR	0X0	Actual duration of the last deep sleep phase measured in low_power_clk clock cycle
0x4000153C	BB_ENBPRESET	(20:10) TWOSC	(20:10) TWOSC	0X0	Time in low power oscillator cycles allowed for stabilization of the high frequency oscillator when the deep-sleep mode has been left due to sleep-timer expiry (DEEPSLWKUP-DEEPSLTIME])
0x40001540	BB_FINECNTCORR	(9:0) FINECNTCORR	(9:0) FINECNTCORR	0X0	Phase correction value for the 625us reference counter (i.e. fine counter) in us
0x40001544	BB_BASETIMECNTCORR	(26:0) BASETIMECNTCORR	(26:0) BASETIMECNTCORR	0X0	Base time counter correction value
0x40001550	BB_DIAGCNTL	(31) DIAG3_EN	(31) DIAG3_EN	0X0	Enable diagnostic port 3 output
		(29:24) DIAG3	(29:24) DIAG3	0X0	
		(23) DIAG2_EN	(23) DIAG2_EN	0X0	Enable diagnostic port 2 output
		(21:16) DIAG2	(21:16) DIAG2	0X0	
		(15) DIAG1_EN	(15) DIAG1_EN	0X0	Enable diagnostic port 1 output
		(13:8) DIAG1	(13:8) DIAG1	0X0	
		(7) DIAG0_EN	(7) DIAG0_EN	0X0	Enable diagnostic port 0 output
		(5:0) DIAG0	(5:0) DIAG0	0X0	
0x40001554	BB_DIAGSTAT	-	(31:24) DIAG3STAT	0X0	Directly connected to ble_dbg3[7:0] output (debug use only)
		-	(23:16) DIAG2STAT	0X0	Directly connected to ble_dbg2[7:0] output (debug use only)
		-	(15:8) DIAG1STAT	0X0	Directly connected to ble_dbg1[7:0] output (debug use only)
		-	(7:0) DIAG0STAT	0X0	Directly connected to ble_dbg0[7:0] output (debug use only)
0x40001558	BB_DEBUGADDMAX	(31:16) REG_ADDMAX	(31:16) REG_ADDMAX	0X0	Upper limit for the register zone indicated by the reg_inzone flag
		(15:0) EM_ADDMAX	(15:0) EM_ADDMAX	0X0	Upper limit for the exchange memory zone indicated by the em_inzone flag
0x4000155C	BB_DEBUGADDMIN	(31:16) REG_ADDMIN	(31:16) REG_ADDMIN	0X0	Lower limit for the register zone indicated by the reg_inzone flag
		(15:0) EM_ADDMIN	(15:0) EM_ADDMIN	0X0	Lower limit for the exchange memory zone indicated by the em_inzone flag

Address	Register Name	Register Write	Register Read	Default	Description
0x40001560	BB_ERRORTYPESTAT	-	(19) RAL_UNDERRUN	0X0	Indicates Resolving Address List engine Under run issue, happens when RAL List parsing not finished on time
		-	(18) RAL_ERROR	0X0	Indicates Resolving Address List engine faced a bad setting
		-	(17) CONCEVTIRQ_ERROR	0X0	Indicates whether two consecutive and concurrent ble_event_irq have been generated, and not acknowledged in time by the RW-BLE software
		-	(16) RXDATA_PTR_ERROR	0X0	Indicates whether Rx data buffer pointer value programmed is null (major failure)
		-	(15) TXDATA_PTR_ERROR	0X0	Indicates whether Tx data buffer pointer value programmed is null during advertising / scanning / initiating events, or during master / slave connections with non-null packet length (major failure)
		-	(14) RXDESC_EMPTY_ERROR	0X0	Indicates whether Rx descriptor pointer value programmed in register is null (major failure)
		-	(13) TXDESC_EMPTY_ERROR	0X0	Indicates whether Tx descriptor pointer value programmed in control structure is null during advertising / scanning / initiating events (major failure)
		-	(12) CSFORMAT_ERROR	0X0	Indicates whether CS-FORMAT has been programmed with an invalid value (major failure)
		-	(11) LLCHMAP_ERROR	0X0	Indicates Link Layer channel map error, happens when actual number of CS-LLCHMAP bit set to one is different from CS-NBCHGOOD at the beginning of frequency hopping process
		-	(10) ADV_UNDERRUN	0X0	Indicates advertising interval under run
		-	(9) IFS_UNDERRUN	0X0	Indicates inter frame space under run, occurs if IFS time is not enough to update and read control structure / descriptors, and/or white list parsing is not finished and/or decryption time is too long to be finished on time
		-	(8) WHITELIST_ERROR	0X0	Indicates white list timeout error, occurs if white list parsing is not finished on time
		-	(7) EVT_CNTL_APFM_ERROR	0X0	Indicates anticipated pre-fetch mechanism error: happens when 2 consecutive events are programmed, and when the first event is not completely finished while second pre-fetch instant is reached
		-	(6) EVT_SCHDL_APFM_ERROR	0X0	Indicates anticipated pre-fetch mechanism error: happens when 2 consecutive events are programmed, and when the first event is not completely finished while second pre-fetch instant is reached
		-	(5) EVT_SCHDL_ENTRY_ERROR	0X0	Indicates event scheduler faced invalid timing programing on two consecutive ET entries (e.g first one with 624us offset and second one with no offset)
		-	(4) EVT_SCHDL_EMACC_ERROR	0X0	Indicates event scheduler exchange memory access error, happens when exchange memory accesses are not served in time, and blocks the exchange table entry read

Address	Register Name	Register Write	Register Read	Default	Description
0x40001564	BB_SWPROFILING	(31:0) SWPROF	(31:0) SWPROF	0X0	Software profiling register: used by RW-BLE software for profiling purpose
0x40001570	BB_RADIOCNTL0	(31:16) SPIPTR	(31:16) SPIPTR	0X0	Pointer to the buffer containing data to be transferred to or received from the SPI port
		(5:4) SPIFREQ	(5:4) SPIFREQ	0X0	SPI clock frequency
		-	(1) SPICOMP	0X1	SPI transfer status
		(0) SPIGO	(0) SPIGO	0X0	Start SPI transfer when writing a 1
0x40001574	BB_RADIOCNTL1	(31) FORCEAGC_EN	(31) FORCEAGC_EN	0X0	Control ATLAS/Ripple AGC force mode based on radioCNTL2-FORCEAGC_LENGTH value
		(30) FORCEBLEIQ	(30) FORCEBLEIQ	0X0	Control Ripple modulation mode in between FM and I and Q
		(27:16) FORCEAGC_LENGTH	(27:16) FORCEAGC_LENGTH	0X0	Control ATLAS/Ripple AGC force mode based on radioCNTL2-FORCEAGC_LENGTH value
		(15) SYNC_PULSE_MODE	(15) SYNC_PULSE_MODE	0X0	Define whether the SYNC_P pulse is generated as pulse or level
		(13) DPCORR_EN	(13) DPCORR_EN	0X0	Enable the use of delayed DC compensated data path in radio correlator block
		(12) JEF_SELECT	(12) JEF_SELECT	0X0	Selects Jitter Elimination FIFO
		(8:4) XRFSEL	(8:4) XRFSEL	0X0	Extended radio selection field
		(3:0) SUBVERSION	(3:0) SUBVERSION	0X0	CSEM RF Sub-version selection
0x40001578	BB_RADIOCNTL2	(15:0) FREQTABLE_PTR	(15:0) FREQTABLE_PTR	0X40	Frequency table pointer
0x40001580	BB_RADIOPWRUPDN0	(23:16) RXPWRUP0	(23:16) RXPWRUP0	0X0	This register holds the length in us of the RX power up phase for the current radio device
		(12:8) TXPWRDN0	(12:8) TXPWRDN0	0X0	This register extends the length in us of the TX power down phase for the current radio device
		(7:0) TXPWRUP0	(7:0) TXPWRUP0	0X0	This register holds the length in us of the TX power up phase for the current radio device
0x40001584	BB_RADIOPWRUPDN1	(23:16) RXPWRUP1	(23:16) RXPWRUP1	0X0	This register holds the length in us of the RX power up phase for the current radio device
		(12:8) TXPWRDN1	(12:8) TXPWRDN1	0X0	This register extends the length in us of the TX power down phase for the current radio device
		(7:0) TXPWRUP1	(7:0) TXPWRUP1	0X0	This register holds the length in us of the TX power up phase for the current radio device
0x40001590	BB_RADIOTXRXTIM0	(31:24) TXPATHDLY0	(31:24) TXPATHDLY0	0X0	
		(20:16) RXPATHDLY0	(20:16) RXPATHDLY0	0X0	
		(14:8) RFRXTMDA0	(14:8) RFRXTMDA0	0X0	
		(6:0) SYNC_POSITION0	(6:0) SYNC_POSITION0	0X0	

Address	Register Name	Register Write	Register Read	Default	Description
0x40001594	BB_RADIOTXRXTIM1	(31:24) TXPATHDLY1	(31:24) TXPATHDLY1	0X0	
		(20:16) RXPATHDLY1	(20:16) RXPATHDLY1	0X0	
		(14:8) RFRXTMDA1	(14:8) RFRXTMDA1	0X0	
		(6:0) SYNC_POSITION1	(6:0) SYNC_POSITION1	0X0	
0x400015A0	BB_SPIPTRCNTL0	(31:16) TXOFFPTR	(31:16) TXOFFPTR	0X0	Pointer to the TxOFF sequence address section
		(15:0) TXONPTR	(15:0) TXONPTR	0X0	Pointer to the TxON sequence address section
0x400015A4	BB_SPIPTRCNTL1	(31:16) RXOFFPTR	(31:16) RXOFFPTR	0X0	Pointer to the RxOFF sequence address section
		(15:0) RXONPTR	(15:0) RXONPTR	0X0	Pointer to the RxON sequence address section
0x400015A8	BB_SPIPTRCNTL2	(15:0) RSSIPTR	(15:0) RSSIPTR	0X0	Pointer to the RSSI read sequence address section
0x400015B0	BB_ADVCHMAP	(2:0) ADVCHMAP	(2:0) ADVCHMAP	0X7	Advertising channel map, defined as per the advertising connection settings. Contains advertising channels index 37 to 39
0x400015C0	BB_ADVTIM	(13:0) ADVINT	(13:0) ADVINT	0X0	Advertising packet interval defines the time interval in between two ADV_xxx packet sent (value in us)
0x400015C4	BB_ACTSCANSTAT	-	(24:16) BACKOFF	0X1	Active scan mode back-off counter initialization value
		-	(8:0) UPPERLIMIT	0X1	Active scan mode upper limit counter value
0x400015D0	BB_WLPUBADDPTR	(15:0) WLPUBADDPTR	(15:0) WLPUBADDPTR	0X0	Start address pointer of the public devices white list
0x400015D4	BB_WLPRIVADDPTR	(15:0) WLPRIVADDPTR	(15:0) WLPRIVADDPTR	0X0	Start address pointer of the private devices white list
0x400015D8	BB_WLNBDEV	(15:8) NBPRIVDEV	(15:8) NBPRIVDEV	0X0	Number of private devices in the white list
		(7:0) NBPUBDEV	(7:0) NBPUBDEV	0X0	Number of public devices in the white list
0x400015E0	BB_AESCNTL	(1) AES_MODE	(1) AES_MODE	0X0	Cipher mode control
		(0) AES_START	(0) AES_START	0X0	Starts AES-128 ciphering/deciphering process
0x400015E4	BB_AESKEY31_0	(31:0) AESKEY31_0	(31:0) AESKEY31_0	0X0	AES encryption 128-bit key (bits 31 down to 0)
0x400015E8	BB_AESKEY63_32	(31:0) AESKEY63_32	(31:0) AESKEY63_32	0X0	AES encryption 128-bit key (bits 63 down to 32)
0x400015EC	BB_AESKEY95_64	(31:0) AESKEY95_64	(31:0) AESKEY95_64	0X0	AES encryption 128-bit key (bits 95 down to 64)
0x400015F0	BB_AESKEY127_96	(31:0) AESKEY127_96	(31:0) AESKEY127_96	0X0	AES encryption 128-bit key (bits 127 down to 96)
0x400015F4	BB_AESPTR	(15:0) AESPTR	(15:0) AESPTR	0X0	Pointer to the memory zone where the block to cipher/decipher using AES-128 is stored.
0x400015F8	BB_TXMICVAL	-	(31:0) TXMICVAL	0X0	AES-CCM plain MIC value. Valid on when MIC has been calculated (in Tx)
0x400015FC	BB_RXMICVAL	-	(31:0) RXMICVAL	0X0	AES-CCM plain MIC value. Valid on once MIC has been extracted from Rx packet

Address	Register Name	Register Write	Register Read	Default	Description
0x40001600	BB_RFTESTCNTL	(31) INFINITERX	(31) INFINITERX	0X0	Applicable in RF test mode only
		(27) RXPKTCNTEN	(27) RXPKTCNTEN	0X0	Applicable in RF test mode only
		(15) INFINITETX	(15) INFINITETX	0X0	Applicable in RF test mode only
		(14) TXLENGTHSRC	(14) TXLENGTHSRC	0X0	Applicable only in Tx/Rx RF test mode
		(13) PRBSTYPE	(13) PRBSTYPE	0X0	Applicable only in Tx/Rx RF test mode
		(12) TXPLDSRC	(12) TXPLDSRC	0X0	Applicable only in Tx/Rx RF test mode
		(11) TXPKTCNTEN	(11) TXPKTCNTEN	0X1	Applicable in RF test mode only
		(8:0) TXLENGTH	(8:0) TXLENGTH	0X0	Tx packet length in number of byte
0x40001604	BB_RFTESTTXSTAT	(31:0) TXPKTCNT	(31:0) TXPKTCNT	0X0	Reports number of transmitted packet during test modes
0x40001608	BB_RFTESTRXSTAT	(31:0) RXPKTCNT	(31:0) RXPKTCNT	0X0	Reports number of correctly received packet during test modes
0x40001610	BB_TIMGENCNTL	(31) APFM_EN	(31) APFM_EN	0X1	Controls the anticipated pre-fetch abort mechanism
		(25:16) PREFETCHABORT_TIME	(25:16) PREFETCHABORT_TIME	0X1FE	Defines the instant in us at which immediate abort is required after anticipated pre-fetch abort
		(8:0) PREFETCH_TIME	(8:0) PREFETCH_TIME	0X96	Defines exchange table pre-fetch instant in us
0x40001614	BB_GROSSTIMTGT	(22:0) GROSSTARGET	(22:0) GROSSTARGET	0X0	Gross timer target value on which a ble_grosstgtim_irq must be generated (precision of 10ms)
0x40001618	BB_FINETIMTGT	(26:0) FINETARGET	(26:0) FINETARGET	0X0	Fine timer target value on which a ble_finetgtim_irq must be generated (precision of 625us)
0x40001620	BB_COEXIFCNTL0	(25:24) MWSSCANFREQMSK	(25:24) MWSSCANFREQMSK	0X0	Determines how mws_scan_frequency impacts BLE Tx and Rx
		(21:20) WLCRXPRIOMODE	(21:20) WLCRXPRIOMODE	0X0	Defines BLE packet ble_rx mode behavior
		(17:16) WLCTXPRIOMODE	(17:16) WLCTXPRIOMODE	0X0	Defines BLE packet ble_tx mode behavior
		(15:14) MWSTXFREQMSK	(15:14) MWSTXFREQMSK	0X0	Determines how MWS Tx Frequency impacts BLE Tx and Rx
		(13:12) MWSRXFREQMSK	(13:12) MWSRXFREQMSK	0X1	Determines how MWS Rx Frequency impacts BLE Tx and Rx
		(11:10) MWSTXMSK	(11:10) MWSTXMSK	0X0	Determines how mws_tx impacts BLE Tx and Rx
		(9:8) MWSRXMSK	(9:8) MWSRXMSK	0X1	Determines how mws_rx impacts BLE Tx and Rx
		(7:6) TXMSK	(7:6) TXMSK	0X0	Determines how TXx impact BLE Tx and Rx
		(5:4) RXMSK	(5:4) RXMSK	0X1	Determines how RXx impact BLE Tx and Rx
		(3) MWSWCI_EN	(3) MWSWCI_EN	0X0	Enable / Disable control of the WCI MWS Coexistence interface / Valid in Dual Mode only
		(2) MWSCOEX_EN	(2) MWSCOEX_EN	0X0	Enable / Disable control of the MWS Coexistence control / Valid in Dual Mode only
		(1) SYNCGEN_EN	(1) SYNCGEN_EN	0X0	Determines whether ble_sync is generated or not
		(0) COEX_EN	(0) COEX_EN	0X0	Enable / disable control of the coexistence control

Address	Register Name	Register Write	Register Read	Default	Description
0x40001624	BB_COEXIFCNTL1	(28:24) WLCPRXTHR	(28:24) WLCPRXTHR	0X0	Determines the threshold for Rx priority setting (applies on ble_rx if WLCRXPRIOMODE equals "10")
		(20:16) WLCPTXTHR	(20:16) WLCPTXTHR	0X0	Determines the threshold for priority setting (applies on ble_tx if WLCTXPRIOMODE equals "10")
		(14:8) WLCPDURATION	(14:8) WLCPDURATION	0X0	Determines how many us the priority information must be maintained (applies on ble_tx and ble_rx if WLCTXPRIOMODE equals "10")
		(6:0) WLCPDELAY	(6:0) WLCPDELAY	0X0	Determines the delay (in us) in Tx/Rx enables rises the time BLE Tx/Rx priority has to be provided (applies on ble_tx and ble_rx if WLCTXPRIOMODE equals "10")
0x40001628	BB_COEXIFCNTL2	(11:8) RX_ANT_DELAY	(11:8) RX_ANT_DELAY	0X0	Time (in us) by which is anticipated bt_rx to be provided before effective Radio receipt operation
		(3:0) TX_ANT_DELAY	(3:0) TX_ANT_DELAY	0X0	Time (in us) by which is anticipated bt_tx to be provided before effective Radio transmit operation
0x4000162C	BB_BBMPRIO0	(31:28) BLEM7	(31:28) BLEM7	0X3	Set priority value for passive scanning
		(27:24) BLEM6	(27:24) BLEM6	0X4	Set priority value for non-connectable advertising
		(23:20) BLEM5	(23:20) BLEM5	0X8	Set priority value for connectable advertising BLE message
		(19:16) BLEM4	(19:16) BLEM4	0X9	Set priority value for active scanning BLE message
		(15:12) BLEM3	(15:12) BLEM3	0XA	Set priority value for initiating (scanning) BLE message
		(11:8) BLEM2	(11:8) BLEM2	0XD	Set priority value for data channel transmission BLE message
		(7:4) BLEM1	(7:4) BLEM1	0XE	Set priority value for LLCP BLE message
		(3:0) BLEM0	(3:0) BLEM0	0XF	Set priority value for initiating (connection request response) BLE message
0x40001630	BB_BBMPRIO1	(31:28) BLEMDEFAULT	(31:28) BLEMDEFAULT	0X3	Set default priority value for other BLE message than those defined above
		(7:4) BLEM9	(7:4) BLEM9	0XD	Set default priority value for ISO Channel first Tx/Rx attempt
		(3:0) BLEM8	(3:0) BLEM8	0XC	Set default priority value for ISO Channel subsequent Tx/Rx attempt
0x40001640	BB_RALPTR	(15:0) RALPTR	(15:0) RALPTR	0X0	Start address pointer of the RAL structure
0x40001644	BB_RALNBDEV	(7:0) RALNBDEV	(7:0) RALNBDEV	0X0	Number of devices in RAL Structure
0x40001648	BB_RAL_LOCAL_RND	(31) LRND_INIT	(31) LRND_INIT	0X0	Writing a 1 initializes of local RPA random number generation LFSR
		(21:0) LRND_VAL	(21:0) LRND_VAL	0X3F0F0F	Initialization value for local RPA random generation when LRDN_INIT is set to 1, else reports the current Local RPA random number LFSR value

Address	Register Name	Register Write	Register Read	Default	Description
0x4000164C	BB_RAL_PEER_RND	(31) PRND_INIT	(31) PRND_INIT	0X0	Writing a 1 initializes of peer RPA random number generation LFSR
		(21:0) PRND_VAL	(21:0) PRND_VAL	0X30F0F0	Initialization value for peer RPA random generation when LRDN_INIT is set to 1, else reports the current Local RPA random number LFSR value
0x40001650	BB_ISOCHANCNTL0	(4) RETXACKEN0	(4) RETXACKEN0	0X0	Generate Tx ACK
		(3) SYNCGEN0	(3) SYNCGEN0	0X0	Enable audio syn_p generation
		(2) ISOCHANEN0	(2) ISOCHANEN0	0X0	Enable ISO channel
		(1:0) ISOTYPE0	(1:0) ISOTYPE0	0X0	ISO Channel Type
0x40001654	BB_ISOMUTECNTL0	(31) TOGO0	(31) TOGO0	0X0	Indicates which buffer is in use (direct copy of ET-ISOBUFSEL)
		(19) MUTE_SINK0	(19) MUTE_SINK0	0X0	HW mute control
		(18) MUTE_SOURCE0	(18) MUTE_SOURCE0	0X0	HW mute control
		(17) INVL0_1	(17) INVL0_1	0X1	SW mute status for ISO buffer 1 (i.e updated when ET-ISOBUFSEL = 0)
		(16) INVL0_0	(16) INVL0_0	0X1	SW mute status for ISO buffer 0 (i.e updated when ET-ISOBUFSEL = 1)
		(7:0) MUTE_PATTERN0	(7:0) MUTE_PATTERN0	0X0	Value of the ISO channel 0 Mute Pattern to be used when HW muting is enabled
0x40001658	BB_ISOCURRENTTXPTR0	(31:16) ISO0TXPTR0	(31:16) ISO0TXPTR0	0X0	Tx ISO Buffer pointer 0 of ISO Channel 0
		(15:0) ISO0TXPTR1	(15:0) ISO0TXPTR1	0X0	Tx ISO Buffer pointer 1 of ISO Channel 0
0x4000165C	BB_ISOCURRENTRXPTR0	(31:16) ISO0RXPTR0	(31:16) ISO0RXPTR0	0X0	Rx ISO Buffer pointer 0 of ISO Channel 0
		(15:0) ISO0RXPTR1	(15:0) ISO0RXPTR1	0X0	Rx ISO Buffer pointer 1 of ISO Channel 0
0x40001660	BB_ISOTRCNL0	(23:16) ISO0RXLEN	(23:16) ISO0RXLEN	0X0	Negotiated, maximum expected number of bytes for ISO Channel 0 Rx payloads
		(7:0) ISO0TXLEN	(7:0) ISO0TXLEN	0X0	Negotiated, number of bytes for ISO Channel 0 Tx payloads
0x40001664	BB_ISOEVTCNTLOFFSETL0	(31:0) EVT_CNT_OFFSETL0	(31:0) EVT_CNT_OFFSETL0	0X0	LSB part of EVT_CNT_OFFSET0[39:0] field
0x40001668	BB_ISOEVTCNTLOFFSETU0	(6:0) EVT_CNT_OFFSETU0	(6:0) EVT_CNT_OFFSETU0	0X0	MSB part of EVT_CNT_OFFSET0[39:0] field
0x40001670	BB_ISOCHANCNTL1	(4) RETXACKEN1	(4) RETXACKEN1	0X0	Generate Tx ACK
		(3) SYNCGEN1	(3) SYNCGEN1	0X0	Enable audio syn_p generation
		(2) ISOCHANEN1	(2) ISOCHANEN1	0X0	Enable ISO channel
		(1:0) ISOTYPE1	(1:0) ISOTYPE1	0X0	ISO Channel Type

Address	Register Name	Register Write	Register Read	Default	Description
0x40001674	BB_ISOMUTECNTL1	(31) TOGO1	(31) TOGO1	0X0	Indicates which buffer is in use (direct copy of ET-ISOBUFSEL)
		(19) MUTE_SINK1	(19) MUTE_SINK1	0X0	HW mute control
		(18) MUTE_SOURCE1	(18) MUTE_SOURCE1	0X0	HW mute control
		(17) INVL1_1	(17) INVL1_1	0X1	SW mute status for ISO buffer 1 (i.e updated when ET-ISOBUFSEL = 0)
		(16) INVL1_0	(16) INVL1_0	0X1	SW mute status for ISO buffer 0 (i.e updated when ET-ISOBUFSEL = 1)
		(7:0) MUTE_PATTERN1	(7:0) MUTE_PATTERN1	0X0	Value of the ISO channel 0 Mute Pattern to be used when HW muting is enabled
0x40001678	BB_ISOCURRENTTXPTR1	(31:16) ISO1TXPTR0	(31:16) ISO1TXPTR0	0X0	Tx ISO Buffer pointer 0 of ISO Channel 1
		(15:0) ISO1TXPTR1	(15:0) ISO1TXPTR1	0X0	Tx ISO Buffer pointer 1 of ISO Channel 1
0x4000167C	BB_ISOCURRENTRXPTR1	(31:16) ISO1RXPTR0	(31:16) ISO1RXPTR0	0X0	Rx ISO Buffer pointer 0 of ISO Channel 1
		(15:0) ISO1RXPTR1	(15:0) ISO1RXPTR1	0X0	Rx ISO Buffer pointer 1 of ISO Channel 1
0x40001680	BB_ISOTRCNL1	(23:16) ISO1RXLEN	(23:16) ISO1RXLEN	0X0	Negotiated, maximum expected number of bytes for ISO Channel 0 Rx payloads
		(7:0) ISO1TXLEN	(7:0) ISO1TXLEN	0X0	Negotiated, number of bytes for ISO Channel 0 Tx payloads
0x40001684	BB_ISOEVTCNTLOFFSETL1	(31:0) EVT_CNT_OFFSETL1	(31:0) EVT_CNT_OFFSETL1	0X0	LSB part of EVT_CNT_OFFSET0[39:0] field
0x40001688	BB_ISOEVTCNTLOFFSETU1	(6:0) EVT_CNT_OFFSETU1	(6:0) EVT_CNT_OFFSETU1	0X0	MSB part of EVT_CNT_OFFSET0[39:0] field
0x40001690	BB_ISOCHANCNTL2	(4) RETXACKEN2	(4) RETXACKEN2	0X0	Generate Tx ACK
		(3) SYNCGEN2	(3) SYNCGEN2	0X0	Enable audio syn_p generation
		(2) ISOCHANEN2	(2) ISOCHANEN2	0X0	Enable ISO channel
		(1:0) ISOTYPE2	(1:0) ISOTYPE2	0X0	ISO Channel Type
0x40001694	BB_ISOMUTECNTL2	(31) TOGO2	(31) TOGO2	0X0	Indicates which buffer is in use (direct copy of ET-ISOBUFSEL)
		(19) MUTE_SINK2	(19) MUTE_SINK2	0X0	HW mute control
		(18) MUTE_SOURCE2	(18) MUTE_SOURCE2	0X0	HW mute control
		(17) INVL2_1	(17) INVL2_1	0X1	SW mute status for ISO buffer 1 (i.e updated when ET-ISOBUFSEL = 0)
		(16) INVL2_0	(16) INVL2_0	0X1	SW mute status for ISO buffer 0 (i.e updated when ET-ISOBUFSEL = 1)
		(7:0) MUTE_PATTERN2	(7:0) MUTE_PATTERN2	0X0	Value of the ISO channel 0 Mute Pattern to be used when HW muting is enabled
0x40001698	BB_ISOCURRENTTXPTR2	(31:16) ISO2TXPTR0	(31:16) ISO2TXPTR0	0X0	Tx ISO Buffer pointer 0 of ISO Channel 2
		(15:0) ISO2TXPTR1	(15:0) ISO2TXPTR1	0X0	Tx ISO Buffer pointer 1 of ISO Channel 2
0x4000169C	BB_ISOCURRENTRXPTR2	(31:16) ISO2RXPTR0	(31:16) ISO2RXPTR0	0X0	Rx ISO Buffer pointer 0 of ISO Channel 2
		(15:0) ISO2RXPTR1	(15:0) ISO2RXPTR1	0X0	Rx ISO Buffer pointer 1 of ISO Channel 2

Address	Register Name	Register Write	Register Read	Default	Description
0x400016A0	BB_ISOTRCNL2	(23:16) ISO2RXLEN	(23:16) ISO2RXLEN	0X0	Negotiated, maximum expected number of bytes for ISO Channel 2 Rx payloads
		(7:0) ISO2TXLEN	(7:0) ISO2TXLEN	0X0	Negotiated, number of bytes for ISO Channel 2 Tx payloads
0x400016A4	BB_ISOEVTCNTLOFFSETL2	(31:0) EVT_CNT_OFFSETL2	(31:0) EVT_CNT_OFFSETL2	0X0	LSB part of EVT_CNT_OFFSET2[39:0] field
0x400016A8	BB_ISOEVTCNTLOFFSETU2	(6:0) EVT_CNT_OFFSETU2	(6:0) EVT_CNT_OFFSETU2	0X0	MSB part of EVT_CNT_OFFSET2[39:0] field
0x400016B0	BB_BBPRIOSCHARB	(15) BLEPRIOMODE	(15) BLEPRIOMODE	0X0	Determine BLE priority scheduling arbitration mode
		(7:0) BLEMARGIN	(7:0) BLEMARGIN	0X0	Determine the decision instant margin for priority scheduling arbitration

A.24 RF FRONT-END 2.4 GHZ

Address	Register Name	Register Write	Register Read	Default	Description
0x40010000	RF_REG00	(31) DATAWHITE_BTLE_DW_BTLE	(31) DATAWHITE_BTLE_DW_BTLE	0X0	If set to 1, the data whitening specified in the Bluetooth LE standard is used. Note that the en_datawhite field of the CODING register has also to be set to 1
		(30:24) DATAWHITE_BTLE_DW_BTLE_RST	(30:24) DATAWHITE_BTLE_DW_BTLE_RST	0X0	Reset value to put on the Bluetooth LE data whitening shift register
		(23) FOURFSK_CODING_EN_FOURFSK_CODING	(23) FOURFSK_CODING_EN_FOURFSK_CODING	0X0	If set to 1 the 4FSK coding is activated
		(22:20) FOURFSK_CODING_TX_FOURFSK_CODING	(22:20) FOURFSK_CODING_TX_FOURFSK_CODING	0X0	Set the 4FSK coding (Tx): bit 0 determine if the sign is given by the Q signal (0) or I signal (1), bit 1 select if the signal is inverted for the sign, it 2 select if the signal is inverted for the abs amplitude
		(18:16) FOURFSK_CODING_RX_FOURFSK_CODING	(18:16) FOURFSK_CODING_RX_FOURFSK_CODING	0X0	Set the 4FSK decoding (Rx): bit 0 determine if the sign is given by the Q signal (0) or I signal (1), bit 1 select if the signal is inverted for the sign, it 2 select if the signal is inverted for the abs amplitude
		(14) MODE2_DIFF_CODING	(14) MODE2_DIFF_CODING	0X0	If set to 1 enables the differential coding/decoding
		(13) MODE2_PSK_NFSK	(13) MODE2_PSK_NFSK	0X0	If set to 1, the PSK mode is selected, FSK otherwise.
		(12:8) MODE2_TESTMODE	(12:8) MODE2_TESTMODE	0X0	set the output testmode
		(7) MODE_NOT_TO_IDLE	(7) MODE_NOT_TO_IDLE	0X0	In FSM mode, if set to 1 indicates to the FSM to go in suspend mode after a Tx or Rx packet
		(5) MODE_EN_FSM	(5) MODE_EN_FSM	0X0	If set to 1 enables the radio FSM
		(4) MODE_EN_DESERIALIZER	(4) MODE_EN_DESERIALIZER	0X0	If set to 1 enables the deserializer
		(3) MODE_EN_SERIALIZER	(3) MODE_EN_SERIALIZER	0X0	If set to 1 enables the serializer
		(2) MODE_TX_NRX	(2) MODE_TX_NRX	0X0	if set to 1 use the Tx, otherwise the Rx
		(1:0) MODE_MODE	(1:0) MODE_MODE	0X0	Select the working mode of the digital baseband: 00) the digital baseband is off (no ck) 01) the clock is generated but the blocks are reset (Tx,Rx,FIFOs and FSM) 10) 10: the digital baseband is freezed 11) working

Address	Register Name	Register Write	Register Read	Default	Description
0x40010004	RF_REG01	(31:24) TAU_PHASE_RECOV_TAU_PHASE_RECOV	(31:24) TAU_PHASE_RECOV_TAU_PHASE_RECOV	0X0	Time constant of the fine carrier recovery block
		(23:16) TAU_ROUGH_RECOV_TAU_ROUGH_RECOV	(23:16) TAU_ROUGH_RECOV_TAU_ROUGH_RECOV	0X0	Time constant of the rough carrier recovery block
		(15) CARRIER_RECOVERY_EN_CORRECT_CFREQ_AFC	(15) CARRIER_RECOVERY_EN_CORRECT_CFREQ_AFC	0X0	If set to 1, enables the automatic AFC correction.
		(14) CARRIER_RECOVERY_CORRECT_CFREQ_IF_NEG	(14) CARRIER_RECOVERY_CORRECT_CFREQ_IF_NEG	0X0	If set to 1, the IF correction is negative
		(13) CARRIER_RECOVERY_EN_CORRECT_CFREQ_IF	(13) CARRIER_RECOVERY_EN_CORRECT_CFREQ_IF	0X0	If set to 1, enables the automatic IF correction
		(12) CARRIER_RECOVERY_AFC_NEG	(12) CARRIER_RECOVERY_AFC_NEG	0X0	If set to 1 correct the AFC negatively
		(11) CARRIER_RECOVERY_STARTER_MODE	(11) CARRIER_RECOVERY_STARTER_MODE	0X0	If set to 1 enables the starter mode, i.e. a 32x faster carrier recovery.
		(10) CARRIER_RECOVERY_EN_AFC	(10) CARRIER_RECOVERY_EN_AFC	0X0	if set to 1 enables the Automatic Frequency Control
		(9) CARRIER_RECOVERY_EN_FINE_RECOV	(9) CARRIER_RECOVERY_EN_FINE_RECOV	0X0	If set to 1 enables the fine carrier recovery
		(8) CARRIER_RECOVERY_EN_ROUGH_RECOV	(8) CARRIER_RECOVERY_EN_ROUGH_RECOV	0X0	If set to 1 enables the rough carrier recovery
		(6) MOD_TX_PULSE_NSYM	(6) MOD_TX_PULSE_NSYM	0X0	If set to 1, the Tx pulse shape is an odd function.
		(5) MOD_TX_EN_INTERP	(5) MOD_TX_EN_INTERP	0X0	If set to 1, enables the Tx CIC interpolator.
		(4:0) MOD_TX_CK_TX_M	(4:0) MOD_TX_CK_TX_M	0X0	Unsigned value that determine the Tx CIC interpolator frequency. The formula is similar to the evaluation of the oversampling frequency.

Address	Register Name	Register Write	Register Read	Default	Description
0x40010008	RF_REG02	(31) FIFO_FIFO_FLUSH_ON_OVFLW	(31) FIFO_FIFO_FLUSH_ON_OVFLW	0X0	If set to 1, stops the Rx and flushes the FIFO in case of overflow
		(30) FIFO_FIFO_FLUSH_ON_ADDR_ERR	(30) FIFO_FIFO_FLUSH_ON_ADDR_ERR	0X0	If set to 1, stops the Rx and flushes the FIFO in case of address error
		(29) FIFO_FIFO_FLUSH_ON_PL_ERR	(29) FIFO_FIFO_FLUSH_ON_PL_ERR	0X0	If set to 1, stops the Rx and flushes the FIFO in case of packet length error
		(28) FIFO_FIFO_FLUSH_ON_CRC_ERR	(28) FIFO_FIFO_FLUSH_ON_CRC_ERR	0X0	If set to 1, stops the Rx and flushes the FIFO in case of CRC error
		(27) FIFO_RX_FIFO_ACK	(27) FIFO_RX_FIFO_ACK	0X0	If set to 1, the Rx FIFO needs an acknowledgement (packet received correctly) to change its state.
		(26:24) FIFO_FIFO_THR	(26:24) FIFO_FIFO_THR	0X0	Threshold indicating the 'almost full' state
		(23:16) DATARATE_OFFSET_DATARATE_OFFSET	(23:16) DATARATE_OFFSET_DATARATE_OFFSET	0X0	Data-rate offset. Is a signed value and the full scale (0x7f) corresponds to a data-rate offset of 12.5%.
		(15:8) TAU_DATARATE_RECOV_TAU_DATARATE_RECOV	(15:8) TAU_DATARATE_RECOV_TAU_DATARATE_RECOV	0X0	Time constant of the data-rate recovery
		(7:0) TAU_CLK_RECOV_TAU_CLK_RECOV	(7:0) TAU_CLK_RECOV_TAU_CLK_RECOV	0X0	Time constant of the clock recovery
0x4001000C	RF_REG03	(31:28) PAD_CONF_2_PAD_3_CONF	(31:28) PAD_CONF_2_PAD_3_CONF	0X0	Configuration of GPIO pad 3
		(27:24) PAD_CONF_2_PAD_2_CONF	(27:24) PAD_CONF_2_PAD_2_CONF	0X0	Configuration of GPIO pad 2
		(23:20) PAD_CONF_1_PAD_1_CONF	(23:20) PAD_CONF_1_PAD_1_CONF	0X0	Configuration of GPIO pad 1
		(19:16) PAD_CONF_1_PAD_0_CONF	(19:16) PAD_CONF_1_PAD_0_CONF	0X0	Configuration of GPIO pad 0
		(15) IRQ_CONF_IRQ_HIGH_Z	(15) IRQ_CONF_IRQ_HIGH_Z	0X0	If set to 1, the pads are set to High-Z when the IRQ is not active.
		(14) IRQ_CONF_IRQ_ACTIVE_LOW	(14) IRQ_CONF_IRQ_ACTIVE_LOW	0X0	If set to 1, the IRQ are active low
		(13:8) IRQ_CONF_IRQS_MASK	(13:8) IRQ_CONF_IRQS_MASK	0X0	Mask to determine which IRQs are enabled (active high)
		(7:5) FIFO_2_FIFO_THR_TX	(7:5) FIFO_2_FIFO_THR_TX	0X0	Threshold indicating the 'almost empty' state
		(4) FIFO_2_WAIT_TXFIFO_WR	(4) FIFO_2_WAIT_TXFIFO_WR	0X0	If set to 1, the FSM will wait a Tx FIFO write before starting the Tx in case of an empty Tx FIFO.
		(3) FIFO_2_STOP_ON_RXFF_OVFLW	(3) FIFO_2_STOP_ON_RXFF_OVFLW	0X0	If set to 1, stops the reception in case of a FIFO overflow.
		(2) FIFO_2_STOP_ON_TXFF_UNFLW	(2) FIFO_2_STOP_ON_TXFF_UNFLW	0X0	If set to 1, stops the transmission in case of a FIFO underflow.
		(1) FIFO_2_RXFF_FLUSH_ON_START	(1) FIFO_2_RXFF_FLUSH_ON_START	0X0	If set to 1, flushes the Rx FIFO when the Rx is enabled, in order to receive a packet with an empty FIFO.
		(0) FIFO_2_TXFF_FLUSH_ON_STOP	(0) FIFO_2_TXFF_FLUSH_ON_STOP	0X0	If set to 1, flushes the Tx FIFO after the end of a packet transmission in order to have an empty FIFO.

Address	Register Name	Register Write	Register Read	Default	Description
0x40010010	RF_REG04	(31:30) MAC_CONF_MAC_TIMER_GR	(31:30) MAC_CONF_MAC_TIMER_GR	0X0	MAC timer granularity. The granularity is given by (2^(2mac_timer_gr))x1us
		(29) MAC_CONF_RX_MAC_ACT	(29) MAC_CONF_RX_MAC_ACT	0X0	If set to 1, the FSM will switch to Rx or Tx after an Rx mode.
		(28) MAC_CONF_RX_MAC_TX_NRX	(28) MAC_CONF_RX_MAC_TX_NRX	0X0	If set to 1, the FSM will switch to Tx after an Rx mode, Rx otherwise.
		(27) MAC_CONF_RX_MAC_START_NSTOP	(27) MAC_CONF_RX_MAC_START_NSTOP	0X0	If set to 1, the MAC timer is activated at the reception of the sync word, at the end of the packet otherwise.
		(26) MAC_CONF_TX_MAC_ACT	(26) MAC_CONF_TX_MAC_ACT	0X0	If set to 1, the FSM will switch to Rx or Tx after a Tx mode.
		(25) MAC_CONF_TX_MAC_TX_NRX	(25) MAC_CONF_TX_MAC_TX_NRX	0X0	If set to 1, the FSM will switch to Tx after an Tx mode, Rx otherwise.
		(24) MAC_CONF_TX_MAC_START_NSTOP	(24) MAC_CONF_TX_MAC_START_NSTOP	0X0	If set to 1, the MAC timer is activated at beginning of the packet, otherwise at the end of the packet transmission.
		(23:20) PAD_CONF_5_PAD_9_CONF	(23:20) PAD_CONF_5_PAD_9_CONF	0X0	Configuration of GPIO pad 9
		(19:16) PAD_CONF_5_PAD_8_CONF	(19:16) PAD_CONF_5_PAD_8_CONF	0X0	Configuration of GPIO pad 8
		(15:12) PAD_CONF_4_PAD_7_CONF	(15:12) PAD_CONF_4_PAD_7_CONF	0X0	Configuration of GPIO pad 7
		(11:8) PAD_CONF_4_PAD_6_CONF	(11:8) PAD_CONF_4_PAD_6_CONF	0X0	Configuration of GPIO pad 6
		(7:4) PAD_CONF_3_PAD_5_CONF	(7:4) PAD_CONF_3_PAD_5_CONF	0X0	Configuration of GPIO pad 5
		(3:0) PAD_CONF_3_PAD_4_CONF	(3:0) PAD_CONF_3_PAD_4_CONF	0X0	Configuration of GPIO pad 4
0x40010014	RF_REG05	(30) CHANNEL_SWITCH_IQ	(30) CHANNEL_SWITCH_IQ	0X0	Switch I and Q channels
		(29:24) CHANNEL_CHANNEL	(29:24) CHANNEL_CHANNEL	0X0	Channel number
		(18) BANK_DATARATE_TX_NRX	(18) BANK_DATARATE_TX_NRX	0X0	Select the data-rate register: 0-> Rx data-rate, 1-> Tx data-rate
		(17:16) BANK_BANK	(17:16) BANK_BANK	0X0	Select the used bank
		(15:8) TX_MAC_TIMER_TX_MAC_TIMER	(15:8) TX_MAC_TIMER_TX_MAC_TIMER	0X0	Time to wait after the Tx mode.
		(7:0) RX_MAC_TIMER_RX_MAC_TIMER	(7:0) RX_MAC_TIMER_RX_MAC_TIMER	0X0	Time to wait after the Rx mode.
0x40010018	RF_CENTER_FREQ	(31) CENTER_FREQ_ADAPT_CFREQ	(31) CENTER_FREQ_ADAPT_CFREQ	0X0	If set to 1, automatically adapt frequency between Tx and Rx.
		(30) CENTER_FREQ_RX_DIV_5_N6	(30) CENTER_FREQ_RX_DIV_5_N6	0X0	If set to 1, the ratio of the pll reference between Tx and Rx is 5 instead of 6.
		(29:0) CENTER_FREQ_CENTER_FREQUENCY	(29:0) CENTER_FREQ_CENTER_FREQUENCY	0X0	Set the center frequency

Address	Register Name	Register Write	Register Read	Default	Description
0x4001001C	`RF_REG07`	(31:16) `CHANNELS_1_CHANNEL_SPACING_LO`	(31:16) `CHANNELS_1_CHANNEL_SPACING_LO`	0X0	channel spacing: the formula that determines this value is the same as for the central frequency. v=ch_sp/144e6*2^25
		(14) `MOD_INFO_RX_EN_DIV_2_N3_RX`	(14) `MOD_INFO_RX_EN_DIV_2_N3_RX`	0X0	If set to 1 the clock divider will provide a clock divided by 2 instead of 3.
		(13) `MOD_INFO_RX_SYMBOL_2BIT_RX`	(13) `MOD_INFO_RX_SYMBOL_2BIT_RX`	0X0	If set to 1, each symbol is composed by 2 bits (OQPSK or 4FSK)
		(12:8) `MOD_INFO_RX_DR_M_RX`	(12:8) `MOD_INFO_RX_DR_M_RX`	0X0	Unsigned value that determine the oversampling frequency and consequently the data-rate. This frequency is the system frequency (16 or 24 MHz) divided by this value+1.
		(6) `MOD_INFO_TX_EN_DIV_2_N3_TX`	(6) `MOD_INFO_TX_EN_DIV_2_N3_TX`	0X0	If set to 1 the clock divider will provide a clock divided by 2 instead of 3.
		(5) `MOD_INFO_TX_SYMBOL_2BIT_TX`	(5) `MOD_INFO_TX_SYMBOL_2BIT_TX`	0X0	If set to 1, each symbol is composed by 2 bits (OQPSK or 4FSK)
		(4:0) `MOD_INFO_TX_DR_M_TX`	(4:0) `MOD_INFO_TX_DR_M_TX`	0X0	Unsigned value that determine the oversampling frequency and consequently the data-rate. This frequency is the system frequency (16 or 24 MHz) divided by this value+1.

Address	Register Name	Register Write	Register Read	Default	Description
0x40010020	RF_REG08	(31:24) PACKET_LENGTH_PACKET_LEN	(31:24) PACKET_LENGTH_PACKET_LEN	0XFF	The packet length in the fixed packet length mode. In the variable packet length mode, it specifies the maximal packet length defined by the standard. In case of error a packet_len_err is raised.
		(23) PACKET_HANDLING_LSB_FIRST	(23) PACKET_HANDLING_LSB_FIRST	0X0	If set to 1, the LSB is the first bit to be sent, the MSB otherwise
		(22) PACKET_HANDLING_EN_CRC	(22) PACKET_HANDLING_EN_CRC	0X0	If set to 1, enables the automatic CRC evaluation and insertion
		(21) PACKET_HANDLING_EN_CRC_ON_PKT LEN	(21) PACKET_HANDLING_EN_CRC_ON_PKT LEN	0X0	If set to 1, enables the CRC calculation on the packet length part of the packet.
		(20) PACKET_HANDLING_EN_PREAMBLE	(20) PACKET_HANDLING_EN_PREAMBLE	0X0	If set to 1, enables the automatic preamble insertion
		(19) PACKET_HANDLING_EN_MULTI_FRAM E	(19) PACKET_HANDLING_EN_MULTI_FRAM E	0X0	If set to 1, enables the multi-frame packet (preamble-pattern-data-CRC-data-CRC-...)
		(18) PACKET_HANDLING_ENB_DW_ON_CRC	(18) PACKET_HANDLING_ENB_DW_ON_CRC	0X0	Enables the data-whitening on the CRC (active low)
		(17) PACKET_HANDLING_EN_PATTERN	(17) PACKET_HANDLING_EN_PATTERN	0X0	If set to 1, enables the automatic pattern insertion and recognition
		(16) PACKET_HANDLING_EN_PACKET	(16) PACKET_HANDLING_EN_PACKET	0X0	If set to 1 enables the packet handler
		(15) CODING_EN_DATAWHITE	(15) CODING_EN_DATAWHITE	0X0	If set to 1 enables the data-whitening
		(14) CODING_I_NQ_DELAYED	(14) CODING_I_NQ_DELAYED	0X0	If set to 1, the channel I is considered 'delayed' in case of a 2bit per symbol modulaton
		(13) CODING_OFFSET	(13) CODING_OFFSET	0X0	If set to 1, an offset (delay) is introduced in one of the two channels (2 bits per symbol modulation).
		(12) CODING_BIT_INVERT	(12) CODING_BIT_INVERT	0X0	If set to 1, it inverts the bit value (Tx and Rx)
		(11) CODING_EVEN_BEFORE_ODD	(11) CODING_EVEN_BEFORE_ODD	0X0	Determines the bit order in case of a 2 bits per symbol modulation: if set to 1 the first bit (bit 0, even) goes to the I path
		(10) CODING_EN_802154_L2F	(10) CODING_EN_802154_L2F	0X0	If set to 1 enables the linear to frequency encoding needed in order to modulate an OQPSK as an MSK.
		(9) CODING_EN_802154_B2C	(9) CODING_EN_802154_B2C	0X0	If set to 1 enables the bit to chips encoding used in the IEEE 802.15.4 standard
		(8) CODING_EN_MANCHESTER	(8) CODING_EN_MANCHESTER	0X0	If set to 1 enables the Manchester encoding
		(7) CHANNELS_2_EN_CHANNEL_SEL	(7) CHANNELS_2_EN_CHANNEL_SEL	0X0	If set to 1 enables the definition of channels
		(3:0) CHANNELS_2_CHANNEL_SPACING_HI	(3:0) CHANNELS_2_CHANNEL_SPACING_HI	0X0	channel spacing: the formula that determines this value is the same as for the central frequency. v=ch_sp/144e6*2^25

Address	Register Name	Register Write	Register Read	Default	Description
0x40010024	RF_REG09	(27) ADDRESS_CONF_ADDRESS_LEN	(27) ADDRESS_CONF_ADDRESS_LEN	0X0	If set to 1 the address length is 16 bits, 8 otherwise.
		(26) ADDRESS_CONF_EN_ADDRESS_RX_BR	(26) ADDRESS_CONF_EN_ADDRESS_RX_BR	0X0	If set to 1 enables the broadcast address detection on Rx.
		(25) ADDRESS_CONF_EN_ADDRESS_RX	(25) ADDRESS_CONF_EN_ADDRESS_RX	0X0	If set to 1 enables the address detection on Rx
		(24) ADDRESS_CONF_EN_ADDRESS_TX	(24) ADDRESS_CONF_EN_ADDRESS_TX	0X0	If set to 1 enables the address insertion on Tx
		(23:16) PREAMBLE_LENGTH_PREAMBLE_LEN	(23:16) PREAMBLE_LENGTH_PREAMBLE_LEN	0X0	Length of the preamble -1
		(15:8) PREAMBLE_PREAMBLE	(15:8) PREAMBLE_PREAMBLE	0X0	Preamble to be inserted
		(6) PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX	(6) PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX	0X0	If set to 1, the packet length is fixed and specified in the PACKET_LEN register
		(5:2) PACKET_LENGTH_OPTS_PACKET_LEN_CORR	(5:2) PACKET_LENGTH_OPTS_PACKET_LEN_CORR	0X0	Signed value that specifies the correction to apply to the specified packet length (due to differences between standards). The packet length here is specified by the byte number after the packet length byte, with the exclusion of the CRC.
		(1:0) PACKET_LENGTH_OPTS_PACKET_LEN_POS	(1:0) PACKET_LENGTH_OPTS_PACKET_LEN_POS	0X0	Unsigned value that specifies the position of the packet length after the pattern
0x40010028	RF_REG0A	(31:16) ADDRESS_BROADCAST_ADDRESS_BR	(31:16) ADDRESS_BROADCAST_ADDRESS_BR	0X0	Broadcast address
		(15:0) ADDRESS_ADDRESS	(15:0) ADDRESS_ADDRESS	0X0	Address of the node
0x4001002C	RF_SYNC_PATTERN	(31:0) PATTERN	(31:0) PATTERN	0X0	Pattern (sync word) to be inserted or recognized.

Address	Register Name	Register Write	Register Read	Default	Description
0x40010030	RF_REG0C	(30:26) CONV_CODES_POLY_CC_POLY_2	(30:26) CONV_CODES_POLY_CC_POLY_2	0X0	polynom of the third convolutional code
		(25:21) CONV_CODES_POLY_CC_POLY_1	(25:21) CONV_CODES_POLY_CC_POLY_1	0X0	polynom of the second convolutional code
		(20:16) CONV_CODES_POLY_CC_POLY_0	(20:16) CONV_CODES_POLY_CC_POLY_0	0X0	polynom of the first convolutional code
		(11:10) CONV_CODES_CONF_CC_VITERBI_LEN	(11:10) CONV_CODES_CONF_CC_VITERBI_LEN	0X0	Set the memory length of the viterbi decoder: 00 => 5, 01 => 10, 10 => 20, 11 => 30
		(9) CONV_CODES_CONF_CC_EN_TX_STOP	(9) CONV_CODES_CONF_CC_EN_TX_STOP	0X0	if set to 1 enables the stop word at the end of the transmission. Necessary in order to keep a stream coherent with the convolutional coding
		(8) CONV_CODES_CONF_EN_CONV_CODE	(8) CONV_CODES_CONF_EN_CONV_CODE	0X0	If set to 1 enablse the convolutional codes
		(7:6) PACKET_EXTRA_STOP_WORD_LEN	(7:6) PACKET_EXTRA_STOP_WORD_LEN	0X0	length of the stop word, same as the pattern word length
		(5) PACKET_EXTRA_EN_STOP_WORD	(5) PACKET_EXTRA_EN_STOP_WORD	0X0	If set to 1 adds the stop word (0x00) after the CRC
		(4) PACKET_EXTRA_PKT_INFO_PRE_NPOST	(4) PACKET_EXTRA_PKT_INFO_PRE_NPOST	0X0	If set to 1 the packet information are sampled at the end of the packet instead of the sync word detection.
		(3:2) PACKET_EXTRA_PATTERN_MAX_ERR	(3:2) PACKET_EXTRA_PATTERN_MAX_ERR	0X0	unsigned value that specifies the maximum number of errors in the pattern recognition
		(1:0) PACKET_EXTRA_PATTERN_WORD_LEN	(1:0) PACKET_EXTRA_PATTERN_WORD_LEN	0X0	Pattern word length: 00 => 8bits, 01 => 16 bits, 10 => 24 bits, 11 => 32 bits
0x40010034	RF_CRC_POLYNOMIAL	(31:0) CRC_POLYNOMIAL_CRC_POLY	(31:0) CRC_POLYNOMIAL_CRC_POLY	0X0	CRC polynomial. It is coded using the Koopman notation, i.e. the nth bit codes the (n+1) coefficient. Example: x^16+x^12+x^5+1 => 0x8810
0x40010038	RF_CRC_RST	(31:0) CRC_RST_CRC_RST	(31:0) CRC_RST_CRC_RST	0X0	CRC reset value
0x4001003C	RF_REG0F	(31:28) RX_FRAC_CONF_RX_FRAC_DEN	(31:28) RX_FRAC_CONF_RX_FRAC_DEN	0X0	
		(27:24) RX_FRAC_CONF_RX_FRAC_NUM	(27:24) RX_FRAC_CONF_RX_FRAC_NUM	0X0	
		(19) FRAC_CONF_TX_FRAC_GAIN	(19) FRAC_CONF_TX_FRAC_GAIN	0X0	
		(18) FRAC_CONF_RX_FRAC_GAIN	(18) FRAC_CONF_RX_FRAC_GAIN	0X0	
		(17) FRAC_CONF_TX_EN_FRAC	(17) FRAC_CONF_TX_EN_FRAC	0X0	
		(16) FRAC_CONF_RX_EN_FRAC	(16) FRAC_CONF_RX_EN_FRAC	0X0	
		(14:10) CONV_CODES_PUNCT_CC_PUNCT_2	(14:10) CONV_CODES_PUNCT_CC_PUNCT_2	0X0	puncture of the third convolutional code
		(9:5) CONV_CODES_PUNCT_CC_PUNCT_1	(9:5) CONV_CODES_PUNCT_CC_PUNCT_1	0X0	puncture of the second convolutional code
		(4:0) CONV_CODES_PUNCT_CC_PUNCT_0	(4:0) CONV_CODES_PUNCT_CC_PUNCT_0	0X0	puncture of the first convolutional code

Address	Register Name	Register Write	Register Read	Default	Description
0x40010040	RF_REG10	(31:29) FRONTEND2_RESAMPLE_PH_GAIN	(31:29) FRONTEND2_RESAMPLE_PH_GAIN	0X0	Gain of the phase resampling block
		(28:26) FRONTEND2_RESAMPLE_RSSI_G2	(28:26) FRONTEND2_RESAMPLE_RSSI_G2	0X0	Gain of the decimator in the RSSI resampling block
		(25:24) FRONTEND2_RESAMPLE_RSSI_G1	(25:24) FRONTEND2_RESAMPLE_RSSI_G1	0X0	Gain of the interpolator in the RSSI resampling block
		(22) FRONTEND_EN_PHADC_DEGLITCH	(22) FRONTEND_EN_PHADC_DEGLITCH	0X0	If set to 1 enables the phADC deglitcher
		(21) FRONTEND_EN_RESAMPLE_RSSI	(21) FRONTEND_EN_RESAMPLE_RSSI	0X0	If set to 1 enables the RSSI resampling
		(20) FRONTEND_EN_RESAMPLE_PHADC	(20) FRONTEND_EN_RESAMPLE_PHADC	0X0	If set to 1 enables the phase resampling
		(19:16) FRONTEND_DIV_PHADC	(19:16) FRONTEND_DIV_PHADC	0X0	Unsigned value that specifies the divider to obtain the phADC clock (and RSSI).
		(15:12) TX_MULT_TX_MULT_EXP	(15:12) TX_MULT_TX_MULT_EXP	0X0	Exponent of the Tx multiplier
		(11:8) TX_MULT_TX_MULT_MAN	(11:8) TX_MULT_TX_MULT_MAN	0X0	Mantissa of the Tx multiplier
		(7:4) TX_FRAC_CONF_TX_FRAC_DEN	(7:4) TX_FRAC_CONF_TX_FRAC_DEN	0X0	
		(3:0) TX_FRAC_CONF_TX_FRAC_NUM	(3:0) TX_FRAC_CONF_TX_FRAC_NUM	0X0	
0x40010044	RF_TX_PULSE0	(31:24) TX_PULSE_SHAPE_1_TX_COEF4	(31:24) TX_PULSE_SHAPE_1_TX_COEF4	0X0	
		(23:16) TX_PULSE_SHAPE_1_TX_COEF3	(23:16) TX_PULSE_SHAPE_1_TX_COEF3	0X0	
		(15:8) TX_PULSE_SHAPE_1_TX_COEF2	(15:8) TX_PULSE_SHAPE_1_TX_COEF2	0X0	
		(7:0) TX_PULSE_SHAPE_1_TX_COEF1	(7:0) TX_PULSE_SHAPE_1_TX_COEF1	0X0	These registers specify the Tx pulse shape. The pulse shape is formed by: coef1-coef16-coef16-coef1. Since the oversampling ratio is 8, the pulse shape is 4 symbols long. Every coefficient is an 8 bits signed.
0x40010048	RF_TX_PULSE1	(31:24) TX_PULSE_SHAPE_2_TX_COEF8	(31:24) TX_PULSE_SHAPE_2_TX_COEF8	0X0	
		(23:16) TX_PULSE_SHAPE_2_TX_COEF7	(23:16) TX_PULSE_SHAPE_2_TX_COEF7	0X0	
		(15:8) TX_PULSE_SHAPE_2_TX_COEF6	(15:8) TX_PULSE_SHAPE_2_TX_COEF6	0X0	
		(7:0) TX_PULSE_SHAPE_2_TX_COEF5	(7:0) TX_PULSE_SHAPE_2_TX_COEF5	0X0	
0x4001004C	RF_TX_PULSE2	(31:24) TX_PULSE_SHAPE_3_TX_COEF12	(31:24) TX_PULSE_SHAPE_3_TX_COEF12	0X0	
		(23:16) TX_PULSE_SHAPE_3_TX_COEF11	(23:16) TX_PULSE_SHAPE_3_TX_COEF11	0X0	
		(15:8) TX_PULSE_SHAPE_3_TX_COEF10	(15:8) TX_PULSE_SHAPE_3_TX_COEF10	0X0	
		(7:0) TX_PULSE_SHAPE_3_TX_COEF9	(7:0) TX_PULSE_SHAPE_3_TX_COEF9	0X0	

Address	Register Name	Register Write	Register Read	Default	Description
0x40010050	RF_TX_PULSE3	(31:24) TX_PULSE_SHAPE_4_TX_COEF16	(31:24) TX_PULSE_SHAPE_4_TX_COEF16	0X0	
		(23:16) TX_PULSE_SHAPE_4_TX_COEF15	(23:16) TX_PULSE_SHAPE_4_TX_COEF15	0X0	
		(15:8) TX_PULSE_SHAPE_4_TX_COEF14	(15:8) TX_PULSE_SHAPE_4_TX_COEF14	0X0	
		(7:0) TX_PULSE_SHAPE_4_TX_COEF13	(7:0) TX_PULSE_SHAPE_4_TX_COEF13	0X0	
0x40010054	RF_RX_PULSE	(31:28) RX_PULSE_SHAPE_RX_COEF8	(31:28) RX_PULSE_SHAPE_RX_COEF8	0X0	
		(27:24) RX_PULSE_SHAPE_RX_COEF7	(27:24) RX_PULSE_SHAPE_RX_COEF7	0X0	
		(23:20) RX_PULSE_SHAPE_RX_COEF6	(23:20) RX_PULSE_SHAPE_RX_COEF6	0X0	
		(19:16) RX_PULSE_SHAPE_RX_COEF5	(19:16) RX_PULSE_SHAPE_RX_COEF5	0X0	
		(15:12) RX_PULSE_SHAPE_RX_COEF4	(15:12) RX_PULSE_SHAPE_RX_COEF4	0X0	
		(11:8) RX_PULSE_SHAPE_RX_COEF3	(11:8) RX_PULSE_SHAPE_RX_COEF3	0X0	
		(7:4) RX_PULSE_SHAPE_RX_COEF2	(7:4) RX_PULSE_SHAPE_RX_COEF2	0X0	
		(3:0) RX_PULSE_SHAPE_RX_COEF1	(3:0) RX_PULSE_SHAPE_RX_COEF1	0X0	These registers specify the Rx pulse shape. The pulse shape is formed by: coef1-coef8-coef8-coef1. Since the oversampling ratio is 8, the pulse shape is 2 symbols long. Coefficients from coef4 to coef8 are unsigned, while coef1 to coef3 are signed.
0x40010058	RF_REG16	(28:25) RX_IF_RESAMPLE_PH_IF	(28:25) RX_IF_RESAMPLE_PH_IF	0X0	IF value for the phase resampler.
		(24:16) RX_IF_IF2_CLK_OS	(24:16) RX_IF_IF2_CLK_OS	0X0	IF value for the carrier recovery
		(15:8) FSK_FCR_AMP_1_FSK_FCR_AMP1	(15:8) FSK_FCR_AMP_1_FSK_FCR_AMP1	0X0	FSK amplitude 1 (lowest): in FSK w/o ISI is used to specify the expected amplitude. In 4FSK is the lowest amplitude (+/-1). in FSK w/ ISI it specify the lowest amplitude (generally it corresponds to a sequence 0-1-0.
		(7:6) FILTER_GAIN_DR_LIMIT	(7:6) FILTER_GAIN_DR_LIMIT	0X0	Set the data-rate recovery limits: 00 => 0%, 01 => 3.125 %, 10 => 6.25 %, 11 => 12.5%
		(5:3) FILTER_GAIN_FILTER_GAIN_M	(5:3) FILTER_GAIN_FILTER_GAIN_M	0X0	Mantissa of the final stage gain of the matched filter
		(2:0) FILTER_GAIN_FILTER_GAIN_E	(2:0) FILTER_GAIN_FILTER_GAIN_E	0X0	Exponent of the final stage gain of the matched filter

Address	Register Name	Register Write	Register Read	Default	Description
0x4001005C	RF_REG17	(31:24) FSK_FCR_AMP_3_FSK_FCR_AMP3	(31:24) FSK_FCR_AMP_3_FSK_FCR_AMP3	0X0	FSK amplitude 3 (highest): in 4FSK is the high amplitude (+/-3). in FSK w/ ISI it specify the highet amplitude (generally it corresponds to a sequence 1-1-1.
		(23:16) FSK_FCR_AMP_2_FSK_FCR_AMP2	(23:16) FSK_FCR_AMP_2_FSK_FCR_AMP2	0X0	FSK amplitude 2 (mid): in 4FSK is the threshold. in FSK w/ ISI it specify the mid amplitude (generally it corresponds to a sequence 0-1-1 or 1-1-0.
		(14:13) CARRIER_RECOVERY_EXTRA_MAX_ER R_IN_DL_SYNC	(14:13) CARRIER_RECOVERY_EXTRA_MAX_ER R_IN_DL_SYNC	0X0	Set the maximum errors in the delay line sync detection
		(12) CARRIER_RECOVERY_EXTRA_EN_SYN C_OK_DELAY_LINE	(12) CARRIER_RECOVERY_EXTRA_EN_SYN C_OK_DELAY_LINE	0X0	If set to 1 uses the pattern_ok signal in delay line to synchronize the deserializer
		(11:9) CARRIER_RECOVERY_EXTRA_NC_SEL _OUT	(11:9) CARRIER_RECOVERY_EXTRA_NC_SEL _OUT	0X0	Select the output position for the 'not-causal processing': 000 => 4 symbol, 001 => 6 symbols, 010 => 8 symbols, 011 => 12 symbols, 100 => 16 symbols, 101 => 24 symbols, 110 => 32 symbols, 111 => 40 symbols
		(8) CARRIER_RECOVERY_EXTRA_EN_NOT _CAUSAL	(8) CARRIER_RECOVERY_EXTRA_EN_NOT _CAUSAL	0X0	if set to 1 enables the not causal processing
		(6:4) CARRIER_RECOVERY_EXTRA_FREQ_L IMIT_MAN	(6:4) CARRIER_RECOVERY_EXTRA_FREQ_L IMIT_MAN	0X0	Mantissa of the carrier recovery frequency limit (unsigned).
		(2:0) CARRIER_RECOVERY_EXTRA_FREQ_L IMIT_EXP	(2:0) CARRIER_RECOVERY_EXTRA_FREQ_L IMIT_EXP	0X0	Exponent of the carrier recovery frequency limit (signed). Formula: carrier_offset_max=(1+m/8)*2^e/ 4*f_sym

Address	Register Name	Register Write	Register Read	Default	Description
0x40010060	RF_REG18	(31:16) CORRECT_CFREQ_IF_CORRECT_CFREQ_IF	(31:16) CORRECT_CFREQ_IF_CORRECT_CFREQ_IF	0X0	Unsigned value that specifies the IF for the Rx mode.
		(15:14) RSSI_BANK_RSSI_TRI_CK_DIV	(15:14) RSSI_BANK_RSSI_TRI_CK_DIV	0X0	Speed on the RSSI triangular dithering signal (cf reg RSSI_TUN)
		(13) RSSI_BANK_FAST_RSSI	(13) RSSI_BANK_FAST_RSSI	0X0	If set to 1, the RSSI filtering is 8x faster
		(12) RSSI_BANK_EN_FAST_PRE_SYNC	(12) RSSI_BANK_EN_FAST_PRE_SYNC	0X0	If the packet mode is set, indicates to switch the fast modes during the preamble reception
		(11:8) RSSI_BANK_TAU_RSSI_FILTERING	(11:8) RSSI_BANK_TAU_RSSI_FILTERING	0X0	Time constant of the RSSI filtering block: 0: 4symbols, 1: 8symbols, 2: 16 symbols, 3: 32symbols, 4: 64symbols, 5: 128symbols, 6: 256symbols, 7: 512symbols, 8: 1024symbols
		(4) DECISION_USE_VIT_SOFT	(4) DECISION_USE_VIT_SOFT	0X0	If set to 1 uses the viterbi soft decoding
		(3:2) DECISION_VITERBI_LEN	(3:2) DECISION_VITERBI_LEN	0X0	Sets the Viterbi path length: 00: 1 bit, 01: 2 bits, 10: 4 bits, 11: 8 bits
		(1) DECISION_VITERBI_POW_NLIN	(1) DECISION_VITERBI_POW_NLIN	0X0	if set to 1, the Viterbi algorithm uses power instead of amplitude to evaluate the error on the path
		(0) DECISION_EN_VITERBI_GFSK	(0) DECISION_EN_VITERBI_GFSK	0X0	If set to 1 enables the Viterbi algorithm for the GFSK decoding; this will override the old ISI correction algorithm.
0x40010064	RF_REG19	(29:28) PLL_BANK_PLL_FILTER_RES_TRIM_TX	(29:28) PLL_BANK_PLL_FILTER_RES_TRIM_TX	0X0	Same as pll_filter_res_trim but for Tx case. Real value in Tx is pll_filter_res_trim xor pll_filter_res_trim_tx. If set to 0, Tx and Rx have the same value.
		(27:24) PLL_BANK_IQ_PLL_0_TX	(27:24) PLL_BANK_IQ_PLL_0_TX	0X0	Charge pump bias for Tx case. Real value in Tx is iq_pll_0 xor iq_pll_0_tx. If set to 0, Tx and Rx have the same value.
		(22) PLL_BANK_LOW_DR_TX	(22) PLL_BANK_LOW_DR_TX	0X0	If set to 1 the Tx will work in low data-rate mode
		(21:20) PLL_BANK_PLL_FILTER_RES_TRIM	(21:20) PLL_BANK_PLL_FILTER_RES_TRIM	0X0	Allow to modify the value of the loop filter resistor R2 when bit 5 is high (TX mode): 00 => normal resistor (R_2_typ), 01 => 123%, 10 => 130% 11 => 170%
		(19:16) PLL_BANK_IQ_PLL_0	(19:16) PLL_BANK_IQ_PLL_0	0X0	Charge pump bias
		(13) PA_PWR_MIN_PA_PWR	(13) PA_PWR_MIN_PA_PWR	0X0	Sets the minimum power during the PA ramp-up: if 0 the ramp-up starts at -3, if 1 the ramp-up starts at -1
		(12:8) PA_PWR_PA_PWR	(12:8) PA_PWR_PA_PWR	0X0	Signed value that sets the PA power: minimum value is -3 (-40dBm), max value is 12 (3.3dBm).
		(7:4) CLK_CH_FILTER_DIV_RSSI	(7:4) CLK_CH_FILTER_DIV_RSSI	0X0	Unsigned value that specifies the division factor for the clock controlling the RSSI.
		(3:0) CLK_CH_FILTER_DIV_FILT	(3:0) CLK_CH_FILTER_DIV_FILT	0X0	Unsigned value that specifies the division factor for the clock controlling the channel filter.

Address	Register Name	Register Write	Register Read	Default	Description
0x40010068	RF_REG1A	(31:28) ATT_CTRL_ATT_CTRL_MAX	(31:28) ATT_CTRL_ATT_CTRL_MAX	0X0	Maximum attenuation level in AGC algorithm
		(27:24) ATT_CTRL_SET_RX_ATT_CTRL	(27:24) ATT_CTRL_SET_RX_ATT_CTRL	0X0	Attuenuation level if the AGC is bypassed
		(23:22) RSSI_CTRL_AGC_DECAY_TAU	(23:22) RSSI_CTRL_AGC_DECAY_TAU	0X0	Time constant of the decay speed; high values corresponds to a slow decay
		(21) RSSI_CTRL_AGC_USE_LNA	(21) RSSI_CTRL_AGC_USE_LNA	0X0	If set to 1 the AGC algorithm uses the LNA bias.
		(20) RSSI_CTRL_AGC_MODE	(20) RSSI_CTRL_AGC_MODE	0X0	Select the AGC algorithm: 0 -> old algorithm, 1 -> new algorithm
		(19:18) RSSI_CTRL_AGC_WAIT	(19:18) RSSI_CTRL_AGC_WAIT	0X0	Sets the wait time of the AGC after switching between states: 00 => don't wait, 01 => wait 1x RSSI filtering period, 10 => wait 2x RSSI filtering period, 11 => wait 3x RSSI filtering period
		(17) RSSI_CTRL_PAYLOAD_BLOCKS_AGC	(17) RSSI_CTRL_PAYLOAD_BLOCKS_AGC	0X0	If set to 1, the AGC is blocked during the payload
		(16) RSSI_CTRL_BYPASS_AGC	(16) RSSI_CTRL_BYPASS_AGC	0X0	If set to 1, the AGC algorithm is bypassed
		(12:8) FILTER_BIAS_IQ_FI_BW	(12:8) FILTER_BIAS_IQ_FI_BW	0X0	Bias for the bandwidth of the channel filter
		(4:0) FILTER_BIAS_IQ_FI_FC	(4:0) FILTER_BIAS_IQ_FI_FC	0X0	Bias for the central frequency of the channel filter
0x4001006C	RF_REG1B	(31) IEEE802154_OPTS_EN_DW_TEST	(31) IEEE802154_OPTS_EN_DW_TEST	0X0	If set to 1 enables the Tx data-whitening before the convolutional code block
		(30:29) IEEE802154_OPTS_BER_CLK_MODE	(30:29) IEEE802154_OPTS_BER_CLK_MODE	0X0	sets the clock output mode for BER mode or RW mode: 00 => data change on falling edge, 01 => data change on rising edge, 10 => clock signal is a toggled signal, 11 => enable signal from clock recovery
		(28) IEEE802154_OPTS_RX_DATA_NOT_S AMPLED	(28) IEEE802154_OPTS_RX_DATA_NOT_S AMPLED	0X0	If set to 1, the signal rx_data in testmodes is not sampled. Used for debug purposes
		(27) IEEE802154_OPTS_EN_L2F_RX	(27) IEEE802154_OPTS_EN_L2F_RX	0X0	if set to 1 enables the frequency to linear conversion in the Rx side (always controlled by the en_802154_l2f configuration bit).
		(26:24) IEEE802154_OPTS_C2B_THR	(26:24) IEEE802154_OPTS_C2B_THR	0X0	Threshold of the chip2bit correlator of the IEEE 802.15.4 protocol.
		(23:20) AGC_PEAK_DET_PEAK_DET_TAU	(23:20) AGC_PEAK_DET_PEAK_DET_TAU	0X0	Time constant of the peak detector monostable circuit; if set to 0 the monostable is bypassed
		(19:18) AGC_PEAK_DET_PEAK_DET_THR_LOW	(19:18) AGC_PEAK_DET_PEAK_DET_THR_LOW	0X0	Threshold for the low level of the peak detector: 0 => 0, 1 => 1, 2 => 2, 3 => N.A.
		(17) AGC_PEAK_DET_PEAK_DET_THR_HIG H	(17) AGC_PEAK_DET_PEAK_DET_THR_HIG H	0X0	Threshold for the high level of the peak detector: 0 => 2, 1 => 3
		(16) AGC_PEAK_DET_EN_AGC_PEAK	(16) AGC_PEAK_DET_EN_AGC_PEAK	0X0	If set to 1 enables the AGC peak detector
		(15:8) AGC_THR_HIGH_AGC_THR_HIGH	(15:8) AGC_THR_HIGH_AGC_THR_HIGH	0X0	AGC threshold high level
		(7:0) AGC_THR_LOW_AGC_THR_LOW	(7:0) AGC_THR_LOW_AGC_THR_LOW	0X0	AGC threshold low level

Address	Register Name	Register Write	Register Read	Default	Description
0x40010070	RF_AGC_LUT1	(31:22) AGC_LUT_1_AGC_LEVEL_2_LO	(31:22) AGC_LUT_1_AGC_LEVEL_2_LO	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
		(21:11) AGC_LUT_1_AGC_LEVEL_1	(21:11) AGC_LUT_1_AGC_LEVEL_1	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
		(10:0) AGC_LUT_1_AGC_LEVEL_0	(10:0) AGC_LUT_1_AGC_LEVEL_0	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
0x40010074	RF_AGC_LUT2	(31:23) AGC_LUT_2_AGC_LEVEL_5_LO	(31:23) AGC_LUT_2_AGC_LEVEL_5_LO	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
		(22:12) AGC_LUT_2_AGC_LEVEL_4	(22:12) AGC_LUT_2_AGC_LEVEL_4	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
		(11:1) AGC_LUT_2_AGC_LEVEL_3	(11:1) AGC_LUT_2_AGC_LEVEL_3	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
		(0) AGC_LUT_2_AGC_LEVEL_2_HI	(0) AGC_LUT_2_AGC_LEVEL_2_HI	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
0x40010078	RF_AGC_LUT3	(31:24) AGC_LUT_3_AGC_LEVEL_8_LO	(31:24) AGC_LUT_3_AGC_LEVEL_8_LO	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
		(23:13) AGC_LUT_3_AGC_LEVEL_7	(23:13) AGC_LUT_3_AGC_LEVEL_7	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
		(12:2) AGC_LUT_3_AGC_LEVEL_6	(12:2) AGC_LUT_3_AGC_LEVEL_6	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
		(1:0) AGC_LUT_3_AGC_LEVEL_5_HI	(1:0) AGC_LUT_3_AGC_LEVEL_5_HI	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
0x4001007C	RF_AGC_LUT4	(31:25) AGC_LUT_4_AGC_LEVEL_11_LO	(31:25) AGC_LUT_4_AGC_LEVEL_11_LO	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
		(24:14) AGC_LUT_4_AGC_LEVEL_10	(24:14) AGC_LUT_4_AGC_LEVEL_10	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
		(13:3) AGC_LUT_4_AGC_LEVEL_9	(13:3) AGC_LUT_4_AGC_LEVEL_9	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
		(2:0) AGC_LUT_4_AGC_LEVEL_8_HI	(2:0) AGC_LUT_4_AGC_LEVEL_8_HI	0X0	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.

Address	Register Name	Register Write	Register Read	Default	Description
0x40010080	RF_REG20	(31:28) TIMINGS_3_T_DLL	(31:28) TIMINGS_3_T_DLL	0X0	Time needed by the DLL blocks to switch on.
		(27:24) TIMINGS_3_T_PLL_TX	(27:24) TIMINGS_3_T_PLL_TX	0X1	Time needed by the PLL blocks in Tx mode to switch on.
		(23:20) TIMINGS_2_T_SUBBAND_TX	(23:20) TIMINGS_2_T_SUBBAND_TX	0XF	Time needed by the subband algorithm to calibrate in Tx.
		(19:16) TIMINGS_2_T_TX_RF	(19:16) TIMINGS_2_T_TX_RF	0XFF	Time needed by the Tx RF blocks to switch on.
		(14:12) TIMINGS_1_T_GRANULARITY_TX	(14:12) TIMINGS_1_T_GRANULARITY_TX	0X0	Fixes the granularity of the timer in Tx mode. The granularity is given by (2^(t_granularity-2))x1us
		(10:8) TIMINGS_1_T_GRANULARITY_RX	(10:8) TIMINGS_1_T_GRANULARITY_RX	0X1	Fixes the granularity of the timer in Rx mode. The granularity is given by (2^(t_granularity))x1us
		(3:0) AGC_LUT_5_AGC_LEVEL_11_HI	(3:0) AGC_LUT_5_AGC_LEVEL_11_HI	0XF	Look up table with the AGC values: agc_level_0 is supposed the lowest attenuation, while agc_level_11 is the one with a maximum of attenuation.
0x40010084	RF_AGC_ATT1	(31:30) AGC_ATT_1_AGC_ATT_AB_LO	(31:30) AGC_ATT_1_AGC_ATT_AB_LO	0X3	
		(29:27) AGC_ATT_1_AGC_ATT_9A	(29:27) AGC_ATT_1_AGC_ATT_9A	0X3	
		(26:24) AGC_ATT_1_AGC_ATT_89	(26:24) AGC_ATT_1_AGC_ATT_89	0X3	
		(23:21) AGC_ATT_1_AGC_ATT_78	(23:21) AGC_ATT_1_AGC_ATT_78	0X3	
		(20:18) AGC_ATT_1_AGC_ATT_67	(20:18) AGC_ATT_1_AGC_ATT_67	0X3	
		(17:15) AGC_ATT_1_AGC_ATT_56	(17:15) AGC_ATT_1_AGC_ATT_56	0X3	
		(14:12) AGC_ATT_1_AGC_ATT_45	(14:12) AGC_ATT_1_AGC_ATT_45	0X3	
		(11:9) AGC_ATT_1_AGC_ATT_34	(11:9) AGC_ATT_1_AGC_ATT_34	0X3	
		(8:6) AGC_ATT_1_AGC_ATT_23	(8:6) AGC_ATT_1_AGC_ATT_23	0X3	
		(5:3) AGC_ATT_1_AGC_ATT_12	(5:3) AGC_ATT_1_AGC_ATT_12	0X3	
		(2:0) AGC_ATT_1_AGC_ATT_01	(2:0) AGC_ATT_1_AGC_ATT_01	0X3	These fields specify the attenuation levels
0x40010088	RF_REG22	(29) TIMING_FAST_RX_EN_FAST_RX_TXFILT	(29) TIMING_FAST_RX_EN_FAST_RX_TXFILT	0X0	If set to 1 enables filter Tx configuration for the fast Rx PLL
		(28) TIMING_FAST_RX_EN_FAST_RX	(28) TIMING_FAST_RX_EN_FAST_RX	0X0	If set to 1 enables the fast Rx PLL
		(27:24) TIMING_FAST_RX_T_RX_FAST_CHP	(27:24) TIMING_FAST_RX_T_RX_FAST_CHP	0X0	Time to switch off the fast CHP in Rx mode
		(23:20) TIMINGS_5_T_RX_RF	(23:20) TIMINGS_5_T_RX_RF	0X0	Time needed by the Rx RF blocks to switch on.
		(19:16) TIMINGS_5_T_RX_BB	(19:16) TIMINGS_5_T_RX_BB	0X0	Time needed by the Rx BB blocks to switch on.
		(15:12) TIMINGS_4_T_SUBBAND_RX	(15:12) TIMINGS_4_T_SUBBAND_RX	0X0	Time needed by the subband algorithm to calibrate in Rx
		(11:8) TIMINGS_4_T_PLL_RX	(11:8) TIMINGS_4_T_PLL_RX	0X0	Time needed by the PLL blocks in Rx mode to switch on.
		(1) AGC_ATT_2_AGC_ATT_1DB	(1) AGC_ATT_2_AGC_ATT_1DB	0X0	If set to 1 the attenuation are specified by 1dB steps from 4dB to 11dB
		(0) AGC_ATT_2_AGC_ATT_AB_HI	(0) AGC_ATT_2_AGC_ATT_AB_HI	0X0	

Address	Register Name	Register Write	Register Read	Default	Description
0x4001008C	RF_REG23	(31:28) BIAS_1_IQ_RXTX_3	(31:28) BIAS_1_IQ_RXTX_3	0X0	PrePA Casc bias
		(27:24) BIAS_1_IQ_RXTX_2	(27:24) BIAS_1_IQ_RXTX_2	0X0	PrePA In bias
		(23:20) BIAS_0_IQ_RXTX_1	(23:20) BIAS_0_IQ_RXTX_1	0X0	PA backoff bias
		(19:16) BIAS_0_IQ_RXTX_0	(19:16) BIAS_0_IQ_RXTX_0	0X0	PA bias
		(14:12) INTERFACE_CONF_APB_WAIT_STATE	(14:12) INTERFACE_CONF_APB_WAIT_STATE	0X0	Select the number of wait states during the APB transaction
		(9:8) INTERFACE_CONF_SPI_SELECT	(9:8) INTERFACE_CONF_SPI_SELECT	0X0	Select the spi mode: 00 legacy spi, 01 advanced spi, 10 BLIM4SME spi
		(7) TIMEOUT_EN_RX_TIMEOUT	(7) TIMEOUT_EN_RX_TIMEOUT	0X0	If set to 1 enables the timeout of the Rx when the system is on FSM mode
		(6:4) TIMEOUT_T_TIMEOUT_GR	(6:4) TIMEOUT_T_TIMEOUT_GR	0X0	Granularity of the timer in timeout Rx mode
		(3:0) TIMEOUT_T_RX_TIMEOUT	(3:0) TIMEOUT_T_RX_TIMEOUT	0X0	Time that has to occur before the timeout.
0x40010090	RF_REG24	(31:28) BIAS_5_IQ_PLL_4_RX	(31:28) BIAS_5_IQ_PLL_4_RX	0X0	VCO bias for Rx
		(27:24) BIAS_5_IQ_PLL_4_TX	(27:24) BIAS_5_IQ_PLL_4_TX	0X0	VCO bias for Tx
		(23:20) BIAS_4_IQ_PLL_2	(23:20) BIAS_4_IQ_PLL_2	0X0	Sub-band comparator bias
		(19:16) BIAS_4_IQ_PLL_1	(19:16) BIAS_4_IQ_PLL_1	0X0	Dynamic divider bias
		(15:12) BIAS_3_IQ_RXTX_8	(15:12) BIAS_3_IQ_RXTX_8	0X0	IFA ctrl_c bias
		(11:8) BIAS_3_IQ_RXTX_7	(11:8) BIAS_3_IQ_RXTX_7	0X0	IFA ctrl_r bias
		(7:4) BIAS_2_IQ_RXTX_6	(7:4) BIAS_2_IQ_RXTX_6	0X0	VCOM_MX bias
		(3:0) BIAS_2_IQ_RXTX_5	(3:0) BIAS_2_IQ_RXTX_5	0X0	VCOM_LO bias
0x40010094	RF_REG25	(31:28) BIAS_9_IQ_BB_6	(31:28) BIAS_9_IQ_BB_6	0X0	Peak detector threshold bias 0
		(27:24) BIAS_9_IQ_BB_5	(27:24) BIAS_9_IQ_BB_5	0X0	Peak detector bias
		(23:20) BIAS_8_IQ_BB_4	(23:20) BIAS_8_IQ_BB_4	0X0	RSSI_D bias
		(19:16) BIAS_8_IQ_BB_3	(19:16) BIAS_8_IQ_BB_3	0X0	RSSI_G bias
		(15:12) BIAS_7_IQ_BB_2	(15:12) BIAS_7_IQ_BB_2	0X0	ACD_L bias
		(11:8) BIAS_7_IQ_BB_1	(11:8) BIAS_7_IQ_BB_1	0X0	ACD_C bias
		(7:4) BIAS_6_IQ_BB_0	(7:4) BIAS_6_IQ_BB_0	0X0	ACD_O bias
		(3:0) BIAS_6_IQ_PLL_3	(3:0) BIAS_6_IQ_PLL_3	0X0	DLL bias

Address	Register Name	Register Write	Register Read	Default	Description
0x40010098	RF_REG26	(28) SD_MASH_MASH_ENABLE	(28) SD_MASH_MASH_ENABLE	0X0	Enable the sigma delta mash
		(27) SD_MASH_MASH_DITHER	(27) SD_MASH_MASH_DITHER	0X0	Enable dithering on the sigma delta mash
		(26:25) SD_MASH_MASH_ORDER	(26:25) SD_MASH_MASH_ORDER	0X0	Order of the sigma delta mash
		(24) SD_MASH_MASH_RSTB	(24) SD_MASH_MASH_RSTB	0X0	Reset of the sigma delta mash (active low)
		(23:20) BIAS_12_LNA_AGC_BIAS_3	(23:20) BIAS_12_LNA_AGC_BIAS_3	0X0	LNA bias for AGC lvl 3
		(19:16) BIAS_12_LNA_AGC_BIAS_2	(19:16) BIAS_12_LNA_AGC_BIAS_2	0X0	LNA bias for AGC lvl 2
		(15:12) BIAS_11_LNA_AGC_BIAS_1	(15:12) BIAS_11_LNA_AGC_BIAS_1	0X0	LNA bias for AGC lvl 1
		(11:8) BIAS_11_LNA_AGC_BIAS_0	(11:8) BIAS_11_LNA_AGC_BIAS_0	0X0	LNA bias for AGC lvl 0
		(7:4) BIAS_10_IQ_BB_8	(7:4) BIAS_10_IQ_BB_8	0X0	Peak detector threshold bias 1
		(3:0) BIAS_10_IQ_BB_7	(3:0) BIAS_10_IQ_BB_7	0X0	Peak detector threshold bias 2
0x4001009C	RF_REG27	(31) CTRL_ADC_ONE_CK_RSSI_PHADC	(31) CTRL_ADC_ONE_CK_RSSI_PHADC	0X0	If set to 1, the RSSI and the phADC share the same clock
		(30:29) CTRL_ADC_PHADC_DELLATCH	(30:29) CTRL_ADC_PHADC_DELLATCH	0X0	phADC delay latch trimming
		(28:24) CTRL_ADC_CTRL_ADC	(28:24) CTRL_ADC_CTRL_ADC	0X0	bits(1:0) => phADC reset delay, bits(3:2) phADC clock delay, bit(4) phADC latch idle
		(19) BIAS_EN_2_EN_PTAT	(19) BIAS_EN_2_EN_PTAT	0X0	Enable PTAT
		(18:16) BIAS_EN_2_EN_BIAS_BB_HI	(18:16) BIAS_EN_2_EN_BIAS_BB_HI	0X0	Bias enable for BB (same order as biases)
		(15:12) BIAS_EN_1_EN_BIAS_BB_LO	(15:12) BIAS_EN_1_EN_BIAS_BB_LO	0X0	Bias enable for BB (same order as biases)
		(11:7) BIAS_EN_1_EN_BIAS_PLL	(11:7) BIAS_EN_1_EN_BIAS_PLL	0X0	Bias enable for PLL (same order as biases)
		(6:0) BIAS_EN_1_EN_BIAS_RXTX	(6:0) BIAS_EN_1_EN_BIAS_RXTX	0X0	Bias enable for RxTx (same order as biases)

Address	Register Name	Register Write	Register Read	Default	Description
0x400100A0	RF_REG28	(31) CTRL_RX_SWITCH_LP	(31) CTRL_RX_SWITCH_LP	0X0	If set to 1 switch the low-pass filter in the Rx chain
		(30) CTRL_RX_USE_PEAK_DETECTOR	(30) CTRL_RX_USE_PEAK_DETECTOR	0X0	If set to 1, the peak detector is powered on during the Rx by the FSM
		(29) CTRL_RX_START_MIX_ON_CAL	(29) CTRL_RX_START_MIX_ON_CAL	0X0	If set to 1, the mixer is enabled during the sub-band selection phase
		(28:24) CTRL_RX_CTRL_RX	(28:24) CTRL_RX_CTRL_RX	0X0	bits(1:0) => resonance 1 LNA, bits(3:2) => resonance 2 LNA, bit(4) => IFA PTAT-R only
		(23:20) SWCAP_FSM_SB_CAP_RX	(23:20) SWCAP_FSM_SB_CAP_RX	0X0	VCO subband selection (Rx in FSM mode)
		(19:16) SWCAP_FSM_SB_CAP_TX	(19:16) SWCAP_FSM_SB_CAP_TX	0X0	VCO subband selection (Tx in FSM mode)
		(10) DLL_CTRL_CK_LAST_SEL_DELAY	(10) DLL_CTRL_CK_LAST_SEL_DELAY	0X0	
		(9) DLL_CTRL_CK_FIRST_SEL_DELAY	(9) DLL_CTRL_CK_FIRST_SEL_DELAY	0X0	
		(8) DLL_CTRL_CK_EXT_SEL	(8) DLL_CTRL_CK_EXT_SEL	0X0	Low: input clock comes from ck_xtal pin (default). High: input clock comes from ck_ext pin
		(7) DLL_CTRL_CK_DIG_EN	(7) DLL_CTRL_CK_DIG_EN	0X0	Debug: enable to use the alternate ck_dig pin to output the PLL reference clock signal
		(6) DLL_CTRL_CK_TEST_EN	(6) DLL_CTRL_CK_TEST_EN	0X0	Debug: enable to output on GPIO the PLL reference clock signal via ck_test pin
		(5) DLL_CTRL_TOO_FAST_ENB	(5) DLL_CTRL_TOO_FAST_ENB	0X0	When low, enable auxiliary wide lock range phase detector when fast mode locking is enabled (fast_enb = 0). When high, only the narrow lock range phase detector is enabled and bit 2 (fast_enb) must be high to avoid false frequency lock (slow mode locking)
		(4) DLL_CTRL_LOCKED_DET_EN	(4) DLL_CTRL_LOCKED_DET_EN	0X0	Enable reference frequency multiplier locked detector. When this signal is high, the dll_locked output goes high when the output multiplied clock is nearly about three times the frequency of the input clock.
		(3) DLL_CTRL_LOCKED_AUTO_CHECK_EN	(3) DLL_CTRL_LOCKED_AUTO_CHECK_EN	0X0	If for some reason the reference frequency multiplier is out of lock (usually because some input clocks from ck_xtal or ck_ext are missing) and this signal is high, the frequency multiplier will try to lock again automatically. Otherwise, a manual reset should be performed via dll_rstb input(see Table 3) to relock the frequency multiplier. This mode only works if bit 4 is also high (locked detector enabled, see below)
		(2) DLL_CTRL_FAST_ENB	(2) DLL_CTRL_FAST_ENB	0X0	Enable, when low, fast mode locking of the reference frequency multiplier (default). Bit 5 must also be set low in this mode of operation (see below)
		(1:0) DLL_CTRL_CK_SEL	(1:0) DLL_CTRL_CK_SEL	0X2	Selection of the clock used as frequency reference of the PLL (also to ck_test and ck_dig outputs): 00 => ref = ck_xtal ot ck_ext (if bit 8 is high), 01 => ref = same as ck_sel = 00 if dll_en = 0, otherwise frequency(ref) = 3x frequency(ck_xtal) or 3x frequency(ck_ext) (if bit 8 is high), 10 => ref = same as ck_sel = 01 but output frequency divided by 2 (used in normal RX mode when dll_en = 0), 11 => ref = same as ck_sel = 01 but output frequency divided by 5 (used for RX mode with external signal at 132 MHz when dll_en = 0)

Address	Register Name	Register Write	Register Read	Default	Description
0x400100A4	RF_PLL_CTRL	(31:24) XTAL_TRIM_XTAL_TRIM	(31:24) XTAL_TRIM_XTAL_TRIM	0X80	trimming of the xtal: 5MSB thermometric, 3LSB direct
		(20) PLL_CTRL_2_PLL_RX_48MEG	(20) PLL_CTRL_2_PLL_RX_48MEG	0X0	If set to 1 the PLL is set to 48MHz in Rx instead of 24MHz (need also to change ck_sel)
		(19) PLL_CTRL_2_SWCAP_TX_SAME_RX	(19) PLL_CTRL_2_SWCAP_TX_SAME_RX	0X0	If set to 1, in case of swcap_fsm=1, the register for Rx and Tx swcap is the same
		(18) PLL_CTRL_2_SWCAP_FSM	(18) PLL_CTRL_2_SWCAP_FSM	0X0	If set to 1 use the swcap_fsm register as reference for the sub-band selection
		(17) PLL_CTRL_2_DLL_RSTB	(17) PLL_CTRL_2_DLL_RSTB	0X0	Reset signal of the DLL (active low)
		(16) PLL_CTRL_2_VCO_SUBBAND_TRIM_HI	(16) PLL_CTRL_2_VCO_SUBBAND_TRIM_HI	0X0	VCO sub-band selection bits
		(15:13) PLL_CTRL_1_VCO_SUBBAND_TRIM_LO	(15:13) PLL_CTRL_1_VCO_SUBBAND_TRIM_LO	0X0	VCO sub-band selection bits
		(12) PLL_CTRL_1_SUB_SEL_OFFS_EN	(12) PLL_CTRL_1_SUB_SEL_OFFS_EN	0X0	Add offset to sub-band selection comparator
		(11) PLL_CTRL_1_DIV2_CLKVCO_TEST_EN	(11) PLL_CTRL_1_DIV2_CLKVCO_TEST_EN	0X0	Debug: VCO signal divided by the programmable divider is divided by a: 0 => division ratio set to 1, 1 => division ratio set to 2; before to be outputted to ck_div_test
		(10) PLL_CTRL_1_VCODIV_CLK_TEST_EN	(10) PLL_CTRL_1_VCODIV_CLK_TEST_EN	0X0	Debug: enable to output on GPIO the VCO signal divided by the programmable divider
		(9) PLL_CTRL_1_EN_LOW_CHP_BIAS	(9) PLL_CTRL_1_EN_LOW_CHP_BIAS	0X0	When high, allow to decrease half time the bias current for the same output pumping current. Should be always high in IcyTRX.
		(8) PLL_CTRL_1_CHP_DEAD_ZONE_EN	(8) PLL_CTRL_1_CHP_DEAD_ZONE_EN	0X0	Debug: enable charge-pump dead zone (degraded PLL characteristics for test)
		(7:6) PLL_CTRL_1_CHP_CURR_OFFSET_TRIM	(7:6) PLL_CTRL_1_CHP_CURR_OFFSET_TRIM	0X0	Debug: charge-pump offset current values selection bits (see bit 6 to enable this mode): 00 => d_phi = 15, 01 => d_phi=22.5, 10 => d_phi = 30, 11 => d_phi = 60. Also sets the bias current of the common mode control block of the charge-pump. Must be sets to 01 to ensure a proper operation of the VCO tuning voltage comparator for sub-band selection, if used
		(5) PLL_CTRL_1_HIGH_BW_FILTER_EN	(5) PLL_CTRL_1_HIGH_BW_FILTER_EN	0X0	Enable the PLL filter high bandwidth needed in TX (must be high together with bit 4 in TX, low in RX)
		(4) PLL_CTRL_1_FAST_CHP_EN	(4) PLL_CTRL_1_FAST_CHP_EN	0X0	Enable the high current output of the charge-pump for PLL TX high bandwidth mode (must be high together with bit 5 in TX, low in RX)
		(3:2) PLL_CTRL_1_CHP_MODE_TRIM	(3:2) PLL_CTRL_1_CHP_MODE_TRIM	0X0	Charge-pump active if 00 else this allow to open the PLL and force the VCO tune voltage to reach: 01 => minimum frequency inside sub-band selection, 10 => medium frequency inside sub-band selection, 11 => maximum frequency inside sub-band selection.
		(1) PLL_CTRL_1_CHP_CMC_EN	(1) PLL_CTRL_1_CHP_CMC_EN	0X0	Enable the common mode control block of the charge-pump. Must be high to ensure proper operation of the VCO tuning voltage comparator for sub-band selection, if used

Address	Register Name	Register Write	Register Read	Default	Description
0x400100A8	RF_REG2A	(28) ENABLES_SEPARATE_PPA_CASC	(28) ENABLES_SEPARATE_PPA_CASC	0X0	If set to 1, the en PPA cascode bit is independent from the en PA
		(27:22) ENABLES_EN_RXTX	(27:22) ENABLES_EN_RXTX	0X0	Enable signals: 0 => LNA, 1 => LNA, 2 => IFA, 3 => Tx, 4 => PA, 5 => PPA casc
		(21:16) ENABLES_EN_BB	(21:16) ENABLES_EN_BB	0X0	Enable signals for the BB: 0 => Filter, 1 => Filter central frequency bias, 2 => Filter bandwidth bias, 3 => ADC, 4 => RSSI, 5 => peak detector
		(15:13) RSSI_TUN_RSSI_TUN_GAIN	(15:13) RSSI_TUN_RSSI_TUN_GAIN	0X3	RSSI tuning for gain
		(12:8) RSSI_TUN_RSSI_ODD_OFFSET	(12:8) RSSI_TUN_RSSI_ODD_OFFSET	0X0	RSSI tuning for odd stages: offset to the even triangular wave
		(7:4) RSSI_TUN_RSSI_EVEN_MAX	(7:4) RSSI_TUN_RSSI_EVEN_MAX	0X7	RSSI tuning for even stages: maximum value of the triangular wave. If max = min, static signal.
		(3:0) RSSI_TUN_RSSI_EVEN_MIN	(3:0) RSSI_TUN_RSSI_EVEN_MIN	0X7	RSSI tuning for even stages: minimum value of the triangular wave

Address	Register Name	Register Write	Register Read	Default	Description
0x400100AC	RF_XTAL_CTRL	(31:28) XTAL_CTRL_XO_THR_HIGH	(31:28) XTAL_CTRL_XO_THR_HIGH	0XC	High threshold for xtal trimming
		(27:24) XTAL_CTRL_XO_THR_LOW	(27:24) XTAL_CTRL_XO_THR_LOW	0X3	Low threshold for xtal trimming
		(23:22) XTAL_CTRL_XO_A_S_CURR_SEL_HIGH	(23:22) XTAL_CTRL_XO_A_S_CURR_SEL_HIGH	0X2	Value of after_startup_curr_sel when level is higher than xo_thr_high
		(21:20) XTAL_CTRL_XO_A_S_CURR_SEL_LOW	(21:20) XTAL_CTRL_XO_A_S_CURR_SEL_LOW	0X0	Value of after_startup_curr_sel when level is lower than xo_thr_low
		(18) XTAL_CTRL_XTAL_CTRL_BYPASS	(18) XTAL_CTRL_XTAL_CTRL_BYPASS	0X0	Bypass the Xtal control algorithm
		(17) XTAL_CTRL_DIG_CLK_IN_SEL	(17) XTAL_CTRL_DIG_CLK_IN_SEL	0X0	If set to 1 selects the clk_in_dig signal for the digital block, otherwise the internal xtal
		(16) XTAL_CTRL_XO_EN_B_REG	(16) XTAL_CTRL_XO_EN_B_REG	0X1	Xtal oscillator enable (active low)
		(15:14) XTAL_CTRL_XTAL_CKDIV	(15:14) XTAL_CTRL_XTAL_CKDIV	0X0	Xtal trimming speed: 00 => 43us, 01 => 85us, 10 => 171us, 11 => 341us
		(13) XTAL_CTRL_CLK_OUT_EN_B	(13) XTAL_CTRL_CLK_OUT_EN_B	0X0	When high, disable the output clock to go to main IP (clk_out output stay low).
		(12) XTAL_CTRL_REG_VALUE_SEL	(12) XTAL_CTRL_REG_VALUE_SEL	0X0	When low, all main ctrl signals are used instead of corresponding ctrl signal or some control bits of xtal_reg. They are: xo_en_b, ext_clk_mode and lp_mode. When high, corresponding ctrl signal and some control bits of xtal_reg are used instead of main ctrl signals. They are: xo_en_b_reg, ext_clk_mode (bit 0) and lp_mode (bit 1).
		(11:10) XTAL_CTRL_AFTERSTARTUP_CURR_SEL	(11:10) XTAL_CTRL_AFTERSTARTUP_CURR_SEL	0X1	Selection of the current before amplitude stabilization but after starting-up in active transistors of the core oscillator: '00': typ. 0.15 mA, '01': typ. 0.24 mA, '10': typ. 0.40 mA, '11': typ. 0.61 mA
		(9:8) XTAL_CTRL_STARTUP_CURR_SEL	(9:8) XTAL_CTRL_STARTUP_CURR_SEL	0X1	Selection of the starting-up current in active transistors of the core oscillator: '00': typ. 0.41 mA, '01': typ. 0.59 mA, '10': typ. 0.88 mA, '11': typ. 1.24 mA
		(7) XTAL_CTRL_INV_CLK_DIG	(7) XTAL_CTRL_INV_CLK_DIG	0X0	Invert clock on clk_dig output
		(6) XTAL_CTRL_INV_CLK_PLL	(6) XTAL_CTRL_INV_CLK_PLL	0X0	Invert clock on clk_pll output
		(5) XTAL_CTRL_FORCE_CLK_READY	(5) XTAL_CTRL_FORCE_CLK_READY	0X0	Debug: allow to force output clocks on clk_pll, clk_dig and clk_out (if these outputs are enabled) and bypass the xtal internal clock detector that gates these clock outputs.
		(4) XTAL_CTRL_CLK_DIG_EN_B	(4) XTAL_CTRL_CLK_DIG_EN_B	0X0	When high, disable the output clock to go to digital (clk_dig output stay low).
		(3) XTAL_CTRL_BUFF_EN_B	(3) XTAL_CTRL_BUFF_EN_B	0X0	When low (and if xtal_en_b(_reg) is low), the xtal buffer is enabled otherwise it is disabled. Could be used to decrease the power consumption of the xtal while maintaining oscillation in the xtal oscillator
		(2) XTAL_CTRL_HP_MODE	(2) XTAL_CTRL_HP_MODE	0X0	When high, bias current in the clock buffer is increased compared to normal operation (high bandwidth mode in 132 MHz clock input buffer).
		(1) XTAL_CTRL_LP_MODE	(1) XTAL_CTRL_LP_MODE	0X0	When high, bias current in the clock buffer is reduced compared to normal operation (low power mode). Usable only if bit 12 is high (see below) otherwise it is bypassed by lp_mode pin input on main interface

Address	Register Name	Register Write	Register Read	Default	Description
0x400100B0	RF_REG2C	(31:24) SUBBAND_OFFSET_SB_OFFSET	(31:24) SUBBAND_OFFSET_SB_OFFSET	0X0	Offset to add in frequency count in order to compensate the offset of the varicap.
		(23:20) SWCAP_LIM_SB_MAX_VAL	(23:20) SWCAP_LIM_SB_MAX_VAL	0X0	maximum subband value in linear search subband (freq and comp)
		(19:16) SWCAP_LIM_SB_MIN_VAL	(19:16) SWCAP_LIM_SB_MIN_VAL	0X0	minimum subband value in linear search subband (freq and comp)
		(15) SUBBAND_CONF_SB_FLL_MODE	(15) SUBBAND_CONF_SB_FLL_MODE	0X0	Enables the FLL mode for the subband selection (overrides other settings)
		(14) SUBBAND_CONF_SB_INV_BAND	(14) SUBBAND_CONF_SB_INV_BAND	0X0	invert the meaning of sb_high and sb_low
		(13:12) SUBBAND_CONF_SB_FREQ_CNT	(13:12) SUBBAND_CONF_SB_FREQ_CNT	0X0	The length to count in frequency mode: 00 => 256 (Rx: 10.7us, Tx: 2.13us),01 => 512 (Rx: 21.3us, Tx: 4.26us),11 => 1024 (Rx: 42.7us, Tx: 8.53us),01 => 4096 (Rx: 171us, Tx: 34.1us)
		(11:10) SUBBAND_CONF_SB_WAIT_T	(11:10) SUBBAND_CONF_SB_WAIT_T	0X0	time to wait to the PLL to settle: 00 => Rx 8us, Tx 2us, 01 => Rx 12us, Tx 3us, 10 => Rx 16us, Tx 4us, 11 => Rx 24us, Tx 6u
		(9:8) SUBBAND_CONF_SB_MODE	(9:8) SUBBAND_CONF_SB_MODE	0X0	sub-band algorithm mode: 00 => SAR w/ comparators, 01 => linear w/ comparators, 00 => SAR w/ frequency ratios, 01 => linear w/ frequency ratios
		(5:4) PA_CONF_SW_CN	(5:4) PA_CONF_SW_CN	0X0	Harmonic 2 notch tuning
		(3) PA_CONF_TX_SWITCHPA	(3) PA_CONF_TX_SWITCHPA	0X0	If set to 1, enables the PA only with the digital block, otherwise it's the RF Tx timing
		(2) PA_CONF_TX_0DBM	(2) PA_CONF_TX_0DBM	0X0	If set to 1 enables the PA, otherwise only the PPA is used (-20dBm)
		(1:0) PA_CONF_CTRL_PA	(1:0) PA_CONF_CTRL_PA	0X0	N.U.

Address	Register Name	Register Write	Register Read	Default	Description
0x400100B4	RF_REG2D	(31) SUBBAND_CORR_SUBBAND_CORR_EN	(31) SUBBAND_CORR_SUBBAND_CORR_EN	0X0	Enable the subband correction
		(30:28) SUBBAND_CORR_SUBBAND_CORR_RX	(30:28) SUBBAND_CORR_SUBBAND_CORR_RX	0X0	Subband correction in Rx
		(26:24) SUBBAND_CORR_SUBBAND_CORR_TX	(26:24) SUBBAND_CORR_SUBBAND_CORR_TX	0X0	Subband correction in Tx
		(23) PLL_CONF_TX_NRX_INV_CLK_PLL_TX	(23) PLL_CONF_TX_NRX_INV_CLK_PLL_TX	0X0	
		(22) PLL_CONF_TX_NRX_INV_CLK_DIG_TX	(22) PLL_CONF_TX_NRX_INV_CLK_DIG_TX	0X0	
		(21:20) PLL_CONF_TX_NRX_CK_SEL_TX	(21:20) PLL_CONF_TX_NRX_CK_SEL_TX	0X3	Xor value between Tx and Rx for the ck_sel field of register DLL_CTRL
		(18:17) PLL_CONF_TX_NRX_CHP_CURR_OFF_TRIM_TX	(18:17) PLL_CONF_TX_NRX_CHP_CURR_OFF_TRIM_TX	0X0	
		(16) PLL_CONF_TX_NRX_CHP_CURR_OFF_EN_TX	(16) PLL_CONF_TX_NRX_CHP_CURR_OFF_EN_TX	0X0	
		(15) PA_RAMPUP_FULL_PA_RAMPUP	(15) PA_RAMPUP_FULL_PA_RAMPUP	0X0	If set to 1, the PA rampup uses the PA backoff enable bit (from -40 dBm)
		(14:12) PA_RAMPUP_DEL_PA_RAMPUP	(14:12) PA_RAMPUP_DEL_PA_RAMPUP	0X0	time to wait to start the ramp-up after the PA enable is detected
		(11:10) PA_RAMPUP_TAU_PA_RAMPUP	(11:10) PA_RAMPUP_TAU_PA_RAMPUP	0X0	time constant of the Ramp-up/Ramp-down
		(9) PA_RAMPUP_EN_PA_RAMPDOWN	(9) PA_RAMPUP_EN_PA_RAMPDOWN	0X0	if set to 1 enables the PA ramp-down. Only valid in case of ramp-up
		(8) PA_RAMPUP_EN_PA_RAMPUP	(8) PA_RAMPUP_EN_PA_RAMPUP	0X0	if set to 1 enables the PA ramp-up
		(7:3) MISC_SPARES	(7:3) MISC_SPARES	0X0	Unused bits
		(2:1) MISC_RSSI_PRE_ATT	(2:1) MISC_RSSI_PRE_ATT	0X0	RSSI pre-attenuator: 00 => 0dB, 01 => 4dB, 10 => 8dB, 11 => 12dB
		(0) MISC_XTAL_LOW_CLK_READY_TH_EN	(0) MISC_XTAL_LOW_CLK_READY_TH_EN	0X0	XTAL: if set to 1, the clk_ready threshold is set to a lower value

Address	Register Name	Register Write	Register Read	Default	Description
0x400100B8	RF_REG2E	(31:24) RSSI_DETECT_ABS_THR_RSSI_DET_ABS_THR	(31:24) RSSI_DETECT_ABS_THR_RSSI_DET_ABS_THR	0X0	Threshold used for absolute RSSI detection
		(23:16) RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR	(23:16) RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR	0X0	Threshold used for differential RSSI detection
		(14) DEMOD_CTRL_EN_DELLINE_SYNC_DET	(14) DEMOD_CTRL_EN_DELLINE_SYNC_DET	0X0	If set to 1 enable the sync word detection in the delay line. This implies that nc_sel_out = 0x7
		(13) DEMOD_CTRL_RSSI_DET_FILT	(13) DEMOD_CTRL_RSSI_DET_FILT	0X0	Add an additional filtering on the RSSI value
		(12) DEMOD_CTRL_EN_FAST_CLK_RECOV	(12) DEMOD_CTRL_EN_FAST_CLK_RECOV	0X0	If set to 1 speed up the clock recovery during the resto of the preamble
		(11) DEMOD_CTRL_EN_MIN_MAX_MF	(11) DEMOD_CTRL_EN_MIN_MAX_MF	0X0	If set to 1 enables the min max algo after the matched filter
		(10) DEMOD_CTRL_EN_PRE_SYNC	(10) DEMOD_CTRL_EN_PRE_SYNC	0X0	If set to 1 enables the sync detection on the non-delayed path; not working in 4FSK
		(9) DEMOD_CTRL_BLOCK_RSSI_DET	(9) DEMOD_CTRL_BLOCK_RSSI_DET	0X0	If set to 1 blocks the rssi detection during the slow-down period
		(8) DEMOD_CTRL_EARLY_FINE_RECOV	(8) DEMOD_CTRL_EARLY_FINE_RECOV	0X0	If set to 1 enables the early fine recovery after the packet detection or pre-sync
		(7:6) RSSI_DETECT_RSSI_DET_CR_LEN	(7:6) RSSI_DETECT_RSSI_DET_CR_LEN	0X0	Number of samples to estimate the carrier offset: 0 -> 32, 1 -> 64, 2 -> 128, 3->256
		(5:4) RSSI_DETECT_RSSI_DET_WAIT	(5:4) RSSI_DETECT_RSSI_DET_WAIT	0X0	Symbols to wait after the RSSI detection: 00 -> 0, 01 -> 1, 10 -> 2, 11 -> 4
		(3:2) RSSI_DETECT_RSSI_DET_DIFF_LL	(3:2) RSSI_DETECT_RSSI_DET_DIFF_LL	0X0	Set the distance between the actual value and the subtracted one (0->1 sample,1->2 samples,etc)
		(1) RSSI_DETECT_RSSI_DET_EN_ABS	(1) RSSI_DETECT_RSSI_DET_EN_ABS	0X0	If set to 1 enables the absolute RSSI detection
		(0) RSSI_DETECT_RSSI_DET_EN_DIFF	(0) RSSI_DETECT_RSSI_DET_EN_DIFF	0X0	If set to 1 enables the differential RSSI detection
0x400100BC	RF_REG2F	(26:24) CK_DIV_1_6_CK_DIV_1_6	(26:24) CK_DIV_1_6_CK_DIV_1_6	0X0	Clock division factor for ck_div_1_6
		(22:0) RESERVED	(22:0) RESERVED	0X0	

Address	Register Name	Register Write	Register Read	Default	Description
0x400100C0	RF_REG30	(31:25) RXFIFO_STATUS_BIST	-	N/A	Start the bist test on the Rx FIFO (code 0x5d)
		-	(31:30) RXFIFO_STATUS_BIST_ERRORS	0X0	Indicate the BIST error: 00 => no error, 01 => error in checkboard test, 10 => error in inversed checkboard test, 11 => error in decoder test
		-	(29) RXFIFO_STATUS_NEAR_UNDERFLOW	0X0	Is set to 1 if the Rx FIFO is close to the underflow
		-	(28) RXFIFO_STATUS_NEAR_OVERFLOW	0X0	Is set to 1 if the Rx FIFO is close to the overflow
		-	(27) RXFIFO_STATUS_UNDERFLOW	0X0	Is set to 1 if there has been an underflow
		-	(26) RXFIFO_STATUS_OVERFLOW	0X0	Is set to 1 if there has been an overflow
		-	(25) RXFIFO_STATUS_FULL	0X0	Is set to 1 if the Rx FIFO is full
		(24) RXFIFO_STATUS_FLUSH	-	N/A	If set to 1 the Rx FIFO is flushed
		-	(24) RXFIFO_STATUS_EMPTY	0X0	Is set to 1 if the Rx FIFO is empty
		(23:17) TXFIFO_STATUS_BIST	-	N/A	Start the bist test on the Tx FIFO (code 0x5d)
		-	(23:22) TXFIFO_STATUS_BIST_ERRORS	0X0	Indicate the BIST error: 00 => no error, 01 => error in checkboard test, 10 => error in inversed checkboard test, 11 => error in decoder test
		-	(21) TXFIFO_STATUS_NEAR_UNDERFLOW	0X0	Is set to 1 if the Tx FIFO is close to the underflow
		-	(20) TXFIFO_STATUS_NEAR_OVERFLOW	0X0	Is set to 1 if the Tx FIFO is close to the overflow
		-	(19) TXFIFO_STATUS_UNDERFLOW	0X0	Is set to 1 if there has been an underflow
		-	(18) TXFIFO_STATUS_OVERFLOW	0X0	Is set to 1 if there has been an overflow
		-	(17) TXFIFO_STATUS_FULL	0X0	Is set to 1 if the Tx FIFO is full
		(16) TXFIFO_STATUS_FLUSH	-	N/A	If set to 1 the Tx FIFO is flushed
		-	(16) TXFIFO_STATUS_EMPTY	0X0	Is set to 1 if the Tx FIFO is empty
		-	(10) FSM_STATUS_TX_NRX	0X0	Is set to 0 if the radio is in Rx mode, to 1 if in Tx mode
		-	(9:8) FSM_STATUS_STATUS	0X0	Status of the FSM: 00 => Idle, 01 => Tx mode, 10 => Rx mode, 11 => Suspend
		(3) FSM_MODE_RESET	-	N/A	If set to 1, the FSM is reset. If mode is set to 0 the FSM is reset abrubtly. If is set to 1 the Tx or Rx (depending on tx_nrx) is stopped gently via the serializer or the deserializer
		(2) FSM_MODE_TX_NRX	-	N/A	Sets the Radio in Tx (1) or Rx (0) mode
		-	(2) FSM_MODE_RX_MODE	0X0	The field stay with value 1 as long as the reception isn't over
		(1:0) FSM_MODE_MODE	-	N/A	Sets the FSM mode: 00: nothing is done, 01: activate, 10: calibrate the PLL, 11: calibrate the PLL then Tx/Rx
		-	(1) FSM_MODE_TX_MODE	0X0	The field keep the value 1 as long as the transmission isn't over

Address	Register Name	Register Write	Register Read	Default	Description
0x400100C4	RF_REG31	-	(31:24) RSSI_MAX_RSSI_MAX	0X0	Maximum RSSI value over a filtering period
		-	(23:16) RSSI_MIN_RSSI_MIN	0X0	Minimum RSSI value over a filtering period
		-	(15:8) RXFIFO_COUNT_RX_COUNT	0X0	Number of bytes in the Rx FIFO
		-	(7:0) TXFIFO_COUNT_TX_COUNT	0X0	Number of bytes in the Tx FIFO
0x400100C8	RF_REG32	-	(30:28) RX_ATT_LEVEL_RX_ATT_LEVEL_PKT_LVL	0X0	Rx attenuation level (AGC level) during the packet reception
		-	(26:24) RX_ATT_LEVEL_RX_ATT_LEVEL	0X0	Rx attenuation level (AGC level)
		-	(23:16) RSSI_AVG_RSSI_AVG	0X0	Filtered RSSI value
		-	(15:8) DR_ERR_IND_DR_ERR_IND	0X0	Data-rate error indicator
		-	(7:0) RSSI_PKT_RSSI_PKT	0X0	Filtered RSSI value sampled during the packet reception
0x400100CC	RF_TXFIFO	(7:0) TXFIFO_TX_DATA	-	N/A	Data to be sent
0x400100D0	RF_RXFIFO	-	(7:0) RXFIFO_RX_DATA	0X0	Received data
0x400100D4	RF_DESER_STATUS	-	(7) DESER_STATUS_SIGNAL_RECEIVING	0X0	Is set to 1 if the deserializer is on
		-	(6) DESER_STATUS_SYNC_DETECTED	0X0	Is set to 1 is the sync word (pattern) has been detected
		-	(5) DESER_STATUS_WAIT_SYNC	0X0	Is set to 1 if the deserializer is waiting the sync word
		-	(4) DESER_STATUS_IS_ADDRESS_BR	0X0	Is set to 1 if the received address is the broadcast address.
		-	(3) DESER_STATUS_PKT_LEN_ERR	0X0	Is set to 1 in case of the packet length is longer than the maximum acceptable packet length
		-	(2) DESER_STATUS_ADDRESS_ERR	0X0	Is set to 1 in case of an address error
		-	(1) DESER_STATUS_CRC_ERR	0X0	Is set to 1 in case of a CRC error
		-	(0) DESER_STATUS_DESER_FINISH	0X0	Is set to 1 when the deserializer has finished
0x400100D8	RF_IRQ_STATUS	-	(5) IRQ_STATUS_FLAG_RXFIFO	0X0	Is set to 1 when the IRQ RXFIFO is active
		-	(4) IRQ_STATUS_FLAG_TXFIFO	0X0	Is set to 1 when the IRQ TXFIFO is active
		-	(3) IRQ_STATUS_FLAG_SYNC	0X0	Is set to 1 when the IRQ SYNC is active
		-	(2) IRQ_STATUS_FLAG_RECEIVED	0X0	Is set to 1 when the IRQ RECEIVED is active
		-	(1) IRQ_STATUS_FLAG_RXSTOP	0X0	Is set to 1 when the IRQ RXSTOP is active
		-	(0) IRQ_STATUS_FLAG_TX	0X0	Is set to 1 when the IRQ TX is active
0x400100DC	RF_REG37	-	(31:16) FEI_PKT_FEI_PKT	0X0	Frequency error indicator sampled during the packet reception.
		-	(15:0) FEI_FEI_OUT	0X0	Frequency error indicator

Address	Register Name	Register Write	Register Read	Default	Description
0x400100E0	RF_REG38	-	(31:24) LINK_QUAL_PKT_LINK_QUALITY_PKT	0X0	Link quality indicator sampled during the packet reception. Note that the Viterbi algorithm as to be enabled.
		-	(23:16) LINK_QUAL_LINK_QUALITY	0X0	Instantaneous link quality indicator. Note that the Viterbi algorithm as to be enabled.
		-	(15:0) FEI_AFC_FEI_AFC	0X0	Frequency error indicator sampled during the AFC.
0x400100E4	RF_REG39	-	(13) ANALOG_INFO_XTAL_FINISH	0X0	If set to 1, the Xtal algorithm has finished
		-	(12) ANALOG_INFO_DLL_LOCKED	0X0	DLL locked signal
		-	(11) ANALOG_INFO_CLK_DIG_READY	0X0	Ready signal of the digital clock
		-	(10) ANALOG_INFO_CLK_PLL_READY	0X0	Ready signal of the PLL clock
		-	(9) ANALOG_INFO_SUBBAND_HI	0X0	Status of the subband comparator Hi
		-	(8) ANALOG_INFO_SUBBAND_LO	0X0	Status of the subband comparator Lo
		-	(7:0) SUBBAND_ERR_SB_FLL_ERR	0X0	distance from the subband center (only available with the FLL method)
0x400100FC	RF_REVISION	-	(29:24) CHIP_ID	0X0	Version of the chip: 0x00: v1, 0x10: v2A, 0x11: v2B, 0x12: v2C, 0x13: v2D, 0x14: v2E, 0x20: v3

A.25 SYSTICK TIMER

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E010	`SysTick_CTRL`	-	(16) `COUNTFLAG`	0X0	Reads as 1 if SYSTICK counter has reached 0 since the last time the timer has reached 0. Clears to 0 on read.
		(2) `CLKSOURCE`	(2) `CLKSOURCE`	0X0	SYSTICK timer clock source
		(1) `TICKINT`	(1) `TICKINT`	0X0	SYSTICK timer interrupt enable
		(0) `ENABLE`	(0) `ENABLE`	0X0	SYSTICK timer enable
0xE000E014	`SysTick_LOAD`	(23:0) `RELOAD`	(23:0) `RELOAD`	0X0	Counter reload value for the SYSTICK timer when it reaches 0
0xE000E018	`SysTick_VAL`	(23:0) `CURRENT`	(23:0) `CURRENT`	0X0	Current value of the SYSTICK counter value. Write to clear counter.
0xE000E01C	`SysTick_CALIB`	-	(31) `NOREF`	0X0	Indicates if a reference clock is available
		-	(30) `SKEW`	0X1	Indicates if calibration value is exactly 10 ms or not
		-	(23:0) `TENMS`	0X139	SYSTICK counter calibration value for 10 ms. A value of 0 means the calibration value is not available

A.26 System Control and ID register not in the SCB

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E004	SCnSCB_ICTR	-	(4:0) INTLINESNUM	0X2	Number of interrupt inputs in steps of 32

A.27 Nested Vector Interrupt Controller

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E100	NVIC_ISER0	(31) DMIC_OUT_OD_IN	(31) DMIC_OUT_OD_IN	0X0	DMIC_OUT_OD_IN interrupt set enable
		(30) UART_ERROR	(30) UART_ERROR	0X0	UART_ERROR interrupt set enable
		(29) UART_TX	(29) UART_TX	0X0	UART_TX interrupt set enable
		(28) UART_RX	(28) UART_RX	0X0	UART_RX interrupt set enable
		(27) I2C	(27) I2C	0X0	I2C interrupt set enable
		(26) SPI1_ERROR	(26) SPI1_ERROR	0X0	SPI1_ERROR interrupt set enable
		(25) SPI1_TX	(25) SPI1_TX	0X0	SPI1_TX interrupt set enable
		(24) SPI1_RX	(24) SPI1_RX	0X0	SPI1_RX interrupt set enable
		(23) SPI0_ERROR	(23) SPI0_ERROR	0X0	SPI0_ERROR interrupt set enable
		(22) SPI0_TX	(22) SPI0_TX	0X0	SPI0_TX interrupt set enable
		(21) SPI0_RX	(21) SPI0_RX	0X0	SPI0_RX interrupt set enable
		(20) WATCHDOG	(20) WATCHDOG	0X0	WATCHDOG interrupt set enable
		(19) DIO3	(19) DIO3	0X0	DIO3 interrupt set enable
		(18) DIO2	(18) DIO2	0X0	DIO2 interrupt set enable
		(17) DIO1	(17) DIO1	0X0	DIO1 interrupt set enable
		(16) DIO0	(16) DIO0	0X0	DIO0 interrupt set enable
		(15) DMA7	(15) DMA7	0X0	DMA7 interrupt set enable
		(14) DMA6	(14) DMA6	0X0	DMA6 interrupt set enable
		(13) DMA5	(13) DMA5	0X0	DMA5 interrupt set enable
		(12) DMA4	(12) DMA4	0X0	DMA4 interrupt set enable
		(11) DMA3	(11) DMA3	0X0	DMA3 interrupt set enable
		(10) DMA2	(10) DMA2	0X0	DMA2 interrupt set enable
		(9) DMA1	(9) DMA1	0X0	DMA1 interrupt set enable
		(8) DMA0	(8) DMA0	0X0	DMA0 interrupt set enable
		(7) TIMER3	(7) TIMER3	0X0	TIMER3 interrupt set enable
		(6) TIMER2	(6) TIMER2	0X0	TIMER2 interrupt set enable
		(5) TIMER1	(5) TIMER1	0X0	TIMER1 interrupt set enable
		(4) TIMER0	(4) TIMER0	0X0	TIMER0 interrupt set enable
		(3) ADC_BATMON	(3) ADC_BATMON	0X0	ADC_BATMON interrupt set enable
		(2) RTC_CLOCK	(2) RTC_CLOCK	0X0	RTC_CLOCK interrupt set enable
		(1) RTC_ALARM	(1) RTC_ALARM	0X0	RTC_ALARM interrupt set enable
		(0) WAKEUP	(0) WAKEUP	0X0	WAKEUP interrupt set enable

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E104	NVIC_ISER1	(31) ASRC_OUT	(31) ASRC_OUT	0X0	ASRC_OUT interrupt set enable
		(30) ASRC_IN	(30) ASRC_IN	0X0	ASRC_IN interrupt set enable
		(29) ASRC_ERROR	(29) ASRC_ERROR	0X0	ASRC_ERROR interrupt set enable
		(28) RF_RXFIFO	(28) RF_RXFIFO	0X0	RF_RXFIFO interrupt set enable
		(27) RF_TXFIFO	(27) RF_TXFIFO	0X0	RF_TXFIFO interrupt set enable
		(26) RF_SYNC	(26) RF_SYNC	0X0	RF_SYNC interrupt set enable
		(25) RF_RECEIVED	(25) RF_RECEIVED	0X0	RF_RECEIVED interrupt set enable
		(24) RF_RXSTOP	(24) RF_RXSTOP	0X0	RF_RXSTOP interrupt set enable
		(23) RF_TX	(23) RF_TX	0X0	RF_TX interrupt set enable
		(22) BLE_COEX_IN_PROCESS	(22) BLE_COEX_IN_PROCESS	0X0	BLE_COEX_IN_PROCESS interrupt set enable
		(21) BLE_COEX_RX_TX	(21) BLE_COEX_RX_TX	0X0	BLE_COEX_RX_TX interrupt set enable
		(20) BLE_SW	(20) BLE_SW	0X0	BLE_SW interrupt set enable
		(19) BLE_FINETGTIM	(19) BLE_FINETGTIM	0X0	BLE_FINETGTIM interrupt set enable
		(18) BLE_GROSSTGTIM	(18) BLE_GROSSTGTIM	0X0	BLE_GROSSTGTIM interrupt set enable
		(17) BLE_ERROR	(17) BLE_ERROR	0X0	BLE_ERROR interrupt set enable
		(16) BLE_CRYPT	(16) BLE_CRYPT	0X0	BLE_CRYPT interrupt set enable
		(15) BLE_EVENT	(15) BLE_EVENT	0X0	BLE_EVENT interrupt set enable
		(14) BLE_RX	(14) BLE_RX	0X0	BLE_RX interrupt set enable
		(13) BLE_SLP	(13) BLE_SLP	0X0	BLE_SLP interrupt set enable
		(12) BLE_CSCNT	(12) BLE_CSCNT	0X0	BLE_CSCNT interrupt set enable
		(11) DSS7	(11) DSS7	0X0	DSS7 interrupt set enable
		(10) DSS6	(10) DSS6	0X0	DSS6 interrupt set enable
		(9) DSS5	(9) DSS5	0X0	DSS5 interrupt set enable
		(8) DSS4	(8) DSS4	0X0	DSS4 interrupt set enable
		(7) DSS3	(7) DSS3	0X0	DSS3 interrupt set enable
		(6) DSS2	(6) DSS2	0X0	DSS2 interrupt set enable
		(5) DSS1	(5) DSS1	0X0	DSS1 interrupt set enable
		(4) DSS0	(4) DSS0	0X0	DSS0 interrupt set enable
		(3) PCM_ERROR	(3) PCM_ERROR	0X0	PCM_ERROR interrupt set enable
		(2) PCM_TX	(2) PCM_TX	0X0	PCM_TX interrupt set enable
		(1) PCM_RX	(1) PCM_RX	0X0	PCM_RX interrupt set enable
		(0) DMIC_OD_ERROR	(0) DMIC_OD_ERROR	0X0	DMIC_OD_ERROR interrupt set enable

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E108	NVIC_ISER2	(8) BLE_AUDIO2	(8) BLE_AUDIO2	0X0	BLE_AUDIO2 interrupt set enable
		(7) BLE_AUDIO1	(7) BLE_AUDIO1	0X0	BLE_AUDIO1 interrupt set enable
		(6) BLE_AUDIO0	(6) BLE_AUDIO0	0X0	BLE_AUDIO0 interrupt set enable
		(5) MEM_ERROR	(5) MEM_ERROR	0X0	MEM_ERROR interrupt set enable
		(4) FLASH_ECC	(4) FLASH_ECC	0X0	FLASH_ECC interrupt set enable
		(3) FLASH_COPY	(3) FLASH_COPY	0X0	FLASH_COPY interrupt set enable
		(2) CLKDET	(2) CLKDET	0X0	CLKDET interrupt set enable
		(1) AUDIO_SINK_PERIOD	(1) AUDIO_SINK_PERIOD	0X0	AUDIO_SINK_PERIOD interrupt set enable
		(0) AUDIO_SINK_DELAY	(0) AUDIO_SINK_DELAY	0X0	AUDIO_SINK_DELAY interrupt set enable

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E180	NVIC_ICER0	(31) DMIC_OUT_OD_IN	(31) DMIC_OUT_OD_IN	0X0	DMIC_OUT_OD_IN interrupt clear enable
		(30) UART_ERROR	(30) UART_ERROR	0X0	UART_ERROR interrupt clear enable
		(29) UART_TX	(29) UART_TX	0X0	UART_TX interrupt clear enable
		(28) UART_RX	(28) UART_RX	0X0	UART_RX interrupt clear enable
		(27) I2C	(27) I2C	0X0	I2C interrupt clear enable
		(26) SPI1_ERROR	(26) SPI1_ERROR	0X0	SPI1_ERROR interrupt clear enable
		(25) SPI1_TX	(25) SPI1_TX	0X0	SPI1_TX interrupt clear enable
		(24) SPI1_RX	(24) SPI1_RX	0X0	SPI1_RX interrupt clear enable
		(23) SPI0_ERROR	(23) SPI0_ERROR	0X0	SPI0_ERROR interrupt clear enable
		(22) SPI0_TX	(22) SPI0_TX	0X0	SPI0_TX interrupt clear enable
		(21) SPI0_RX	(21) SPI0_RX	0X0	SPI0_RX interrupt clear enable
		(20) WATCHDOG	(20) WATCHDOG	0X0	WATCHDOG interrupt clear enable
		(19) DIO3	(19) DIO3	0X0	DIO3 interrupt clear enable
		(18) DIO2	(18) DIO2	0X0	DIO2 interrupt clear enable
		(17) DIO1	(17) DIO1	0X0	DIO1 interrupt clear enable
		(16) DIO0	(16) DIO0	0X0	DIO0 interrupt clear enable
		(15) DMA7	(15) DMA7	0X0	DMA7 interrupt clear enable
		(14) DMA6	(14) DMA6	0X0	DMA6 interrupt clear enable
		(13) DMA5	(13) DMA5	0X0	DMA5 interrupt clear enable
		(12) DMA4	(12) DMA4	0X0	DMA4 interrupt clear enable
		(11) DMA3	(11) DMA3	0X0	DMA3 interrupt clear enable
		(10) DMA2	(10) DMA2	0X0	DMA2 interrupt clear enable
		(9) DMA1	(9) DMA1	0X0	DMA1 interrupt clear enable
		(8) DMA0	(8) DMA0	0X0	DMA0 interrupt clear enable
		(7) TIMER3	(7) TIMER3	0X0	TIMER3 interrupt clear enable
		(6) TIMER2	(6) TIMER2	0X0	TIMER2 interrupt clear enable
		(5) TIMER1	(5) TIMER1	0X0	TIMER1 interrupt clear enable
		(4) TIMER0	(4) TIMER0	0X0	TIMER0 interrupt clear enable
		(3) ADC_BATMON	(3) ADC_BATMON	0X0	ADC_BATMON interrupt clear enable
		(2) RTC_CLOCK	(2) RTC_CLOCK	0X0	RTC_CLOCK interrupt clear enable
		(1) RTC_ALARM	(1) RTC_ALARM	0X0	RTC_ALARM interrupt clear enable
		(0) WAKEUP	(0) WAKEUP	0X0	WAKEUP interrupt clear enable

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E184	NVIC_ICER1	(31) ASRC_OUT	(31) ASRC_OUT	0X0	ASRC_OUT interrupt clear enable
		(30) ASRC_IN	(30) ASRC_IN	0X0	ASRC_IN interrupt clear enable
		(29) ASRC_ERROR	(29) ASRC_ERROR	0X0	ASRC_ERROR interrupt clear enable
		(28) RF_RXFIFO	(28) RF_RXFIFO	0X0	RF_RXFIFO interrupt clear enable
		(27) RF_TXFIFO	(27) RF_TXFIFO	0X0	RF_TXFIFO interrupt clear enable
		(26) RF_SYNC	(26) RF_SYNC	0X0	RF_SYNC interrupt clear enable
		(25) RF_RECEIVED	(25) RF_RECEIVED	0X0	RF_RECEIVED interrupt clear enable
		(24) RF_RXSTOP	(24) RF_RXSTOP	0X0	RF_RXSTOP interrupt clear enable
		(23) RF_TX	(23) RF_TX	0X0	RF_TX interrupt clear enable
		(22) BLE_COEX_IN_PROCESS	(22) BLE_COEX_IN_PROCESS	0X0	BLE_COEX_IN_PROCESS interrupt clear enable
		(21) BLE_COEX_RX_TX	(21) BLE_COEX_RX_TX	0X0	BLE_COEX_RX_TX interrupt clear enable
		(20) BLE_SW	(20) BLE_SW	0X0	BLE_SW interrupt clear enable
		(19) BLE_FINETGTIM	(19) BLE_FINETGTIM	0X0	BLE_FINETGTIM interrupt clear enable
		(18) BLE_GROSSTGTIM	(18) BLE_GROSSTGTIM	0X0	BLE_GROSSTGTIM interrupt clear enable
		(17) BLE_ERROR	(17) BLE_ERROR	0X0	BLE_ERROR interrupt clear enable
		(16) BLE_CRYPT	(16) BLE_CRYPT	0X0	BLE_CRYPT interrupt clear enable
		(15) BLE_EVENT	(15) BLE_EVENT	0X0	BLE_EVENT interrupt clear enable
		(14) BLE_RX	(14) BLE_RX	0X0	BLE_RX interrupt clear enable
		(13) BLE_SLP	(13) BLE_SLP	0X0	BLE_SLP interrupt clear enable
		(12) BLE_CSCNT	(12) BLE_CSCNT	0X0	BLE_CSCNT interrupt clear enable
		(11) DSS7	(11) DSS7	0X0	DSS7 interrupt clear enable
		(10) DSS6	(10) DSS6	0X0	DSS6 interrupt clear enable
		(9) DSS5	(9) DSS5	0X0	DSS5 interrupt clear enable
		(8) DSS4	(8) DSS4	0X0	DSS4 interrupt clear enable
		(7) DSS3	(7) DSS3	0X0	DSS3 interrupt clear enable
		(6) DSS2	(6) DSS2	0X0	DSS2 interrupt clear enable
		(5) DSS1	(5) DSS1	0X0	DSS1 interrupt clear enable
		(4) DSS0	(4) DSS0	0X0	DSS0 interrupt clear enable
		(3) PCM_ERROR	(3) PCM_ERROR	0X0	PCM_ERROR interrupt clear enable
		(2) PCM_TX	(2) PCM_TX	0X0	PCM_TX interrupt clear enable
		(1) PCM_RX	(1) PCM_RX	0X0	PCM_RX interrupt clear enable
		(0) DMIC_OD_ERROR	(0) DMIC_OD_ERROR	0X0	DMIC_OD_ERROR interrupt clear enable

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E188	NVIC_ICER2	(8) BLE_AUDIO2	(8) BLE_AUDIO2	0X0	BLE_AUDIO2 interrupt clear enable
		(7) BLE_AUDIO1	(7) BLE_AUDIO1	0X0	BLE_AUDIO1 interrupt clear enable
		(6) BLE_AUDIO0	(6) BLE_AUDIO0	0X0	BLE_AUDIO0 interrupt clear enable
		(5) MEM_ERROR	(5) MEM_ERROR	0X0	MEM_ERROR interrupt clear enable
		(4) FLASH_ECC	(4) FLASH_ECC	0X0	FLASH_ECC interrupt clear enable
		(3) FLASH_COPY	(3) FLASH_COPY	0X0	FLASH_COPY interrupt clear enable
		(2) CLKDET	(2) CLKDET	0X0	CLKDET interrupt clear enable
		(1) AUDIO_SINK_PERIOD	(1) AUDIO_SINK_PERIOD	0X0	AUDIO_SINK_PERIOD interrupt clear enable
		(0) AUDIO_SINK_DELAY	(0) AUDIO_SINK_DELAY	0X0	AUDIO_SINK_DELAY interrupt clear enable

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E200	NVIC_ISPR0	(31) DMIC_OUT_OD_IN	(31) DMIC_OUT_OD_IN	0X0	DMIC_OUT_OD_IN interrupt set pending
		(30) UART_ERROR	(30) UART_ERROR	0X0	UART_ERROR interrupt set pending
		(29) UART_TX	(29) UART_TX	0X0	UART_TX interrupt set pending
		(28) UART_RX	(28) UART_RX	0X0	UART_RX interrupt set pending
		(27) I2C	(27) I2C	0X0	I2C interrupt set pending
		(26) SPI1_ERROR	(26) SPI1_ERROR	0X0	SPI1_ERROR interrupt set pending
		(25) SPI1_TX	(25) SPI1_TX	0X0	SPI1_TX interrupt set pending
		(24) SPI1_RX	(24) SPI1_RX	0X0	SPI1_RX interrupt set pending
		(23) SPI0_ERROR	(23) SPI0_ERROR	0X0	SPI0_ERROR interrupt set pending
		(22) SPI0_TX	(22) SPI0_TX	0X0	SPI0_TX interrupt set pending
		(21) SPI0_RX	(21) SPI0_RX	0X0	SPI0_RX interrupt set pending
		(20) WATCHDOG	(20) WATCHDOG	0X0	WATCHDOG interrupt set pending
		(19) DIO3	(19) DIO3	0X0	DIO3 interrupt set pending
		(18) DIO2	(18) DIO2	0X0	DIO2 interrupt set pending
		(17) DIO1	(17) DIO1	0X0	DIO1 interrupt set pending
		(16) DIO0	(16) DIO0	0X0	DIO0 interrupt set pending
		(15) DMA7	(15) DMA7	0X0	DMA7 interrupt set pending
		(14) DMA6	(14) DMA6	0X0	DMA6 interrupt set pending
		(13) DMA5	(13) DMA5	0X0	DMA5 interrupt set pending
		(12) DMA4	(12) DMA4	0X0	DMA4 interrupt set pending
		(11) DMA3	(11) DMA3	0X0	DMA3 interrupt set pending
		(10) DMA2	(10) DMA2	0X0	DMA2 interrupt set pending
		(9) DMA1	(9) DMA1	0X0	DMA1 interrupt set pending
		(8) DMA0	(8) DMA0	0X0	DMA0 interrupt set pending
		(7) TIMER3	(7) TIMER3	0X0	TIMER3 interrupt set pending
		(6) TIMER2	(6) TIMER2	0X0	TIMER2 interrupt set pending
		(5) TIMER1	(5) TIMER1	0X0	TIMER1 interrupt set pending
		(4) TIMER0	(4) TIMER0	0X0	TIMER0 interrupt set pending
		(3) ADC_BATMON	(3) ADC_BATMON	0X0	ADC_BATMON interrupt set pending
		(2) RTC_CLOCK	(2) RTC_CLOCK	0X0	RTC_CLOCK interrupt set pending
		(1) RTC_ALARM	(1) RTC_ALARM	0X0	RTC_ALARM interrupt set pending
		(0) WAKEUP	(0) WAKEUP	0X0	WAKEUP interrupt set pending

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E204	NVIC_ISPR1	(31) ASRC_OUT	(31) ASRC_OUT	0X0	ASRC_OUT interrupt set pending
		(30) ASRC_IN	(30) ASRC_IN	0X0	ASRC_IN interrupt set pending
		(29) ASRC_ERROR	(29) ASRC_ERROR	0X0	ASRC_ERROR interrupt set pending
		(28) RF_RXFIFO	(28) RF_RXFIFO	0X0	RF_RXFIFO interrupt set pending
		(27) RF_TXFIFO	(27) RF_TXFIFO	0X0	RF_TXFIFO interrupt set pending
		(26) RF_SYNC	(26) RF_SYNC	0X0	RF_SYNC interrupt set pending
		(25) RF_RECEIVED	(25) RF_RECEIVED	0X0	RF_RECEIVED interrupt set pending
		(24) RF_RXSTOP	(24) RF_RXSTOP	0X0	RF_RXSTOP interrupt set pending
		(23) RF_TX	(23) RF_TX	0X0	RF_TX interrupt set pending
		(22) BLE_COEX_IN_PROCESS	(22) BLE_COEX_IN_PROCESS	0X0	BLE_COEX_IN_PROCESS interrupt set pending
		(21) BLE_COEX_RX_TX	(21) BLE_COEX_RX_TX	0X0	BLE_COEX_RX_TX interrupt set pending
		(20) BLE_SW	(20) BLE_SW	0X0	BLE_SW interrupt set pending
		(19) BLE_FINETGTIM	(19) BLE_FINETGTIM	0X0	BLE_FINETGTIM interrupt set pending
		(18) BLE_GROSSTGTIM	(18) BLE_GROSSTGTIM	0X0	BLE_GROSSTGTIM interrupt set pending
		(17) BLE_ERROR	(17) BLE_ERROR	0X0	BLE_ERROR interrupt set pending
		(16) BLE_CRYPT	(16) BLE_CRYPT	0X0	BLE_CRYPT interrupt set pending
		(15) BLE_EVENT	(15) BLE_EVENT	0X0	BLE_EVENT interrupt set pending
		(14) BLE_RX	(14) BLE_RX	0X0	BLE_RX interrupt set pending
		(13) BLE_SLP	(13) BLE_SLP	0X0	BLE_SLP interrupt set pending
		(12) BLE_CSCNT	(12) BLE_CSCNT	0X0	BLE_CSCNT interrupt set pending
		(11) DSS7	(11) DSS7	0X0	DSS7 interrupt set pending
		(10) DSS6	(10) DSS6	0X0	DSS6 interrupt set pending
		(9) DSS5	(9) DSS5	0X0	DSS5 interrupt set pending
		(8) DSS4	(8) DSS4	0X0	DSS4 interrupt set pending
		(7) DSS3	(7) DSS3	0X0	DSS3 interrupt set pending
		(6) DSS2	(6) DSS2	0X0	DSS2 interrupt set pending
		(5) DSS1	(5) DSS1	0X0	DSS1 interrupt set pending
		(4) DSS0	(4) DSS0	0X0	DSS0 interrupt set pending
		(3) PCM_ERROR	(3) PCM_ERROR	0X0	PCM_ERROR interrupt set pending
		(2) PCM_TX	(2) PCM_TX	0X0	PCM_TX interrupt set pending
		(1) PCM_RX	(1) PCM_RX	0X0	PCM_RX interrupt set pending
		(0) DMIC_OD_ERROR	(0) DMIC_OD_ERROR	0X0	DMIC_OD_ERROR interrupt set pending

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E208	NVIC_ISPR2	(8) BLE_AUDIO2	(8) BLE_AUDIO2	0X0	BLE_AUDIO2 interrupt set pending
		(7) BLE_AUDIO1	(7) BLE_AUDIO1	0X0	BLE_AUDIO1 interrupt set pending
		(6) BLE_AUDIO0	(6) BLE_AUDIO0	0X0	BLE_AUDIO0 interrupt set pending
		(5) MEM_ERROR	(5) MEM_ERROR	0X0	MEM_ERROR interrupt set pending
		(4) FLASH_ECC	(4) FLASH_ECC	0X0	FLASH_ECC interrupt set pending
		(3) FLASH_COPY	(3) FLASH_COPY	0X0	FLASH_COPY interrupt set pending
		(2) CLKDET	(2) CLKDET	0X0	CLKDET interrupt set pending
		(1) AUDIO_SINK_PERIOD	(1) AUDIO_SINK_PERIOD	0X0	AUDIO_SINK_PERIOD interrupt set pending
		(0) AUDIO_SINK_DELAY	(0) AUDIO_SINK_DELAY	0X0	AUDIO_SINK_DELAY interrupt set pending

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E280	NVIC_ICPR0	(31) DMIC_OUT_OD_IN	(31) DMIC_OUT_OD_IN	0X0	DMIC_OUT_OD_IN interrupt clear pending
		(30) UART_ERROR	(30) UART_ERROR	0X0	UART_ERROR interrupt clear pending
		(29) UART_TX	(29) UART_TX	0X0	UART_TX interrupt clear pending
		(28) UART_RX	(28) UART_RX	0X0	UART_RX interrupt clear pending
		(27) I2C	(27) I2C	0X0	I2C interrupt clear pending
		(26) SPI1_ERROR	(26) SPI1_ERROR	0X0	SPI1_ERROR interrupt clear pending
		(25) SPI1_TX	(25) SPI1_TX	0X0	SPI1_TX interrupt clear pending
		(24) SPI1_RX	(24) SPI1_RX	0X0	SPI1_RX interrupt clear pending
		(23) SPI0_ERROR	(23) SPI0_ERROR	0X0	SPI0_ERROR interrupt clear pending
		(22) SPI0_TX	(22) SPI0_TX	0X0	SPI0_TX interrupt clear pending
		(21) SPI0_RX	(21) SPI0_RX	0X0	SPI0_RX interrupt clear pending
		(20) WATCHDOG	(20) WATCHDOG	0X0	WATCHDOG interrupt clear pending
		(19) DIO3	(19) DIO3	0X0	DIO3 interrupt clear pending
		(18) DIO2	(18) DIO2	0X0	DIO2 interrupt clear pending
		(17) DIO1	(17) DIO1	0X0	DIO1 interrupt clear pending
		(16) DIO0	(16) DIO0	0X0	DIO0 interrupt clear pending
		(15) DMA7	(15) DMA7	0X0	DMA7 interrupt clear pending
		(14) DMA6	(14) DMA6	0X0	DMA6 interrupt clear pending
		(13) DMA5	(13) DMA5	0X0	DMA5 interrupt clear pending
		(12) DMA4	(12) DMA4	0X0	DMA4 interrupt clear pending
		(11) DMA3	(11) DMA3	0X0	DMA3 interrupt clear pending
		(10) DMA2	(10) DMA2	0X0	DMA2 interrupt clear pending
		(9) DMA1	(9) DMA1	0X0	DMA1 interrupt clear pending
		(8) DMA0	(8) DMA0	0X0	DMA0 interrupt clear pending
		(7) TIMER3	(7) TIMER3	0X0	TIMER3 interrupt clear pending
		(6) TIMER2	(6) TIMER2	0X0	TIMER2 interrupt clear pending
		(5) TIMER1	(5) TIMER1	0X0	TIMER1 interrupt clear pending
		(4) TIMER0	(4) TIMER0	0X0	TIMER0 interrupt clear pending
		(3) ADC_BATMON	(3) ADC_BATMON	0X0	ADC_BATMON interrupt clear pending
		(2) RTC_CLOCK	(2) RTC_CLOCK	0X0	RTC_CLOCK interrupt clear pending
		(1) RTC_ALARM	(1) RTC_ALARM	0X0	RTC_ALARM interrupt clear pending
		(0) WAKEUP	(0) WAKEUP	0X0	WAKEUP interrupt clear pending

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E284	NVIC_ICPR1	(31) ASRC_OUT	(31) ASRC_OUT	0X0	ASRC_OUT interrupt clear pending
		(30) ASRC_IN	(30) ASRC_IN	0X0	ASRC_IN interrupt clear pending
		(29) ASRC_ERROR	(29) ASRC_ERROR	0X0	ASRC_ERROR interrupt clear pending
		(28) RF_RXFIFO	(28) RF_RXFIFO	0X0	RF_RXFIFO interrupt clear pending
		(27) RF_TXFIFO	(27) RF_TXFIFO	0X0	RF_TXFIFO interrupt clear pending
		(26) RF_SYNC	(26) RF_SYNC	0X0	RF_SYNC interrupt clear pending
		(25) RF_RECEIVED	(25) RF_RECEIVED	0X0	RF_RECEIVED interrupt clear pending
		(24) RF_RXSTOP	(24) RF_RXSTOP	0X0	RF_RXSTOP interrupt clear pending
		(23) RF_TX	(23) RF_TX	0X0	RF_TX interrupt clear pending
		(22) BLE_COEX_IN_PROCESS	(22) BLE_COEX_IN_PROCESS	0X0	BLE_COEX_IN_PROCESS interrupt clear pending
		(21) BLE_COEX_RX_TX	(21) BLE_COEX_RX_TX	0X0	BLE_COEX_RX_TX interrupt clear pending
		(20) BLE_SW	(20) BLE_SW	0X0	BLE_SW interrupt clear pending
		(19) BLE_FINETGTIM	(19) BLE_FINETGTIM	0X0	BLE_FINETGTIM interrupt clear pending
		(18) BLE_GROSSTGTIM	(18) BLE_GROSSTGTIM	0X0	BLE_GROSSTGTIM interrupt clear pending
		(17) BLE_ERROR	(17) BLE_ERROR	0X0	BLE_ERROR interrupt clear pending
		(16) BLE_CRYPT	(16) BLE_CRYPT	0X0	BLE_CRYPT interrupt clear pending
		(15) BLE_EVENT	(15) BLE_EVENT	0X0	BLE_EVENT interrupt clear pending
		(14) BLE_RX	(14) BLE_RX	0X0	BLE_RX interrupt clear pending
		(13) BLE_SLP	(13) BLE_SLP	0X0	BLE_SLP interrupt clear pending
		(12) BLE_CSCNT	(12) BLE_CSCNT	0X0	BLE_CSCNT interrupt clear pending
		(11) DSS7	(11) DSS7	0X0	DSS7 interrupt clear pending
		(10) DSS6	(10) DSS6	0X0	DSS6 interrupt clear pending
		(9) DSS5	(9) DSS5	0X0	DSS5 interrupt clear pending
		(8) DSS4	(8) DSS4	0X0	DSS4 interrupt clear pending
		(7) DSS3	(7) DSS3	0X0	DSS3 interrupt clear pending
		(6) DSS2	(6) DSS2	0X0	DSS2 interrupt clear pending
		(5) DSS1	(5) DSS1	0X0	DSS1 interrupt clear pending
		(4) DSS0	(4) DSS0	0X0	DSS0 interrupt clear pending
		(3) PCM_ERROR	(3) PCM_ERROR	0X0	PCM_ERROR interrupt clear pending
		(2) PCM_TX	(2) PCM_TX	0X0	PCM_TX interrupt clear pending
		(1) PCM_RX	(1) PCM_RX	0X0	PCM_RX interrupt clear pending
		(0) DMIC_OD_ERROR	(0) DMIC_OD_ERROR	0X0	DMIC_OD_ERROR interrupt clear pending

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E288	NVIC_ICPR2	(8) BLE_AUDIO2	(8) BLE_AUDIO2	0X0	BLE_AUDIO2 interrupt clear pending
		(7) BLE_AUDIO1	(7) BLE_AUDIO1	0X0	BLE_AUDIO1 interrupt clear pending
		(6) BLE_AUDIO0	(6) BLE_AUDIO0	0X0	BLE_AUDIO0 interrupt clear pending
		(5) MEM_ERROR	(5) MEM_ERROR	0X0	MEM_ERROR interrupt clear pending
		(4) FLASH_ECC	(4) FLASH_ECC	0X0	FLASH_ECC interrupt clear pending
		(3) FLASH_COPY	(3) FLASH_COPY	0X0	FLASH_COPY interrupt clear pending
		(2) CLKDET	(2) CLKDET	0X0	CLKDET interrupt clear pending
		(1) AUDIO_SINK_PERIOD	(1) AUDIO_SINK_PERIOD	0X0	AUDIO_SINK_PERIOD interrupt clear pending
		(0) AUDIO_SINK_DELAY	(0) AUDIO_SINK_DELAY	0X0	AUDIO_SINK_DELAY interrupt clear pending

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E300	NVIC_IABR0	-	(31) DMIC_OUT_OD_IN	0X0	Set the DMIC_OUT_OD_IN interrupt as active
		-	(30) UART_ERROR	0X0	Set the UART_ERROR interrupt as active
		-	(29) UART_TX	0X0	Set the UART_TX interrupt as active
		-	(28) UART_RX	0X0	Set the UART_RX interrupt as active
		-	(27) I2C	0X0	Set the I2C interrupt as active
		-	(26) SPI1_ERROR	0X0	Set the SPI1_ERROR interrupt as active
		-	(25) SPI1_TX	0X0	Set the SPI1_TX interrupt as active
		-	(24) SPI1_RX	0X0	Set the SPI1_RX interrupt as active
		-	(23) SPI0_ERROR	0X0	Set the SPI0_ERROR interrupt as active
		-	(22) SPI0_TX	0X0	Set the SPI0_TX interrupt as active
		-	(21) SPI0_RX	0X0	Set the SPI0_RX interrupt as active
		-	(20) WATCHDOG	0X0	Set the WATCHDOG interrupt as active
		-	(19) DIO3	0X0	Set the DIO3 interrupt as active
		-	(18) DIO2	0X0	Set the DIO2 interrupt as active
		-	(17) DIO1	0X0	Set the DIO1 interrupt as active
		-	(16) DIO0	0X0	Set the DIO0 interrupt as active
		-	(15) DMA7	0X0	Set the DMA7 interrupt as active
		-	(14) DMA6	0X0	Set the DMA6 interrupt as active
		-	(13) DMA5	0X0	Set the DMA5 interrupt as active
		-	(12) DMA4	0X0	Set the DMA4 interrupt as active
		-	(11) DMA3	0X0	Set the DMA3 interrupt as active
		-	(10) DMA2	0X0	Set the DMA2 interrupt as active
		-	(9) DMA1	0X0	Set the DMA1 interrupt as active
		-	(8) DMA0	0X0	Set the DMA0 interrupt as active
		-	(7) TIMER3	0X0	Set the TIMER3 interrupt as active
		-	(6) TIMER2	0X0	Set the TIMER2 interrupt as active
		-	(5) TIMER1	0X0	Set the TIMER1 interrupt as active
		-	(4) TIMER0	0X0	Set the TIMER0 interrupt as active
		-	(3) ADC_BATMON	0X0	Set the ADC_BATMON interrupt as active
		-	(2) RTC_CLOCK	0X0	Set the RTC_CLOCK interrupt as active
		-	(1) RTC_ALARM	0X0	Set the RTC_ALARM interrupt as active
		-	(0) WAKEUP	0X0	Set the WAKEUP interrupt as active

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E304	NVIC_IABR1	-	(31) ASRC_OUT	0X0	Set the ASRC_OUT interrupt as active
		-	(30) ASRC_IN	0X0	Set the ASRC_IN interrupt as active
		-	(29) ASRC_ERROR	0X0	Set the ASRC_ERROR interrupt as active
		-	(28) RF_RXFIFO	0X0	Set the RF_RXFIFO interrupt as active
		-	(27) RF_TXFIFO	0X0	Set the RF_TXFIFO interrupt as active
		-	(26) RF_SYNC	0X0	Set the RF_SYNC interrupt as active
		-	(25) RF_RECEIVED	0X0	Set the RF_RECEIVED interrupt as active
		-	(24) RF_RXSTOP	0X0	Set the RF_RXSTOP interrupt as active
		-	(23) RF_TX	0X0	Set the RF_TX interrupt as active
		-	(22) BLE_COEX_IN_PROCESS	0X0	Set the BLE_COEX_IN_PROCESS interrupt as active
		-	(21) BLE_COEX_RX_TX	0X0	Set the BLE_COEX_RX_TX interrupt as active
		-	(20) BLE_SW	0X0	Set the BLE_SW interrupt as active
		-	(19) BLE_FINETGTIM	0X0	Set the BLE_FINETGTIM interrupt as active
		-	(18) BLE_GROSSTGTIM	0X0	Set the BLE_GROSSTGTIM interrupt as active
		-	(17) BLE_ERROR	0X0	Set the BLE_ERROR interrupt as active
		-	(16) BLE_CRYPT	0X0	Set the BLE_CRYPT interrupt as active
		-	(15) BLE_EVENT	0X0	Set the BLE_EVENT interrupt as active
		-	(14) BLE_RX	0X0	Set the BLE_RX interrupt as active
		-	(13) BLE_SLP	0X0	Set the BLE_SLP interrupt as active
		-	(12) BLE_CSCNT	0X0	Set the BLE_CSCNT interrupt as active
		-	(11) DSS7	0X0	Set the DSS7 interrupt as active
		-	(10) DSS6	0X0	Set the DSS6 interrupt as active
		-	(9) DSS5	0X0	Set the DSS5 interrupt as active
		-	(8) DSS4	0X0	Set the DSS4 interrupt as active
		-	(7) DSS3	0X0	Set the DSS3 interrupt as active
		-	(6) DSS2	0X0	Set the DSS2 interrupt as active
		-	(5) DSS1	0X0	Set the DSS1 interrupt as active
		-	(4) DSS0	0X0	Set the DSS0 interrupt as active
		-	(3) PCM_ERROR	0X0	Set the PCM_ERROR interrupt as active
		-	(2) PCM_TX	0X0	Set the PCM_TX interrupt as active
		-	(1) PCM_RX	0X0	Set the PCM_RX interrupt as active
		-	(0) DMIC_OD_ERROR	0X0	Set the DMIC_OD_ERROR interrupt as active

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E308	NVIC_IABR2	-	(8) BLE_AUDIO2	0X0	Set the BLE_AUDIO2 interrupt as active
		-	(7) BLE_AUDIO1	0X0	Set the BLE_AUDIO1 interrupt as active
		-	(6) BLE_AUDIO0	0X0	Set the BLE_AUDIO0 interrupt as active
		-	(5) MEM_ERROR	0X0	Set the MEM_ERROR interrupt as active
		-	(4) FLASH_ECC	0X0	Set the FLASH_ECC interrupt as active
		-	(3) FLASH_COPY	0X0	Set the FLASH_COPY interrupt as active
		-	(2) CLKDET	0X0	Set the CLKDET interrupt as active
		-	(1) AUDIO_SINK_PERIOD	0X0	Set the AUDIO_SINK_PERIOD interrupt as active
		-	(0) AUDIO_SINK_DELAY	0X0	Set the AUDIO_SINK_DELAY interrupt as active
0xE000E400	NVIC_IP0	(31:29) ADC_BATMON	(31:29) ADC_BATMON	0X0	Configure the ADC_BATMON interrupt priority
		(23:21) RTC_CLOCK	(23:21) RTC_CLOCK	0X0	Configure the RTC_CLOCK interrupt priority
		(15:13) RTC_ALARM	(15:13) RTC_ALARM	0X0	Configure the RTC_ALARM interrupt priority
		(7:5) WAKEUP	(7:5) WAKEUP	0X0	Configure the WAKEUP interrupt priority
0xE000E404	NVIC_IP1	(31:29) TIMER3	(31:29) TIMER3	0X0	Configure the TIMER3 interrupt priority
		(23:21) TIMER2	(23:21) TIMER2	0X0	Configure the TIMER2 interrupt priority
		(15:13) TIMER1	(15:13) TIMER1	0X0	Configure the TIMER1 interrupt priority
		(7:5) TIMER0	(7:5) TIMER0	0X0	Configure the TIMER0 interrupt priority
0xE000E408	NVIC_IP2	(31:29) DMA3	(31:29) DMA3	0X0	Configure the DMA3 interrupt priority
		(23:21) DMA2	(23:21) DMA2	0X0	Configure the DMA2 interrupt priority
		(15:13) DMA1	(15:13) DMA1	0X0	Configure the DMA1 interrupt priority
		(7:5) DMA0	(7:5) DMA0	0X0	Configure the DMA0 interrupt priority
0xE000E40C	NVIC_IP3	(31:29) DMA7	(31:29) DMA7	0X0	Configure the DMA7 interrupt priority
		(23:21) DMA6	(23:21) DMA6	0X0	Configure the DMA6 interrupt priority
		(15:13) DMA5	(15:13) DMA5	0X0	Configure the DMA5 interrupt priority
		(7:5) DMA4	(7:5) DMA4	0X0	Configure the DMA4 interrupt priority
0xE000E410	NVIC_IP4	(31:29) DIO3	(31:29) DIO3	0X0	Configure the DIO3 interrupt priority
		(23:21) DIO2	(23:21) DIO2	0X0	Configure the DIO2 interrupt priority
		(15:13) DIO1	(15:13) DIO1	0X0	Configure the DIO1 interrupt priority
		(7:5) DIO0	(7:5) DIO0	0X0	Configure the DIO0 interrupt priority
0xE000E414	NVIC_IP5	(31:29) SPI0_ERROR	(31:29) SPI0_ERROR	0X0	Configure the SPI0_ERROR interrupt priority
		(23:21) SPI0_TX	(23:21) SPI0_TX	0X0	Configure the SPI0_TX interrupt priority
		(15:13) SPI0_RX	(15:13) SPI0_RX	0X0	Configure the SPI0_RX interrupt priority
		(7:5) WATCHDOG	(7:5) WATCHDOG	0X0	Configure the WATCHDOG interrupt priority

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E418	NVIC_IP6	(31:29) I2C	(31:29) I2C	0X0	Configure the I2C interrupt priority
		(23:21) SPI1_ERROR	(23:21) SPI1_ERROR	0X0	Configure the SPI1_ERROR interrupt priority
		(15:13) SPI1_TX	(15:13) SPI1_TX	0X0	Configure the SPI1_TX interrupt priority
		(7:5) SPI1_RX	(7:5) SPI1_RX	0X0	Configure the SPI1_RX interrupt priority
0xE000E41C	NVIC_IP7	(31:29) DMIC_OUT_OD_IN	(31:29) DMIC_OUT_OD_IN	0X0	Configure the DMIC_OUT_OD_IN interrupt priority
		(23:21) UART_ERROR	(23:21) UART_ERROR	0X0	Configure the UART_ERROR interrupt priority
		(15:13) UART_TX	(15:13) UART_TX	0X0	Configure the UART_TX interrupt priority
		(7:5) UART_RX	(7:5) UART_RX	0X0	Configure the UART_RX interrupt priority
0xE000E420	NVIC_IP8	(31:29) PCM_ERROR	(31:29) PCM_ERROR	0X0	Configure the PCM_ERROR interrupt priority
		(23:21) PCM_TX	(23:21) PCM_TX	0X0	Configure the PCM_TX interrupt priority
		(15:13) PCM_RX	(15:13) PCM_RX	0X0	Configure the PCM_RX interrupt priority
		(7:5) DMIC_OD_ERROR	(7:5) DMIC_OD_ERROR	0X0	Configure the DMIC_OD_ERROR interrupt priority
0xE000E424	NVIC_IP9	(31:29) DSS3	(31:29) DSS3	0X0	Configure the DSS3 interrupt priority
		(23:21) DSS2	(23:21) DSS2	0X0	Configure the DSS2 interrupt priority
		(15:13) DSS1	(15:13) DSS1	0X0	Configure the DSS1 interrupt priority
		(7:5) DSS0	(7:5) DSS0	0X0	Configure the DSS0 interrupt priority
0xE000E428	NVIC_IP10	(31:29) DSS7	(31:29) DSS7	0X0	Configure the DSS7 interrupt priority
		(23:21) DSS6	(23:21) DSS6	0X0	Configure the DSS6 interrupt priority
		(15:13) DSS5	(15:13) DSS5	0X0	Configure the DSS5 interrupt priority
		(7:5) DSS4	(7:5) DSS4	0X0	Configure the DSS4 interrupt priority
0xE000E42C	NVIC_IP11	(31:29) BLE_EVENT	(31:29) BLE_EVENT	0X0	Configure the BLE_EVENT interrupt priority
		(23:21) BLE_RX	(23:21) BLE_RX	0X0	Configure the BLE_RX interrupt priority
		(15:13) BLE_SLP	(15:13) BLE_SLP	0X0	Configure the BLE_SLP interrupt priority
		(7:5) BLE_CSCNT	(7:5) BLE_CSCNT	0X0	Configure the BLE_CSCNT interrupt priority
0xE000E430	NVIC_IP12	(31:29) BLE_FINETGTIM	(31:29) BLE_FINETGTIM	0X0	Configure the BLE_FINETGTIM interrupt priority
		(23:21) BLE_GROSSTGTIM	(23:21) BLE_GROSSTGTIM	0X0	Configure the BLE_GROSSTGTIM interrupt priority
		(15:13) BLE_ERROR	(15:13) BLE_ERROR	0X0	Configure the BLE_ERROR interrupt priority
		(7:5) BLE_CRYPT	(7:5) BLE_CRYPT	0X0	Configure the BLE_CRYPT interrupt priority
0xE000E434	NVIC_IP13	(31:29) RF_TX	(31:29) RF_TX	0X0	Configure the RF_TX interrupt priority
		(23:21) BLE_COEX_IN_PROCESS	(23:21) BLE_COEX_IN_PROCESS	0X0	Configure the BLE_COEX_IN_PROCESS interrupt priority
		(15:13) BLE_COEX_RX_TX	(15:13) BLE_COEX_RX_TX	0X0	Configure the BLE_COEX_RX_TX interrupt priority
		(7:5) BLE_SW	(7:5) BLE_SW	0X0	Configure the BLE_SW interrupt priority

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E438	NVIC_IP14	(31:29) RF_TXFIFO	(31:29) RF_TXFIFO	0X0	Configure the RF_TXFIFO interrupt priority
		(23:21) RF_SYNC	(23:21) RF_SYNC	0X0	Configure the RF_SYNC interrupt priority
		(15:13) RF_RECEIVED	(15:13) RF_RECEIVED	0X0	Configure the RF_RECEIVED interrupt priority
		(7:5) RF_RXSTOP	(7:5) RF_RXSTOP	0X0	Configure the RF_RXSTOP interrupt priority
0xE000E43C	NVIC_IP15	(31:29) ASRC_OUT	(31:29) ASRC_OUT	0X0	Configure the ASRC_OUT interrupt priority
		(23:21) ASRC_IN	(23:21) ASRC_IN	0X0	Configure the ASRC_IN interrupt priority
		(15:13) ASRC_ERROR	(15:13) ASRC_ERROR	0X0	Configure the ASRC_ERROR interrupt priority
		(7:5) RF_RXFIFO	(7:5) RF_RXFIFO	0X0	Configure the RF_RXFIFO interrupt priority
0xE000E440	NVIC_IP16	(31:29) FLASH_COPY	(31:29) FLASH_COPY	0X0	Configure the FLASH_COPY interrupt priority
		(23:21) CLKDET	(23:21) CLKDET	0X0	Configure the CLKDET interrupt priority
		(15:13) AUDIO_SINK_PERIOD	(15:13) AUDIO_SINK_PERIOD	0X0	Configure the AUDIO_SINK_PERIOD interrupt priority
		(7:5) AUDIO_SINK_DELAY	(7:5) AUDIO_SINK_DELAY	0X0	Configure the AUDIO_SINK_DELAY interrupt priority
0xE000E444	NVIC_IP17	(31:29) BLE_AUDIO1	(31:29) BLE_AUDIO1	0X0	Configure the BLE_AUDIO1 interrupt priority
		(23:21) BLE_AUDIO0	(23:21) BLE_AUDIO0	0X0	Configure the BLE_AUDIO0 interrupt priority
		(15:13) MEM_ERROR	(15:13) MEM_ERROR	0X0	Configure the MEM_ERROR interrupt priority
		(7:5) FLASH_ECC	(7:5) FLASH_ECC	0X0	Configure the FLASH_ECC interrupt priority
0xE000E448	NVIC_IP18	(7:5) BLE_AUDIO2	(7:5) BLE_AUDIO2	0X0	Configure the BLE_AUDIO2 interrupt priority
0xE000EF00	NVIC_STIR	(7:0) INTID	-	N/A	The interrupt number to trigger

A.28 SYSTEM CONTROL BLOCK

Address	Register Name	Register Write	Register Read	Default	Description
0xE000ED00	SCB_CPUID	-	(31:24) IMPLEMENTER	0X41	Implementer code
		-	(23:20) VARIANT	0X2	Implementation variant
		-	(19:16) ARCHITECTURE	0XF	Architecture number
		-	(15:4) PARTNO	0XC23	Part number
		-	(3:0) REVISION	0X1	Revision code
0xE000ED04	SCB_ICSR	(31) NMIPENDSET	(31) NMIPENDSET	0X0	Indicates the NMI exception is pending
		(28) PENDSVSET	(28) PENDSVSET	0X0	Write 1 to set pending status for PendSV exception. Read indicates PendSV pending status
		(27) PENDSVCLR	-	N/A	Write 1 to clear PendSV exception pending status
		(26) PENDSTSET	(26) PENDSTSET	0X0	Write 1 to set pending status for SYSTICK exception. Read value indicates SYSTICK pending status
		(25) PENDSTCLR	-	N/A	Write 1 to clear SYSTICK exception pending status
		-	(23) ISRPREEMPT	0X0	Indicates that a pending interrupt is going to be active in the next step
		-	(22) ISRPENDING	0X0	Indicates that an external interrupt is pending
		-	(21:12) VECTPENDING	0X0	Pending external interrupt number
		-	(11) RETTOBASE	0X0	Set to 1 when the an exception handler is being run.
		-	(9:0) VECTACTIVE	0X0	Number of current running interrupt service routine
0xE000ED08	SCB_VTOR	(31:7) TBLOFF	(31:7) TBLOFF	0X0	Table offset value in code or RAM region. Must be multiple of 128.
0xE000ED0C	SCB_AIRCR	(31:16) VECTKEY	(31:16) VECTKEY	0XFA05	Access key for writing this register. Must be set to 0x05FA to write the other register fields.
		-	(15) ENDIANESS	0X0	Indicates endianess for data.
		(10:8) PRIGROUP	(10:8) PRIGROUP	0X7	Priority group setting. Controls how many bits in the priority register are used for pre-empty priority vs. sub-priority.
		(2) SYSRESETREQ	-	N/A	Requests a chip-level system reset
		(1) VECTCLRACTIVE	-	N/A	Clears all active exception state information
		(0) VECTRESET	-	N/A	Reset the CM3
0xE000ED10	SCB_SCR	(4) SEVONPEND	(4) SEVONPEND	0X0	Set to 1 to cause the WFE to wake up if a new interrupt is pended
		(2) SLEEPDEEP	(2) SLEEPDEEP	0X0	Enable SLEEPDEEP output signal when entering sleep mode
		(1) SLEEPONEXIT	(1) SLEEPONEXIT	0X0	Enable sleep on exit feature when returning from handler to thread mode

Address	Register Name	Register Write	Register Read	Default	Description
0xE000ED14	SCB_CCR	(9) STKALIGN	(9) STKALIGN	0X0	Force stack to be double-word aligned
		(8) BFHFNMIGN	(8) BFHFNMIGN	0X0	Ignore data bus faults in hard fault and NMI exceptions
		(4) DIV_0_TRP	(4) DIV_0_TRP	0X0	Trap on divide by zero
		(3) UNALIGN_TRP	(3) UNALIGN_TRP	0X0	Trap on unaligned data access
		(1) USERSETMPEND	(1) USERSETMPEND	0X0	Allow user code to write to software trigger interrupt register
		(0) NONBASETHRDENA	(0) NONBASETHRDENA	0X0	Non base thread mode enable
0xE000ED18	SCB_SHP0	(23:16) NVIC_USAGE_FAULT_PRIORITY	(23:16) NVIC_USAGE_FAULT_PRIORITY	0X0	Configure the usage fault priority
		(15:8) NVIC_BUS_FAULT_PRIORITY	(15:8) NVIC_BUS_FAULT_PRIORITY	0X0	Configure the bus fault priority
		(7:0) NVIC_MEM_FAULT_PRIORITY	(7:0) NVIC_MEM_FAULT_PRIORITY	0X0	Configure the memory fault priority
0xE000ED1C	SCB_SHP1	(31:24) NVIC_SVC_PRIORITY	(31:24) NVIC_SVC_PRIORITY	0X0	Configure the SVC priority
0xE000ED20	SCB_SHP2	(31:24) NVIC_SYSTICK_PRIORITY	(31:24) NVIC_SYSTICK_PRIORITY	0X0	Configure the SysTick interrupt priority
		(23:16) NVIC_PENDSV_PRIORITY	(23:16) NVIC_PENDSV_PRIORITY	0X0	Configure the PendSV priority
		(7:0) NVIC_MONITOR_PRIORITY	(7:0) NVIC_MONITOR_PRIORITY	0X0	Configure the monitor priority
0xE000ED24	SCB_SHCSR	(18) USGFAULTENA	(18) USGFAULTENA	0X0	Usage fault handler enable
		(17) BUSFAULTENA	(17) BUSFAULTENA	0X0	Bus fault handler enable
		(16) MEMFAULTENA	(16) MEMFAULTENA	0X0	Memory management fault handler enable
		(15) SVCALLPENDED	(15) SVCALLPENDED	0X0	SVCall is pending or was started and replaced by a higher priority exception
		(14) BUSFAULTPENDED	(14) BUSFAULTPENDED	0X0	Bus fault is pending or was started and replaced by a higher priority exception
		(13) MEMFAULTPENDED	(13) MEMFAULTPENDED	0X0	Memory management fault is pending or was started and replaced by a higher priority exception
		(12) USGFAULTPENDED	(12) USGFAULTPENDED	0X0	Usage fault is pending or was started and replaced by a higher priority exception
		(11) SYSTICKACT	(11) SYSTICKACT	0X0	SYSTICK exception handler is active
		(10) PENDSVACT	(10) PENDSVACT	0X0	PendSV exception handler is active
		(8) MONITORACT	(8) MONITORACT	0X0	Debug monitor exception handler is active
		(7) SVCALLACT	(7) SVCALLACT	0X0	SVCall exception handler is active
		(3) USGFAULTACT	(3) USGFAULTACT	0X0	Usage fault exception handler is active
		(1) BUSFAULTACT	(1) BUSFAULTACT	0X0	Bus fault exception handler is active
		(0) MEMFAULTACT	(0) MEMFAULTACT	0X0	Memory management fault exception handler is active

Address	Register Name	Register Write	Register Read	Default	Description
0xE000ED28	SCB_CFSR	(25) DIVBYZERO	(25) DIVBYZERO	0X0	Indicates divide by zero will take place
		(24) UNALIGNED	(24) UNALIGNED	0X0	Indicates unaligned access will take place
		(19) NOCP	(19) NOCP	0X0	Attempt to execute a coprocessor instruction
		(18) INVPC	(18) INVPC	0X0	Attempt to do exception with bad value in EXC_RETURN number
		(17) INVSTATE	(17) INVSTATE	0X0	Attempt to switch to invalid (e.g. ARM) state
		(16) UNDEFINSTR	(16) UNDEFINSTR	0X0	Attempt to execute an undefined instruction
		-	(15) BFARVALID	0X0	Indicates if bus fault address register is valid
		(12) STKERR	(12) STKERR	0X0	Indicates stacking error
		(11) UNSTKERR	(11) UNSTKERR	0X0	Indicates unstacking error
		(10) IMPRECISERR	(10) IMPRECISERR	0X0	Indicates imprecise data access violation
		(9) PRECISERR	(9) PRECISERR	0X0	Indicates precise data access violation
		(8) IBUSERR	(8) IBUSERR	0X0	Indicates instruction access violation
		-	(7) MMARVALID	0X0	Indicates if memory management fault address register is valid
		(4) MSTKERR	(4) MSTKERR	0X0	Indicates stacking error
		(3) MUNSTKERR	(3) MUNSTKERR	0X0	Indicates unstacking error
		(1) DACCVIOL	(1) DACCVIOL	0X0	Indicates data access violation
		(0) IACCVIOL	(0) IACCVIOL	0X0	Indicates instruction access violation
0xE000ED2C	SCB_HFSR	(31) DEBUGEVT	(31) DEBUGEVT	0X0	Indicates hard fault is triggered by debug event
		(30) FORCED	(30) FORCED	0X0	Indicates hard fault is taken because of a lower priority (e.g., bus, memory management or usage) fault
		(1) VECTBL	(1) VECTBL	0X0	Indicates hard fault is taken due to failed vector fetch
0xE000ED30	SCB_DFSR	(4) EXTERNAL	(4) EXTERNAL	0X0	Indicates external debug request signal asserted
		(3) VCATCH	(3) VCATCH	0X0	Indicates vector fetch occurred
		(2) DWTTRAP	(2) DWTTRAP	0X0	Indicates DWT match occurred
		(1) BKPT	(1) BKPT	0X0	Indicates BKPT instruction executed
		(0) HALTED	(0) HALTED	0X0	Indicates halt requested by NVIC
0xE000ED34	SCB_MMFAR	-	(31:0) NVIC_MMAR	0X0	
0xE000ED38	SCB_BFAR	-	(31:0) NVIC_BFAR	0X0	

A.29 DEBUG CONTROLLER

Address	Register Name	Register Write	Register Read	Default	Description
0xE000EDF0	DEBUG_DHCSR	(31:16) DBGKEY	-	N/A	Debug key must be written to this field in order to write the rest of the register
		-	(25) S_RESET_ST	0X0	Core has been reset or is being rest. Bit is cleared on read.
		-	(24) S_RETIRE_ST	0X0	
		-	(19) S_LOCKUP	0X0	Indicates if core is in lockup state
		-	(18) S_SLEEP	0X0	Indicates if core is in sleep mode
		-	(17) S_HALT	0X0	Indicates is core is halted
		-	(16) S_REGRDY	0X0	Indicates register read/write operation is completed
		(5) C_SNAPSTALL	(5) C_SNAPSTALL	0X0	Set to break a stalled memory access
		(3) C_MASKINTS	(3) C_MASKINTS	0X0	Mask interrupts while stepping
		(2) C_STEP	(2) C_STEP	0X0	Single step the processor
		(1) C_HALT	(1) C_HALT	0X0	Halt the processor
		(0) C_DEBUGEN	(0) C_DEBUGEN	0X1	Enable halt mode debugging
0xE000EDF4	DEBUG_DCRSR	(16) REGWnR	-	N/A	Indicates direction of register transfer
		(4:0) REGSEL	-	N/A	Indicates register to be accessed
0xE000EDF8	DEBUG_DCRDR	(31:0) DEBUG_REGDATA	(31:0) DEBUG_REGDATA	0X0	Register read/write data for debugging
0xE000EDFC	DEBUG_DEMCR	(24) TRCENA	(24) TRCENA	0X0	Trace system enable
		(19) MON_REQ	(19) MON_REQ	0X0	Indicates that the debug monitor is caused by a manual pending request rather than a hardware event
		(18) MON_STEP	(18) MON_STEP	0X0	Single step the processor
		(17) MON_PEND	(17) MON_PEND	0X0	Pend the monitor exception request
		(16) MON_EN	(16) MON_EN	0X0	Enable the debug monitor exception
		(10) VC_HARDERR	(10) VC_HARDERR	0X0	Debug trap on hard faults
		(9) VC_INTERR	(9) VC_INTERR	0X0	Debug trap on interrupt service errors
		(8) VC_BUSERR	(8) VC_BUSERR	0X0	Debug trap on bus faults
		(7) VC_STATERR	(7) VC_STATERR	0X0	Debug trap on usage fault state errors
		(6) VC_CHKERR	(6) VC_CHKERR	0X0	Debug trap on fault-enabled checking errors (e.g. unaligned access, divide by zero, etc.)
		(5) VC_NOCPERR	(5) VC_NOCPERR	0X0	Debug trap on usage fault no coprocessor errors
		(4) VC_MMERR	(4) VC_MMERR	0X0	Debug trap on memory management fault
		(0) VC_CORERESET	(0) VC_CORERESET	0X0	Debug trap on core reset

APPENDIX B
Glossary

The following abbreviations and terms are used in this manual:

ACS	analog control system
ADC	analog-to-digital converter
AFE	analog front-end
CRC	cyclic redundancy check
DAC	digital-to-analog converter
DIO	digital input/output
DMA	direct memory access
ECC	error correcting code
GAP	generic access profile
GPIO	general-purpose input/output
HIZ	high impedance
I^2C	inter-IC communication protocol
I^2S	inter-IC sound protocol
INL	integral non-linearity
JTAG	joint test action group (developer of IEEE standard 1149.1-1990)
L2CAP	logical link control and adaptation protocol
LDO	low dropout voltage regulator
LSB	least significant bit
MCU	microcontroller unit
MSB	most significant bit
MUX	multiplexer, selector of one signal from many
NVIC	nested vectored interrupt controller
PCM	pulse code modulation

PLL	phase-locked loop
PMU	power management unit
PWM	pulse width modulation
POR	power-on-reset
RAM	random-access memory
RFFE	radio-frequency front-end
ROM	read-only memory
RTC	real-time clock
SCL	serial clock (part of I^2C bus)
SDA	serial data (part of I^2C bus)
SPI	serial peripheral interface
SWD	serial wire debug, two-wire interface used for communication with Arm cores
SWJ-DP	serial wire and JTAG debug port
UART	universal asynchronous receiver-transmitter
VCO	voltage-controlled oscillator
VDDA	supply voltage for the non-RF analog blocks and flash memory
VDDC	supply voltage for the digital logic
VDDM	supply voltage for the memories
VDDO	input supply for the digital I/O pads, including the debug port (SWJ-DP)
VDDPA	optional supply voltage for the power amplifier from the RF front-end
VDDRF	supply voltage for the RF front-end
WDF	wave digital filter
XTAL	crystal, generally quartz-based

Sections of this manual relating to the Arm Cortex-M3 processor have been republished from the *Cortex-M3 Technical Reference* (version r2p1) with permission.Sections of this manual relating to the Arm Cortex-M3 processor have been republished from the *Cortex-M3 Technical Reference* (version r2p1) with permission.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Literature Distribution Center for ON Semiconductor
19521 E. 32nd Pkwy, Aurora, Colorado 80011 USA
Phone: 303-675-2175 or 800-344-3860 Toll Free USA/Canada
Fax: 303-675-2176 or 800-344-3867 Toll Free USA/Canada
Email: orderlit@onsemi.com

N. American Technical Support: 800-282-9855 Toll Free USA/Canada
Europe, Middle East and Africa Technical Support: Phone: 421 33 790 2910

ON Semiconductor Website: www.onsemi.com

Order Literature: http://www.onsemi.com/orderlit

For additional information, please contact your local Sales Representative